ALIBABA CLOUD

阿里云

视频点播

上传SDK

文档版本：20200916

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.使用说明

阿里云视频点播上传SDK是阿里视频云端到云到端服务的重要一环，为您提供上传媒体文件到点播存储的开发工具包。集成点播上传SDK，就可以快捷上传包括视频、音频、图片、字幕等在内的各种媒体文件。同时提供服务端、Web端、移动端等多种版本SDK，全面适配各个主流平台和运行环境。

功能介绍

视频点播上传SDK支持上传多种格式的媒体文件，以及上传时的各种附加设置，如上传时设置媒体元数据(标题等)、视频封面、转码参数、事件通知和消息回调等，详情参考媒体上传概述，其中也说明了视频上传后何时能播放等热点问题。

此外，服务端上传SDK还支持网络文件的上传，指定文件URL即可自动下载并上传到点播存储，以及支持m3u8视频的上传等。

客户端上传SDK还支持文件列表管理、网络切换（移动端3G/4G<->WiFi切换）、上传控制等，详情可参考各版本上传SDK。

准备工作

服务开通

- 确认您已 注册 阿里云账号，并完成了 实名认证。
- 确认您已开通 视频点播服务，并完成了相关配置，详情参考快速入门。

账号准备

准备好访问点播服务使用的Access Key，相关概念和介绍请参考账号和授权。

- 服务端上传：推荐使用子账号Access Key，在RAM访问控制台 创建子账号，并授予VOD权限（如 `Aliyun VODFullAccess` ）后进行相应操作，子账号配置参考RAM子账号访问。
- 客户端上传：先配置RAM子账号，后续通过其Access Key访问媒体上传接口获取上传地址和凭证，再下发给客户端进行上传。

> 说明 同时也支持主账号Access Key 和STS方式上传，但主账号权限过大、一旦泄露风险巨大；而STS配置又较为繁琐，都不推荐使用。

开发环境

- 客户端上传SDK，支持Android、iOS、移动端&PC端Web浏览器上传。
- 服务端上传SDK，支持Linux、Windows、Mac等多种平台和运行环境。

> 说明 请提前安装好对应开发语言的编译器或解释器，并完成相关环境配置，具体版本要求可参考各上传SDK的说明文档。

客户端上传

在客户端上传媒体文件时，会直接将文件上传到点播存储（基于OSS），不会再经服务端进行中转，故客户端上传必须进行鉴权，也就是需要您在应用服务器上部署授权服务。

目前客户端上传SDK支持两种授权方式：

- 推荐使用上传地址和凭证上传，建议使用服务端SDK来获取上传地址和凭证。

- 同时也支持使用STS方式上传，STS配置参考STS临时授权访问。相对于上传地址和凭证，STS方式配置较为繁琐，不建议使用，二者对比可参考凭证和STS对比。

当前提供三种客户端上传SDK：

- Android上传SDK，注意先完成工程配置。
- iOS上传SDK，注意先完成iOS工程配置。
- Web端上传SDK，使用JavaScript开发，支持PC端和移动端Web浏览器上传等。

服务端上传

服务端上传SDK封装了获取上传地址和凭证、解析凭证上传文件到点播存储等底层细节，您只需要部署在应用服务器上，指定Access Key、文件路径即可进行上传。

当前提供四种服务端上传SDK：

- Java上传SDK
- Python上传SDK
- PHP上传SDK
- C_C++上传SDK

更多语言版本的上传SDK，如 .NET、Node.js、Go等都在排期或开发中，后续会放出，敬请期待。

2.客户端上传

2.1. 使用上传地址和凭证上传

简介

背景

在开发指南-客户端上传里，介绍了客户端提供了两种授权方式上传到点播：

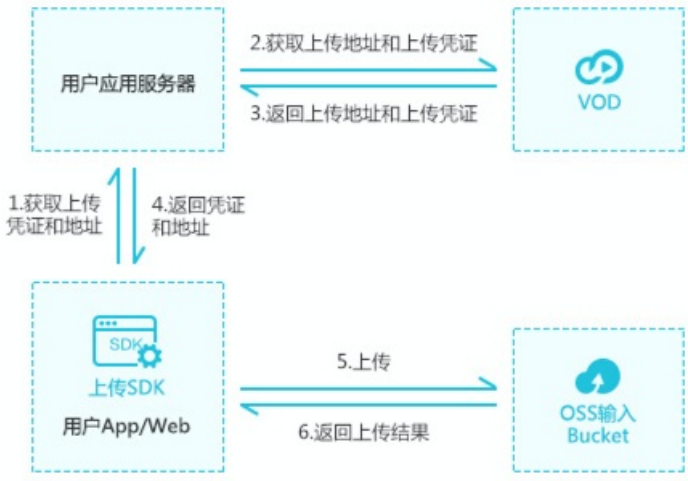
- 使用上传地址和凭证上传
- 使用STS方式上传

本文重点介绍使用上传地址和凭证进行上传的实现过程，这也是**点播服务默认推荐的上传方式**，相比STS方式拥有诸多优势。

上传地址和凭证介绍

上传地址和凭证 是点播服务下发的上传媒体文件到点播存储的授权凭证和文件地址。详情参考上传地址和凭证。

基本流程



实现过程

1. 完成准备工作，建议使用子账号AccessKey访问点播服务，注意务必要授予点播权限（AliyunVODFullAccess）。
2. 在应用服务器上，部署授权服务，获取上传地址和凭证，建议使用服务端SDK方式。
3. 客户端上传时，从应用服务器获取上传地址和凭证。
4. 客户端添加本地文件，并设置上传地址和凭证，开始上传。

设置上传地址和凭证

上传地址和凭证是跟每个文件一一对应的，所以建议在开始上传的回调里面再去用户应用服务器（AppServer）进行获取并设置给上传实例。具体代码如下：

iOS示例代码

```
// create VODUploadClient object
self.uploader = [VODUploadClient new];
// weakself
__weak typeof(self) weakSelf = self;
// setup callback
OnUploadFinishedListener FinishCallbackFunc = ^(UploadFileInfo* fileInfo, VodUploadResult* result){
    NSLog(@"upload finished callback videoid:%@, imageurl:%@", result.videoId, result.imageUrl);
};
OnUploadFailedListener FailedCallbackFunc = ^(UploadFileInfo* fileInfo, NSString *code, NSString* mes
sage){
    NSLog(@"upload failed callback code = %@, error message = %@", code, message);
};
OnUploadProgressListener ProgressCallbackFunc = ^(UploadFileInfo* fileInfo, long uploadedSize, long
totalSize) {
    NSLog(@"upload progress callback uploadedSize：%li, totalSize：%li", uploadedSize, totalSize);
};
OnUploadTokenExpiredListener TokenExpiredCallbackFunc = ^{
    NSLog(@"upload token expired callback.");
 // token过期，设置新的上传凭证，继续上传
};
OnUploadRertyListener RetryCallbackFunc = ^{
    NSLog(@"upload retry begin callback.");
};
OnUploadRertyResumeListener RetryResumeCallbackFunc = ^{
    NSLog(@"upload retry end callback.");
};
OnUploadStartedListener UploadStartedCallbackFunc = ^(UploadFileInfo* fileInfo) {
    NSLog(@"upload upload started callback.");
    // 设置上传地址 和 上传凭证
```

```
    [weakSelf.uploader setUploadAuthAndAddress:fileInfo uploadAuth:`upload auth` uploadAddress:`upload address`];
};

VODUploadListener *listener = [[VODUploadListener alloc] init];
listener.finish = FinishCallbackFunc;
listener.failure = FailedCallbackFunc;
listener.progress = ProgressCallbackFunc;
listener.expire = TokenExpiredCallbackFunc;
listener.retry = RetryCallbackFunc;
listener.retryResume = RetryResumeCallbackFunc;
listener.started = UploadStartedCallbackFunc;
// init with upload address and upload auth
[self.uploader init:listener];
```

Android示例代码

```
VODUploadClient uploader = new VODUploadClientImpl(getApplicationContext());
VODUploadCallback callback = new VODUploadCallback() {
 public void onUploadSucceed(UploadFileInfo info) {
  //上传成功
    }

 public void onUploadFailed(UploadFileInfo info, String code, String message) {
    //上传失败
}

    public void onUploadProgress(UploadFileInfo info, long uploadedSize, long totalSize) {
  //上传进度
}

 public void onUploadTokenExpired() {
  //上传凭证过期，需要调用刷新凭证接口。
}

 public void onUploadRetry(String code, String message) {
  //重试回调
}

 public void onUploadRetryResume() {

}

 public void onUploadStarted(UploadFileInfo uploadFileInfo) {
  OSSLog.logError("onUploadStarted ---------- ");
  //TODO:从业务服务器获取 “上传凭证和地址”
  //设置：uploadAuth, uploadAddress
  uploader.setUploadAuthAndAddress(uploadFileInfo, uploadAuth, uploadAddress);
 }
};

uploader.init(callback);
```

H5 JS示例代码

```
var uploader = new AliyunUpload.Vod({
    partSize: 1048576,//分片大小默认1M，不能小于100K
    parallel: 5,//并行上传分片个数，默认5
    retryCount: 3,//网络原因失败时，重新上传次数，默认为3
    retryDuration: 2,//网络原因失败时，重新上传间隔时间，默认为2秒
    'onUploadstarted': function (uploadInfo) {
    uploader.setUploadAuthAndAddress(
     uploadInfo,
   uploadAuth,
   uploadAddress,
   videoId);
 }
 ..... //其他回调
});
```

