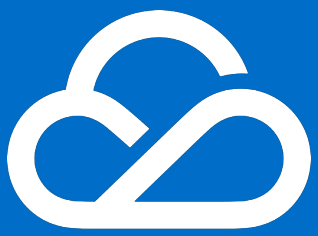
微视频

开发者手册

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

API

1 基本概念

概念	解释
appid	接入微视频时,生成为唯一id, 用于唯一标识接入业务, 获取地址: 密钥配置
Authorization	签名，具体生成参见鉴权签名方法.
bucket_name	bucket名称，bucket创建参见创建Bucket

2 鉴权

腾讯云•微视频通过签名来验证请求的合法性。开发者通过将签名授权给客户端，使其具备上传下载及管理指定资源的能力。

签名分为单次签名和多次签名, 区别为: 如果针对资源进行写操作(资源删除), 那么这个签名必须是单次有效的.重复使用该签名则会返回签名失败.如果是上传或下载资源,签名必须是多次有效的.有效时长最多为三个月。

开发者可以通过服务器SDK文档
生成签名，也可以参考我们的签名函数自行生成签名，具体生成方式详见鉴权签名方法。

3 目录操作

3.1 创建目录

功能: 在指定路径下创建目录。
接口:web.video.myqcloud.com/files/v1/[appid]/[bucket_name]/[dirName]/ <- 有文件夹斜杠 /
要求: 父目录存在
方法: POST
请求参数HTTP头部信息:

参数名称	必选	类型	描述
Host	是	String	微视频服务器域名，固定为 web.video.myqcloud.co

参数名称	必选	类型	描述
			m
Content-Type	是	String	application/json
Authorization	是	String	多次有效签名,用于鉴权，具体生成方式详见鉴权签名方法

请求包体 (json):

参数名称	必选	类型	描述
op	是	String	操作类型.固定填”create”
biz_attr	否	String	目录属性，业务端维护

返回包体(json):

参数名称	子属性	必选	类型	描述
code	-	是	Int	服务端返回码
message	-	是	String	服务端提示内容
data	-	是	集合	服务器返回的应答数据
	ctime	是	Unix时间戳	创建时间
	resource_path	是	String	资源路径. 不包含/[appid]/[bucket_name]

3.2 创建视频:(完整上传)

功能: 在指定路径下创建视频。

接口:web.video.myqcloud.com/files/v1/[appid]/[bucket_name]/[DirName]/[file_name] <-没有文件夹斜杠 /

方法: POST

请求参数HTTP头部信息:

参数名称	必选	类型	描述
Content-Length	是	Int	整个multipart/form-data内容的总长度，单位：字节（Byte）
Content-Type	是	String	固定为multipart/form-data
Authorization	是	String	多次有效签名,用于鉴权，具体生成 方式详见鉴权签名方法

请求包体信息 (multipart/form-data):

参数名称	必选	类型	描述
op	是	String	固定填upload
filecontent	是	Binary	视频文件内容
sha	否	String	视频文件的sha值
biz_attr	否	String	文件属性，业务端维护
video_cover	否	String	视频封面的URL
video_title	否	String	视频标题
video_desc	否	String	视频描述
magicContext	否	String	转码成功后,用于透传回调用者的业务后台

返回包体信息(json):

参数名称	子属性	必选	类型	描述
code	-	是	Int	服务端返回码
message	-	是	String	服务端提示内容
data	-	是	集合	服务器返回的应答数据
	access_url	是	String	生成的文件下载url
	url	是	String	操作文件的url
	resource_path	是	String	资源路径. 不包

				含/[appid]/[bucket_name]

3.3 创建视频:(分片上传, 第一片)

功能: 在指定路径下创建视频。

接口:web.video.myqcloud.com/files/v1/[appid]/[bucket_name]/[DirName]/[file_name] <-没有文件夹斜杠 /

方法: POST

请求参数HTTP头部信息:

参数名称	必选	类型	描述
Content-Length	是	Int	整个multipart/form-data内容的总长度，单位：字节（Byte）
Content-Type	是	String	固定为multipart/form-data
Authorization	是	String	多次有效签名,用于鉴权，具体生成方式详见鉴权签名方法

请求包体信息 (multipart/form-data):

参数名称	必选	类型	描述
op	是	String	固定填upload_slice
filesize	是	Int 64	视频文件总大小
sha	否	String	文件的sha值,必须提供
biz_attr	否	String	视频属性，业务端维护
video_cover	否	String	视频封面的URL
session	否	String	如果想要断点续传,则带上上一次的session id
video_title	否	String	视频标题
video_desc	否	String	视频描述
magicContext	否	String	转码成功后,用于透传回调

参数名称	必选	类型	描述
			用者的业务后台

返回包体信息(json):

参数名称	子属性	必选	类型	描述
code	-	是	Int	服务端返回码
message	-	是	String	服务端提示内容
data	-	是	集合	服务器返回的应答数据
	session	否(非秒传的大部分情况会有)	String	唯一标识此视频文件传输过程的id
	offset	否(非秒传的大部分情况会有)	Int 64	开始传输的位移
	slice_size	否(非秒传的大部分情况会有)	Int	分片大小
	access_url	否(上一次已传完/秒传成功)	String	生成的文件下载url
	url	否(上一次已传完/秒传成功)	String	操作文件的url
	resource_path	否(上一次已传完/秒传成功)	String	资源路径. 不包含/[appid]/[bucket_name]

3.4 创建视频:(分片上传, 后续分片)

功能: 在指定路径下创建视频。

接口:web.video.myqcloud.com/files/v1/[appid]/[bucket_name]/[DirName]/[file_name] <- 有文件夹斜杠/

方法: POST

请求参数HTTP头部信息:

参数名称	必选	类型	描述

参数名称	必选	类型	描述
Content-Length	是	Int	整个multipart/form-data内容的总长度，单位：字节（Byte）
Content-Type	是	String	固定为multipart/form-data
Authorization	是	String	多次有效签名,用于鉴权，具体生成 方式详见鉴权签名方法

请求包体信息 (multipart/form-data):

参数名称	必选	类型	描述
op	是	String	固定填upload_slice
filecontent	是	Binary	视频文件内容
sha	否	String	本次文件分片的sha值,可以提供用于校验(暂时未启用)
session	是	String	唯一标识此视频文件传输过程的id, 由后台下发, 调用方透传
offset	是	Int 64	本次分片位移

返回包体信息(json):

参数名称	子属性	必选	类型	描述
code	-	是	Int	服务端返回码
message	-	是	String	服务端提示内容
data	-	是	集合	服务器返回的应答数据
	session	否(非秒传的大部分情况)	String	唯一标识此视频文件传输过程的id
	offset	否(非秒传的大部分情况)	Int 64	请求包体里的传输的位移,调用

				方如果用多线程等方式传输,可以用来唯一确定本次分片结果
	access_url	否(上一次已传完/秒传成功)	String	生成的文件下载url
	url	否(上一次已传完/秒传成功)	String	操作文件的url
	resource_path	否(上一次已传完/秒传成功)	String	资源路径. 不包含/[appid]/[bucket_name]

3.5 目录列表,前缀搜索

功能: 在指定路径下搜索视频或目录。 接口: web.video.myqcloud.com/files/v1/[appid]/[bucket_name]/
web.video.myqcloud.com/files/v1/[appid]/[bucket_name]/[DirName]/
web.video.myqcloud.com/files/v1/[appid]/[bucket_name]/[DirName]/[prefix] <- 如果填写prefix,则列出含此前缀的所有文件

方法: GET

Request Param(query string):

参数名称	必选	类型	描述
op	否	String	操作类型.可以不填,如果要填,固定填”list”
num	是	Int	拉取的总数
pattern	否	String	eListBoth, eListDirOnly, eListFileOnly (默认eListBoth)
order	否	Int	默认正序(=0), 填1为反序,
context	否	String	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ord

参数名称	必选	类型	描述
			er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。
Authorization	是	String	多次有效签名,用于鉴权，具体生成方式详见鉴权签名方法

Response (json):

参数名称	子属性	其他属性	必选	类型	描述
code	-	-	是	Int	服务端返回码
message	-	-	是	String	服务端提示内容
data	-	-	是	集合	服务器返回的应答数据
	context	-	是	String	透传字段,用于翻页,前端不需理解,需要往前/往后翻页则透传回来
	has_more	-	是	Bool	是否有内容可以继续往前/往后翻页
	dircount	-	是	Int	子目录数量(总)
	filecount	-	是	Int	子视频文件数量(总)
	infos	-	是	Array	可以为空
		name	是	String	视频文件名
		biz_attr	否	String	目录/视频文件属性，业

			务端维护
video_cover	否(当类型为视频文件时返回)	String	视频封面的URL
filesize	否(当类型为视频文件时返回)	Int	视频文件大小
filelen	否(当类型为视频文件时返回)	Int	文件已传输大小(通过与filesize对比可知文件传输进度)
sha	否(当类型为视频文件时返回)	String	文件sha
ctime	是	Unix时间戳	创建时间
mtime	是	Unix时间戳	修改时间
access_url	否(当类型为视频文件时返回)	String	生成的资源可访问的url
trans_status	否，视频才有，目录没有	Json	

SDK

SDK下载

1. Android SDK

版本号：Android SDK V1.1.4.1

发布时间：2016-07-11

版本说明：

优化代码，修改bug

下载地址：Android SDK V1.1.4.1（含说明文档）

接入文档：微视频Android-SDK文档

版本号：Android SDK V1.1.3.3

发布时间：2015-12-17

版本说明：

视频信息可以上传视频封面

优化代码，修改bug

下载地址：Android SDK V1.1.3.3（含说明文档）| Android 体验 Demo

版本号：Android SDK V1.1.3

发布时间：2015-07-24

版本说明：

1. 全新发布

   下载地址：Android SDK V1.1.3

   接入文档：微视频Android-SDK文档

2. iOS SDK

版本号：iOS SDK V1.1.3.4

发布时间：2016-07-11

版本说明：

修正部分bug

下载地址：iOS SDK V1.1.3.4

接入文档：微视频iOS-SDK文档

版本号：iOS SDK V1.1.3.3
发布时间：2015-12-17
版本说明：
视频信息可以上传视频封面（TXYVideoFileInfo增加coverUrl 属性）
上传SDK添加支持了 bitecode 版本
下载地址：iOS SDK V1.1.3.3（含说明文档）| iOS 体验 Demo

版本号：iOS SDK V1.1.3
发布时间：2015-08-07
版本说明：

1. 全新发布
   下载地址：iOS SDK V1.1.3
   接入文档：微视频iOS-SDK文档

3. 服务器SDK

3.1 Java SDK

版本号：mvs-java-sdk v1.0.0
发布时间：2015-07-31
下载地址：https://github.com/tencentyun/mvs-java-sdk.git

3.2 PHP SDK

版本号：mvs-php-sdk v1.0.0
发布时间：2015-07-31
下载地址：https://github.com/tencentyun/mvs-php-sdk.git

3.3 Python SDK

版本号：mvs-python-sdk v1.0.0

发布时间：2015-07-31

下载地址：https://github.com/tencentyun/mvs-python-sdk.git

3.4 Nodejs SDK

版本号：mvs-nodejs-sdk v1.0.0

发布时间：2015-07-31

下载地址：https://github.com/tencentyun/mvs-nodejs-sdk.git

3.5 C++ SDK

版本号：mvs-cpp-sdk v1.0.0

发布时间：2015-08-03

下载地址：https://github.com/tencentyun/mvs-cpp-sdk.git

3.6 C# SDK

版本号：mvs-dotnet-sdk v1.0.0

发布时间：2015-08-03

下载地址：https://github.com/tencentyun/mvs-dotnet-sdk.git

Android-SDK说明

1 注册APP

在腾讯云页面上注册APP信息，获取APPID。

2 工程配置

2.1 导入SDK

将SDK包中的libs目录合并到本地工程的libs目录，然后配置工程导入所有jar包。

上传SDK的libs目录如下：

PS：如果工程中含有armeabi-
v7a，则上述so也需要拷贝一份到此目录下，否则由于android系统的问题在安装apk之后会找不到so。

so兼容x86架构，若项目需要兼容x86，则将sdk中的so复制一份放入x86目录下即可。

下载SDK的libs目录如下：

2.2 配置manifest

SDK需要网络访问相关的一些权限，需要在manifest中进行权限声明如下所示：

```
<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"/>
```

3 上传SDK

3.1 初始化

在使用上传功能之前需要先初始化，目前支持文件、图片和视频三种业务，用户根据需求进行注册，初始化分为两步:

1. 创建UploadManager对象

- 原型

```
/**
 * 业务类型
 * File-文件业务，Photo-图片业务，Video视频业务，Audio暂时未支持
 *
 */
public enum FileType {
   File, Photo, Audio, Video, Other
}
/**
 * 构造方法
 * @param context
 * @param appid     腾讯云注册的APPID
 * @param fileType  业务类型
 * @param persistenceId  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继续进行上传；传
 *                       入为Null，则不会进行持久化保存
 */
```

```
public
 UploadManager(Context context, String appid, FileType fileType, String persiste
nceId)
```

- 示例

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

// 实例化视频业务上传管理类
UploadManager  videoUploadMgr = null;
videoUploadMgr = new
 UploadManager(context, APPID,  FileType.Video, "qcloudvideo");
```

3.2 视频上传

上传视频的步骤如下:

1. 创建FileUploadTask对象

2. 调用UploadManager的upload方法，将FileUploadTask对象传入

3. 原型

```
public VideoUploadTask(
String bucket, String srcFilePath, String destFilePath, String bizAttr, VideoAtt
r videoAttr,int to_over_write, IUploadTaskListener listener);

/** 上传回调监听器 */
public interface IUploadTaskListener {
```

```
    // 上传成功
    void onUploadSucceed(FileInfo result);

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

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

    // 上传任务状态变化
    void onUploadStateChange(TaskState state);
}
```

- 示例

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

FileUploadTask task = new
 VideoUploadTask(String bucket, String srcFilePath, String destFilePath, String
bizAttr, VideoAttr videoAttr,int to_over_write, IUploadTaskListener listener)
    new IUploadTaskListener() {

    @Override
    public void onUploadSucceed(final FileInfo result) {
        Log.i("Demo", "upload succeed: " + result.url);
    }

    @Override
    public void onUploadStateChange(TaskState state) {
    }

    @Override
    public void onUploadProgress(long totalSize, long sendSize){
```

```
            long p = (long) ((sendSize * 100) / (totalSize * 1.0f));
            Log.i("Demo", "上传进度: " + p + "%");
    }

    @Override
    public void onUploadFailed(final int errorCode, final String errorMsg){
            Log.i("Demo", "上传结果:失败! ret:" + errorCode + " msg:" + errorMsg);
    }
    }
);
task.setAuth(VIDEO_SIGN);
videoUploadMgr.upload(task);  // 开始上传
```

3.3 暂停、恢复、取消上传

上传任务可以暂停、恢复或者取消，只需要传入相应的taskId即可，上传任务的状态变化会通过IUploadTaskListener通知。

- 原型

```
/**
 * 暂停指定的上传任务
 * @param taskId 对应UploadTask的任务ID
 * @return 成功返回True，失败返回False；如果传入的taskId没有对应的UploadTask，则失败
 */
public boolean pause(int taskId);

/**
 * 恢复指定的上传任务，重新发送
 * @param taskId 对应UploadTask的任务ID
 * @return 成功返回True，失败返回False；如果传入的taskId没有对应的UploadTask，则失败
 */
public boolean resume(int taskId);
```

```
/**
 * 取消指定的上传任务
 * @param taskId 对应UploadTask的任务ID
 * @return 成功返回True，失败返回False；如果传入的taskId没有对应的UploadTask，则失败
 */
public boolean cancel(int taskId);
```

- 示例

```
int taskId= task.getTaskId();
videoUploadMgr.pause(taskId);  // 暂停上传
videoUploadMgr.resume(taskId); // 恢复上传
videoUploadMgr.cancel(taskId); // 取消上传
```

3.4 视频文件查询

查询视频文件的详细信息，步骤如下:

1. 通过文件url创建FileStatTask对象
2. 调用UploadManager的sendCommand方法，将FileStatTask对象传入
3. 在FileStatTask.Ilistener的回调中获取查询结果

```
/**
 * 文件查询任务，视频业务使用
 * @param file_id    查询的文件ID
 * @param fileType   查询的文件类型
 * @param bucket     查询的文件所属Bucket
 * @param listener   结果回调
 */
public
 FileStatTask(String file_id, FileType fileType, String bucket, IListener listen
```

er)

- 原型

```
// FileInfo -- 文件详细信息Key
String _file_url = "file_url";                    //文件url
String _file_fileid = "file_fileid";              //文件key
String _file_upload_time = "file_upload_time";    //上传时间
String _file_size = "file_size";                  //文件大小
String _file_md5 = "file_md5";                    //存储中文件md5
String _file_type = "file_type";                  //视频文件
String _video_status = "video_status";            //视频状态VideoStatus
String _video_play_time = "video_play_time";      //视频播放时长
String _video_title = "video_title";              //视频标题
String _video_desc = "video_desc";                //视频描述
String _video_cover_url = "video_cover_url";      //视频封面url
```

- 示例

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

FileStatTask task = new FileStatTask(fileId, FileType.Video, BUCKET,
    new FileStatTask.IListener() {
        @Override
        public void onSuccess(final FileInfo result) {
            Log.i("Demo", "MD5:" + result.extendInfo.get("_file_md5")
                + "\nWidth : " + result.extendInfo.get("_video_play_time")
                + "\nHeight: " + result.extendInfo.get("_video_title"));
        }

        @Override
    public void onFailure(final int ret, final String msg) {
```

```
        Log.e("Demo", "查询结果:失败! ret:" + ret + " msg:" + msg);
    }
});

task.setAuth(VIDEO_SIGN);
videoUploadMgr.upload(task)
```

3.5 文件删除

删除文件步骤如下:

1. 通过文件url创建FileDeleteTask对象
2. 调用UploadManager的sendCommand方法，将FileDeleteTask对象传入
3. 在FileDeleteTask.Ilistener的回调中获取查询结果文件复制
4. 原型

```
/**
 * 文件删除任务，视频业务使用
 * @param file_id  删除的文件ID
 * @param fileType 删除的文件类型
 * @param bucket   删除的文件所属Bucket
 * @param listener 删除结果回调
 */
public
 FileDeleteTask(String file_id, FileType fileType, String bucket, IListener list
ener)
```

- 示例

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

FileDeleteTask task= new FileDeleteTask(fileId, FileType.Video, BUCKET,
```

```
    new FileDeleteTask.IListener() {
    @Override
    public void onSuccess(Void result) {
        Log.e("Demo", "删除结果:成功!");
    }

    @Override
    public void onFailure(final int ret, final String msg) {
        Log.e("Demo", "删除结果:失败! ret:" + ret + " msg:" + msg);
    }
});

task.setAuth(VIDEO_SIGN);
videoUploadMgr.upload(task);
```

3.6 文件复制

复制文件步骤如下:

1. 通过文件url创建FileCopyTask对象
2. 调用UploadManager的sendCommand方法，将FileCopyTask对象传入
3. 在FileCopyTask.Ilistener的回调中获取查询结果文件复制
4. 原型

```
/**
*  文件复制任务，视频业务使用
 * @param fileType     文件类型
 * @param bucket       文件所在Bucket
 * @param src_fileId   源文件的fileid
 * @param dst_fileId   复制后新文件的fileid
 * @param listener
 */
public
```

```
 FileCopyTask(FileType fileType, String bucket, String src_fileId, String dst_fi
leId, IListener listener)
```

- 示例

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

FileCopyTask task = new FileCopyTask(FileType.Video, BUCKET, fileId,
        fileId + "_copy", new FileCopyTask.IListener() {
    @Override
    public void onSuccess(final String result) {
        Log.e("Demo", "复制结果:成功! url:" + result);
    }

    @Override
    public void onFailure(final int ret, final String msg) {
        Log.e("Demo", "复制结果:失败! ret:" + ret + " msg:" + msg);
    }
});

task.setAuth(VIDEO_SIGN);
videoUploadMgr.upload(task);
```

3.7 日志上报

SDK会将上传过程中的日志保存到本地文件中，以便当用户上传过程中遇到问题时可以直接通过日志文件进行详细定位分析，SDK提供了日志上报接口，可以将指定日期的日志上报到腾讯云后台。

- 原型

```
/**
 * 上报formDate到toDate对应的日志
```

```
 * @param appid      腾讯云注册的APPID
 * @param fromDate   开始上报时间（时间单位：天）
 * @param toDate     结束上报时间（时间单位：天）
 * @return 成功返回True，失败返回False
 */
public static boolean uploadLog(String appid, Date fromDate, Date toDate);
```

- 示例

```
// 上报一天前的日志
Date beginDate = new Date(System.currentTimeMillis() - 24 * 3600 * 1000);
Date endDate = beginDate;
videoUploadMgr.uploadLog(APPID, beginDate, endDate);
```

4 下载SDK

4.1 初始化

- 函数原型

```
/**
 * 构造方法
 * @param context         Android Context
 * @param appid           腾讯云注册的APPID
 * @param persistenceId 每个Download实例需要分配一个唯一的id，该ID用于区分临时缓存
 *                              目录
 */
public Downloader(Context context, String appid, String persistenceId);
```

- 示例

```
// 实例化下载管理类
Downloader  downloader = new Downloader(this, APPID, "TestDownloader");
```

4.2 下载并发数

可以指定下载器最大并发数。

- 函数原型

```
/**
 * 指定下载队列的最大大并发数
 * @param count 并发数; 调用下载接口之后再修改则无效
 */
public void setMaxConcurrent(int count);
```

- 示例

```
// 设置最大并发数
downloader.setMaxConcurrent(3);
```

4.3 长连接/断点续传

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。

- 函数原型

```
/**
 * 指定下载器是否支持断点续传
 * @param enable True -- 支持; False -- 不支持
 */
public void enableHTTPRange(boolean enable);
```

```
/**
 * 指定下载器是否支持HTTP长连接
 * @param keepalive True -- 支持; False -- 不支持
 */
public void enableKeepAlive(boolean keepalive);
```

- 示例

```
// 启动断点续传功能
downloader.enableHTTPRange(true);
// 启动长连接功能
downloader.enableKeepAlive(true);
```

4.4 视频下载

文件下载是采用异步模式进行下载，下载的进度/成功/失败/取消等信息通过DownloadListener进行回调通知。

- 函数原型

```
/**
 * 异步下载请求
 * @param url          要下载的文件URL
 * @param listener     下载结果监听器
 * @return
 */
public boolean download(String url, DownloadListener listener);
```

- 监听器定义

```
/** □□□□□ */
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);

}
```

- 示例

```
DownloadListener listener = new DownloadListener() {
    @Override
    public void onDownloadSucceed(String url, DownloadResult result) {
        Log.i("Demo", "□□□□: " + result.getPath());
    }

    @Override
    public void onDownloadProgress(String url, long totalSize, float progress)
    {
        long nProgress = (int) (progress * 100);
        Log.i("Demo", "□□□□: " + nProgress + "%");
     }

     @Override
     public void onDownloadFailed(String url, DownloadResult result) {
         Log.i("Demo", "□□□□: " + result. getErrorCode());
     }
```

```
    @Override
    public void onDownloadCanceled(String url) {
        Log.i("Demo", "下载任务被取消");
    }
};

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

4.5 取消下载

- 函数原型

```
/**
 * 取消指定的下载任务
 * @param url        下载文件地址
 * @param listener   下载结果监听器
 */
public void cancel(String url, DownloadListener listener);

/**
 * 取消所有下载任务
 */
public void cancelAll();
```

- 示例

```
// 取消单个下载任务: listener 为创建下载任务的时候传入的监听器
downloader.cancel(url, listener);
// 取消全部下载任务
downloader.cancelAll();
```

4.6 缓存查询

Downloader下载组件支持本地文件缓存

- 函数原型

```
/**
 * 判断指定URL对应的本地缓存文件是否存在
 * @param url 文件地址
 * @return 有本地缓存文件则返回True，否则False
 */
public boolean hasCache(String url);

/**
 * 获取指定URL的本地缓存文件
 * @param url 文件地址
 * @return 缓存存在则返回对应的File，否则返回Null
 */
public File  getCacheFile(String url);

/**
 * 清除SDK所有缓存文件
 */
public void cleanCache();
```

- 示例

```
//先检查是否有本地缓存文件
if (mDownloader.hasCache(url)) {
    File file = mDownloader.getCacheFile(url);
    if (file != null && file.exists()) {
        Log.i("Demo", "命中缓存");
           return;
```

```
    }
}

// 如果需求清空本地缓存
mDownloader.cleanCache();
```

4.7 日志上报

SDK会将下载过程中的日志保存到本地文件中，以便当用户上传过程中遇到问题时可以直接通过日志文件进行详细定位分析，SDK提供了日志上报接口，可以将指定日期的日志上报到腾讯云后台。

- 原型

```
/**
 * 上报formDate到toDate对应的日志
 * @param appid      腾讯云注册的APPID
 * @param fromDate   开始上报时间（时间单位：天）
 * @param toDate     结束上报时间（时间单位：天）
 * @return 成功返回True，失败返回False
 */
public static boolean uploadLog(String appid, Date fromDate, Date toDate);
```

- 示例

```
// 上报一天前的日志
Date beginDate = new Date(System.currentTimeMillis() - 24 * 3600 * 1000);
Date endDate = beginDate;
Downloader.uploadLog(APPID, beginDate, endDate);
```

5 文件SDK

5.1 初始化

见3.1初始化章节

- 示例

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

// 实例化Video业务上传管理类
UploadManager  videoUploadMgr = null;
videoUploadMgr = new
 UploadManager(context, APPID, FileType.Video, "qcloudvideo");
```

5.2 创建目录

创建目录步骤如下:

1. 通过文件path创建DirCreateTask对象
2. 调用UploadManager的sendCommand方法，将DirCreateTask对象传入
3. 在DirCreateTask.Ilistener的回调中获取查询结果文件复制
4. 原型

```
/**
 * 文件删除任务，视频业务使用
 * @param fileType       业务类型
 * @param bucket         bucket
 * @param path           相对路径
 * @param bizAttr        目录属性
 * @param listener       结果回调
 */
public DirCreateTask (FileType fileType, String bucket, String path,
                      String bizAttr, IListener listener)
```

- 示例

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

DirCreateTask task = null
task = new DirCreateTask(FileType.Video, BUCKET, path, bizAttr,
    new DirCreateTask.IListener(){
    @Override
    public void onSuccess(final DirCreateTask.CmdTaskRsp result) {
        Log.e("Demo", "□□□□□□:□□! accessUrl:" + result. accessUrl);
    }

    @Override
    public void onFailure(final int ret, final String msg) {
        Log.e("Demo", "□□□□□□:□□! ret:" + ret + " msg:" + msg);
    }
});

task.setAuth(VIDEO_SIGN);
videoUploadMgr.sendCommand(task);
```

5.3 拉取目录

拉取目录步骤如下:

1. 通过path创建DirListTask对象
2. 调用UploadManager的sendCommand方法，将DirListTask对象传入
3. 在DirListTask.Ilistener的回调中获取查询结果文件复制
4. 原型

```
public class Dentry {
```

```
    public final static int MORE = -1;                                  //更多...
    public final static int DIR = ObjectType._eObjectDir;               //目录对象
    public final static int FILE = ObjectType._eObjectDir;              //文件对象
    public final static int BUCKET = ObjectType._eObjectBucket; //bucket对象
    public final static int VIDEO = ObjectType._eObjectVideo;           //视频对象

    //  拉取文件选项
    public final static int ListBoth = ListPattern._eListBoth;      //拉取文件和目录
    public final static int ListDirOnly = ListPattern._eListDirOnly; //只拉取目录
    public final static int ListFileOnly = ListPattern._eListFileOnly;//只拉取文件

    public Dentry();      //构造DIR对象类型
    public Dentry(int type);
    public VideoInfo getVideoInfo();//  如果是VIDEO对象类型,
    //  则返回视频相关信息，否则返回null

    public int type = 0;                    //对象类型
    public String sha = "";                   //sha信息
    public String path = "";                //相对路径
    public String name = "";                //对象名称
    public String accessUrl = "";           //访问URL
    public String attribute = "";           //对象属性

    public long fileSize = 0;               //文件长度
    public long fileLength = 0;             //文件长度
    public long createTime = 0;             //创建时间
    public long modifyTime = 0;             //修改时间
}
```

- 原型

```
//  视频相关属性
public class VideoAttr {
```

```
    public String title = "";           // 视频title
    public String desc = "";            // 视频描述
    public long timeLen = 0;            // 视频时长
    public boolean isCheck = true;     // 0: 先发后审，1: 先审后发;
}

// 视频相关信息
public class VideoInfo {

    //微视频新增
    //--------------
    //f10: 手机
    //f20: 标清
    //f30: 高清
    //--------------
    public static final int F0 = VideoFormat._eF0;
    public static final int F10 = VideoFormat._eF10;
    public static final int F20 = VideoFormat._eF20;
    public static final int F30 = VideoFormat._eF30;

    //------------------------
    //- 描述: 原视频状态
    //------------------------
    public static final int DEFAULT = CosVideoStatus._eDEFAULT;          //默认状态
    public static final int UPLOADING = CosVideoStatus._eUPLOADING;    //视频入库中
    public static final int CHECKPASS = CosVideoStatus._eCHECKPASS;   //审核通过
    public static final int CHECKNOTPASS = CosVideoStatus._eCHECKNOTPASS;//审核不通过
    public static final int CHECKFAIL = CosVideoStatus._eCHECKFAIL; //审核失败

    //------------------------
    //- 描述: 不同码率视频状态
    //------------------------
    public static final int INVALID = TranscodeStat._eINVALID;     //默认状态
    public static final int TRANSCODING = TranscodeStat._eTRANSCODING; //转码中
```

```
    public static final int
 TRANSCODEDONE = TranscodeStat._eTRANSCODEDONE; //转码成功
    public static final int
 TRANSCODEFAIL = TranscodeStat._eTRANSCODEFAIL; //转码失败

    //------------------------
    //- 描述：属性修改标记
    //------------------------
    public static final int MaskBizAttr = FileModifyFlag._eMaskBizAttr;
    public static final int MaskTitle = FileModifyFlag._eMaskTitle;
    public static final int MaskDesc = FileModifyFlag._eMaskDesc;
    public static final int MaskAll = FileModifyFlag._eMaskAll;

    public int videoStatus = VideoInfo.DEFAULT;      // 视频状态
    public VideoAttr videoAttr;                      // 视频属性
    public Map<Integer, String> playUrlList;         // 转码视频url列表
    public Map<Integer, Integer> transStatus;        // 不同码率转码状态
}
```

- 原型

```
/**
 * 拉取目录任务，视频业务使用
 * @param fileType       业务类型
 * @param bucket         bucket
 * @param path           相对路径
 * @param num            拉多少记录
 * @param pattern        拉取方式,目录和文件一起拉的时候，先拉取目录再拉取文件
 * @param order          0-正序   1-反序
 * @param content        首次拉取必须清空，拉取下一页时需要传入上次一拉去时返回的content
 * @param listener       结果回调
 */
public DirListTask (FileType fileType, String bucket, String path, int num,
```

```
                int pattern, boolean order, String content, IListener listener)
```

- 示例

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

String content = ""; // 第一次拉取目录传入空，拉取path下一页时填入result.content
DirListTask task = null
task = new
 DirListTask(FileType.Video, BUCKET, path, 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;
            }
        Log.e("Demo", "目录拉取结果:成功! \n" + strEntryList);
    }

    @Override
    public void onFailure(final int ret, final String msg) {
        Log.e("Demo", "目录拉取结果:失败! ret:" + ret + " msg:" + msg);
    }
});
task.setAuth(VIDEO_SIGN);
videoUploadMgr.sendCommand(task);
```

5.4 前缀搜索

前缀搜索步骤如下:

1. 通过path+前缀创建DirListTask对象
2. 通过DirListTask对象的setPrefixSearch(true)开启前缀搜索
3. 调用UploadManager的sendCommand方法，将DirListTask对象传入
4. 在DirListTask.Ilistener的回调中获取查询结果文件复制
5. 原型

```
/**
 * DirListTask任务前缀搜索开关
 * @param prefixSearch      true-开启   false-关闭
*/
public void setPrefixSearch(boolean prefixSearch)
```

- 示例

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

String content = ""; // 第一次拉取目录传入空，拉取path下一页时填入result.content
DirListTask filetask = null
task = new
 DirListTask(FileType.Video, BUCKET,
path, 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;
            }
        Log.e("Demo", "目录拉取结果:成功! \n" + strEntryList);
    }

    @Override
    public void onFailure(final int ret, final String msg) {
        Log.e("Demo", "目录拉取结果:失败! ret:" + ret + " msg:" + msg);
    }
});

task .setPrefixSearch(true);
task.setAuth(VIDEO_SIGN);
videoUploadMgr.sendCommand(task);
```

5.5 文件上传

上传文件的步骤如下:

1. 创建FileUploadTask对象
2. 调用UploadManager的upload方法，将FileUploadTask对象传入
3. 原型

```
/** 上传回调监听器 */
public interface IUploadTaskListener {
    // 上传成功
    void onUploadSucceed(FileInfo result);

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

    // 上传进度
```

```
        void onUploadProgress(long totalSize, long recvDataSize);

    // 上传任务状态变化
        void onUploadStateChange(TaskState state);
}
```

- 原型

```
/**
 * 文件任务
 * @param bucket         bucket
 * @param srcFilePath    本地绝对路径
 * @param destFilePath   远程相对路径
 * @param bizAttr        文件私有属性
 * @param listener       上传结果回调
 */

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

- 示例

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

FileUploadTask task = new FileUploadTask(BUCKET, filePath, destPath, "",
    new IUploadTaskListener() {
    @Override
    public void onUploadSucceed(final FileInfo result) {
        Log.i("Demo", "upload succeed: " + result.url);
    }
```

```
    @Override
    public void onUploadStateChange(TaskState state) {

        }

    @Override
    public void onUploadProgress(long totalSize, long sendSize){
            long p = (long) ((sendSize * 100) / (totalSize * 1.0f));
            Log.i("Demo", "上传进度: " + p + "%");
    }

    @Override
    public void onUploadFailed(final int errorCode, final String errorMsg) {
            Log.i("Demo", "上传结果:失败! ret:" + errorCode + " msg:" + errorMsg);
    }
});

task.setAuth(VIDEO_SIGN);
videoUploadMgr.upload(task);  // 开始上传
```

5.6 视频上传

上传视频的步骤如下:

1. 创建VideoUploadTask对象
2. 调用UploadManager的upload方法，将VideoUploadTask对象传入
3. 原型

```
/**
 * 视频上传任务
 * @param bucket         bucket
 * @param srcFilePath    本地绝对路径
 * @param destFilePath   远程相对路径
```

```
 * @param bizAttr        文件私有属性，选填
 * @param videoAttr      视频私有属性，选填
 * @param listener       文件上传结果监听器，选填
 */
public VideoUploadTask(String bucket, String srcFilePath, String destFilePath,
        String bizAttr, VideoAttr videoAttr, IUploadTaskListener listener)
```

- 视频上传示例

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

VideoAttr videoAttr = new VideoAttr();
videoAttr.isCheck = false;
videoAttr.title = fileName;
videoAttr.desc = "cos-video-desc" + fileName;

VideoUploadTask task = new VideoUploadTask(BUCKET, filePath, destPath,
    "", videoAttr, new IUploadTaskListener() {
    @Override
    public void onUploadSucceed(final FileInfo result) {
        Log.i("Demo", "upload succeed: " + result.url);
    }

    @Override
    public void onUploadStateChange(TaskState state) {

        }

    @Override
    public void onUploadProgress(long totalSize, long sendSize){
```

```
            long p = (long) ((sendSize * 100) / (totalSize * 1.0f));
            Log.i("Demo", "上传进度: " + p + "%");
        }