更多参考

更加详细的说明请阅读下面的文档：包括如何集成，上传文件队列管理，上传成功回调处理，凭证过期回调处理等可以阅读相关平台的上传SDK文档。

- iOS文件上传文档
- Android文件上传文档
- H5文件上传文档

2.2. 使用STS方式上传

简介

背景

在开发指南-客户端上传里，介绍了客户端提供了两种授权方式上传到点播：

- 使用上传地址和凭证上传
- 使用STS方式上传

本文重点介绍使用STS方式进行上传的实现过程，但点播服务默认推荐使用上传地址和凭证上传，其相比STS方式拥有诸多优势。

STS介绍

STS (Security Token Service) 是阿里云通用的鉴权方式，通过STS方式上传，上传SDK内部会封装所有上传的细节，包括：获取上传地址和上传凭证的API调用，整体流程上用户只需要关注STS的获取、STS过期刷新以及文件上传完成回调即可。具体对STS的介绍文档请参考STS介绍。

基本流程

注意事项

- 步骤2中业务服务APPServer发送请求从阿里云RAM/STS服务获取的STS，为了提高获取效率和避免RAM/STS服务对请求的流控限制需要在业务服务APPServer中对STS进行缓存。
- 具体STS针对点播服务权限的相关配置详见下面的文档：点播服务的STS配置。
- 使用STS上传视频文件到点播时，必须在点播服务上创建好一个模版ID。

实现过程

1. 控制台配置STS点播权限。详见：点播服务的STS配置。
2. 业务服务APPServer从STS服务获取STS。相关示例代码详见：点播服务STS获取示例代码。
3. 客户端从业务服务APPServer获取STS凭证。
4. 客户端添加本地文件，并设置STS，开始上传。

设置STS

STS在初始化上传实例的时候进行设置，下面是iOS、Android、Web平台的相关初始化代码。

iOS示例代码

```
[self.uploader
  init:`STS Key Id`
  accessKeySecret:`STS Key Secret`
  secretToken:STS Secret Token`
  expireTime:`STS Expire Time`
  listener:listener
];
```

Android示例代码

```
VODUploadClient uploader = new VODUploadClientImpl(getApplicationContext());
uploader.init(accessKeyId,
    accessKeySecret,
    secretToken,
    expireTime,
    callback);
```

H5 JS 示例代码

```
var uploader = new AliyunUpload.Vod({
   partSize: 1048576,//分片大小默认1M，不能小于100K
   parallel: 5,//并行上传分片个数，默认5
   retryCount: 3,//网络原因失败时，重新上传次数，默认为3
   retryDuration: 2,//网络原因失败时，重新上传间隔时间，默认为2秒
   'onUploadstarted': function (uploadInfo) {
   uploader.setSTSToken(uploadInfo, accessKeyId, accessKeySecret,secretToken);
 }
 ..... //其他回调
});
```

更多参考

上传文件列表管理，上传成功回调处理，凭证过期回调处理等可以阅读相关平台的上传SDK文档。

- iOS文件上传文档
- Android文件上传文档
- H5文件上传文档

2.3. JavaScript上传SDK

- 安装
- 整体步骤
- 请求上传地址加凭证或STS
  - 请求上传地址加凭证
  - 请求STS
- 初始化
  - 上传地址和凭证方式(推荐使用)
  - STS方式
- 列表管理

- 上传控制
- 断点续传

安装

在页面上引入下面三个JS脚本，见 视频上传SDK下载。

```
<!-- IE需要es6-promise -->
<script src="../lib/es6-promise.min.js"></script>
<script src="../lib/aliyun-oss-sdk6.10.0.min.js"></script>
<script src="../aliyun-vod-upload-sdk1.5.2.min.js"></script>
```

整体步骤

1. 请求上传地址加凭证或STS，相关概念请参见相关文档；
2. 初始化上传实例，实例化上传有两种方式：上传地址加凭证和STS方式；
3. 回调设置，所有的上传状态包括进度，上传成功，上传失败，凭证过期都在这里进行处理；
4. 添加上传文件进入上传列表，目前主要支持视频文件和图片文件的上传；
5. 启动上传；
6. 回调处理；

请求上传地址加凭证或STS

请求上传地址加凭证

上传图片和上传视频获取上传地址和凭证所请求的API是不同的。 **客户端上传视频：**需要请求向AppServer发送请求，AppServer通过OpenApi向阿里云点播服务发送 `CreateUploadVideo` 请求。请求成功将返回上传地址，上传凭证以及 `VideoId` ，AppServer需要将结果返回给客户端。 **客户端上传图片：**需要请求向AppServer发送请求，AppServer通过OpenApi向阿里云点播服务发送 `CreateUploadImage` 请求。请求成功将返回上传地址，上传凭证以及 `ImageURL` ，AppServer需要将结果返回给客户端，然后通过上传地址，上传凭证上传图片，后面流程和上传视频一样，不过不需要处理断点续传，因为图片没有续传功能。

请求STS

通过STS方式，客户端需要向AppServer发送请求，AppServer向阿里云STS服务请求临时STS凭证。请求成功将返回STS凭证，AppServer需要将结果返回给客户端。

初始化

首先，我们需要声明一个 `AliyunUpload.Vod` 初始化回调

```
var uploader = new AliyunUpload.Vod({
    //阿里账号ID，必须有值 ，值的来源https://help.aliyun.com/knowledge_detail/37196.html
    userId:"122"
  //上传到点播的地域， 默认值为'cn-shanghai',//eu-central-1,ap-southeast-1
  region:"",
    //分片大小默认1M，不能小于100K
    partSize: 1048576,
  //并行上传分片个数，默认5
    parallel: 5,
  //网络原因失败时，重新上传次数，默认为3
  retryCount: 3,
  //网络原因失败时，重新上传间隔时间，默认为2秒
  retryDuration: 2,
   // 开始上传
   'onUploadstarted': function (uploadInfo) {
   }
   // 文件上传成功
   'onUploadSucceed': function (uploadInfo) {
   },
   // 文件上传失败
   'onUploadFailed': function (uploadInfo, code, message) {
   },
   // 文件上传进度，单位：字节
   'onUploadProgress': function (uploadInfo, totalSize, loadedPercent) {
   },
   // 上传凭证超时
   'onUploadTokenExpired': function (uploadInfo) {
   },
  //全部文件上传结束
  'onUploadEnd':function(uploadInfo){
    }
});
```

上传地址和凭证方式(推荐使用)

首先请求获取的上传地址和凭证初始化时无需设置，而是在开始上传开始后触发的 `onUploadStarted` 回调中调用 `setUploadAuthAndAddress(uploadFileInfo, uploadAuth, uploadAddress,videoId);` 方法进行设置。当token超时，会触发 `onUploadTokenExpired` 回调，需要调用 `resumeUploadWithAuth(uploadAuth)` 方法，设置新的上传凭证继续上传。

示例代码

```
var uploader = new AliyunUpload.Vod({
    //阿里账号ID，必须有值 ，值的来源https://help.aliyun.com/knowledge_detail/37196.html
    userId:"122"
    //分片大小默认1M，不能小于100K
    partSize: 1048576,
   //并行上传分片个数，默认5
    parallel: 5,
   //网络原因失败时，重新上传次数，默认为3
   retryCount: 3,
   //网络原因失败时，重新上传间隔时间，默认为2秒
   retryDuration: 2,
 //是否上报上传日志到点播，默认为true
 enableUploadProgress: true,
    // 开始上传
    'onUploadstarted': function (uploadInfo) {
      log("onUploadStarted:" + uploadInfo.file.name + ", endpoint:" + uploadInfo.endpoint + ", bucket:"
+ uploadInfo.bucket + ", object:" + uploadInfo.object);
   //上传方式1, 需要根据uploadInfo.videoId是否有值，调用点播的不同接口获取uploadauth和uploadAddress，
如果videoId有值，调用刷新视频上传凭证接口，否则调用创建视频上传凭证接口
   if (uploadInfo.videoId) {
         // 如果 uploadInfo.videoId 存在, 调用 刷新视频上传凭证接口(https://help.aliyun.com/document_det
ail/55408.html)
       }
    else{
         // 如果 uploadInfo.videoId 不存在,调用 获取视频上传地址和凭证接口(https://help.aliyun.com/docume
nt_detail/55407.html)
       }
    //从点播服务获取的uploadAuth、uploadAddress和videoId,设置到SDK里
      uploader.setUploadAuthAndAddress(uploadInfo, uploadAuth, uploadAddress,videoId);
    }
    // 文件上传成功
    'onUploadSucceed': function (uploadInfo) {
      log("onUploadSucceed: " + uploadInfo.file.name + ", endpoint:" + uploadInfo.endpoint + ", bucket:"
+ uploadInfo.bucket + ", object:" + uploadInfo.object);
    },
    // 文件上传失败
    'onUploadFailed': function (uploadInfo, code, message) {
      log("onUploadFailed: file:" + uploadInfo.file.name + ",code:" + code + ", message:" + message);
    },
```

```
  // 文件上传进度，单位：字节
    'onUploadProgress': function (uploadInfo, totalSize, loadedPercent) {
       log("onUploadProgress:file:" + uploadInfo.file.name + ", fileSize:" + totalSize + ", percent:" + Mat
h.ceil(loadedPercent * 100) + "%");
    },
    // 上传凭证超时
    'onUploadTokenExpired': function (uploadInfo) {
       console.log("onUploadTokenExpired");
    //实现时，根据uploadInfo.videoId调用刷新视频上传凭证接口重新获取UploadAuth
    //https://help.aliyun.com/document_detail/55408.html
    //从点播服务刷新的uploadAuth,设置到SDK里

    uploader.resumeUploadWithAuth(uploadAuth);
    },
  //全部文件上传结束
  'onUploadEnd':function(uploadInfo){
       console.log("onUploadEnd: uploaded all the files");
    }
});
```

STS方式

首先请求获取的STS初始化时无需设置，而是在开始上传开始后触发的 `onUploadStarted` 回调中调用 `setSTSToken(uploadInfo, accessKeyId, accessKeySecret, secretToken);` 方法进行设置。 当token超时，会触发 `onUploadTokenExpired` 回调，需要调用 `resumeUploadWithSTSToken(accessKeyId, accessKeySecret, secretToken)` 方法，设置新的Token继续上传。

```
var uploader = new AliyunUpload.Vod({
      //分片大小默认1M，不能小于100K
     partSize: 1048576,
   //并行上传分片个数，默认5
     parallel: 5,
   //网络原因失败时，重新上传次数，默认为3
   retryCount: 3,
   //网络原因失败时，重新上传间隔时间，默认为2秒
   retryDuration: 2,
 //是否上报上传日志到点播，默认为true
 enableUploadProgress: true,
    // 开始上传
    'onUploadstarted': function (uploadInfo) {
     log("onUploadStarted:" + uploadInfo.file.name + ", endpoint:" + uploadInfo.endpoint + ", bucket:"
+ uploadInfo.bucket + ", object:" + uploadInfo.object);
```

```
    //获取STS Token,设置到SDK
      uploader.setSTSToken(uploadInfo, accessKeyId, accessKeySecret, secretToken);
    }
    // 文件上传成功
    'onUploadSucceed': function (uploadInfo) {
     log("onUploadSucceed: " + uploadInfo.file.name + ", endpoint:" + uploadInfo.endpoint + ", bucket:"
+ uploadInfo.bucket + ", object:" + uploadInfo.object);
    },
    // 文件上传失败
    'onUploadFailed': function (uploadInfo, code, message) {
     log("onUploadFailed: file:" + uploadInfo.file.name + ",code:" + code + ", message:" + message);
    },
    // 文件上传进度，单位：字节
    'onUploadProgress': function (uploadInfo, totalSize, loadedPercent) {
       log("onUploadProgress:file:" + uploadInfo.file.name + ", fileSize:" + totalSize + ", percent:" + Mat
h.ceil(loadedPercent * 100) + "%");
    },
    // 上传凭证超时
    'onUploadTokenExpired': function (uploadInfo) {
       console.log("onUploadTokenExpired");
       //重新获取STS token，恢复上传
    uploader.resumeUploadWithSTSToken(accessKeyId, accessKeySecret, secretToken);
    },
   //全部文件上传结束
   'onUploadEnd':function(uploadInfo){
        console.log("onUploadEnd: uploaded all the files");
     }
 });
```

列表管理

添加上传文件

> 注意 支持的文件大小<=10G。

需要使用标准的input方式让用户选择文件。

```javascript

<form action="">
 <input type="file" name="file" id="files" multiple/>
</form>
userData = '';
document.getElementById("files")
 .addEventListener('change', function (event) {
  for(var i=0; i<event.target.files.length; i++) {
    // 逻辑代码
  }
 });

```

获取到用户选择的文件后，添加到上传列表中。

```javascript

uploader.addFile(event.target.files[i], null, null, null, paramData);

```

STS方式上传时，可以选择是否启用水印和优先级，paramData是一个json对象字符串。第一级的Vod是必须的，Vod下面添加属性，paramData支持的属性参考点播服务的createUploadVideo获取视频上传地址和凭证，接口示例如下：

```javascript

var paramData = '{"Vod":{"Title":"test","CateId":"234"}}';

```

paramData只有在STS的上传方式是需要在SDK指定， 如果是上传凭证的方式，则在获取上传凭证createUploadVideo接口的参数里指定，无需在SDK里指定paramData参数

删除上传文件

index，对应listFiles接口返回列表中元素的索引。

```
uploader.deleteFile(index);
```

取消单个文件上传

```
uploader.cancelFile(index);
```

恢复单个文件上传

```
uploader.resumeFile(index);
```

获取上传文件列表

```
uploader.listFiles();
var list = uploader.listFiles();
for (var i=0; i<list.length; i++) {
   log("file:" + list[i].file.name + ", status:" +      list[i].state + ", endpoint:" + list[i].endpoint + ", bucket:
" + list[i].bucket + ", object:" + list[i].object);
}
```

清理上传文件列表

```
uploader.cleanList();
```

上传控制

开始上传

```
uploader.startUpload();
```

停止上传

```
uploader.stopUpload();
```

上传凭证失效后恢复上传

```
uploader.resumeUploadWithAuth(uploadAuth);
```

设置上传地址和上传凭证

设置上传地址和上传凭证方法在onUploadstarted回调里调用，此回调的参数包含uploadInfo的值。

```
uploader.setUploadAuthAndAddress(uploadInfo,uploadAuth, uploadAddress, videoId);
```

> **说明** 获取上传凭证和上传地址，请查看相关文档 。

设置STS Token

设置STS Token方法在onUploadstarted回调里调用，此回调的参数包含uploadInfo的值。

```
uploader.setSTSToken(uploadInfo, accessKeyId, accessKeySecret,secretToken);
```

上传STS Token失效后恢复上传

```
uploader.resumeUploadWithSTSToken(accessKeyId, accessKeySecret, secretToken, expireTime);
```

断点续传

用户在上传过程中，由于某种原因没有上传完成，在下次选择同一个文件上传时， SDK会从上次完成的位置继续上传，并且在onUploadstarted回调中，获取上传凭证， 如果使用的时上传方式1（通过uploadAuth和uploadAddress）上传时，用户需要根据回调返回的videoId的值，调用点播的不同接口， 如下：

- 获取视频上传地址和凭证
- 刷新视频上传凭证
- STS SDK参考

```
onUploadstarted': function (uploadInfo) {
if (上传方式1) {
 if(!uploadInfo.videoId)//这个文件没有上传异常
 {
  //实际环境中调用调用点播的获取上传凭证接口
  uploader.setUploadAuthAndAddress(uploadInfo, uploadAuth, uploadAddress,videoId);
 }
 else//如果videoId有值，根据videoId刷新上传凭证
 {
  //实际环境中调用点播的刷新上传凭证接口，获取凭证
  uploader.setUploadAuthAndAddress(uploadInfo, uploadAuth, uploadAddress);
 }
}
else(上传方式2)
{
  //实际环境中调用获取STS接口，获取STS的值
  uploader.setSTSToken(uploadInfo, accessKeyId, accessKeySecret,secretToken);
}
```

获取断点信息

参数为file对象

```
```

uploader.getCheckpoint(file);
```
```

2.4. Android上传SDK

2.4.1. 工程配置

系统支持

上传SDK自身没有系统版本限制，建议2.3及以上Android系统版本使用。

运行环境

推荐开发者使用Android Studio 作为自己的开发工具，本开发文档也是基于Android Studio开发环境下进行编写的。

如何导入

方式一 Maven依赖

通过maven的形式导入，分别引入以下两个SDK。

- 在Maven项目中加入依赖项（推荐方式）

```
dependencies {
 compile 'com.aliyun.video.android:upload:1.5.5'
}
```

另外您需要增加阿里云maven仓库地址，在您的根目录下的build.gradle文件增加

```
allprojects {
   repositories {
      maven { url "https://maven.aliyun.com/nexus/content/repositories/releases" }
   }
}
```

- OSS Android SDK，具体SDK说明和下载地址请查看:OSS产品的 Android-SDK 安装。

方式二 本地jar包依赖

通过jar包的形式导入，分别引入以下两个SDK。

- VODUpload Android SDK，见视频上传SDK下载。 解压后在libs目录下得到jar包，目前包括aliyun-vod-upload-android-sdk-xxx.jar、gson-xxx.jar、jsr305-xxx.jar将以上3个jar包导入工程的libs目录。
- OSS Android SDK，具体SDK说明和下载地址请查看:OSS产品的 Android-SDK 安装。

2.4.2. 文件上传

整体步骤

1. 请求上传地址加凭证或STS，相关概念请参见相关文档；
2. 初始化上传实例，实例化上传有两种方式：上传地址加凭证和STS方式；
3. 回调设置，所有的上传状态包括进度，上传成功，上传失败，凭证过期都在这里进行处理；
4. 添加上传文件进入上传列表，目前主要支持视频文件和图片文件的上传；
5. 启动上传；
6. 回调处理；

1. 请求上传地址加凭证或STS

1.1 请求上传地址加凭证

上传图片和上传视频获取上传地址和凭证所请求的API是不同的。**客户端上传视频：**需要请求向AppServer发送请求，AppServer通过OpenApi向阿里云点播服务发送 `CreateUploadVideo` 请求。请求成功将返回上传地址，上传凭证以及 `VideoId` ，AppServer需要将结果返回给客户端。**客户端上传图片：**需要请求向AppServer发送请求，AppServer通过OpenApi向阿里云点播服务发送 `CreateUploadImage` 请求。请求成功将返回上传地址，上传凭证以及 `ImageURL` ，AppServer需要将结果返回给客户端。

1.2 请求STS

通过STS方式，客户端需要向AppServer发送请求，AppServer向阿里云STS服务请求临时STS凭证。请求成功将返回STS凭证，AppServer需要将结果返回给客户端。

2. 初始化

首先，我们需要声明一个 `VODUploadClient` 初始化回调

```
uploader = new VODUploadClientImpl(getApplicationContext());
```

2.1 上传地址和凭证方式

上传地址和凭证方式调用 `init` 方法初始化。第一步请求获取的上传地址和凭证初始化时无需设置，而是在开始上传开始后触发的 `onUploadStarted` 回调中调用 `setUploadAuthAndAddress(uploadFileInfo, uploadAuth, uploadAddress)` 方法进行设置。当token超时，会触发 `onUploadTokenExpired` 回调，需要调用 `resumeWithAuth(uploadAuth)` 方法，设置新的上传凭证继续上传。

示例代码

```
// create VODUploadClientfinal VODUploadClient uploader = new VODUploadClientImpl(getApplication Context());// setup callbackVODUploadCallback callback = new VODUploadCallback() { public voi d onUploadSucceed(UploadFileInfo info) { OSSLog.logDebug("onsucceed ------------------" + inf o.getFilePath()); } public void onUploadFailed(UploadFileInfo info, String code, String mess age) { OSSLog.logError("onfailed ------------------ " + info.getFilePath() + " " + code + " " + mess age); } public void onUploadProgress(UploadFileInfo info, long uploadedSize, long totalSiz e) { OSSLog.logDebug("onProgress ------------------ " + info.getFilePath() + " " + uploadedSize + " " + totalSize); } } } public void onUploadTokenExpired() { OSS Log.logError("onExpired ------------- "); // 重新刷新上传凭证:RefreshUploadVideo u ploadAuth = "此处需要设置重新刷新凭证之后的值"; uploader.resumeWithAuth(uploadAuth); } public void onUploadRetry(String code, String message) { OSSLog.logError("onUploadR etry ------------- "); } public void onUploadRetryResume() { OSSLog.logError("onUpl oadRetryResume ------------- "); } public void onUploadStarted(UploadFileInfo uploadFileInf o) { OSSLog.logError("onUploadStarted ------------- "); uploader.setUploadAuthAndAd dress(uploadFileInfo, uploadAuth, uploadAddress); } };//上传初始化uploader.init(callback);
```