    @Override
    public void onUploadFailed(final int errorCode, final String errorMsg) {
            Log.i("Demo", "上传结果:失败! ret:" + errorCode + " msg:" + errorMsg);
    }
});


task.setAuth(VIDEO_SIGN);
videoUploadMgr.upload(task);  // 开始上传
```

5.7 暂停、恢复、取消上传

见3.3暂停、恢复、取消上传章节

5.8 对象查询

查询Bucket、目录、文件等对像的详细信息，步骤如下:

1. 通过path和ObjectType创建ObjectStatTask对象
2. 调用UploadManager的sendCommand方法，将ObjectStatTask对象传入
3. 在ObjectStatTask.Ilistener的回调中获取查询结果。
4. 原型

```
/**
 * 文件查询任务，视频业务使用
 * @param fileType  业务类型
 * @param bucket    bucket
 * @param path      相对路径
 * @param type      对像类型
```

```
 * @param listener  □□□□
 */
public ObjectStatTask(FileType fileType, String bucket, String path,
                      int type, IListener listener);
```

- 示例

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

ObjectStatTask task = null;
task = new ObjectStatTask(FileType.Video, BUCKET, path, Dentry.Dir,
    new ObjectStatTask.IListener() {
        @Override
        public void onSuccess(final ObjectStatTask.CmdTaskRsp result) {
            Dentry dentry = result.inode;
        String info = "name:" + dentry.name;
        info += " sha:" + dentry. sha;
        info += " path:" + dentry. path;
        info += " type:" + dentry. type;
            info += " accessUrl:" + dentry.accessUrl;
            info += " attribute:" + dentry. attribute;
        info += " fileSize:" + dentry. fileSize;
            info += " fileLength:" + dentry.fileLength;
            info += " createTime:" + dentry.createTime;
            info += " modifyTime:" + dentry.modifyTime;
            Log.i("Demo", info);
        }

    @Override
    public void onFailure(final int ret, final String msg) {
        Log.e("Demo", "□□□□:□□! ret:" + ret + " msg:" + msg);
    }
});
```

```
task.setAuth(VIDEO_SIGN);
videoUploadMgr.sendCommand(task);
```

5.9 对象更新

更新Bucket、目录、文件等对像步骤如下:

1. 通过文件path、ObjectType和attr创建ObjectUpdateTask对象
2. 调用UploadManager的sendCommand方法，将ObjectUpdateTask对象传入
3. 在ObjectUpdateTask.Ilistener的回调中获取删除对像结果。
4. 原型

```
/**
 * 文件删除任务，File和Video业务使用
 * @param fileType   业务类型
 * @param bucket     bucket
 * @param path         相对路径
 * @param type         对像类型
 * @param attr         对像属性
 * @param listener   结果回调
 */
public
 ObjectUpdateTask(FileType fileType, String bucket, String path, String attr,
```

- 示例

```
import com.tencent.upload.task.impl.ObjectUpdateTask;
ObjectUpdateTask task = null; task = new
 ObjectUpdateTask (FileType.Video, BUCKET, path, Dentry.Dir, attr,
   new ObjectUpdateTask.IListener() {
   @Override
   public void onSuccess(ObjectDeleteTask.CmdTaskRsp result) {
```

```
        Log.e("Demo", "更新结果:成功!");
    }

    @Override
    public void onFailure(final int ret, final String msg) {
        Log.e("Demo", "更新结果:失败! ret:" + ret + " msg:" + msg);
    }
});
task.setAuth(VIDEO_SIGN); videoUploadMgr.sendCommand(task);
```

5.10 对象删除

删除Bucket、目录、文件等对像步骤如下:

1. 通过文件path和ObjectType创建ObjectDeleteTask对象
2. 调用UploadManager的sendCommand方法，将ObjectDeleteTask对象传入
3. 在ObjectDeleteTask.Ilistener的回调中获取删除对像结果。
4. 原型

```
/**
 * 文件删除任务，视频业务使用
 * @param fileType   业务类型
 * @param bucket     bucket
 * @param path         相对路径
 * @param type         对像类型
 * @param listener   结果回调
 */
public ObjectDeleteTask
(FileType fileType, String bucket, String path, int type, IListener listener)
```

- 示例

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

ObjectDeleteTask task= null;
task = new ObjectDeleteTask(FileType.Video, BUCKET, path, Dentry.Dir,
    new ObjectDeleteTask.IListener() {

    @Override
    public void onSuccess(ObjectDeleteTask.CmdTaskRsp result) {
        Log.e("Demo", "□□□□:□□!");
    }

    @Override
    public void onFailure(final int ret, final String msg) {
        Log.e("Demo", "□□□□:□□! ret:" + ret + " msg:" + msg);
    }
});

task.setAuth(VIDEO_SIGN);
videoUploadMgr.sendCommand(filetask);
```

6 返回码定义

错误码	含义
-5999	参数错误
-5998	签名格式错误
-5997	后端网络错误
-5996	HTTP请求方法错误
-5995	文件大小错误
-5994	url参数解析不匹配
-5993	multipart/formdata参数错误
-5992	请求参数错误
-5991	分片过大

-5990	找不到filecontent
-5989	上传失败
-5988	cgi初始化错误
-5987	wup编码失败
-5986	wup解码失败
-5985	获取路由失败
-5984	sha1不匹配
-5983	错误的session
-5982	建立连接错误
-5981	建立连接错误

iOS-SDK说明

1 开发准备

1.1 前期准备

1. iOS 5.0+；
2. 手机必须要有网络（GPRS、3G、Wifi网络等）；
3. 在腾讯云微视频页面上添加空间（bucket），获取项目ID（APPID）。

1.2 导入SDK

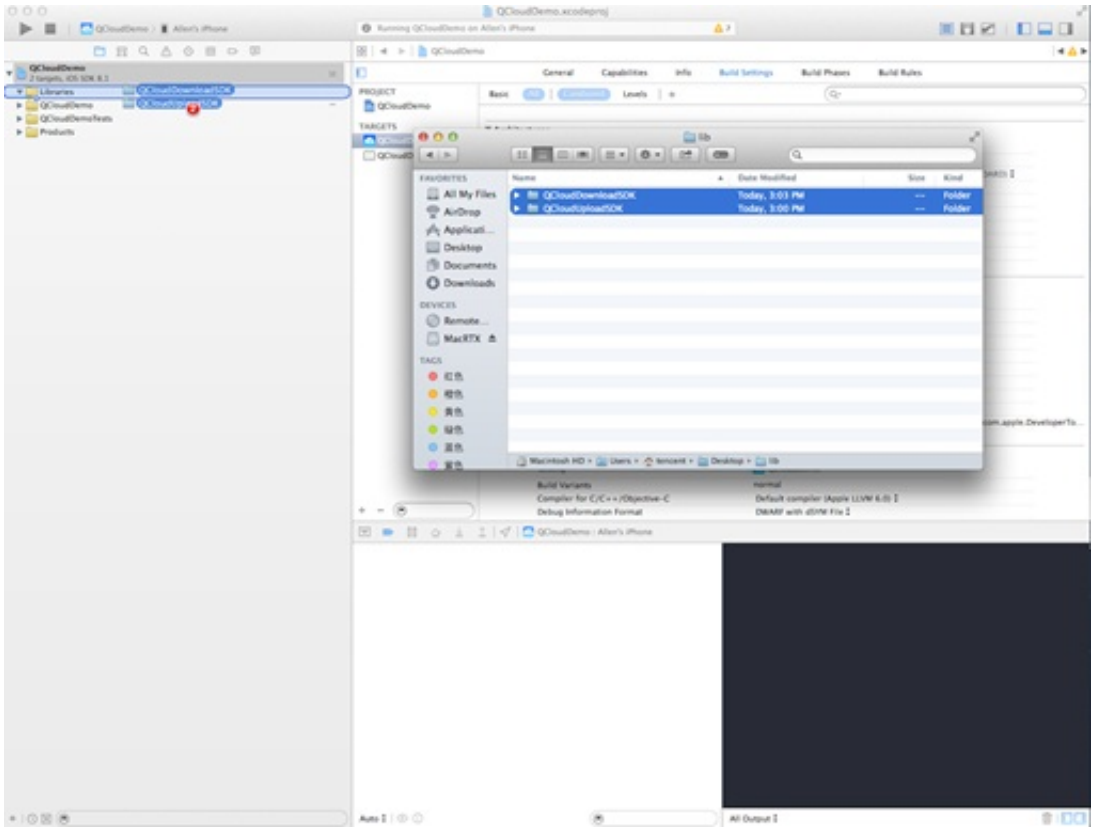
微视频 iOS SDK其中包括上传SDK和下载SDK，上传SDK压缩包QCloudUploadSDK.zip,下载SDK压缩包QCloudDownloadSDK.zip.
上传和下载SDK压缩包中分别包含了一个.a 静态库和一个包含头文件的文件夹Headers，解压后的内容如下：

上传SDK：

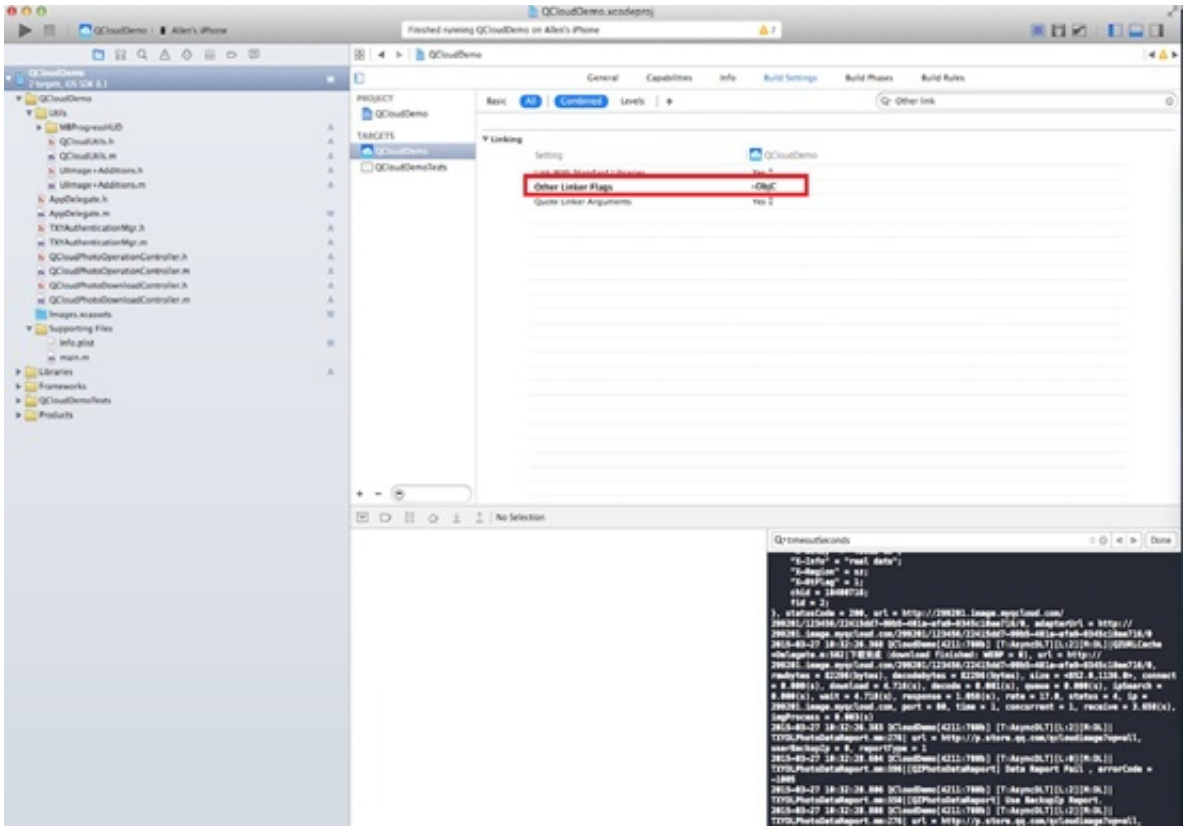
下载SDK：

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

注：如果只需要上传或下载功能，则只拖入对应的SDK即可。



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



在build Phases->Link Binary With Libraries中加入以下几个依赖库

1) SystemConfiguration.framework

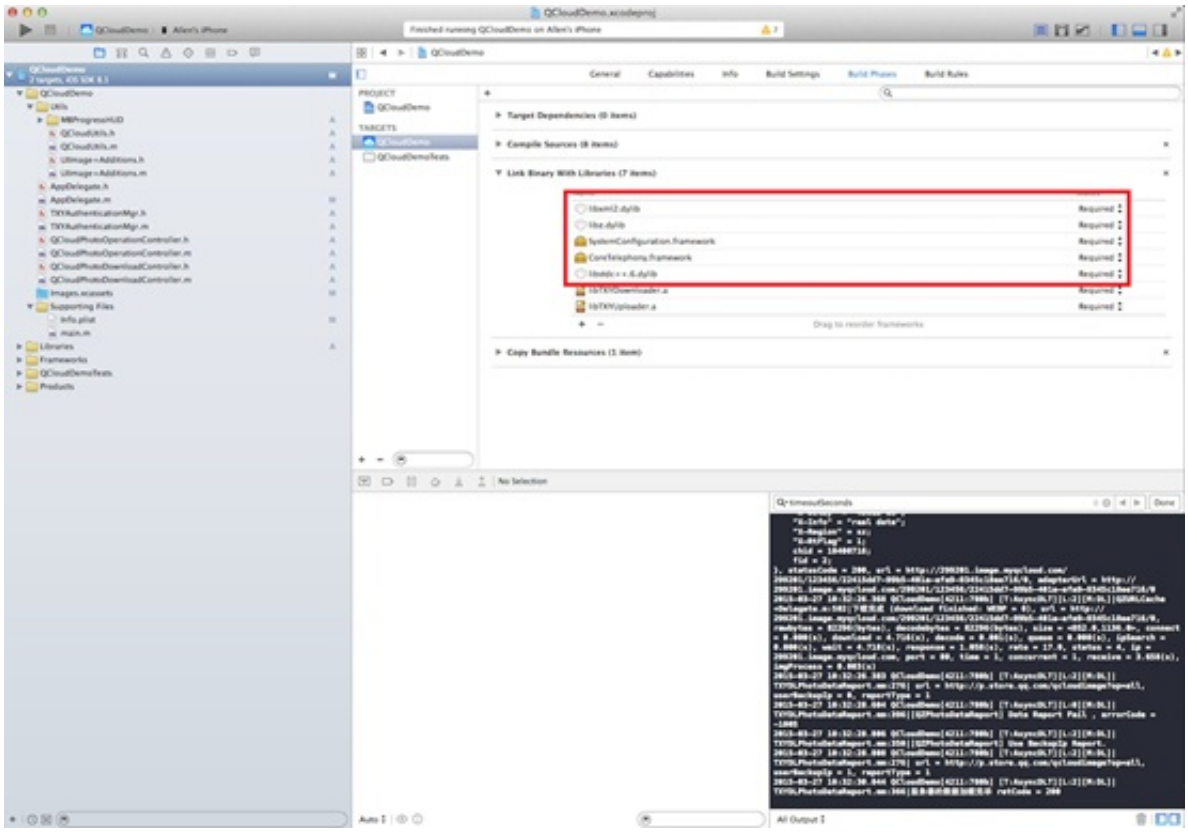
2) CoreTelephony.framework

3) MobileCoreServices.framework

4) libxml2.dylib

5) libz.dylib

6) libstdc++.6.dylib



注：如果只需要上传或下载功能，则只需要引入对应的动态库：

上传SDK依赖的系统动态库有：

1) SystemConfiguration.framework

2) CoreTelephony.framework

3) libstdc++.6.dylib

下载SDK依赖的系统动态库有：

1) SystemConfiguration.framework

2) CoreTelephony.framework

3) MobileCoreServices.framework

4) libxml2.dylib

5) libz.dylib

2 上传SDK

2.1 初始化

先引入上传SDK的头文件“TXYUploadManager.h”，创建TXYUploadManager对象，需要执行上传类型（图片云、文件云或视频云，在微视频服务中固定为视频云），appId，userId和签名信息。
原型

```
/*!
* @brief TXYUploadManager构造函数
* @param cloudType, 文件云、图片云、视频云
* @param persistenceId TXYUploadManager实例对应的持久化id,id必须全局唯
一,persistenceId为nil时，上传任务不持久化
```

```
* @appId 用户注册的appId
* @userId 用户注册的appId
* @sigin 签名信息
* @return TXYUploadManager实例
*/
- (instancetype)initWithPersistenceId:(TXYCloudType)cloudType
persistenceId:(NSString *)persistenceId appId:(NSString*)appId
userId:(NSString*)userId sign:(NSString*)sign;
```

示例

```
_uploadFileManager = [[TXYUploadManager alloc]
initWithPersistenceId:TXYCloudTypeForVideo persistenceId:@"TestDemo"
appId:appId userId:userId sign:[TXYAuthenticationMgr
shareInstance].fileSignature];
```

2.2 视频上传

上传视频的步骤如下:
1. 创建TXYFileUploadTask对象，指定文件上传的全路径（如appid/bucket），文件自定义属性，比如只读（readonly=true），如果是视频文件，可以制定视频标题、视频描述和是否需要先审后发。
2. 调用TXYUploadManager的upload方法，将TXYFileUploadTask对象传入

原型

```
@interface TXYFileUploadTask: TXYUploadTask &lt;NSCoding&gt;
/** 视频文件上传目录，用户选填 */
@property (nonatomic, readonly) NSString *directory;
/** 用户自定义属性，用户选填 */
@property (nonatomic, readonly) NSString *attrs;
/** 上传视频的属性，用户选填 */
```

```
@property(nonatomic, strong) TXYVideoFileInfo *videoInfo;
/*!
* @brief 视频文件上传任务初始化函数
* @param filePath 视频文件路径，必填
* @param attrs 视频文件属性，选填
* @param filename 文件名称，必填
* @param uploadDirectory 上传视频文件到哪个目录
* @param videoInfo 视频信息
* @param msgContext 通知用户业务后台的信息，选填
* @param insert 上传动作是插入还是覆盖
* @return TXYFileUploadTask实例
*/
- (instancetype)initWithPath:(NSString *)filePath
    sign:(NSString*)sign
    bucket:(NSString *)bucket
    fileName:(NSString *)fileName
    customAttribute:(NSString *)attrs
    uploadDirectory:(NSString*)directory
    videoFileInfo:(TXYVideoFileInfo*)videoInfo
    msgContext:(NSString *)msgContext
    insertOnly:(BOOL)insert;
@end
```

示例

```
//初始化视频文件上传对象
TXYFileUploadTask *videoTask = [[TXYFileUploadTask alloc] initWithPath:path
customAttribute:@"company=tencent"
uploadDirectory:@"/299201/ba/myfolder/"
videoFileInfo:nil msgContext:nil];
[uploadManager upload:videoTask sign:nil
complete:^(TXYTaskRsp *resp, NSDictionary *context) {
```

```
TXYFileUploadTaskRsp *fileResp = (TXYFileUploadTaskRsp *)resp;
NSLog(@"upload return=%d,%@",fileResp.retCode,fileResp.fileUrl);
}
progress:^(int64_t totalSize, int64_t sendSize, NSDictionary *context) {
}
stateChange:^(TXYUploadTaskState state, NSDictionary *context) {
    switch (state) {
        case TXYUploadTaskStateWait:
        NSLog(@"任务等待中");
        break;
    case TXYUploadTaskStateConnecting:
        NSLog(@"任务连接中");
        break;
    case TXYUploadTaskStateFail:
        NSLog(@"任务失败");
        break;
    case TXYUploadTaskStateSuccess:
        NSLog(@"任务成功");
        break;
    default:
        break;
}}];
```

2.3 恢复历史任务

上传过程中，程序如果意外退出，那么在下次启动时，可以通过TXYUploadManager的uploadTasks获取历史任务，然后从新调 上传API来恢复历史任务。

示例

```
//获取上次上传的历史任务后，重新调用upload上传
NSArray *histroyTasks =[uploadManager uploadTasks];
```

2.4 暂停、恢复、取消上传

上传任务可以暂停、恢复或者取消，只需要传入响应的taskId即可，上传任务的状态变化会通过TXYUpStateChangeHandler回调通知

原型

```
/*!
* @brief 暂停指定上传任务
* @param taskId 上传任务id @see &lt;TXYUploadTask&gt; 里的taskId
* @return 暂停成功返回YES，添加失败返回NO
*/
- (BOOL)pause:(int64_t)taskId;
/*!
* @brief 重新发送指定上传任务
* @param taskId 上传任务id @see &lt;TXYUploadTask&gt; 里的taskId
*/
- (void)resume:(int64_t)taskId;
/*!
* @brief 取消上传任务
* @param taskId 上传任务id，@see &lt;TXYUploadTask&gt; 里的taskId
* @return 取消成功返回YES，添加失败返回NO
*/
- (BOOL)cancel:(int64_t)taskId;
```

示例

```
TXYUploadTask* task = [taskModels objectAtIndex:index];
[uploadManager puase:task.taskId];//暂停上传
[uploadManager resume:task.taskId];//恢复上传
[uploadManager cancel:task.taskId];//取消上传
```

3 目录/视频管理SDK

3.1 创建目录

创建目录步骤如下：

1. 创建TXYCreateDir对象，必填的输入参数是目录的全路径，比如/appId/bucketId/MyDocument/
2. 调用TXYUploadManager的sendCommand方法，将TXYCreateDir对象传入
3. 在sendCommand传入的TXYUpCommandCompletionHandler回调中获取访问url

原型

```
@interface TXYCreateDir&amp;nbsp;: TXYCommandTask
//是否覆盖相同名字的目录或文件
@property(nonatomic,readonly) BOOL overwrite;
//用户自定义的属性
@property(nonatomic,readonly) NSString * attrs;
- (instancetype) initWithPath:(NSString*)path
needOverWrite:(BOOL)overwrite customAttribute:(NSString*)attrs;
@end
```

示例

```
TXYCreateDir *createDirCommand = [[TXYCreateDir alloc]
initWithURL:@”/appId/bucketId/MyDocument/TestFolder”];
    [self.uploadManager sendCommand:createDirCommand sign:nil
complete:^(TXYTaskRsp *resp) {
        if (resp.retCode &gt;= 0)
        {
            NSLog(@"创建目录成功，code:%d desc:%@", resp.retCode, resp.descMsg);
        }
        else
        {
```

```
            NSLog(@"创建目录失败，code:%d desc:%@", resp.retCode, resp.descMsg);
        }
    }];
```

3.2 查询视频或目录

查询视频或者目录的详细信息，步骤如下：

1. 创建TXYStat对象，必填的输入参数是视频或者目录的全路径，比如/appId/bucketId/MyDocument/
2. 调用TXYUploadManager的sendCommand方法，将TXYFileStat对象传入
3. 在sendCommand传入的TXYUpCommandCompletionHandler回调中获取查询结果FileDirInfo，如果是视频的话，会返回视频不同码率信息

原型

```
@interface TXYStat : TXYCommandTask
//对象类型，目录/文件/Bucket
@property(nonatomic,readonly)TXYObjectType objectType;
- (instancetype) initWithPath:(NSString*)path
objectType:(TXYObjectType)objectType;


@interface TXYFileDirInfo : NSObject
//文件名和目录名
@property(nonatomic,strong) NSString *name;
//文件的大小
@property(nonatomic,assign) long long fileSize;
//文件的长度
@property(nonatomic,assign) long long fileLength;
//文件自定义属性
@property(nonatomic,strong) NSString *attrs;
//文件的摘要sha
@property(nonatomic,strong) NSString *sha;
//文件的创建时间
```

```
@property(nonatomic,assign) NSUInteger ctime;
//文件的修改时间
@property(nonatomic,assign) NSInteger mtime;
//文件的访问url
@property(nonatomic,strong) NSString *accessUrl;
//文件目录类型
@property(nonatomic,assign) TXYObjectType objectType;
//文件目录云端的路径
@property(nonatomic,strong) NSString *startPath;
//视频的信息
@property(nonatomic,strong) TXYVideoListInfo *videoListInfo;
@end


@interface TXYVideoListInfo : NSObject
//  不同码率转码状态
@property(nonatomic,strong) NSDictionary *transcodeStatus;
//  视频文件状态
@property(nonatomic,assign) TXYCosVideoStatus videoStatus;
//  视频时长
@property(nonatomic,assign) NSInteger timeLength;
//  转码视频url列表
@property(nonatomic,strong) NSDictionary *playUrl;
//  视频信息
@property(nonatomic,strong) TXYVideoFileInfo *videoInfo;
@end
```

3.3 删除视频或目录

删除视频或目录步骤如下：

1. 创建TXYDelete对象，必填的输入参数是视频文件目录的全路径，比如/appId/bucketId/MyDocument/，删除对象的类型，比如command.type=TXYObjectDir

2. 调用TXYUploadManager的sendCommand方法，将TXYDelete对象传入

3. 在sendCommand传入的TXYUpCommandCompletionHandler回调

原型

```
@interface TXYDelete&amp;nbsp;: TXYCommandTask
//对象类型，目录/视频/Bucket
@property(nonatomic,readonly)TXYObjectType objectType;
- (instancetype) initWithPath:(NSString*)path;
- (instancetype) initWithPath:(NSString*)path
objectType:(TXYObjectType)objectType;
@end
```

示例

```
TXYDelete *command = [[TXYDelete alloc]
initWithURL:@”appId/bucketId/MyDocument”];
    command.type = TXYObjectDir;
    [self.uploadManager sendCommand:command sign:nil complete:^(TXYTaskRsp
*resp) {
        if (resp.retCode &gt;= 0)
        {
            NSLog(@"删除目录成功，code:%d desc:%@", resp.retCode, resp.descMsg);
        }
        else
        {
            NSLog(@"删除目录失败，code:%d desc:%@", resp.retCode, resp.descMsg);
        }
    }];
```

3.4 更新视频或目录

更新视频或目录步骤如下：

1. 创建TXYUpdate对象，必填的输入参数：1）全路径，比如/appId/bucketId/MyDocument/，2）属性值，比如readonly=true

2. 调用TXYUploadManager的sendCommand方法，将TXYUpdate对象传入

3. 在sendCommand传入的TXYUpCommandCompletionHandler回调中获取访问url

原型

```
@interface TXYUpdate: TXYCommandTask
//用户自定义的属性,比如: company=tencent,readonly=true
@property(nonatomic,strong) NSString *attrs;
//对象类型，目录/视频/Bucket
@property(nonatomic,assign)TXYObjectType objectType;
- (instancetype) initWithPath:(NSString*)path
objectType:(TXYObjectType)objectType customAttribute:(NSString*)attrs;
@end
```

示例

```
TXYUpdate *command = [[TXYUpdate alloc] initWithURL:@”/appId/bucketId/dir”];
    command.type = TXYObjectDir;
    command.attrs = @”readOnly=true”;
    [self.uploadManager sendCommand:command sign:nil complete:^(TXYTaskRsp
*resp) {
        if (resp.retCode &gt;= 0)
        {
            NSLog(@"更新目录成功，code:%d desc:%@", resp.retCode, resp.descMsg);
        }
        else
        {
            NSLog(@"更新目录失败，code:%d desc:%@", resp.retCode, resp.descMsg);
        }
```

```
    }];
```

3.5 浏览目录

浏览目录步骤如下：

1. 创建TXYListDir对象，必填的输入参数是视频或目录的全路径，比如/appId/bucketId/MyDocument/
2. 调用TXYUploadManager的sendCommand方法，将TXYListDir对象传入
3. 在sendCommand传入的TXYUpCommandCompletionHandler回调中获取视频目录属性FileDirInfo列表

原型

```
@interface TXYListDir: TXYCommandTask
//一次拉取多少条记录
@property(nonatomic,readonly) NSUInteger num;
//拉取方式,1）拉取目录和视频 2）拉取目录 3）拉取视频
@property(nonatomic,readonly) TXYListPattern pattern;
//0正序，1反序
@property(nonatomic,readonly) BOOL order;
- (instancetype) initWithPath:(NSString*)path number:(NSUInteger)num
listPattern:(TXYListPattern)pattern order:(BOOL)order;
@end
```

示例

```
TXYListDir *task = [[TXYListDir alloc] initWithPath: @"/appId/bucketId/dir"
number:10 listPattern:TXYListBoth order:NO pageContext:nil];
command.num = 10;//一次查询视频目录的个数
    [self.uploadManager sendCommand:command sign:nil complete:^(TXYTaskRsp
*resp) {
        if (resp.retCode &gt;= 0)
        {
```

```
            TXYListDirCommandRsp *listResp = (TXYListDirCommandRsp *)resp;
            [strongSelf showListDirResult:listResp.fileDirInfoList];
        }
        else
        {
            NSLog(@"更新目录失败，code:%d desc:%@", resp.retCode, resp.descMsg);
        }
    }];
```

3.6 前缀搜索

前缀搜索目录视频的步骤如下：

1. 创建TXYSearch对象，必填的输入参数是目录的前缀匹配字符串，比如/appId/bucketId/MyDocument/a，以a大头的目录及视频
2. 调用TXYUploadManager的sendCommand方法，将TXYSearch对象传入
3. 在sendCommand传入的TXYUpCommandCompletionHandler回调中获取视频目录属性FileDirInfo列表

原型

```
@interface TXYSearch: TXYCommandTask
//一次拉取多少条记录
@property(nonatomic,readonly) NSUInteger num;
//分页浏览的上下文
@property(nonatomic,readonly) NSString *pageContext;
- (instancetype) initWithPath:(NSString*)path number:(NSUInteger)num
pageContext:(NSString*)context;
@end
```

示例

```
TXYSearch *task = [[TXYSearch alloc]
```

```
initWithPath:@”/appid/bucket/myfolder/a” number:10 pageContext:nil];
    [self.uploadManager sendCommand:command sign:nil complete:^(TXYTaskRsp
*resp) {
        if (resp.retCode &gt;= 0)
        {
            TXYListDirCommandRsp *listResp = (TXYListDirCommandRsp *)resp;
            [strongSelf showListDirResult:listResp.fileDirInfoList];
        }
        else
        {
            NSLog(@"更新目录失败，code:%d desc:%@", resp.retCode, resp.descMsg);
        }
    }];
```

4 下载SDK

4.1 初始化

原型

```
/*!
* @brief 对应用程序的id,初始化一次即可
* @param appId 应用程序id,必填
* @param userId 用户id,选填
* @return 成功返回YES，失败返回NO
*/
+ (BOOL)authorize:(NSString *)appId userId:(NSString *)userId;
```

示例

```
//注册签名
[TXYDownLoder authorize:APPID userId:USERID sign:SIGN];
//实例化上传管理类
downloder = [[TXYDownloader sharedInstanceWithPersistenceId:nil type:
TXYDownloadTypeFile]
```

4.2 下载并发数

可以指定下载器最大并发数。

原型

```
/*!
* @brief 指定下载队列的最大并发数
* @param count 下载队列最大并发数,调用下载接口再修改则无效
* @return 成功返回YES，失败返回NO
*/
- (void)setMaxConcurrent:(int)count;
```

示例

```
//设置最大并发数
[downloader setMaxConcurrent:3];
```

4.3 长连接/断点续传

下载器提供开关，可以设定是否开启长连接和断电续传功能。

原型

```
/*!
* @brief 指定下载是否支持断点续传
* @param enable YES表示支持，为NO表示不支持，默认为YES
* @return 成功返回YES，失败返回NO
*/
- (void)enableHTTPRange:(BOOL)enable;
/*!
* @brief 指定是否支持HTTP长连接
* @param flag YES表示支持，为NO表示不支持，默认为YES
* @return 成功返回YES，失败返回NO
*/
- (void)enableKeepAlive:(BOOL)enable;
```

示例

```
//启动断点续传功能
[downloader enableHTTPRange:YES];
//启动长连接功能
[downloader enableKeepAlive:YES];
```

4.4 文件下载

文件下载是采用异步模式进行下载，下载的进度/成功/失败/取消等信息通过回调通知。

原型

```
/*!
* @brief 下载指定url的数据，并自动缓存到磁盘中，然后回调通知Target
* @param url 视频资源地址
* @param target 通知的对象
* @param succBlock 成功通知
```

```
* @param failBlock 失败通知
* @param progressBlock 进度通知,当前下载百分比
* @param 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
* @see &lt;TXYDownloaderParam&gt; 其中param中的key可以按照TXYDownloaderParam枚举指定
*/
- (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;
```

示例

```
[[TXYDownloader sharedInstanceWithPersistenceId:nil] download:_url
target:self
succBlock:^(NSString *url, NSData *data, NSDictionary *info) {
} failBlock:^(NSString *url, NSError *error) {
} progressBlock:^(NSString *url, NSNumber *value) {
} param:self.params];
```

4.5 取消下载

原型

```
/*!
* @brief 取消target对应的下载请求
* @param url 资源地址
* @param target 通知的对象
*/
- (void)cancel:(NSString *)url target:(id)target;
/*!
```

```
* @brief 取消所有的下载请求
*/
- (void)cancelAll;
```

示例

```
//取消单个下载任务
[[TXYDownloader sharedInstanceWithPersistenceId:nil] cancel: url
target:self];
//取消全部下载任务
[[TXYDownloader sharedInstanceWithPersistenceId:nil] cancelAll
```

4.6 缓存查询

TXYDownloader下载组件支持本地文件缓存，查询/获取/清除缓存文件。

原型

```
/*!
* @brief 判断指定url数据对应的本地缓存路径是否存在
* @param url 资源地址
* @return 成功返回YES，失败返回NO
*/
- (BOOL)hasCache:(NSString *)url;
/*!
* @brief 获取指定url的对应的二进制对象
* @param url 资源地址
* @return 找到了返回资源对应的NSData对象，否则返回nil
*/
- (NSData *)getCacheData:(NSString *)url;
/*!
```

```
* @brief 获取指定url的对应的本地缓存路径
* @param url 资源地址
* @return 找到了返回资源对应的本地缓存路径，否则返回nil
*/
- (NSString *)getCachePath:(NSString *)url;
/*!
* @brief 清除指定url对应的缓存
* @param url 资源地址
* @return 成功返回YES，失败返回NO
*/
- (BOOL)clearCache:(NSString *)url;
/*!
* @brief 清除所有缓存
* @return 成功返回YES，失败返回NO
*/
- (BOOL)clearCache;
//
```

示例

```
//先检查是否有本地缓存文件，有的话直接读取缓存文件并显示
if([[TXYDownloader sharedInstanceWithPersistenceId:nil]
hasCache:self.url]){
NSData *data = [[TXYDownloader
sharedInstanceWithPersistenceId:nil]getCacheData:self.url]
cacheImage = [UIImage imageWithData:data];
}
```

4.7 取消回调通知

可以根据制定的url取消下载回调通知，比如下载页面不可见或者消失时，取消回调后继续下载。

原型

```
/*!
* @brief 清除target下指定url的通知，不取消下载任务，继续下载
* @param urlPath 资源地址
* @param target 通知的对象
*/
- (void)clearTarget:(id)target url:(NSString *)url;
/*!
* @brief 清除指定Target所有的通知，不取消下载任务，继续下载
* @param target 通知的对象
*/
- (void)clearTarget:(id)target;
```

Java-SDK说明

1 开发准备

微视频服务的java sdk的下载地址： https://github.com/tencentyun/mvs-java-sdk.git

1.1 前期准备

1. sdk采用1.8版本的jdk开发，
推荐使用相同的版本。如果使用其他版本，建议不要直接导入jar包，自行编译为佳；
2. 通过项目设置获取appid，secret_id和secret_key；
3. Sdk开发采用netbeans，本文档以netbeans为例，其他IDE请适当调整。

1.2 导入SDK

1. 下载java sdk
如果安装了git命令行，执行git clone https://github.com/tencentyun/mvs-java-sdk.git
或者直接在github下载zip包。
2. 导入项目
在IDE中导入jar包（如果代码不支持，可以直接复制代码文件）

3. 参照api说明和sdk中提供的demo，开发代码。

1.3 https支持

修改VideoCloud.java中VIDEO_CGI_URL的值为：https://web.video.myqcloud.com/files/v1/

2 API详细说明

2.1 生成签名

1． 接口说明

签名生成方法，可以在服务端生成签名，供移动端app使用。

签名分为2种：

多次有效签名（有一定的有效时间）

单次有效签名（绑定资源url，只能生效一次）

签名的详细描述及使用场景参见鉴权服务技术方案

2． 方法

多次有效签名

String appSign(int appId, String secretId, String secretKey, long expired, String bucketName)

单次有效签名

String appSignOnce(int appId, String secretId, String secretKey, String resourcePath, String bucketName)

3． 参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
appId	String	是	无	开发者的授权appid
secret_id	String	是	无	开发者的授权secret_id
secret_key	String	是	无	开发者的授权secret_key，以上三项获取参见项目设置
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
resourcePath	String	是	无	视频文件唯一的标识，视频上传时会返回，格式/appid/bucketname/filepath/filename，其中/filepath/filename为视频文

				件在此bucketname下的全路径，

返回值：签名字符串

示例代码：