2.2 STS方式

STS方式调用init(accessKeyId, accessKeySecret, secretToken, expireTime, callback);方法初始化，初始化参数即是第一步请求获取的临时STS凭证。当token过期时，触发OnUploadTokenExpired回调，需要调用resumeWithToken(accessKeyId, accessKeySecret, secretToken, expireTime);方法，设置新的STS继续上传。

```
// create VODUploadClient objectuploader = new VODUploadClientImpl(getApplicationContext());// set up callback// setup callbackVODUploadCallback callback = new VODUploadCallback() { public voi d onUploadSucceed(UploadFileInfo info) { OSSLog.logDebug("onsucceed ------------------" + inf o.getFilePath()); } public void onUploadFailed(UploadFileInfo info, String code, String mess age) { OSSLog.logError("onfailed ------------------ " + info.getFilePath() + " " + code + " " + mess age); } public void onUploadProgress(UploadFileInfo info, long uploadedSize, long totalSiz e) { OSSLog.logDebug("onProgress ------------------ " + info.getFilePath() + " " + uploadedSize + " " + totalSize); } } } public void onUploadTokenExpired() { OSS Log.logError("onExpired ------------- "); // 重新获取STS之后调用resumeWithToken u ploader.resumeWithToken(accessKeyId, accessKeySecret, secretToken, expireTime); } pub lic void onUploadRetry(String code, String message) { OSSLog.logError("onUploadRetry -------- ------ "); } public void onUploadRetryResume() { OSSLog.logError("onUploadRetryRe sume ------------- "); } public void onUploadStarted(UploadFileInfo uploadFileInfo) { OSSLog.logError("onUploadStarted ------------- "); } };// 初始化，临时账号过期时，在onUploadT okenExpired事件中，用resumeWithToken更新临时账号，上传默认支持断点续传uploader.init(accessKeyId, ac cessKeySecret, secretToken, expireTime, callback);
```

3. 回调设置

从初始化代码片段中可以看到，两种方法初始化都需要设置VODUploadCallback对象，该对象是上传状态的回调类，需要设置下列回调方法：

```
/** 上传完成回调 @param info 上传文件信息 */void onUploadSucceed(UploadFileInfo info);/** 上传失败回调 @param info 上传文件信息 @param code 错误码 @param message 错误描述 */ void onUploadFailed(UploadFileInfo info, String code, String message);/** 上传进度回调 @param fileInfo 上传文件信息 @param uploadedSize 已上传大小 @param totalSize 总大小 */ void onUploadProgress(UploadFileInfo info, long uploadedSize, long totalSize);/** token过期回调 上传地址和凭证方式上传需要调用resumeWithAuth方法继续上传 STS方式上传需要调用resumeWithToken方法继续上传 */ void onUploadTokenExpired();/** 上传开始重试回调 */ void onUploadRetry(String code, String message);/** 上传结束重试，继续上传回调 */ void onUploadRetryResume ();/** 开始上传回调 上传地址和凭证方式上传需要调用setUploadAuthAndAddress:uploadAuth:uploadAddress:方法设置上传地址和凭证 @param fileInfo 上传文件信息 */  void onUploadStarted(UploadFileInfo uploadFileInfo);
```

4. 添加文件到上传列表

4.1 添加文件

添加视频

```
String filePath = "文件地址";VodInfo vodInfo = new VodInfo();vodInfo.setTitle("标题" + index);vodInfo.setDesc("描述." + index);vodInfo.cateId (19);vodInfo.tags("sports");uploader.addFile(filePath,vodInfo);
```

添加图片``javaString filePath = “图片文件地址”;VodInfo vodInfo = new VodInfo();vodInfo.setTitle(“标题” + index);vodInfo.setDesc(“描述.” + index);vodInfo.cateId (19);vodInfo.tags(“sports”);uploader.addFile(filePath,vodInfo);

> 注意：支持的文件大小<=4G。VodInfo具体结构如下：

//标题String title;//标签List tags;//描述String desc;//分类idInteger cateId;//封面url （完整的URL https://)String coverUrl;

添加文件后，SDK会将待上传文件封装为`UploadFileInfo`对象，具体结构如下：```java//文件本地路径String filePath;//endpointString endpoint;//bucketString bucket;//objectString object;//VodInfoVodInfo vodInfo;

4.2 管理上传队列

VODUploadClient支持添加多个文件顺序上传，提供了以下方法管理上传队列。从队列中删除上传文件。如果待删除的文件正在上传中，则取消上传并自动上传下一个文件：

```
void deleteFile(int index)
```

清空上传队列，如果有文件在上传，则取消上传：

```
void clearFiles()
```

获取上传文件队列：

```
List<UploadFileInfo> listFiles()
```

将文件标记为取消，文件任保留在上传列表中。如果待取消的文件正在上传中，则取消上传并自动上传下一个文件：

```
cancelFile(int index)
```

恢复已取消的上传文件，并自动开始上传：

```
resumeFile(int index)
```

> 说明 尽管VODUploadClient支持多文件上传，如果使用上传凭证和地址方式上传，每个文件还是需要单独设置。基于多文件上传代码复杂度的考虑，建议只添加单文件上传。

5. 上传控制

开始上传：

```
void start();
```

> 说明 该方法调用后，会触发onUploadStarted回调。注意，如果通过上传地址和凭证方式上传，需要在该回调方法中设置上传地址和凭证。代码如下：
>
> ```
> void setUploadAuthAndAddress(UploadFileInfo uploadFileInfo, String uploadAuth, String uploadAddress)
> ```

停止上传，如果有文件正在上传中，则取消上传：

```
void stop();
```

> 说明 stop后恢复上传需要调用resumeFile恢复待上传文件，或者清空队列后重新添加文件上传。

暂停上传：

```
void pause();
```

恢复上传：

```
void resume();
```

5. 回调处理

5.1 上传进度

每上传一个分片，会触发onUploadProgress回调，回调参数包括已上传文件大小uploadedSize和总文件大小totalSize。

5.2 上传成功

上传成功时，会触发onUploadSucceed回调。回调包含上传结果videoId，imageUrl属性

> 说明 视频上传成功后会返回videoId作为视频id，拿到videoId之后需要获取播放地址进行播放。相关文档可以参考获取播放地址播放。图片上完成后会返回imageUrl，开启URL鉴权后imageUrl会有过期时间。相关配置可以参考URL鉴权。

5.3 上传失败

上传失败时，会触发onUploadFailed(String code, String message)回调。在该回调方法中，我们可以通过code和message查看具体原因，页面上给予用户提示。错误码参见：点播错误码和oss错误码。

5.4 凭证过期处理

上传凭证和STS过期，会触发onSTSTokenExpried回调。在该回调方法中，我们可以向AppServer重新请求新的STS凭证，并调用以下方法继续上传：

```
refreshSTSToken(accessKeyId,accessKeySecret,securityToken,expriedTime);
```

5.5 超时处理

上传超时，会触发uploadRetry回调并自动重试。在该回调方法中，我们可以在页面上给予用户提示或者调用cancel方法停止上传。此外，可以设置maxRetryCount属性，指定最大重试次数。超时重试发现可以继续上传时，会触发uploadRetryResume回调并恢复上传。

高级设置

VODUploadClient支持以下高级设置：

```
/** 上传文件到服务端是否转码,默认值YES */void setTranscodeMode(boolean bool);/** 分片大小，默认值1024 * 1024*/void setPartSize(long partSize);/*** 指定视频文件的存储区域*/void setStorageLocation(String storageLocation);/*** 设置转码模版组Id*/void setTemplateGroupId(String templateGroupId);
```

2.4.3. 短视频上传

简介

短视频上传 `VODSVideoUploadClient` 主要针对需要同时上传封面图片+视频的场景，旨在简化用户的接口调用，内部是在 `VODUploadClient` 的基础上封装实现的。短视频上传只支持STS方式上传，要想通过上传地址加凭证方式上传短视频请使用 `VODUploadClient` 分别上传封面图片+视频。

整体步骤

1. 请求STS，相关概念请参见相关文档；
2. 初始化上传实例；
3. 回调设置，所有的上传状态包括进度，上传成功，上传失败，凭证过期都在这里进行处理；
4. 启动上传；
5. 回调处理；

1. 请求STS

通过STS方式，客户端需要向AppServer发送请求，AppServer向阿里云STS服务请求临时STS凭证。请求成功将返回STS凭证，AppServer需要将结果返回给客户端。