```
long expired = System.currentTimeMillis() / 1000 + 2592000;
String
 sign = Sign.a
ppSign(m_appid, m_secret_id, m_secret_key, expired, "myBucketName");
String resourcePath= "/myFloder/myVideo.mp4";
String
 sign
 = Sign.ap
pSignOnce(m_appid, m_secret_id, m_secret_key, resourcePath, "myBucketName");
```

2.2 目录操作

2.2.1 创建目录

1．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。

2．方法

String createFolder(String bucketName,String remotePath, String bizAttribute)

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketNam	String	是	无	bucket名称，bucket创

e				建参见创建Bucket
remotePath	String	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
bizAttribute	String	否	无	目录绑定的属性信息，业务自行维护

返回值,json格式字符串：

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	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

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/";
String result = video.createFolder("bucketname", remotePath,"bizAttribute");
```

2.2.2 目录属性更新

1． 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。

2．方法

String updateFolder(String bucketName, String remotePath, String bizAttribute)

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	String	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
bizAttribute	String	是	无	新的目录绑定的属性信息

返回值,json格式字符串：

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

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/";
String result = video.updateFolder("bucketname", remotePath,"bizAttribute_new");
```

2.2.3 目录查询

1． 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。

2． 方法

String getFolderStat(String bucketName, String remotePath)

3． 参数和返回值

参数说明：

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

返回值,json格式字符串：

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

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/";
String result = video.getFolderStat("bucketname", remotePath);
```

2.2.4 目录删除

1．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视频文件或目录，将不能删除。

2．方法

String deleteFolder(String bucketName, String remotePath)

3．参数和返回值

参数说明：

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

返回值,json格式字符串：

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

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/";
String result = video.deleteFolder("bucketname", remotePath);
```

2.2.5 列举目录下视频&目录

1．接口说明

用于列举目录下视频和目录，调用者可以通过此接口查询目录下的视频和目录属性。

2．方法

String getFolderList(String bucketName, String remotePath, int num, String context, int order, FolderPattern pattern)

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	String	是	无	目录的全路径，以"/"开头,以"/"结尾，api会补齐
num	int	是	无	要查询的目录/视频文件数量
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	FolderPa	是	无	pattern File:只是视频文

	ttern			件，Folder:只是视频文件夹，Both:全部

返回值,json格式字符串：

参数名	类型	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	子视频数量(总)
data.infos	Array	是	视频文件、目录集合，可以为空
data.infos.name	String	是	视频文件名或目录名
data.infos.biz_attr	String	是	目录或视频属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或视频文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为视频文件时返回)	视频文件大小
data.infos.filelen	Int	否(当类型为	视频文件已传输大小(通过与filesize

		视频文件时返回)	对比可知视频文件传输进度)
data.infos.sha	String	否(当类型为视频文件时返回)	视频文件sha
data.infos.access_url	String	否(当类型为视频文件时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频文件时返回)	转码状态,如: { “f10” : 0, “f20” : 1} 等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_status	Int	否(当类型为视频文件时返回)	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频文件时返回)	视频播放时长, 只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频文件时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频文件时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频文件时返回)	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3] 等

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/";
String result = video.getFolderList("bucketname",remotePath,100,"",0
,FolderPattern.Both);
```

2.2.6 列举目录下指定前缀视频&目录

1．接口说明

用于列举目录下指定前缀的视频和目录，调用者可以通过此接口查询目录下的指定前缀的视频和目录信息。

2．方法

String getFolderList(String bucketName, String remotePath, String prefix, int num, String context, int order, FolderPattern pattern)

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	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	是	无	读取视频文件/目录前缀
num	int	是	无	要查询的目录/视频文件数量
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	FolderPattern	是	无	pattern File:只是视频文件，Folder:只是视频文件夹，Both:全部

返回值,json格式字符串：

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	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	子视频文件数量(总)
data.infos	Array	是	视频文件、目录集合，可以为空
data.infos.name	String	是	视频文件或目录名
data.infos.biz_attr	String	是	目录或视频文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或视频文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为	视频文件大小

		视频文件时返回)	
data.infos.filelen	Int	否(当类型为视频文件时返回)	视频文件已传输大小(通过与filesize对比可知视频文件传输进度)
data.infos.sha	String	否(当类型为视频文件时返回)	视频文件sha
data.infos.access_url	String	否(当类型为视频文件时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频文件时返回)	转码状态,如: { “f10” : 0, “f20” : 1} 等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_status	Int	否(当类型为视频文件时返回)	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频文件时返回)	视频播放时长, 只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频时返回)	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3] 等

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/";
String result = video.getFolderList("bucketname", remotePath,
"20150701_", 100, "", 0, FolderPattern.Both);
```

2.3 视频文件

2.3.1 视频上传

1．接口说明

用于较小视频(一般小于8MB)的上传，调用者可以通过此接口上传较小的视频并获得视频的url，较大的视频请使用分片上传接口。

2．方法

String uploadFile(String bucketName, String remotePath,String localPath)

String uploadFile(String bucketName, String remotePath,String localPath,String videoCover,String bizAttribute,String title,String desc,String magicContext)

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	String	是	无	视频在微视频服务端的全路径，不包括/appid/bucketname
localPath	String	是	无	本地要上传视频的全路径
videoCover	String	否	无	视频封面的URL
bizAttribute	String	否	无	视频属性，业务端维护
title	String	否	无	视频的标题
desc	String	否	无	视频的描述
magicContext	String	否	无	透传字段，微视频会将此字段信息透传给业务设定

				的回 调url，具体参见回调设置

返回值,json格式字符串：

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	生成的视频下载url
data.url	String	是	操作视频的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/myVideo.mp4";
String
 result = video.upload(b
ucketname, remotePath,localPath,"http://video_cover.jpg","biz_attr",
"title","desc","magic_context");
```

2.3.2 视频分片上传

1． 接口说明

用于较大视频(一般大于8MB)的上传，调用者可以通过此接口上传较大视频并获得视频的url和唯一标识resour

ce_path（用于调用其他api）。

2．方法

String sliceUpload(String bucketName,String remotePath,String localPath)

String sliceUpload(String bucketName,String remotePath,String localPath,String videoCover,String bizAttribute,String title,String desc,String magicContext,int sliceSize)

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	String	是	无	视频在微视频服务端的全路径，不包括/appid/bucketname
localPath	String	是	无	本地要上传视频的全路径
videoCover	String	否	无	视频封面的URL
bizAttribute	String	否	无	视频属性，业务端维护
title	String	否	无	视频的标题
desc	String	否	无	视频的描述
magicContext	String	否	无	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置
sliceSize	Int	否	512*1024字节	分片大小，用户可以根据网络状况自行设置

返回值,json格式字符串：

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	生成的视频下载url
data.url	String	是	操作视频的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/myVideo.mp4";
String
 result = video.sli
ceUpload(bucketname, remotePath,localP
ath,"http://video_cover.jpg","biz_attr","title","desc",
"magic_context",1024*1024);
```

2.3.3 视频属性更新

1． 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。

2． 方法

String updateFile(String bucketName, String remotePath, String videoCover, String bizAttribute,String title,String desc,String magicContext);

3． 参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketNam	String	是	无	bucket名称，bucket创

e				建参见创建Bucket
remotePath	String	是	无	视频在微视频服务端的全路径，不包括/appid/bucketname
videoCover	String	否	无	视频封面的URL
bizAttribute	String	否	无	待更新的视频属性信息
title	String	否	无	视频的标题
desc	String	否	无	视频的描述

返回值,json格式字符串：

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

示例代码：

```
>>VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/myVideo.mp4";
String
 result = video.updateFile(buc
ketname, remotePath,"http://video_cover.jpg","biz_attr_new",
"title_new","desc_new");
```

2.3.4 视频查询

1． 接口说明

用于视频的查询，调用者可以通过此接口查询视频的各项属性信息。

2．方法

String getFileStat(String bucketName, String remotePath)

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	String	是	无	视频在微视频服务端的全路径，不包括/appid/bucketname

返回值,json格式字符串：

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	视频属性数据
data.name	String	是	视频文件名或目录名
data.biz_attr	String	是	视频属性，业务端维护
data.video_cover	String	是	视频封面的URL
data.ctime	String	是	视频的创建时间，unix时间戳
data.mtime	String	是	视频的修改时间，unix时间戳
data.filesize	Int	是	视频文件大小
data.filelen	Int	是	视频文件已传输大小(通过与filesize对比可知文件传输进度)
data.sha	String	是	视频文件sha
data.access_url	String	是	生成的视频下载url
data.trans_status	Array	是	转码状态,如: {“f10”：0, “f20”：1}

			等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.video_status	Int	是	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.video_play_time	Int	是	视频播放时长, 只有使用视频转码的业务才有
data.video_title	String	是	视频标题
data.video_desc	String	是	视频描述
data.video_play_url	Array	是	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3]等

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/myVideo.mp4";
String result = video.getFileStat("bucketname", remotePath);
```

2.3.5 视频删除

1． 接口说明

用于视频的删除，调用者可以通过此接口删除已经上传的视频。

2． 方法

String deleteFile(String bucketName, String remotePath)

3． 参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	String	是	无	视频在微视频服务端的全路径，不包括/appid/bucketname

返回值,json格式字符串：

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

示例代码：

```
VideoCloud video = new VideoCloud(APP_ID, SECRET_ID, SECRET_KEY);
String remotePath = "/myFolder/myVideo.mp4";
String result = video.deleteFile("bucketname", remotePath);
```

PHP-SDK说明

1 开发准备

1、微视频服务的PHP SDK的下载地址： https://github.com/tencentyun/uvs-php-sdk.git

2、composer项目名： tencentyun/uvs-php-sdk

1.1 前期准备

前期准备

1. sdk依赖php 5.3.0版本及以上， 推荐使用相同的版本。

2. 获取项目ID(appid)，bucket，secret_id和secret_key；

1.2 获取SDK

1、直接下载github上提供的源代码，集成到您的开发环境。

2、composer安装：直接执行命令：php composer.phar require tencentyun/uvs-php-sdk

在您使用sdk之前，加载include.php即可。

```
require('./include.php'); <br>
use Qcloud_video\Auth; <br>
use Qcloud_video\Video; <br>
```

1.3 https支持

修改conf.php中API_VIDEO_END_POINT的值为：https://web.video.myqcloud.com/files/v1/

2 API详细说明

2.1 生成签名

1． 接口说明

签名生成方法，可以在服务端生成签名，供移动端app使用。

签名分为2种：

多次有效签名（有一定的有效时间）

单次有效签名（绑定资源url，只能生效一次）

签名的详细描述及使用场景参见鉴权服务技术方案

2． 方法

多次有效签名

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

单次有效签名

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

3． 参数和返回值

参数说明：

参数名	类型	必须	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
path	String	是	无	视频唯一的标识，视频文件上传时会返回，格式/filepath/filename，文件在

				此bucketname下的全路径，

返回值：签名字符串

示例代码：

```
$expired = time() + 60;
$sign = Auth::appSign($expired, $bucketName);
$resourcePath= "/myFloder/myVideo.mp4";
$sign = Auth::appSign_once($path, $bucketName);
```

2.2 目录操作

2.2.1 创建目录

1．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。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	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
bizAttr	String	否	null	目录绑定的属性信息，业务自行维护

返回值,json格式：

参数名	类型	参数描述
httpcode	Int	http响应码，请求正常时为200
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

示例代码：

```
$bucketName = "myBucket";
$path = "/myFolder/";
$bizAttr = "attr_folder";
$result  = Video::createFolder($bucketName, $path,$bizAttr)
```

2.2.2 目录属性更新

1．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。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	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
bizAttr	String	是	无	新的目录绑定的属性信息

返回值,json格式：

参数名	类型	参数描述
httpcode	Int	http响应码，请求正常时为200
code	Int	错误码，成功时为0
message	String	错误信息

示例代码：

```
$bucketName = "myBucket";
$path = "/myFolder/";
$bizAttr = "attr_folder_new";
$result  = Video::updateFolder($bucketName, $path,$bizAttr)
```

2.2.3 目录查询

1．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。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	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

返回值,json格式：

参数名	类型	参数描述
httpcode	Int	http响应码，请求正常时为200
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.video_cover	String	视频封面的URL
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

示例代码：

```
$bucketName = "myBucket";
$path = "/myFolder/";
$result = Video::statFolder($bucketName, $path);
```

2.2.4 目录删除

1．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视频文件或目录，将不能删除。

2．方法

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

3． 参数和返回值

参数说明：

参数名	类型	必须	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

返回值,json格式：

参数名	类型	参数描述
httpcode	Int	http响应码，请求正常时为200
code	Int	错误码，成功时为0
message	String	错误信息

示例代码：

```
$bucketName = "myBucket";
$path = "/myFolder/";
$result = Video::delFolder($bucketName, $path);
```

2.2.5 列举目录下视频&目录

1．接口说明

用于列举目录下文件和目录，调用者可以通过此接口查询目录下的文件和目录属性。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	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
num	int	否	20	要查询的目录/视频文件数量
context	String	否	null	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	eListBoth,eListDirOnly, eListFileOnly

				默认eListBoth

返回值,json格式：

参数名	类型	必然返回	参数描述
httpcode	Int	是	http响应码，请求正常时为200
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	子视频文件数量(总)
data.infos	Array	是	视频文件、目录集合，可以为空
data.infos.name	String	是	视频文件或目录名
data.infos.biz_attr	String	是	目录或文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
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(当类型为视频文件时返回)	视频文件大小
data.infos.filelen	Int	否(当类型为视频文件时返回)	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(当类型为视频文件时返回)	文件sha
data.infos.access_url	String	否(当类型为视频文件时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为文件时返回)	转码状态,如: { “f10” : 0, “f20” : 1} 等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_status	Int	否(当类型为视频时返回)	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频时返回)	视频播放时长, 只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为文件时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频时返回)	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3] 等

示例代码：

```
$bucketName = "myBucket";
$path = "/myFolder/";
$result = Video::listFolder($bucketName, $path, 20, 'eListBoth',0);
```

2.2.6 列举目录下指定前缀视频&目录

1． 接口说明

用于列举目录下指定前缀的文件和目录，调用者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。

2． 方法

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

3． 参数和返回值

参数说明：

参数名	类型	必须	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视频文件(带全路径)
num	int	否	20	要查询的目录/视频文件数量
context	String	否	null	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	eListBoth,eListDirOnly, eListFileOnly 默认eListBoth

返回值,json格式：

参数名	类型	必然返回	参数描述
httpcode	Int	是	http响应码，请求正常时为200
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	子视频文件数量(总)
data.infos	Array	是	视频文件、目录集合，可以为空
data.infos.name	String	是	视频文件或目录名
data.infos.biz_attr	String	是	目录或视频文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
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(当类型为视频文件时返回)	视频文件大小
data.infos.filelen	Int	否(当类型为	文件已传输大小(通过与filesize对比

		视频文件时返回)	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(当类型为视频文件时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频文件时返回)	转码状态,如: { “f10” : 0, “f20” : 1} 等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_status	Int	否(当类型为文件时返回)	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频时返回)	视频播放时长, 只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频时返回)	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3] 等

示例代码：

```
$bucketName = "myBucket";
$prefix= "/myFolder/2015-";
$result = Video::listFolder($bucketName, $path, 20, 'eListBoth',0);
```

2.3 视频文件

2.3.1 视频上传

1．接口说明

用于较小视频(一般小于8MB)的上传，调用者可以通过此接口上传较小的视频并获得视频的url，较大的视频请使用分片上传接口。

2．方法

```
public static function upload($srcPath, $bucketName, $dstPath, $videoCover = nul
l, $bizAttr = null, $title = null, $desc = null, $magicContext = null)
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
srcPath	String	是	无	本地要上传视频文件的全路径
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
dstPath	String	是	无	文件在微视频服务端的全路径，不包括/appid/bucketname
videoCover	String	否	null	视频封面的URL
bizAttr	String	否	null	视频文件属性，业务端维护
title	String	否	null	视频的标题
desc	String	否	null	视频的描述
magicConte	String	否	null	透传字段，微视频会将此

xt				字段信息透传给业务设定的回调url，具体参见回调设置

返回值,json格式：

参数名	类型	必然返回	参数描述
httpcode	Int	是	http响应码，请求正常时为200
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

示例代码：

```
$srcPath= "/data/test.mp4";
$bucketName = "myBucket";
$dstPath = "/myFolder/";
$result = Video::upload($
srcPath,$bucketName,dstPath , "http://cover-url.jpg", "biz_attr","title",
"desc","magic_context");
```

2.3.2 视频分片上传

1． 接口说明

用于较大视频(一般大于8MB)的上传，调用者可以通过此接口上传较大视频并获得视频的url和唯一标识resource_path（用于调用其他api）。

2． 方法

```
public static function upload_slice(
            $srcPath, $bucketName, $dstPath,
            $videoCover = null, $bizAttr = null, $title = null, $desc = null,
            $magicContext = null, $sliceSize = self::DEFAULT_SLICE_SIZE, $sessio
n = null)
```

3． 参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
srcPath	String	是	无	本地要上传视频文件的全路径
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
dstPath	String	是	无	文件在微视频服务端的全路径，不包括/appid/bucketname
videoCover	String	否	null	视频封面的URL
bizAttr	String	否	null	视频文件属性，业务端维护
title	String	否	null	视频的标题
desc	String	否	null	视频的描述
magicContext	String	否	null	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置

sliceSize	Int	否	512*1024字节	分片大小，用户可以根据网络状况自行设置
session	String	否	null	如果是断点续传, 则带上(唯一标识此视频文件传输过程的id, 由后台下发, 调用方透传)

返回值,json格式：

参数名	类型	必然返回	参数描述
httpcode	Int	是	http响应码，请求正常时为200
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

示例代码：

```
$srcPath= "/data/test.mp4";
$bucketName = "myBucket";
$dstPath = "/myFolder/";
$result = Video::upl
oad_slice($srcPath,$bucketName,dstPath ,"http://cover-
url.jpg","biz_attr","title","desc","magic_context");
```

2.3.3 视频属性更新

1． 接口说明

用于文件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。

2． 方法

```
 public static function update($bucketName, $path, $videoCover = null, $bizAttr = null,$title = null, $desc = null)
```

3． 参数和返回值

参数说明：

参数名	类型	必须	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	文件在微视频服务端的全路径，不包括/appid/bucketname
videoCover	String	否	null	视频封面的URL
bizAttr	String	否	null	待更新的视频文件属性信息
title	String	否	null	视频的标题
desc	String	否	null	视频的描述

返回值,json格式：

参数名	类型	必然返回	参数描述
httpcode	Int	是	http响应码，请求正常时为200
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例代码：

```
$bucketName = "myBucket";
$path = "/myFolder/test.mp4";
$result = Video::update($bucketName,
$path,"http://cover-url.jpg","attr_file_new"□"title_new"□"desc_new");
```

2.3.4 视频查询

1．接口说明

用于视频的查询，调用者可以通过此接口查询视频的各项属性信息。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketNam	String	是	无	bucket名称，bucket创

e				建参见创建Bucket
path	String	是	无	文件在微视频服务端的全路径，不包括/appid/bucketname

返回值,json格式：

参数名	类型	必然返回	参数描述
httpcode	Int	是	http响应码，请求正常时为200
code	Int	是	错误码，成功时为0
message	String	是	错误信息
data	Array	是	视频属性数据
data.name	String	是	视频文件或目录名
data.biz_attr	String	是	视频属性，业务端维护
data.video_cover	String	是	视频封面的URL
data.ctime	String	是	视频的创建时间，unix时间戳
data.mtime	String	是	视频的修改时间，unix时间戳
data.filesize	Int	是	视频文件大小
data.filelen	Int	是	视频文件已传输大小(通过与filesize对比可知文件传输进度)
data.sha	String	是	视频文件sha
data.access_url	String	是	生成的视频下载url
data.trans_status	Array	是	转码状态,如:["f10":0,"f20":1,"f30":0] 等
data.video_status	Int	是	视频状态码
data.video_play_time	Int	是	视频播放时长,只有使用视频转码的业务才有
data.video_title	String	是	视频标题
data.video_desc	String	是	视频描述
data.video_play_url	Array	是	各码率的播放url,如:["f10":url1,"f20":url2,"f30":url3]

			等

示例代码：

```
$bucketName = "myBucket";
$path = "/myFolder/test.mp4";
$result = Video::stat($bucketName, $path);
```

2.3.5 视频删除

1．接口说明

用于视频的删除，调用者可以通过此接口删除已经上传的视频。

2．方法

```
public static function del($bucketName, $path)
```

3．参数和返回值

参数说明：

参数名	类型	必须	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	文件在微视频服务端的全路径，不包括/appid/bucketname

返回值,json格式：

参数名	类型	必然返回	参数描述
httpcode	Int	是	http响应码，请求正常时为200
code	Int	是	错误码，成功时为0
message	String	是	错误信息

示例代码：

```
$bucketName = "myBucket";
$path = "/myFolder/test.mp4";
$result = Video::del($bucketName, $path);
```

Python-SDK说明

1 开发准备

微视频服务的Python2 sdk的下载地址： https://github.com/tencentyun/mvs-python-sdk.git

1.1 前期准备

1. sdk采用Python2.7开发， 推荐使用相同的版本。如果使用其他版本，建议不要直接导入，自行调试为佳；
2. 通过项目设置获取appid，secret_id和secret_key；

1.2 导入SDK

1. 下载Python sdk
方法一：使用 pip install qcloud_video 安装
方法二：如果安装了git命令行，执行git clone https://github.com/tencentyun/mvs-python-sdk.git
或者直接在github下载zip包。
注意：SDK依赖requests包，使用方法二需自行安装。
2. 导入项目
在IDE中导入qcloud_video包

```
import qcloud_video
```

3. 参照api说明和sdk中提供的demo，开发代码。

1.3 HTTPS支持

如果想使用https协议上传，则将qcloud_video/conf.py文件中变量API_VIDEO_END_POINT中的http修改为https即可。

2 API详细说明

2.1 生成签名

1．接口说明

签名生成方法，可以在服务端生成签名，供移动端app使用。

签名分为2种：

多次有效签名（有一定的有效时间）

单次有效签名（绑定资源url，只能生效一次）

签名的详细描述及使用场景参见鉴权服务技术方案

2．方法

签名类构造函数

```
def __init__(self, secret_id, secret_key)
```

多次有效签名

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

单次有效签名

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

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
secret_id	String	是	无	开发者的授权secret_id
secret_key	String	是	无	开发者的授权secret_key，以上两项获取参见项目设置
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
fileid	String	是	无	视频文件唯一的标识，格式/appid/bucketname/filepath/filename，其中/filepath/filename为视频文件在此bucketname下的全路径，

返回值：签名字符串

示例代码：

```
import qcloud_video
qcloud_video.conf.set_app_info(appid, secret_id, secret_key)
auth = qcloud_video.Auth(secret_id, secret_key)
sign = auth.sign_more('bucketname', time.time() + 86400)
sign = auth.sign_once('bucketname', '/appid/bucketname/myFolder/myVideo.mp4')
```

2.2 目录操作

2.2.1 创建目录

1．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。

2．方法

```
def createFolder(self, bucket, path, bizattr='')
```

3．参数和返回值

参数说明：

参数名	类型	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	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
bizattr	String	否	空串	目录绑定的属性信息，业务自行维护

返回值,json格式：

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

示例代码：

```
video  = qcloud_video.Video(appid, secret_id, secret_key)
result = video.createFolder('bucketname', 'myFolder/', 'bizAttribute')
```

2.2.2 目录属性更新

1．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。

2．方法

```
def updateFolder(self, bucket, path, bizattr)
```

3．参数和返回值

参数说明：

参数名	类型	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	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
bizattr	String	是	无	新的目录绑定的属性信息

返回值,json格式：

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

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
result = video.updateFolder('bucketname', 'myFolder/', 'bizAttribute')
```

2.2.3 目录查询

1． 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。

2． 方法

```
def statFolder(self, bucket, path)
```

3． 参数和返回值

参数说明：

参数名	类型	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，api会补齐

返回值,json格式：

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.video_cover	String	视频封面的URL
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

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
result = video.statFolder('bucketName', 'myFolder/')
```

2.2.4 目录删除

1． 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视频文件或目录，将不能删除。

2． 方法

```
def deleteFolder(self, bucket, path)
```

3． 参数和返回值

参数说明：

参数名	类型	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，api会补齐

返回值,json格式：

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

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
result = video.deleteFolder('bucketname', 'myFolder/')
```

2.2.5 列举目录下视频文件&目录

1． 接口说明

用于列举目录下视频文件和目录，调用者可以通过此接口查询目录下的视频文件和目录属性。

2． 方法

```
def list(self, bucket, path, num=20, pattern='eListBoth', order=0, context='')
```

3．参数和返回值

参数说明：

参数名	类型	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，api会补齐
num	int	否	20	要查询的目录/视频文件数量
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:只是视频文件，ListDirOnly:只是文件夹，eListBoth:全部

返回值,json格式：

参数名	类型	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	子视频文件数量(总)
data.infos	Array	是	视频文件、目录集合，可以为空
data.infos.name	String	是	视频文件或目录名
data.infos.biz_attr	String	是	目录或视频文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或视频文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为视频文件时返回)	视频文件大小
data.infos.filelen	Int	否(当类型为视频文件时返回)	视频文件已传输大小(通过与filesize对比可知视频文件传输进度)
data.infos.sha	String	否(当类型为视频文件时返回)	视频文件sha
data.infos.access_url	String	否(当类型为视频文件时返回)	生成的视频下载url

data.infos.trans_status	Array	否(当类型为视频文件时返回)	转码状态,如: { “f10” : 0, “f20” : 1} 等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_status	Int	否(当类型为视频文件时返回)	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频文件时返回)	视频播放时长, 只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频文件时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频文件时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频文件时返回)	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3] 等

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
result = video.list('bucketname', 'myFolder/', 30, 'eListBoth', 0, '')
```

2.2.6 列举目录下指定前缀的视频文件&目录

1． 接口说明

用于列举目录下指定前缀的视频文件和目录，调用者可以通过此接口查询目录下的指定前缀的视频文件和目录

信息。

2．方法

```
def prefixSearch(self, bucket, path, prefix='', num=20, pattern=
'eListBoth', order=0, context='')
```

3．参数和返回值

参数说明：

参数名	类型	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	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	读取视频文件/目录前缀
num	int	否	20	要查询的目录/视频文件数量
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:只是视频文件，ListDirOnly:只是文件夹，eListBoth:

				全部

返回值,json格式：

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	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	子视频文件数量(总)
data.infos	Array	是	视频文件、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	目录或视频文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或视频文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为视频文件时返回)	视频文件大小
data.infos.filelen	Int	否(当类型为视频文件时返	视频文件已传输大小(通过与filesize对比可知视频文件传输进度)

		回)	
data.infos.sha	String	否(当类型为视频文件时返回)	视频文件sha
data.infos.access_url	String	否(当类型为视频文件时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频文件时返回)	转码状态,如: { “f10” : 0, “f20” : 1} 等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_status	Int	否(当类型为视频文件时返回)	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频文件时返回)	视频播放时长, 只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频文件时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频文件时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频文件时返回)	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3] 等

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
```

```
result = video.prefixSearch('bucketname', 'myFolder/', '20150606_', 30,
'eListBoth', 0, '')
```

2.3 视频文件

2.3.1 视频上传

1．接口说明

用于较小视频(一般小于8MB)的上传，调用者可以通过此接口上传较小的视频并获得视频的url，较大的视频请使用分片上传接口。

2．方法

```
def upload(self
, filepath, bucket, dstpath, videoCover=None, title=None, desc=None, bizattr=Non
e, magiccontext=None)
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
dstpath	String	是	无	视频文件在微视频服务端的全路径，不包括/appid/bucketname
filepath	String	是	无	本地要上传视频的全路径
bizattr	String	否	None	视频文件属性，业务端维护
title	String	否	None	视频的标题
desc	String	否	None	视频的描述

magicconte xt	String	否	None	透传字段，微视频会将此字段信息透传给业务设定的回 调url，具体参见回调设置

返回值,json格式：

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	生成的视频文件下载url
data.url	String	是	操作视频文件的url
data.resource_pat h	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
result = video.upload('test.mp4', 'bucketName',
'myFolder/myVideo.mp4', 'http://video-cover.jpg', 'title', 'desc')
```

2.3.2 视频分片上传

1． 接口说明

用于较大视频(一般大于8MB)的上传，调用者可以通过此接口上传较大视频并获得视频的url和唯一标识resource_path（用于调用其他api）。

2．方法

```
def upload_slice(self
, f
ilepat
h, bucket, d
stpath, title=None, desc
=None, bizattr=None, slice_size=0, session='', magiccontext=None)
```

3．参数和返回值

参数说明：

参数名	类型	必须	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	视频文件在微视频服务端的全路径，不包括/appid/bucketname
filepath	String	是	无	本地要上传视频文件的全路径
bizattr	String	否	None	视频文件属性，业务端维护
videoCover	String	否	None	视频封面的URL
title	String	否	None	视频的标题
desc	String	否	None	视频的描述
magiccontext	String	否	None	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置
slice_size	Int	否	512*1024字节	分片大小，用户可以根据网络状况自行设置,传0代

				表使用默认值。
session	String	否	空串	续传时透传的session，一般不设置。

返回值,json格式：

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	生成的视频文件下载url
data.url	String	是	操作视频文件的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
result = video.upload_slice('test.mp4', 'bucketName',
'myFolder/myVideo.mp4', 'http://video-cover.jpg', 'title', 'desc',
'bizattr', 2*1024*1024)
```

2.3.3 视频属性更新

1． 接口说明

用于视频属性的更新，调用者可以通过此接口更新视频的Title，Desc和自定义属性字段。

2． 方法

```
def updateFile(self
, bucket, path, videoCover=None, title=None, desc=None, bizattr=None)
```

3．参数和返回值

参数说明：

参数名	类型	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	视频文件在微视频服务端的全路径，不包括/appid/bucketname
videoCover	String	否	None	视频封面的URL
bizattr	String	否	None	待更新的视频文件属性信息
title	String	否	None	视频的标题
desc	String	否	None	视频的描述

返回值,json格式：

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

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
result = video.updateFile('bucketname',
'myFolder/myVideo.mp4', 'http://video-cover.jpg', 'title', 'desc', 'bizattr')
```

2.3.4 视频查询

1． 接口说明

用于视频的查询，调用者可以通过此接口查询视频的各项属性信息。

2． 方法

```
def statFile(self, bucket, path)
```

3． 参数和返回值

参数说明：

参数名	类型	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	视频文件在微视频服务端的全路径，不包括/appid/bucketname

返回值,json格式：

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	视频属性数据
data.name	String	是	视频文件或目录名
data.biz_attr	String	是	视频属性，业务端维护
data.video_cover	String	是	视频封面的URL
data.ctime	String	是	视频的创建时间，unix时间戳
data.mtime	String	是	视频的修改时间，unix时间戳
data.filesize	Int	是	视频文件大小
data.filelen	Int	是	视频文件已传输大小(通过与filesize对比可知视频文件传输进度)
data.sha	String	是	视频文件sha
data.access_url	String	是	生成的视频下载url
data.trans_status	Array	是	转码状态,如:["f10":0,"f20":1,"f30":0] 等
data.video_status	Int	是	视频状态码
data.video_play_time	Int	是	视频播放时长,只有使用视频转码的业务才有
data.video_title	String	是	视频标题
data.video_desc	String	是	视频描述
data.video_play_url	Array	是	各码率的播放url,如:["f10":url1,"f20":url2,"f30":url3]等

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
result = video.statFile('bucketname', 'myFolder/myVideo.mp4')
```

2.3.5 视频删除

1．接口说明

用于视频的删除，调用者可以通过此接口删除已经上传的视频。

2．方法

```
def deleteFile(self, bucket, path)
```

3．参数和返回值

参数说明：

参数名	类型	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	视频文件在微视频服务端的全路径，不包括/appid/bucketname

返回值json格式：

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

示例代码：

```
video = qcloud_video.Video(appid, secret_id, secret_key)
```

```
result = video.deleteFile('bucketname', 'myFolder/myVideo.mp4')
```

Nodejs-SDK说明

1 开发准备

微视频服务的Node.js sdk的下载地址： https://github.com/tencentyun/mvs-nodejs-sdk.git

1.1 前期准备

1. sdk采用Node.js v0.10.29版本开发， 推荐使用相同的版本。
2. 通过项目设置获取appid，secret_id和secret_key；

1.2 导入SDK

1. 下载Node.js sdk
方法一：执行 npm install qcloud_video 直接安装。
方法二：执行git clone https://github.com/tencentyun/mvs-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。
2. 导入项目
在IDE中导入qcloud_video包

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

3. 参照api说明和sdk中提供的demo，开发代码。

1.3 HTTPS支持

如果想使用https协议上传，则将qcloud_video/lib/conf.js文件中变量API_VIDEO_END_POINT中的http修改为https即可。

2 API详细说明

2.1 生成签名

1．接口说明

签名生成方法，可以在服务端生成签名，供移动端app使用。

签名分为2种：

多次有效签名（有一定的有效时间）

单次有效签名（绑定资源url，只能生效一次）

签名的详细描述及使用场景参见鉴权服务技术方案

2．方法

多次有效签名

```
function signMore(bucket, expired);
```

单次有效签名

```
function signOnce(bucket, fileid);
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
secret_id	String	是	无	开发者的授权secret_id，非传入参数，从conf.js中获取，使用前需初始化好conf

secret_key	String	是	无	开发者的授权secret_key，非传入参数，从conf.js中获取，使用前需初始化好conf，以上两项获取参见项目设置
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
fileid	String	是	无	视频文件唯一的标识，格式/appid/bucketname/filepath/filename，其中/filepath/filename为视频文件在此bucketname下的全路径

返回值：签名字符串

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//□□conf.js□□□□□□APPID□SECRET_KEY□□□□□□□□
var expired = parseInt(Date.now() / 1000) + conf.EXPIRED_SECONDS;
var sign1  = qcloud.auth.signMore(bucket, expired);
var
 sign2  = qcloud.auth.signOnce(bucket, '/'+conf.APPID+'/'
+bucketname+'/'+remoteFilepath);
```

2.2 目录操作

2.2.1 创建目录

1．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。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	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	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。

Callback参数,json格式：

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

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.video.createFolder('bucketname', '/myFolder/', function(ret) {
//deal with ret});
```

2.2.2 目录属性更新

1．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。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	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	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。

Callback参数,json格式：

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

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.video.updateFolder('bucketname', '/myFolder/',
'bizAttribute', function(ret) {//deal with ret});
```

2.2.3 目录查询

1． 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。

2． 方法

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

3． 参数和返回值

参数说明：

参数名	类型	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。

Callback参数,json格式：

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

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.video.statFolder('bucketname', '/myFolder/', function(ret) {
//deal with ret});
```