2. 初始化

首先，我们需要声明一个 `VODSVideoUploadClient` 属性，注意不能是局部变量。

```
//1.初始化短视频上传对象
VODSVideoUploadClient vodsVideoUploadClient = new VODSVideoUploadClientImpl(this.getApplication
Context());
vodsVideoUploadClient.init();
```

```
//2.构建上传参数
//参数请确保存在，如不存在SDK内部将会直接将错误throw Exception
// 文件路径保证存在之外因为Android 6.0之后需要动态获取权限，请开发者自行实现获取"文件读写权限".
VodHttpClientConfig vodHttpClientConfig = new VodHttpClientConfig.Builder()
        .setMaxRetryCount(2)//重试次数
        .setConnectionTimeout(15 * 1000)//连接超时
        .setSocketTimeout(15 * 1000)//socket超时
        .build();
 //构建短视频VideoInfo,常见的描述，标题，详情都可以设置
 SvideoInfo svideoInfo = new SvideoInfo();
 svideoInfo.setTitle(new File(videoPath).getName());//标题
 svideoInfo.setDesc("");//文件详情
 svideoInfo.setCateId(1);//分类id
 //构建点播上传参数(重要)
 VodSessionCreateInfo vodSessionCreateInfo =new    VodSessionCreateInfo.Builder()
.setImagePath(imagePath)//图片地址
.setVideoPath(videoPath)//视频地址
.setAccessKeyId(accessKeyId)//临时accessKeyId
.setAccessKeySecret(accessKeySecret)//临时accessKeySecret
.setSecurityToken(securityToken)//securityToken
.setExpriedTime(expriedTime)//STStoken过期时间
.setRequestID(requestID)//requestID，开发者可以传将获取STS返回的requestID设置也可以不设.
.setIsTranscode(true)//是否转码.如开启转码请AppSever务必监听服务端转码成功的通知
.setSvideoInfo(svideoInfo)//短视频视频信息
.setVodHttpClientConfig(vodHttpClientConfig)//网络参数
                .build();
```

3. 启动上传并设置上传的回调

从以下上传代码片段中可以看到，启动上传需要设置上传回调，需要实现 `VODSVideoUploadCallback` 回调：

```
vodsVideoUploadClient.uploadWithVideoAndImg(vodSessionCreateInfo, new VODSVideoUploadCallbac
k() {
    @Override
    public void onUploadSucceed(String videoId, String imageUrl) {
    //上传成功返回视频ID和图片URL.
    Log.d(TAG,"onUploadSucceed"+ "videoId:"+ videoId + "imageUrl" + imageUrl);
            }
            @Override
            public void onUploadFailed(String code, String message) {
               //上传失败返回错误码和message.错误码有详细的错误信息请开发者仔细阅读
               Log.d(TAG,"onUploadFailed" + "code" + code + "message" + message);
            }
            @Override
            public void onUploadProgress(long uploadedSize, long totalSize) {
               //上传的进度回调,非UI线程
               Log.d(TAG,"onUploadProgress" + uploadedSize * 100 / totalSize);
               progress = uploadedSize * 100 / totalSize;
               handler.sendEmptyMessage(0);
            }
            @Override
            public void onSTSTokenExpried() {
               Log.d(TAG,"onSTSTokenExpried");
               //STS token过期之后刷新STStoken，如正在上传将会断点续传
               vodsVideoUploadClient.refreshSTSToken(accessKeyId,accessKeySecret,securityToken,ex
priedTime);
            }
            @Override
            public void onUploadRetry(String code, String message) {
               //上传重试的提醒
               Log.d(TAG,"onUploadRetry" + "code" + code + "message" + message);
            }
            @Override
            public void onUploadRetryResume() {
               //上传重试成功的回调.告知用户重试成功
               Log.d(TAG,"onUploadRetryResume");
            }
         });
```

4. 暂停、恢复 、取消上传

常见的比如退后台的暂停，进入前台继续上传的逻辑Demo中有体现。开发者可以自由的组装暂停上传，恢复上传和取消上传的逻辑。

暂停上传：

```
//需要保证跟resume成对出现
vodsVideoUploadClient.pause();
```

恢复上传：

```
//需要保证跟pause成对出现
vodsVideoUploadClient.resume();
```

取消上传：

```
//取消上传之后就认为上传流程结束了,不能再调用resume恢复
vodsVideoUploadClient.cancel();
```

5. 回调处理

5.1 上传进度

每上传一个分片，会触发 `onUploadProgress` 回调，回调参数包括已上传文件大小 `uploadedSize` 和总文件大小 `totalSize` 。

5.2 上传成功

上传成功时，会触发 `onUploadSucceed` 回调。回调包含上传结果 `videoId` ， `imageUrl` 属性

> **说明** 视频上传成功后会返回videoId作为视频id，拿到videoId之后需要获取播放地址进行播放。相关文档可以参考**获取播放地址播放**。 图片上完成后会返回imageUrl，开启URL鉴权后imageUrl会有过期时间。相关配置可以参考**URL鉴权**。

5.3 上传失败

上传失败时，会触发 `onUploadFailed(String code, String message)` 回调。在该回调方法中，我们可以通过 `code` 和 `message` 查看具体原因，页面上给予用户提示。 错误码参见：点播错误码和oss错误码。

5.4 凭证过期处理

上传凭证和STS过期，会触发 `onSTSTokenExpried` 回调。在该回调方法中，我们可以向AppServer重新请求新的STS凭证，并调用以下方法继续上传：

```
refreshSTSToken(accessKeyId,accessKeySecret,securityToken,expriedTime);
```

5.5 超时处理

上传超时，会触发 uploadRetry 回调并自动重试。在该回调方法中，我们可以在页面上给予用户提示或者调用 cancel 方法停止上传。此外，可以设置 maxRetryCount 属性，指定最大重试次数。 超时重试发现可以继续上传时，会触发 uploadRetryResume 回调并恢复上传。

2.5. iOS上传SDK

2.5.1. 工程配置

系统版本

上传SDK支持iOS7及以上系统

SDK集成

pod方式集成

1. 在Podfile文件中添加VODUpload库依赖

```
pod 'VODUpload'
```

2. 更新pod repo

```
pod repo update
```

3. 安装VODUpload库

```
pod install
```

手动方式集成

1. 下载 VODUpload iOS SDK,下载地址请参考上传SDK下载。
2. 下载 OSS iOS SDK，下载地址请参考OSS iOS SDK。
3. 在Xcode中，直接把 VODUpload.framework 和 AliyunOSSiOS.framework 拖入项目Target下即可，在弹出框勾选Copy items if needed。
4. 添加以下系统依赖库： AVFoundation.framework ，CoreMedia.framework ，SystemConfiguration.framework ，MobileCoreServices.framework ，libresolv.9.tbd 。

项目配置

SDK集成后，打开项目工程并修改以下配置： 配置Build Setting — Linking — Other Linker Flags，添加-ObjC。

2.5.2. 文件上传

简介

点播文件上传 VODUploadClient ，支持上传地址加凭证或STS方式上传，推荐使用上传地址加凭证方式上传。

整体步骤

1. 请求上传地址加凭证或STS，相关概念请参见相关文档。
2. 初始化上传实例，实例化上传有两种方式：上传地址加凭证和STS方式。
3. 回调设置，所有的上传状态包括进度，上传成功，上传失败，凭证过期都在这里进行处理。
4. 添加上传文件进入上传列表，目前主要支持视频文件和图片文件的上传。
5. 启动上传。
6. 回调处理。

1. 请求上传地址加凭证或STS

1.1 请求上传地址加凭证

上传图片和上传视频获取上传地址和凭证所请求的API是不同的。

客户端上传视频：需要请求向AppServer发送请求，AppServer通过OpenApi向阿里云点播服务发送 `CreateUploadVideo` 请求。请求成功将返回上传地址，上传凭证以及 `VideoId` ，AppServer需要将结果返回给客户端。

客户端上传图片：需要请求向AppServer发送请求，AppServer通过OpenApi向阿里云点播服务发送 `CreateUploadImage` 请求。请求成功将返回上传地址，上传凭证以及 `ImageURL` ，AppServer需要将结果返回给客户端。

1.2 请求STS

通过STS方式，客户端需要向AppServer发送请求，AppServer向阿里云STS服务请求临时STS凭证。请求成功将返回STS凭证，AppServer需要将结果返回给客户端。

2. 初始化

首先，声明 `VODUploadClient` 属性，注意不能是局部变量。

```
@property (nonatomic, strong) VODUploadClient *uploader;
```

2.1 上传地址和凭证方式

上传地址和凭证方式调用 `init:` 方法初始化。

第一步请求获取的上传地址和凭证初始化时无需设置，而是在开始上传开始后触发的 `OnUploadStartedListener` 回调中调用 `setUploadAuthAndAddress: uploadAuth:uploadAddress:` 方法进行设置。

当token超时，会触发 `OnUploadTokenExpiredListener` 回调，需要调用 `resumeWithAuth:` 方法，设置新的上传凭证继续上传。

```
// create VODUploadClient object
self.uploader = [VODUploadClient new];
// weakself
__weak typeof(self) weakSelf = self;
// setup callback
OnUploadFinishedListener FinishCallbackFunc = ^(UploadFileInfo* fileInfo, VodUploadResult* result){
```

```
    NSLog(@"upload finished callback videoid:%@, imageurl:%@", result.videoId, result.imageUrl);
};
OnUploadFailedListener FailedCallbackFunc = ^(UploadFileInfo* fileInfo, NSString *code, NSString* mes
sage){
    NSLog(@"upload failed callback code = %@, error message = %@", code, message);
};
OnUploadProgressListener ProgressCallbackFunc = ^(UploadFileInfo* fileInfo, long uploadedSize, long
totalSize) {
    NSLog(@"upload progress callback uploadedSize : %li, totalSize : %li", uploadedSize, totalSize);
};
OnUploadTokenExpiredListener TokenExpiredCallbackFunc = ^{
    NSLog(@"upload token expired callback.");
 // token过期，设置新的上传凭证，继续上传
    [weakSelf.uploader resumeWithAuth:`new upload auth`];
};
OnUploadRertyListener RetryCallbackFunc = ^{
    NSLog(@"upload retry begin callback.");
};
OnUploadRertyResumeListener RetryResumeCallbackFunc = ^{
    NSLog(@"upload retry end callback.");
};
OnUploadStartedListener UploadStartedCallbackFunc = ^(UploadFileInfo* fileInfo) {
    NSLog(@"upload upload started callback.");
    // 设置上传地址 和 上传凭证
    [weakSelf.uploader setUploadAuthAndAddress:fileInfo uploadAuth:`upload auth` uploadAddress:`
upload address`];
};

VODUploadListener *listener = [[VODUploadListener alloc] init];
listener.finish = FinishCallbackFunc;
listener.failure = FailedCallbackFunc;
listener.progress = ProgressCallbackFunc;
listener.expire = TokenExpiredCallbackFunc;
listener.retry = RetryCallbackFunc;
listener.retryResume = RetryResumeCallbackFunc;
listener.started = UploadStartedCallbackFunc;
// init with upload address and upload auth
[self.uploader init:listener];
```

2.2 STS方式

STS方式调用 `init: accessKeySecret: secretToken: expireTime: listener:` 方法初始化，初始化参数即是第一步请求获取的临时STS凭证。

当token过期时，触发 `OnUploadTokenExpiredListener` 回调，需要调用 `resumeWithToken: accessKeySecret: secretToken: expireTime:` 方法，设置新的STS继续上传。

```
// create VODUploadClient object
self.uploader = [VODUploadClient new];
// weakself
__weak typeof(self) weakSelf = self;
// setup callback
OnUploadFinishedListener FinishCallbackFunc = ^(UploadFileInfo* fileInfo, VodUploadResult* result){
    NSLog(@"upload finished callback videoid:%@, imageurl:%@", result.videoId, result.imageUrl);
};
OnUploadFailedListener FailedCallbackFunc = ^(UploadFileInfo* fileInfo, NSString *code, NSString* mes
sage){
    NSLog(@"upload failed callback code = %@, error message = %@", code, message);
};
OnUploadProgressListener ProgressCallbackFunc = ^(UploadFileInfo* fileInfo, long uploadedSize, long
totalSize) {
    NSLog(@"upload progress callback uploadedSize：%li, totalSize：%li", uploadedSize, totalSize);
};
OnUploadTokenExpiredListener TokenExpiredCallbackFunc = ^{
    NSLog(@"upload token expired callback.");
 // token过期，设置新的STS，继续上传
    [weakSelf.uploader resumeWithToken:`STS Key Id` accessKeySecret:`STS Key Secret` secretToken:
`STS Secret Token` expireTime:`STS Expire Time`];
};
OnUploadRertyListener RetryCallbackFunc = ^{
    NSLog(@"upload retry begin callback.");
};
OnUploadRertyResumeListener RetryResumeCallbackFunc = ^{
    NSLog(@"upload retry end callback.");
};
OnUploadStartedListener UploadStartedCallbackFunc = ^(UploadFileInfo* fileInfo) {
    NSLog(@"upload upload started callback.");
};
// init
VODUploadListener *listener = [[VODUploadListener alloc] init];
listener.finish = FinishCallbackFunc;
listener.failure = FailedCallbackFunc;
listener.progress = ProgressCallbackFunc;
```

```
listener.expire = TokenExpiredCallbackFunc;
listener.retry = RetryCallbackFunc;
listener.retryResume = RetryResumeCallbackFunc;
listener.started = UploadStartedCallbackFunc;
// init with upload address and upload auth
[self.uploader init:`STS Key Id` accessKeySecret:`STS Key Secret` secretToken:STS Secret Token` expi
reTime:`STS Expire Time` listener:listener];
```

3. 回调设置

从初始化代码片段中可以看到，两种方法初始化都需要设置 `VODUploadListener` 对象，该对象是上传状态的回调类，需要设置下列回调方法：

```
/**
 上传完成回调

 @param fileInfo 上传文件信息
 @param result 上传结果信息
 */
typedef void (^OnUploadFinishedListener) (UploadFileInfo* fileInfo, VodUploadResult* result);

/**
 上传失败回调

 @param fileInfo 上传文件信息
 @param code 错误码
 @param message 错误描述
 */
typedef void (^OnUploadFailedListener) (UploadFileInfo* fileInfo, NSString *code, NSString * message)
;

/**
 上传进度回调

 @param fileInfo 上传文件信息
 @param uploadedSize 已上传大小
 @param totalSize 总大小
 */
typedef void (^OnUploadProgressListener) (UploadFileInfo* fileInfo, long uploadedSize, long totalSize
);

/**
```

```
/**
 token过期回调
 上传地址和凭证方式上传需要调用resumeWithAuth:方法继续上传
 STS方式上传需要调用resumeWithToken:accessKeySecret:secretToken:expireTime:方法继续上传
 */
typedef void (^OnUploadTokenExpiredListener) ();

/**
 上传开始重试回调
 */
typedef void (^OnUploadRertyListener) ();

/**
 上传结束重试，继续上传回调
 */
typedef void (^OnUploadRertyResumeListener) ();

/**
 开始上传回调
 上传地址和凭证方式上传需要调用setUploadAuthAndAddress:uploadAuth:uploadAddress:方法设置上传地址和凭证
 @param fileInfo 上传文件信息
 */
typedef void (^OnUploadStartedListener) (UploadFileInfo* fileInfo);
```

4. 添加文件到上传列表

4.1 添加文件

添加视频

```
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"test" ofType:@"mp4"];
VodInfo *vodInfo = [[VodInfo alloc] init];
vodInfo.title = @"title";
vodInfo.desc =@"desc";
vodInfo.cateId = @(19);
vodInfo.tags = @"sports";
[self.uploader addFile:filePath vodInfo:vodInfo];
```

添加图片

```
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"test" ofType:@"jpg"];
VodInfo *imageInfo = [[VodInfo alloc] init];
imageInfo.title = @"title";
imageInfo.desc =@"desc";
imageInfo.cateId = @(19);
imageInfo.tags = @"sports";
[self.uploader addFile:filePath vodInfo:imageInfo];
```

> 注意 支持的文件大小<=4G。

VodInfo具体结构如下：

```
//标题
@property (nonatomic, copy) NSString* title;
//标签
@property (nonatomic, copy) NSString* tags;
//描述
@property (nonatomic, copy) NSString* desc;
//分类id
@property (nonatomic, strong) NSNumber* cateId;
//封面url（完整的URL https://）
@property (nonatomic, copy) NSString* coverUrl;
```

添加文件后，SDK会将待上传文件封装为 `UploadFileInfo` 对象，具体结构如下：

```
//文件本地路径
@property (nonatomic, copy) NSString* filePath;
//endpoint
@property (nonatomic, copy) NSString* endpoint;
//bucket
@property (nonatomic, copy) NSString* bucket;
//object
@property (nonatomic, copy) NSString* object;
//VodInfo
@property (nonatomic, strong) VodInfo* vodInfo;
```

4.2 管理上传队列

VODUploadClient 支持添加多个文件顺序上传，提供了以下方法管理上传队列。 从队列中删除上传文件。如果待删除的文件正在上传中，则取消上传并自动上传下一个文件：

```
- (BOOL)deleteFile:(int) index;
```

清空上传队列，如果有文件在上传，则取消上传：

```
- (BOOL)clearFiles;
```

获取上传文件队列：

```
- (NSMutableArray<UploadFileInfo *> *)listFiles;
```

将文件标记为取消，文件任保留在上传列表中。如果待取消的文件正在上传中，则取消上传并自动上传下一个文件：

```
- (BOOL)cancelFile:(int)index;
```

恢复已取消的上传文件，并自动开始上传：

```
- (BOOL)resumeFile:(int)index;
```

> 说明 尽管 VODUploadClient 支持多文件上传，如果使用上传凭证和地址方式上传，每个文件还是需要单独设置。基于多文件上传代码复杂度的考虑，建议只添加单文件上传。

5. 上传控制

开始上传：

```
[self.uploader start];
```

该方法调用后，会触发 OnUploadStartedListener 回调。

> 注意 如果通过上传地址和凭证方式上传，需要在该回调方法中设置上传地址和凭证。

```
[weakSelf.uploader setUploadAuthAndAddress:fileInfo uploadAuth:weakSelf.uploadAuth uploadAddr
ess:weakSelf.uploadAddress];
```

停止上传，如果有文件正在上传中，则取消上传：

```
- (BOOL)stop;
```

> 说明 stop后恢复上传需要调用 resumeFile: 恢复待上传文件，或者清空队列后重新添加文件上传。

暂停上传：

```
- (BOOL)pause;
```

恢复上传：

```
- (BOOL)resume;
```

6. 回调处理

6.1 上传进度

每上传一个分片，会触发 OnUploadProgressListener 回调，回调参数包括已上传文件大小 uploadedSize 和总文件大小 totalSize 。

6.2 上传成功

上传成功时，会触发 OnUploadFinishedListener 回调。回调包含上传文件信息 UploadFileInfo 和上传结果 VodUploadResult ， VodUploadResult 包含以下属性：

```
@property (nonatomic, copy) NSString* videoId;
@property (nonatomic, copy) NSString* imageUrl;
```

需要注意， videoId 只在STS方式上传视频成功后才有返回值， imageUrl 只在STS方式上传图片成功后才有返回值。如果使用上传地址加凭证方式上传， videoId 和 imageUrl 不会返回，相应的值在请求上传地址加凭证时就可以获取到。

> 说明 视频上传成功后会返回videoId作为视频id，拿到videoId之后需要获取播放地址进行播放。相关文档可以参考获取播放地址播放。 图片上完成后会返回imageUrl，开启URL鉴权后imageUrl会有过期时间。相关配置可以参考URL鉴权。

6.3 上传失败

上传失败时，会触发 OnUploadFailedListener 回调。在该回调方法中，我们可以通过 code 和 message 查看具体原因，页面上给予用户提示。 错误码参见：点播错误码和oss错误码。

6.4 凭证过期处理

上传凭证和STS过期，会触发 OnUploadTokenExpiredListener 回调。在该回调方法中，我们可以向AppServer重新请求新的上传凭证或STS，并调用以下方法继续上传：

重新设置上传凭证

```
- (BOOL)resumeWithAuth:(NSString *)uploadAuth;
```

重新设置STS

```
- (BOOL)resumeWithToken:(NSString *)accessKeyId
      accessKeySecret:(NSString *)accessKeySecret
        secretToken:(NSString *)secretToken
         expireTime:(NSString *)expireTime;
```