2.2.4 目录删除

1． 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视频文件或目录，将不能删除。

2． 方法

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

3． 参数和返回值

参数说明：

参数名	类型	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。

Callback参数,json格式：

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

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.video.deleteFolder('bucketname', '/myFolder/', function(ret) {
//deal with ret});
```

2.2.5 列举目录下视频文件&目录

1． 接口说明

用于列举目录下视频文件和目录，调用者可以通过此接口查询目录下的视频文件和目录属性。

2． 方法

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

3． 参数和返回值

参数说明：

参数名	类型	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，api会补齐
num	int	否	20	要查询的目录/视频文件数量
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:只是视频文件，ListDirOnly:只是文件夹，eListBoth:全部
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。

Callback参数,json格式：

参数名	类型	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	子视频文件数量(总)
data.infos	Array	是	视频文件、目录集合，可以为空
data.infos.name	String	是	视频文件或目录名
data.infos.biz_attr	String	是	目录或视频文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或视频文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为视频文件时返回)	视频文件大小
data.infos.filelen	Int	否(当类型为视频文件时返回)	视频文件已传输大小(通过与filesize对比可知视频文件传输进度)
data.infos.sha	String	否(当类型为视频文件时返回)	视频文件sha
data.infos.access_url	String	否(当类型为视频文件时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频文件时返回)	转码状态,如:["f10":0,"f20":1,"f30":0] 等
data.infos.video_status	Int	否(当类型为视频文件时返	视频状态码

		回)	
data.infos.video_play_time	Int	否(当类型为视频文件时返回)	视频播放时长,只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频文件时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频文件时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频文件时返回)	各码率的播放url,如:["f10":url1,"f20":url2,"f30":url3]等

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//□□conf.js□□□□□□APPID□SECRET_KEY□□□□□□□□
qcloud.video.list('bucketname', '/myFolder/', 20, 'eListBoth', 0, '',
function(ret) {//deal with ret});
```

2.2.6 列举目录下指定前缀的视频文件&目录

1． 接口说明

用于列举目录下指定前缀的视频文件和目录，调用者可以通过此接口查询目录下的指定前缀的视频文件和目录信息。

2． 方法

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

3．参数和返回值

参数说明：

参数名	类型	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	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	读取视频文件/目录前缀
num	int	否	20	要查询的目录/视频文件数量
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:只是视频文件，ListDirOnly:只是文件夹，eListBoth:全部
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。

Callback参数,json格式：

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	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	子视频文件数量(总)
data.infos	Array	是	视频文件、目录集合，可以为空
data.infos.name	String	是	视频文件或目录名
data.infos.biz_attr	String	是	目录或视频文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或视频文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为视频文件时返回)	视频文件大小
data.infos.filelen	Int	否(当类型为视频文件时返回)	视频文件已传输大小(通过与filesize对比可知视频文件传输进度)
data.infos.sha	String	否(当类型为	视频文件sha

		视频文件时返回)	
data.infos.access_url	String	否(当类型为视频文件时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频文件时返回)	转码状态,如:["f10":0,"f20":1,"f30":0] 等
data.infos.video_status	Int	否(当类型为视频文件时返回)	视频状态码
data.infos.video_play_time	Int	否(当类型为视频文件时返回)	视频播放时长,只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频文件时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频文件时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频文件时返回)	各码率的播放url,如:["f10":url1,"f20":url2,"f30":url3]等

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//□□conf.js□□□□□□APPID□SECRET_KEY□□□□□□□□
qcloud.video.prefixSearch('bucketname', '/myFolder/', '20150606_', 20,
'eListBoth', 0, '', function(ret) {//deal with ret});
```

2.3 视频文件

2.3.1 视频上传

1．接口说明

用于较小视频(一般小于8MB)的上传，调用者可以通过此接口上传较小的视频并获得视频的url，较大的视频请使用分片上传接口。

2．方法

```
function upload
(filePath, bucket, dstpath, videocover, bizattr, title, desc, magiccontext, call
back);
```

3．参数和返回值

参数说明：

参数名	类型	必须	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	视频文件在微视频服务端的全路径，不包括/appid/bucketname
filepath	String	是	无	本地要上传视频文件的全路径
bizattr	String	否	null	视频文件属性，业务端维护
videocover	String	否	null	视频封面的URL
title	String	否	null	视频的标题
desc	String	否	null	视频的描述

magiccontext	String	否	null	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置
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。

Callback参数json格式：

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	生成的视频文件下载url
data.url	String	是	操作视频文件的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//□□conf.js□□□□□□APPID□SECRET_KEY□□□□□□□□
qcloud.video.upload('test.mp4', 'bucketName',
'/myFolder/myVideo.mp4', 'http://video-cover.jpg', 'bizattr', 'title',
'desc', 'magiccontext', function(ret) {//deal with ret});
```

2.3.2 视频分片上传

1． 接口说明

用于较大视频(一般大于8MB)的上传，调用者可以通过此接口上传较大视频并获得视频的url和唯一标识resource_path（用于调用其他api）。

2． 方法

```
function upload_slice
(filePath, bucket, dstpath, videocover, bizattr, title, desc, magiccontext, slice_size, session, callback);
```

3． 参数和返回值

参数说明：

参数名	类型	必须	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	视频文件在微视频服务端的全路径，不包括/appid/bucketname
filepath	String	是	无	本地要上传视频文件的全路径
bizattr	String	否	null	视频文件属性，业务端维护
videocover	String	否	null	视频封面的URL
title	String	否	null	视频的标题
desc	String	否	null	视频的描述
magiccontext	String	否	null	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置

slice_size	Int	否	512*1024字节	分片大小，用户可以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。
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。

Callback参数json格式：

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	生成的视频文件下载url
data.url	String	是	操作视频文件的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//□□conf.js□□□□□□APPID□SECRET_KEY□□□□□□□□
qcloud.video.upload_slice('test.mp4', 'bucketName',
'/myFolder/myVideo.mp4', 'http://video-cover.jpg', 'bizattr', 'title',
'desc', 'magiccontext', 2*1024*1024, null, function(ret) {//deal with ret});
```

2.3.3 视频属性更新

1． 接口说明

用于视频属性的更新，调用者可以通过此接口更新视频的Title，Desc和自定义属性字段。

2． 方法

```
function updateFile(bucket, path, title, desc, bizattr, videocover, callback);
```

3． 参数和返回值

参数说明：

参数名	类型	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	视频文件在微视频服务端的全路径，不包括/appid/bucketname
bizattr	String	否	null	待更新的视频文件属性信息
videocover	String	否	null	视频封面的URL
title	String	否	null	视频的标题
desc	String	否	null	视频的描述
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。

Callback参数json格式：

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

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//□□conf.js□□□□□□APPID□SECRET_KEY□□□□□□□□
qcloud.video.updateFile('bucketName', '/myFolder/test.mp4', 'title',
'desc', 'bizattr', 'http://video-cover.jpg', function(ret) {//deal with ret});
```

2.3.4 视频查询

1． 接口说明

用于视频的查询，调用者可以通过此接口查询视频的各项属性信息。

2． 方法

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

3． 参数和返回值

参数说明：

参数名	类型	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	视频文件在微视频服务端

				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。

Callback参数,json格式：

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	视频属性数据
data.name	String	是	视频文件或目录名
data.biz_attr	String	是	视频属性，业务端维护
data.video_cover	String	是	视频封面的URL
data.ctime	String	是	视频的创建时间，unix时间戳
data.mtime	String	是	视频的修改时间，unix时间戳
data.filesize	Int	是	视频文件大小
data.filelen	Int	是	视频文件已传输大小(通过与filesize对比可知视频文件传输进度)
data.sha	String	是	视频文件sha
data.access_url	String	是	生成的视频下载url
data.trans_status	Array	是	转码状态,如:["f10":0,"f20":1,"f30":0] 等
data.video_status	Int	是	视频状态码
data.video_play_time	Int	是	视频播放时长,只有使用视频转码的业务才有
data.video_title	String	是	视频标题
data.video_desc	String	是	视频描述
data.video_play_url	Array	是	各码率的播放url,如:["f10":url1,"f20":url2,"f30":url3]

			等

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.video.statFile('bucketName', '/myFolder/test.mp4', function(ret) {
//deal with ret});
```

2.3.5 视频删除

1．接口说明

用于视频的删除，调用者可以通过此接口删除已经上传的视频。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	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	视频文件在微视频服务端的全路径，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。

Callback参数json格式：

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

示例代码：

```
var qcloud = require('qcloud_video');
qcloud.conf.setAppInfo('10000','xxxxxx','xxxxxx');
//如果conf.js中已经设置好APPID和SECRET_KEY则不需要此步骤。
qcloud.video.deleteFile('bucketName', '/myFolder/test.mp4', function(ret) {
//deal with ret});
```

C++-SDK说明

1 开发准备

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

1.1 前期准备

前期准备
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/

1.2 集成SDK

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

需要将sample.cpp里的appid、secretId、secretKey、bucket等信息换成你自己的。
生成的sample就可以直接运行，试用，生成的静态库，名称为:libuvsdk.a。

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

例如我的项目里只有一个sample.cpp,项目目录和sdk在同级目录，
copy libuvsdk.a 到项目所在目录那么编译命令为:
g++ -o sample sample.cpp -I ./include/ -L. -L../uvs-cpp-sdk/lib/ -luvsdk -lcurl -lcrypto -lssl -lrt -ljsoncpp

1.3 https支持

修改video.cpp中API_VIDEO_END_POINT的值为：https://web.video.myqcloud.com/files/v1/

2 API详细说明

2.1 生成签名

1． 接口说明
签名生成方法，可以在服务端生成签名，供移动端app使用。
签名分为2种：
多次有效签名（有一定的有效时间）
单次有效签名（绑定资源url，只能生效一次）
签名的详细描述及使用场景参见鉴权技术服务方案
2． 方法
多次有效签名

```
static string appSign(
    const uint64_t appId, const string &secretId, const string &secretKey,
    const uint64_t expired, const string &bucketName);
```

单次有效签名

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

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
expired	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
path	String	是	无	视频路径，以斜杠开头，例如/filepath/filename，为视频在此bucketname下的全路径

返回值：签名字符串

示例代码：

```
uint64_t expired = time(NULL) + 60;
string sign = Auth::appSign(10000000, "SecretId", "SecretKey"
, expired , "bucketName");
string resourcePath= "/myFloder/myFile.rar";
sign = Auth::appSign_once(path, bucketName);
```

2.2 目录操作

2.2.1 创建目录

1． 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。

2． 方法

```
int createFolder(const string &bucketName, const string &path, const string &biz_attr ="");
```

3． 参数和返回值

参数说明：

参数名	类型	必须	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
biz_attr	String	否	空	目录绑定的属性信息，业务自行维护

通过类的成员变量Json::Value retJson返回请求结果：

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

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string bucketName = "myBucket";
string path = "/myFolder/";
string bizAttr = "attr_folder";
int result  = video.createFolder(bucketName, path,bizAttr);
video.dump_res();
```

2.2.2 目录属性更新

1．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。

2．方法

```
 int updateFolder(const string &bucketName, const string &path, const string &biz_attr = "");
```

3．参数和返回值

参数说明：

参数名	类型	必须	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
biz_attr	String	是	空	新的目录绑定的属性信息

通过类的成员变量Json::Value retJson返回请求结果：

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

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string bucketName = "myBucket";
string path = "/myFolder/";
string bizAttr = "attr_folder";
int result  = video.updateFolder(bucketName, path,bizAttr);
video.dump_res();
```

2.2.3 目录查询

1．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。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	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

通过类的成员变量Json::Value retJson返回请求结果：

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.video_cover	String	视频封面的URL
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

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string bucketName = "myBucket";
string path = "/myFolder/";
int result  = video.statFolder(bucketName, path);
video.dump_res();
```

2.2.4 目录删除

1．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视频或目录，将不能删除。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketNam	String	是	无	bucket名称，bucket创

e				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

通过类的成员变量Json::Value retJson返回请求结果：

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

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string bucketName = "myBucket";
string path = "/myFolder/";
int result  = video.delFolder(bucketName, path);
video.dump_res();
```

2.2.5 列举目录下视频&目录

1．接口说明

用于列举目录下视频和目录，调用者可以通过此接口查询目录下的视频和目录属性。

2．方法

```
int listFolder(const string &bucketName, const string &path, const int num = 20
```

```
, const string &pattern = "eListBoth",
const int order = 0, const string &context = "");
```

3．参数和返回值

参数说明：

参数名	类型	必须	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
num	int	否	20	要查询的目录/视频数量
context	String	否	空格	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	eListBoth,eListDirOnly, eListFileOnly 默认eListBoth

通过类的成员变量Json::Value retJson返回请求结果：

参数名	类型	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	子视频数量(总)
data.infos	Array	是	视频、目录集合，可以为空
data.infos.name	String	是	视频或目录名
data.infos.biz_attr	String	是	目录或视频属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或视频的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为视频时返回)	视频大小
data.infos.filelen	Int	否(当类型为视频时返回)	视频已传输大小(通过与filesize对比可知视频传输进度)
data.infos.sha	String	否(当类型为视频时返回)	视频sha
data.infos.access_url	String	否(当类型为视频时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频时返回)	转码状态,如: {“f10”：0, “f20”：1}等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_s	Int	否(当类型为	视频状态码

tatus		视频时返回)	0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频时返回)	视频播放时长,只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频时返回)	各码率的播放url,如:["f10":url1,"f20":url2,"f30":url3]等

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string bucketName = "myBucket";
string path = "/myFolder/";
int result  = video.listFolder(bucketName, path);
video.dump_res();
```

2.2.6 列举目录下指定前缀视频&目录

1． 接口说明

用于列举目录下指定前缀的视频和目录，调用者可以通过此接口查询目录下的指定前缀的视频和目录信息。

2． 方法

```
 int prefixSearch(const string &bucketName, const string &path, const int num =
20, const string &pattern = "eListBoth",
```

```
const int order = 0, const string &context= "");
```

3．参数和返回值

参数说明：

参数名	类型	必须	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视频(带全路径)
num	int	否	20	要查询的目录/视频数量
context	String	否	空	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	eListBoth,eListDirOnly, eListFileOnly 默认eListBoth

通过类的成员变量Json::Value retJson返回请求结果：

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	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	子视频数量(总)
data.infos	Array	是	视频、目录集合，可以为空
data.infos.name	String	是	视频或目录名
data.infos.biz_attr	String	是	目录或文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为视频时返回)	视频大小
data.infos.filelen	Int	否(当类型为视频时返回)	视频已传输大小(通过与filesize对比可知视频传输进度)
data.infos.sha	String	否(当类型为视频时返回)	视频sha
data.infos.access_url	String	否(当类型为视频时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频时返回)	转码状态,如: {“f10”:0, “f20”:1}等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_s	Int	否(当类型为	视频状态码

tatus		视频时返回)	0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频时返回)	视频播放时长,只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频时返回)	各码率的播放url,如:["f10":url1,"f20":url2,"f30":url3]等

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string bucketName = "myBucket";
string path = "/myFolder/";
int result  = video.prefixSearch(bucketName, path);
video.dump_res();
```

2.3 视频操作

2.3.1 视频上传

1． 接口说明

用于较小视频(一般小于8MB)的上传，调用者可以通过此接口上传较小的视频并获得视频的url，较大的视频请使用分片上传接口。

2． 方法

```
int upload(const string &amp;srcPath, const string
 &amp;bucketName, const string &amp;dstPath, const string &amp;videoCover,
const string &amp;bizAttr = "", const string &amp;title = "",const string
 &amp;desc = "",const string &amp;magicContext = "");
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
srcPath	String	是	无	本地要上传视频的全路径
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
dstPath	String	是	无	视频在微视频服务端的全路径，不包括/appid/bucketname
bizAttr	String	否	空	视频属性，业务端维护
videoCover	String	否	空	视频封面的URL
title	String	否	空	视频的标题
desc	String	否	空	视频的描述
magicContext	String	否	空	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置

通过类的成员变量Json::Value retJson返回请求结果：

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	生成的视频下载url
data.url	String	是	操作视频的url
data.resource_path	String	是	资源路径. 不包含/appid/bucket

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string srcPath= "/data/test.mp4";
string bucketName = "myBucket";
string dstPath = "/myFolder/test.mp4";
int
 result  = video.upload(srcPa
th,bucketName,dstPath,"attr","title","desc","magicContext");
video.dump_res();
```

2.3.2 视频分片上传

1．接口说明

用于较大视频(一般大于8MB)的上传，调用者可以通过此接口上传较大视频并获得视频的url和唯一标识resource_path（用于调用其他api）。

2．方法

```
int upload_slice(const string &amp;srcPath, const string
 &amp;bucketName, const string &amp;dstPath, const string
 &amp;videoCover, const string &amp;bizAttr = "",
const string &amp;title = "",const string &amp;desc = "",const string
```

```
&amp;magicContext = "", const int sliceSize = 0, const string
&amp;session = "");
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
srcPath	String	是	无	本地要上传视频的全路径
bucketName	String	是	无	bucket名称，bucket创建参见创建Bucket
dstPath	String	是	无	视频在微视频服务端的全路径，不包括/appid/bucketname
bizAttr	String	否	空	视频属性，业务端维护
videoCover	String	否	空	视频封面的URL
title	String	否	空	视频的标题
desc	String	否	空	视频的描述
magicContext	String	否	空	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置
sliceSize	Int	否	512*1024字节	分片大小，用户可以根据网络状况自行设置
session	String	否	空	如果是断点续传，则带上(唯一标识此视频传输过程的id，由后台下发，调用方透传)

通过类的成员变量Json::Value retJson返回请求结果：

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	生成的视频下载url
data.url	String	是	操作视频的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string srcPath= "/data/test.mp4";
string bucketName = "myBucket";
string dstPath = "/myFolder/test.mp4";
int
 result  = video.upload
_slice(srcPath,bucketName,dstPath,"attr","title","desc","magicContext");
video.dump_res();
```

2.3.3 视频属性更新

1． 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。

2． 方法

```
 int update(const string &bucketName, const string &path, const string &videoCov
er, const string &biz_attr = "", const string &title = "", const string &desc =
```

"");

3．参数和返回值

参数说明：

参数名	类型	必须	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	视频在微视频服务端的全路径，不包括/appid/bucketname
biz_attr	String	否	空	待更新的视频属性信息
videoCover	String	否	空	视频封面的URL
title	String	否	空	视频的标题
desc	String	否	空	视频的描述

通过类的成员变量Json::Value retJson返回请求结果：

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

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string path = "/data/test.mp4";
string bucketName = "myBucket";
int result = video.update(bucketName,path,"attr","title","desc");
video.dump_res();
```

2.3.4 视频查询

1． 接口说明

用于视频的查询，调用者可以通过此接口查询视频的各项属性信息。

2． 方法

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

3． 参数和返回值

参数说明：

参数名	类型	必须	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	视频在微视频服务端的全路径，不包括/appid/bucketname