6.5 超时处理

上传超时，会触发 `OnUploadRertyListener` 回调并自动重试。在该回调方法中，我们可以在页面上给予用户提示或者调用 `stop` 方法停止上传。此外，可以设置 `maxRetryCount` 属性，指定最大重试次数。 超时重试发现可以继续上传时，会触发 `OnUploadRertyResumeListener` 回调并恢复上传。

高级设置

`VODUploadClient` 支持以下高级设置：

```
/**
 上传文件到服务端是否转码,默认值YES
 */
@property (nonatomic, assign) BOOL transcode;

/**
 最大超时重试次数，默认值INT_MAX
 */
@property (nonatomic, assign) uint32_t maxRetryCount;

/**
 超时时间
 */
@property (nonatomic, assign) NSTimeInterval timeoutIntervalForRequest;

/**
 缓存文件夹位置
 */
@property (nonatomic, copy) NSString * recordDirectoryPath;

/**
 是否记录上传进度(断点续传)，默认值YES
 */
@property (nonatomic, assign) BOOL recordUploadProgress;

/**
 分片大小，默认值1024 * 1024
*/
@property (nonatomic, assign) NSInteger uploadPartSize;

/**
 vod region，默认值"cn-shanghai"
 */
@property (nonatomic, copy) NSString *region;
```

2.5.3. 短视频上传

简介

短视频上传 `VODUploadSVideoClient` 主要针对需要同时上传封面图片+视频的场景，旨在简化用户的接口调用，内部是在 `VODUploadClient` 的基础上封装实现的。短视频上传只支持STS方式上传，要想通过上传地址加凭证方式上传短视频请使用 `VODUploadClient` 分别上传封面图片+视频。

整体步骤

1. 请求STS，相关概念请参见相关文档。
2. 初始化上传实例。
3. 回调设置，所有的上传状态包括进度，上传成功，上传失败，凭证过期都在这里进行处理。
4. 启动上传。
5. 回调处理。

1. 请求STS

通过STS方式，客户端需要向AppServer发送请求，AppServer向阿里云STS服务请求临时STS凭证。请求成功将返回STS凭证，AppServer需要将结果返回给客户端。

2. 初始化

首先，声明 `VODUploadSVideoClient` 属性，注意不能是局部变量。

```
@property (nonatomic, strong) VODUploadSVideoClient *client;
```

初始化并设置代理回调。

```
self.client = [[VODUploadSVideoClient alloc] init];
self.client.delegate = self;
```

3. 回调设置

从初始化代码片段中可以看到，需要设置delegate代理，代理需要实现 `VODUploadSVideoClientDelegate` 协议，包含以下代理方法：

```
/**
 上传成功回调

 @param result 上传结果
 */
- (void)uploadSuccessWithResult:(VodSVideoUploadResult *)result;

/**
 上传失败回调

 @param code 错误码
 @param message 错误描述
 */
- (void)uploadFailedWithCode:(NSString *)code message:(NSString *)message;

/**
 上传进度回调

 @param uploadedSize 已上传文件大小
 @param totalSize 文件总大小
 */
- (void)uploadProgressWithUploadedSize:(long long)uploadedSize totalSize:(long long)totalSize;

/**
 token过期回调
 */
- (void)uploadTokenExpired;

/**
 上传开始重试回调
 */
- (void)uploadRetry;

/**
 上传结束重试，继续上传回调
 */
- (void)uploadRetryResume;
```

4. 启动上传

短视频上传不支持上传列表，每次上传短视频只需调用上传接口：

```
NSString *videoPath = [[NSBundle mainBundle] pathForResource:@"svideo" ofType:@"mp4"];
    NSString *imagePath = [[NSBundle mainBundle] pathForResource:@"cover" ofType:@"png"];
    VodSVideoInfo *info = [VodSVideoInfo new];
    info.title = @"short video";
    info.desc = @"desc";
    info.cateId = @(10);
    info.tags = @"game";
    [self.client uploadWithVideoPath:videoPath imagePath:imagePath svideoInfo:info accessKeyId:self.
keyId accessKeySecret:self.keySecret accessToken:self.token];
```

从示例代码可以看到，调用上传方法需要传入待上传的视频加封面路径，同时还有 `VodSVideoInfo` 和第一步请求获取的STS凭证

> 注意 支持的文件大小<=4G。

`VodSVideoInfo` 具体结构如下：

```
//标题
@property (nonatomic, copy) NSString* title;
//标签
@property (nonatomic, copy) NSString* tags;
//描述
@property (nonatomic, copy) NSString* desc;
//分类id
@property (nonatomic, strong) NSNumber* cateId;
```

其他上传控制方法，暂停上传：

```
- (void)pause;
```

恢复上传：

```
- (BOOL)resume;
```

取消上传：

```
- (BOOL)cancel;
```

5. 回调处理

5.1 上传进度

每上传一个分片，会触发 `uploadProgressWithUploadedSize: totalSize:` 回调，回调参数包括已上传文件大小 `uploadedSize` 和总文件大小 `totalSize` 。

5.2 上传成功

上传成功时，会触发 `uploadSuccessWithResult:` 回调。回调包含上传结果 `VodSVideoUploadResult` ， `VodSVideoUploadResult` 包含以下属性：

```
@property (nonatomic, copy) NSString* videoId;
@property (nonatomic, copy) NSString* imageUrl;
```

> **说明** 视频上传成功后会返回videoId作为视频id，拿到videoId之后需要获取播放地址进行播放。相关文档可以参考获取播放地址播放。
>
> 图片上完成后会返回imageUrl，开启URL鉴权后imageUrl会有过期时间。相关配置可以参考URL鉴权。

5.3 上传失败

上传失败时，会触发 `uploadFailedWithCode: message:` 回调。在该回调方法中，我们可以通过 `code` 和 `message` 查看具体原因，页面上给予用户提示。 错误码参见：点播错误码和oss错误码。

5.4 凭证过期处理

上传凭证和STS过期，会触发 `uploadTokenExpired` 回调。在该回调方法中，我们可以向AppServer重新请求新的STS凭证，并调用以下方法继续上传：

```
- (void)refreshWithAccessKeyId:(NSString *)accessKeyId
          accessKeySecret:(NSString *)accessKeySecret
            accessToken:(NSString *)accessToken
             expireTime:(NSString *)expireTime;
```

5.5 超时处理

上传超时，会触发 `uploadRetry` 回调并自动重试。在该回调方法中，我们可以在页面上给予用户提示或者调用 `cancel` 方法停止上传。此外，可以设置 `maxRetryCount` 属性，指定最大重试次数。

超时重试发现可以继续上传时，会触发 `uploadRetryResume` 回调并恢复上传。

高级设置

`VODUploadSVideoClient` 支持以下高级设置：

```
/**
 上传文件到服务端是否转码,默认值YES
 */
@property (nonatomic, assign) BOOL transcode;

/**
 最大超时重试次数，默认值INT_MAX
 */
@property (nonatomic, assign) uint32_t maxRetryCount;

/**
 超时时间
 */
@property (nonatomic, assign) NSTimeInterval timeoutIntervalForRequest;

/**
 缓存文件夹位置
 */
@property (nonatomic, copy) NSString * recordDirectoryPath;

/**
 是否记录上传进度(断点续传)，默认值YES
 */
@property (nonatomic, assign) BOOL recordUploadProgress;

/**
 分片大小，默认值1024 * 1024
*/
@property (nonatomic, assign) NSInteger uploadPartSize;

/**
 vod region，默认值"cn-shanghai"
 */
@property (nonatomic, copy) NSString *region;
```

3.服务端上传

3.1. Java上传SDK

1. SDK简介

概述

点播服务(VoD)基于对象存储(OSS)构建，开通VoD时会自动分配独立的系统Bucket，以存储各种媒体文件，包括上传的视频、音频、图片等源文件，以及转码后的输出文件、截图和封面等等，并作为点播加速域名的源站。使用VoD上传SDK能方便、快速实现媒体文件的上传。支持的文件格式参考 媒体上传文件支持。

功能介绍

使用此上传SDK可实现以下功能：

主要功能

- 上传本地音视频到点播，默认使用分片上传，最大支持48.8TB的单个文件；支持断点续传。
- 上传网络音视频到点播，指定URL地址，即可自动下载并上传到点播，最大支持48.8TB的单个文件。
- 上传本地图片到点播，指定本地文件路径，即可自动上传到点播。
- 上传网络图片到点播，指定URL地址，即可自动下载并上传到点播。
- 上传本地m3u8音视频（包括所有分片文件）到点播，需指定本地m3u8索引文件和分片文件目录。
- 上传网络m3u8音视频（包括所有分片文件）到点播，需指定网络m3u8索引文件和分片文件的URL地址。
- 上传本地辅助媒资文件到点播，指定本地文件路径，即可自动上传到点播。
- 上传网络辅助媒资文件到点播，指定URL地址，即可自动下载并上传到点播。

其他功能

- 上传进度条功能，支持SDK默认进度回调和自定义进度回调，m3u8文件上传暂不支持。
- 可指定上传脚本部署的ECS区域，如果与点播存储（OSS）区域相同，则自动使用内网上传文件至存储，上传更快且更省公网流量（由于点播API只提供外网域名访问，因此部署上传脚本的ECS服务器必须具有访问外网的权限）。
- 可指定点播中心（默认为上海）和存储区域，便于海外上传。

- 支持上传时设置元数据(标题等)，以及StorageLocation、UserData、转码模板、点播工作流等。
- 支持STS方式接入，需实现传递和刷新STS的接口，当文件上传时间超过STS过期时间时，SDK内部定期获取新的STS信息，进行后续上传操作。