通过类的成员变量Json::Value retJson返回请求结果：

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	视频属性数据
data.name	String	是	视频或目录名
data.biz_attr	String	是	视频属性，业务端维护
data.video_cover	String	是	视频封面的URL
data.ctime	String	是	视频的创建时间，unix时间戳
data.mtime	String	是	视频的修改时间，unix时间戳
data.filesize	Int	是	视频文件大小
data.filelen	Int	是	视频文件已传输大小(通过与filesize对比可知文件传输进度)
data.sha	String	是	视频文件sha
data.access_url	String	是	生成的视频文件下载url
data.infos.trans_status	Array	否(当类型为视频时返回)	转码状态,如: {“f10”:0, “f20”:1}等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_status	Int	否(当类型为视频时返回)	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频时返回)	视频播放时长, 只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频文件时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频文件时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频文件时返	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3]

		回)	等

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string bucketName = "myBucket";
string path = "/myFolder/test.mp4";
int result = video.stat(bucketName, path);
video.dump_res();
```

3.3.5 视频删除

1．接口说明

用于视频的删除，调用者可以通过此接口删除已经上传的视频。

2．方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	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	视频在微视频服务端的全路径，不包括/appid/bucketname

通过类的成员变量Json::Value retJson返回请求结果：

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

示例代码：

```
Video video("your appid", "your secretId", "your secretKey",
"interface timeout");
string bucketName = "myBucket";
string path = "/myFolder/test.mp4";
int result = video.del(bucketName, path);
video.dump_res();
```

CSharp-SDK说明

1 开发准备

微视频服务的C# SDK的下载地址： https://github.com/tencentyun/mvs-dotnet-sdk

1.1 前期准备

前期准备

1. sdk依赖C# 4.0版本及以上， 推荐使用相同的版本。
2. 通过项目设置获取appid，bucket，secret_id和secret_key；

1.2 获取SDK

直接下载github上提供的源代码，集成到您的开发环境。

1.3 HTTPS支持

如果想使用https协议上传，则将video_dotnet_sdk/Api/VideoCloud.cs文件中变量VIDEOAPI_CGI_URL中的http修改为https即可。

2 API详细说明

2.1 生成签名

1． 接口说明

签名生成方法，可以在服务端生成签名，供移动端app使用。

签名分为2种：

多次有效签名（有一定的有效时间）

单次有效签名（绑定资源url，只能生效一次）

签名的详细描述及使用场景参见鉴权服务技术方案

2．方法

多次有效签名

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

单次有效签名

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

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
appId	int	是	无	AppId
secretId	string	是	无	Secret Id
secretKey	string	是	无	Secret Key，以上三项通过项目设置获取
expired	long	是	无	过期时间，Unix时间戳
bucketName	string	是	无	bucket名称，bucket创建参见创建Bucket

remotePath	string	是	无	视频文件唯一的标识，格式/appid/bucketname/filepath/filename，其中/filepath/filename为视频文件在此bucketname下的全路径，

返回值：签名字符串

示例代码：

```
var
 sign =
 Sign.Signature(appId, secretId, secretKey, expired, bucketName); //多次有效签名调用例子
var
 s
ign
= Sign.S
ignatureOnce(app
Id, secretId, secretKey, (remote
Path.StartsWith("/")&amp;nbsp;? ""&amp;nbsp;: "/"
) + remotePath, bucketName); //单次有效签名调用例子
```

2.2 目录操作

2.2.1 创建目录

1．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。

2．方法

```
public string CreateFolder(string bucketName, string remotePath, string bizAttr
ibute = "")
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketNam e	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
bizAttribute	string	否	空字符串	目录绑定的属性信息，业务自行维护

返回值,json格式字符串：

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	返回数据
data.ctime	String	目录的创建时间，unix时间戳
data.resour ce_path	String	目录的资源路径

示例代码：

```
var result = video.CreateFolder("myBucket", "/sdk/");
```

2.2.2 目录属性更新

1．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。

2．方法

```
 public string UpdateFolder(string bucketName, string remotePath, string bizAttr
ibute)
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketNam e	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	需要创建目录的全路径，以"/"开头,以"/"结尾，api会补齐
bizAttribute	string	是	无	新的目录绑定的属性信息

返回值,json格式字符串：

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

示例代码：

```
var result = video.UpdateFolder("myBucket", "/sdk/", "test folder");
```

2.2.3 目录查询

1． 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。

2． 方法

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

3． 参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	目录的全路径，以"/"开

				头,以"/"结尾，api会补齐

返回值,json格式字符串：

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.video_cover	String	视频封面的URL
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

示例代码：

```
var result = video.GetFolderStat("myBucket", "/sdk/");
```

2.2.4 目录删除

1． 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视频文件或目录，将不能删除。

2． 方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	目录的全路径，以"/"开头,以"/"结尾，api会补齐

返回值,json格式字符串：

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

示例代码：

```
var result = video.DeleteFolder("myBucket", "/sdk/");
```

2.2.5 列举目录下视频文件&目录

1．接口说明

用于列举目录下视频文件和目录，调用者可以通过此接口查询目录下的视频文件和目录属性。

2．方法

```
public string GetFolderList(string bucketName, string remotePath, int num, stri
ng context, int order, FolderPattern pattern, string prefix = "")
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketNam e	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	目录的全路径，以"/"开头,以"/"结尾，api会补齐
num	int	是	无	要查询的目录/视频文件数量
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	FolderPa ttern	是	无	枚举类型，值：File,Folder,Both

prefix	string	否	空字符	搜索前缀

返回值,json格式字符串：

参数名	类型	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	子视频文件数量(总)
data.infos	Array	是	视频文件、目录集合，可以为空
data.infos.name	String	是	视频文件或目录名
data.infos.biz_attr	String	是	目录或视频文件属性，业务端维护
data.infos.video_cover	String	是	视频封面的URL
data.infos.ctime	String	是	目录或视频文件的创建时间，unix时间戳
data.infos.mtime	String	是	目录或视频文件的修改时间，unix时间戳
data.infos.filesize	Int	否(当类型为视频文件时返回)	视频文件大小
data.infos.filelen	Int	否(当类型为视频文件时返	视频文件已传输大小(通过与filesize对比可知视频文件传输进度)

		回)	
data.infos.sha	String	否(当类型为视频文件时返回)	视频文件sha
data.infos.access_url	String	否(当类型为视频文件时返回)	生成的视频下载url
data.infos.trans_status	Array	否(当类型为视频文件时返回)	转码状态,如: { “f10” : 0, “f20” : 1} 等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.infos.video_status	Int	否(当类型为视频文件时返回)	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.infos.video_play_time	Int	否(当类型为视频文件时返回)	视频播放时长, 只有使用视频转码的业务才有
data.infos.video_title	String	否(当类型为视频文件时返回)	视频标题
data.infos.video_desc	String	否(当类型为视频文件时返回)	视频描述
data.infos.video_play_url	Array	否(当类型为视频文件时返回)	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3] 等

示例代码：

```
var result = video.GetFolderList("myBucket", "/", 20, "", 0
```

```
, FolderPattern.Both);
```

2.3 视频文件

2.3.1 视频上传

1．接口说明

用于较小视频(一般小于8MB)的上传，调用者可以通过此接口上传较小的视频并获得视频的url，较大的视频请使用分片上传接口。

2．方法

```
public string UploadFile(string bucketName, string remotePath, string localPath,
 string videoCover, string bizAttribute = "", string title = "", string desc = "
", string magicContext = "")
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	视频在服务端的全路径，不包括/appid/bucketname
localPath	string	是	无	视频本地路径
bizAttribute	string	否	空串	视频属性，业务端维护
videoCover	string	否	空串	视频封面的URL
title	string	否	空串	视频的标题

desc	string	否	空串	视频的描述
magicContext	string	否	空串	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置

返回值,json格式字符串：

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	生成的视频下载url
data.url	String	是	操作视频的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
var result = video.UploadFile("myBucket", "/test.mp4", "F:\\test.mp4");
```

2.3.2 视频分片上传

1． 接口说明

用于较大视频(一般大于8MB)的上传，调用者可以通过此接口上传较大视频并获得视频的url和唯一标识resource_path（用于调用其他api）。

2． 方法

```
public string SliceUploadFile(string bucketName, string
 remotePath, string localPath, string videoCover = "", string
 bizAttribute = "", string title = "", string desc = "", string
 magicContext = "", int sliceSize = 512 * 1024)
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	视频在服务端的全路径，不包括/appid/bucketname
localPath	string	是	无	视频本地路径
bizAttribute	string	否	空串	视频属性，业务端维护
videoCover	string	否	空串	视频封面的URL
title	string	否	空串	视频的标题
desc	string	否	空串	视频的描述
magicContext	string	否	空串	透传字段，微视频会将此字段信息透传给业务设定的回调url，具体参见回调设置
sliceSize	int	否	512*1024字节	分片大小，用户可以根据网络状况自行设置

返回值,json格式字符串：

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	生成的视频下载url
data.url	String	是	操作视频的url
data.resource_path	String	是	资源路径. 格式:/appid/bucket/xxx

示例代码：

```
var result = video.SliceUploadFile("myBucket", "/test.mp4",
"F:\\test.mp4", 512 * 1024);
```

2.3.3 视频属性更新

1．接口说明

用于视频属性的更新，调用者可以通过此接口更新视频的Title，Desc和自定义属性字段。

2．方法

```
 public string UpdateFile(string bucketName, string remotePath, string bizAttrib
ute, string videoCover = null, string title = "", string desc = "")
```

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	视频文件在视频服务端的全路径，不包括/appid/bucketname
bizAttribute	string	是	无	待更新的视频文件属性信息
videoCover	string	是	无	视频封面的URL
title	string	否	空串	视频的标题
desc	string	否	空串	视频的描述

返回值,json格式字符串：

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

示例代码：

```
result = video.UpdateFile("myBucket", "/sdk/xx.mp4", "test file");
```

2.3.4 视频查询

1． 接口说明

用于视频文件的查询，调用者可以通过此接口查询视频文件的各项属性信息。

2． 方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	视频文件在视频服务端的全路径，不包括/appid/bucketname

返回值,json格式字符串：

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	视频文件属性数据
data.name	String	是	视频文件或目录名
data.biz_attr	String	是	视频文件属性，业务端维护
data.video_cover	String	是	视频封面的URL
data.ctime	String	是	视频文件的创建时间，unix时间戳
data.mtime	String	是	视频文件的修改时间，unix时间戳
data.filesize	Int	是	视频文件大小
data.filelen	Int	是	视频文件已传输大小(通过与filesize

			对比可知视频文件传输进度)
data.sha	String	是	视频文件sha
data.access_url	String	是	生成的视频文件下载url
data.trans_status	Array	是	转码状态,如: { “f10” : 0, “f20” : 1} 等 f10:低清，f20:标清，f30:高清 状态码：0,初始化中，1，转码中;2，转码成功;3，转码失败;
data.video_status	Int	是	视频状态码 0,初始化中，1，视频入库中;2，上传成功;
data.video_play_time	Int	是	视频播放时长, 只有使用视频转码的业务才有
data.video_title	String	是	视频标题
data.video_desc	String	是	视频描述
data.video_play_url	Array	是	各码率的播放url, 如:["f10":url1,"f20":url2,"f30":url3] 等

示例代码：

```
var result = video.GetFileStat("myBucket", "/sdk/xx.mp4");
```

2.3.5 视频删除

1． 接口说明

用于视频的删除，调用者可以通过此接口删除已经上传的视频。

2． 方法

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

3．参数和返回值

参数说明：

参数名	类型	必须	默认值	参数描述
bucketName	string	是	无	bucket名称，bucket创建参见创建Bucket
remotePath	string	是	无	视频文件在视频服务端的全路径，不包括/appid/bucketname

返回值,json格式字符串：

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

示例代码：

```
var result = video.DeleteFile("myBucket", "/sdk/xx.mp4");
```

鉴权及签名文档

1 签名与鉴权

腾讯移动服务通过签名来验证请求的合法性。开发者通过将签名授权给客户端，使其具备上传下载及管理指定资源的能力。签名分为多次有效签名和单次有效签名：

多次有效签名：

签名中不绑定文件fileid，需要设置大于当前时间的有效期，有效期内此签名可多次使用，有效期最长可设置三个月。

单次有效签名：

签名中绑定文件fileid，有效期必须设置为0，此签名只可使用一次，且只能应用于被绑定的文件。

具体适用场景参见签名适用场景。

2 签名算法

2.1 获取签名所需信息

生成签名所需信息包括项目ID（appid），空间名称（bucket,视频资源的组织管理单元），项目的Secret ID和Secret Key。获取这些信息的方法如下：

1) 登录 微视频-视频空间, 进入视频空间；

2) 如开发者未创建视频空间，可添加视频空间，空间名称（bucket）由用户自行输入；

3) 登陆视频空间-项目设置，进入项目设置；

4) 如果开发者未添加密钥，则需添加密钥，获取项目的Secret ID和Secret Key，每个项目最多添加两对密钥；如果已经添加过密钥则直接获取项目的Appid、Secret ID和Secret Key。

注：

(1) 添加视频空间可参考添加视频空间；

(2) 密钥管理可参考添加密钥。

2.2 拼接签名串

拼接多次有效签名串：

a=[appid]&b=[bucket]&k=[SecretID]&e=[expiredTime]&t=[currentTime]&r=[rand]&f=

拼接单次有效签名串：

a=[appid]&b=[bucket]&k=[SecretID]&e=[expiredTime]&t=[currentTime]&r=[rand]&f=[fileid]

签名串中各字段含义如下：

字段	解释
a	开发者的项目ID，接入微视频创建空间时系统生成的唯一标识项目的ID，即Appid
b	视频空间名称bucket
k	项目的Secret ID
e	签名的有效期，是一个符合UNIX Epoch时间戳规范的数值，单位为秒；单次签名时，e必须设置为0
t	当前时间戳，是一个符合UNIX Epoch时间戳规范的数值，单位为秒，多次签名时，e应大于t
r	随机串，无符号10进制整数，用户需自行生成，最长10位
fileid	资源存储的唯一标识，格式为"/appid/bucketname/用户自定义路径或资源名"，并且需要对其中非'/'字符进行urlencode编码

注：

拼接单次有效签名的签名串时，过期时间e必须设置为0

，以保证此签名只能只能针对固定资源且只能使用一次；

删除文件必须使用单次有效签名；

上传必须使用多次有效签名；

具体适用场景参见签名适用场景。

2.3 生成签名

1. 微视频使用 HMAC-SHA1 算法对请求进行签名；
2. 签名串需要使用 Base64 编码。

   即生成签名的公式如下：

```
SignTmp = HMAC-SHA1(SecretKey, orignal)
Sign = Base64(SignTmp.orignal)
```

其中SecretKey为2.1节获取的Secret Key，orignal为2.2节中拼接好的签名串，首先对orignal使用HMAC-SHA1算法进行签名，然后将orignal附加到签名结果的末尾，再进行Base64编码，得到最终的sign。
注：此处使用的是标准的Base64编码，不是urlsafe的Base64编码，请注意。

3 实例

本节介绍生成签名的算法实例，实例中使用PHP语言，如果开发者使用其他与开发，请使用对应的算法。

3.1 获取签名所需信息

获取得到的签名所需信息如下。

项目ID：200001
空间名称（bucket）：newbucket
Secret ID：AKIDUfLUEUigQiXqm7CVSspKJnuaiIKtxqAv
Secret Key：bLcPnl88WU30VY57ipRhSePfPdOfSruK

3.2 拼接签名串

拼接的多次有效签名串如下：
a=200001&b=newbucket&k=AKIDUfLUEUigQiXqm7CVSspKJnuaiIKtxqAv&e=1438669115&t=1436077115&r=11162&f=
拼接的单次有效签名串如下：
a=10001290&b=tencentyun&k=AKIDgaoOYh2kOmJfWVdH4lpfxScG2zPLPGoK&e=0&t=1436077115&r=11162&f=tencentyunSignTest

```
$appid = "200001";
$bucket = "newbucket";
$secret_id = "AKIDUfLUEUigQiXqm7CVSspKJnuaiIKtxqAv";
$secret_key = "bLcPnl88WU30VY57ipRhSePfPdOfSruK";
$expired = time() + 60;
```

```
$onceExpired = 0;
$current = time();
$rdm = rand();
$userid = "0";
$fileid = "/200001/newbucket/tencent_test.jpg";

$srcStr = 'a='.$appid.'&b='.$bucket.'&k='.$secret_id.'&e='.$expired.'&t='
.$current.'&r='.$rdm.'&f=';

$srcStrOnce= 'a='.$appid.'&b='.$bucket.'&k='.$secret_id.'&e='
.$onceExpired .'&t='.$current.'&r='.$rdm
.'&f='.$fileid;
```

3.3 生成签名

```
$signStr = base64_encode(hash_hmac('SHA1', $srcStr, $secret_key, true).$srcStr);

$signStrOnce = base64_encode(hash_hmac('SHA1',$srcStrOnce,$secret_key, true).
$srcStrOnce);

echo $signStr."\n";

echo $signStrOnce."\n";
```

最终得到的多次有效签名为：

vxzLR6vzMNhBMUVzMTWKUB+LMeVhPTIwMDAwMSZrPUFLSURVZkxVRVVpZ1FpWHFtN0
NWU3NwS0pudWFpSUt0eHFBdiZlPTE0Mzc5OTU3MDQmdD0xNDM3OTk1NjQ0JnI9MjA4
MTY2MDQyMSZmPSZiPW5ld2J1Y2tldA==

单次有效签名为：

f11dDSuw86CR02Ko1INzsZstbRlhPTIwMDAwMSZrPUFLSURVZkxVRVVpZ1FpWHFtN0
NWU3NwS0pudWFpSUt0eHFBdiZlPTAmdD0xNDM3OTk1NjQ1JnI9MTE2NjcxMDc5MiZm

PS8yMDAwMDEvbmV3YnVja2V0L3RlbmNlbnRfdGVzdC5qcGcmYj1uZXdidWNrZXQ=

4 签名适用场景

微视频对签名的适用场景做了如下限制：

场景	适用签名
下载（不开启token防盗链）	不验证签名
上传	多次有效签名
查询目录、文件	
创建目录	
下载（开启token防盗链）	
删除目录、文件	单次有效签名
更新目录、文件	