2. SDK安装

环境要求

Java版本1.8.0

安装SDK

以 1.4.12 版本为例，步骤如下：

1.下载Java示例代码VODUploadDemo-java-1.4.12.zip开发包(包含示例代码和所需jar包），见服务端上传SDK；

2.将解压后lib目录下的所有jar文件拷贝至您的项目中；

3.SDK依赖的jar包版本说明

注意：以下列举出部分依赖jar包的版本，您可直接在您的项目中添加maven依赖，也可以将VODUploadDemo-java-1.4.12.zip包中的所有jar包引入您的项目中使用。其中，aliyun-java-vod-upload-1.4.12.jar 还未正式开源，请您直接引入jar包至您的项目中使用。

```
<dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-core</artifactId>
    <version>4.5.1</version>
</dependency>
<dependency>
    <groupId>com.aliyun.oss</groupId>
    <artifactId>aliyun-sdk-oss</artifactId>
    <version>3.1.0</version>
</dependency>
 <dependency>
    <groupId>com.aliyun</groupId>
    <artifactId>aliyun-java-sdk-vod</artifactId>
    <version>2.15.11</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.28</version>
</dependency>
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20170516</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.2</version>
</dependency>
```

4.使用IDE开发时引用jar包的方法，以Eclipse和IntelliJ IDEA为例说明如下：

4.1 在Eclipse中选择您的工程，右击 -> Properties -> Java Build Path -> Add JARs;

4.2 在IntelliJ IDEA中打开您的工程，File -> Project Structure -> Modules -> 右侧Dependencies -> + -> JARs or directories

5.选中您在第一步拷贝的所有jar文件；

经过以上几步，您就可以在Eclipse或IntelliJ IDEA项目中使用VODUpload Java SDK。

3. 示例代码

上传SDK示例

将VODUploadDemo-java-1.4.12.zip开发包解压后, 在sample目录下的UploadVideoDemo.java为文件上传示例程序, 如下:

```
public class UploadVideoDemo {
    //账号AK信息请填写(必选)
    private static final String accessKeyId = "";
    //账号AK信息请填写(必选)
    private static final String accessKeySecret = "";

    public static void main(String[] args) {
        //1.音视频上传-本地文件上传
        //视频标题(必选)
        String title = "测试标题";
        //本地文件上传和文件流上传时，文件名称为上传文件绝对路径，如:/User/sample/文件名称.mp4 (必选)
        //文件名必须包含扩展名
        String fileName = "测试文件名称.mp4";
        //本地文件上传
        testUploadVideo(accessKeyId, accessKeySecret, title, fileName);

        //2.图片上传-本地文件上传
        testUploadImageLocalFile(accessKeyId, accessKeySecret);
    }

    /**
     * 本地文件上传接口
     *
     * @param accessKeyId
     * @param accessKeySecret
     * @param title
     * @param fileName
     */
    private static void testUploadVideo(String accessKeyId, String accessKeySecret, String title, String fileName) {
        UploadVideoRequest request = new UploadVideoRequest(accessKeyId, accessKeySecret, title, fileName);
        /* 可指定分片上传时每个分片的大小，默认为1M字节 */
        request.setPartSize(1 * 1024 * 1024L);
        /* 可指定分片上传时的并发线程数，默认为1，(注：该配置会占用服务器CPU资源，需根据服务器情况指定）*/
        request.setTaskNum(1);
        /* 是否开启断点续传, 默认断点续传功能关闭。当网络不稳定或者程序崩溃时，再次发起相同上传请求，可以继续
```

```
未完成的上传任务，适用于超时3000秒仍不能上传完成的大文件。
    注意: 断点续传开启后，会在上传过程中将上传位置写入本地磁盘文件，影响文件上传速度，请您根据实际情况选择是否开启*/
    request.setEnableCheckpoint(false);
    /* OSS慢请求日志打印超时时间，是指每个分片上传时间超过该阈值时会打印debug日志，如果想屏蔽此日志，请调整该阈值。单位: 毫秒，默认为300000毫秒*/
    //request.setSlowRequestsThreshold(300000L);
    /* 可指定每个分片慢请求时打印日志的时间阈值，默认为300s*/
    //request.setSlowRequestsThreshold(300000L);
    /* 是否使用默认水印(可选)，指定模板组ID时，根据模板组配置确定是否使用默认水印*/
    //request.setIsShowWaterMark(true);
    /* 自定义消息回调设置(可选)，参数说明参考文档 https://help.aliyun.com/document_detail/86952.html#UserData */
    // request.setUserData("{\"Extend\":{\"test\":\"www\",\"localId\":\"xxxx\"},\"MessageCallback\":{\"CallbackURL\":\"http://test.test.com\"}}");

    /* 视频分类ID(可选) */
    //request.setCateId(0);
    /* 视频标签,多个用逗号分隔(可选) */
    //request.setTags("标签1,标签2");
    /* 视频描述(可选) */
    //request.setDescription("视频描述");
    /* 封面图片(可选) */
    //request.setCoverURL("http://cover.sample.com/sample.jpg");
    /* 模板组ID(可选) */
    //request.setTemplateGroupId("8c4792cbc8694*****d5330e56a33d");
    /* 存储区域(可选) */
    //request.setStorageLocation("in-2017032*****18266-5sejdln9o.oss-cn-shanghai.aliyuncs.com");
    /* 开启默认上传进度回调 */
    // request.setPrintProgress(true);
    /* 设置自定义上传进度回调 (必须继承 ProgressListener) */
    // request.setProgressListener(new PutObjectProgressListener());
    UploadVideoImpl uploader = new UploadVideoImpl();
    UploadVideoResponse response = uploader.uploadVideo(request);
    System.out.print("RequestId=" + response.getRequestId() + "\n");  //请求视频点播服务的请求ID
    if (response.isSuccess()) {
      System.out.print("VideoId=" + response.getVideoId() + "\n");
    } else {
      /* 如果设置回调URL无效，不影响视频上传，可以返回VideoId同时会返回错误码。其他情况上传失败时，VideoId为空，此时需要根据返回错误码分析具体错误原因 */
      System.out.print("VideoId=" + response.getVideoId() + "\n");
```

```
            System.out.print("ErrorCode=" + response.getCode() + "\n");
            System.out.print("ErrorMessage=" + response.getMessage() + "\n");
        }
    }

    /**
     * 图片上传接口，本地文件上传示例
     * 参数参考文档 https://help.aliyun.com/document_detail/55619.html
     *
     * @param accessKeyId
     * @param accessKeySecret
     */
    private static void testUploadImageLocalFile(String accessKeyId, String accessKeySecret) {
        // 图片类型（必选）取值范围：default（默认），cover（封面），watermark(水印)
        String imageType = "cover";
        UploadImageRequest request = new UploadImageRequest(accessKeyId, accessKeySecret, imageT
ype);
        /* 图片文件扩展名（可选）取值范围：png，jpg，jpeg */
        //request.setImageExt("png");
        /* 图片标题（可选）长度不超过128个字节，UTF8编码 */
        //request.setTitle("图片标题");
        /* 图片标签（可选）单个标签不超过32字节，最多不超过16个标签，多个用逗号分隔，UTF8编码 */
        //request.setTags("标签1,标签2");
        /* 存储区域（可选）*/
        //request.setStorageLocation("out-4f3952f78c021*****013e7.oss-cn-shanghai.aliyuncs.com");
        /* 流式上传时，InputStream为必选，fileName为源文件名称，如:文件名称.png(可选)*/
        //request.setFileName("测试文件名称.png");
        /* 开启默认上传进度回调 */
        // request.setPrintProgress(true);
        /* 设置自定义上传进度回调 (必须继承 ProgressListener) */
        // request.setProgressListener(new PutObjectProgressListener());

        UploadImageImpl uploadImage = new UploadImageImpl();
        UploadImageResponse response = uploadImage.upload(request);
        System.out.print("RequestId=" + response.getRequestId() + "\n");
        if (response.isSuccess()) {
            System.out.print("ImageId=" + response.getImageId() + "\n");
            System.out.print("ImageURL=" + response.getImageURL() + "\n");
        } else {
            System.out.print("ErrorCode=" + response.getCode() + "\n");
```

```
            System.out.print("ErrorMessage=" + response.getMessage() + "\n");
        }


    }
}
```

上传进度条示例

将VODUploadDemo-java-1.4.12.zip开发包解压后，在sample目录下的PutObjectProgressListener.java为上传进度回调函数示例程序，该类必须继承VoDProgressListener类。其中，ProgressEvent是通过OSS上传文件时产生的进度回调通知，您可自定义各个事件通知的业务处理逻辑，示例代码如下：

```
import com.aliyun.oss.event.ProgressEvent;
import com.aliyun.oss.event.ProgressEventType;

/**
 * 上传进度回调方法类
 * 当您开启上传进度回调时该事件回调才会生效。
 * OSS分片上传成功或失败均触发相应的回调事件，您可根据业务逻辑处理相应的事件回调。
 * 当创建音视频信息成功后，此上传进度回调中的videoId为本次上传生成的视频ID，您可以根据视频ID进行音视频管理。
 * 当创建图片信息成功后，此上传进度回调中的ImageId为本次上传生成的图片ID，您可以根据视频ID进行图片管理。
 */

public class PutObjectProgressListener implements VoDProgressListener {
    /**
     * 已成功上传至OSS的字节数
     */
    private long bytesWritten = 0;
    /**
     * 原始文件的总字节数
     */
    private long totalBytes = -1;
    /**
     * 本次上传成功标记
     */
    private boolean succeed = false;
    /**
     * 视频ID
     */
    private String videoId;
```

```
private String videoId;
/**
 * 图片ID
 */
private String imageId;

public void progressChanged(ProgressEvent progressEvent) {
    long bytes = progressEvent.getBytes();
    ProgressEventType eventType = progressEvent.getEventType();
    switch (eventType) {
        // 开始上传事件
        case TRANSFER_STARTED_EVENT:
            if (videoId != null) {
                System.out.println("Start to upload videoId " + videoId + "......");
            }
            if (imageId != null) {
                System.out.println("Start to upload imageId " + imageId + "......");
            }
            break;
        // 计算待上传文件总大小事件通知，只有调用本地文件方式上传时支持该事件
        case REQUEST_CONTENT_LENGTH_EVENT:
            this.totalBytes = bytes;
            System.out.println(this.totalBytes + "bytes in total will be uploaded to OSS.");
            break;
        // 已经上传成功文件大小事件通知
        case REQUEST_BYTE_TRANSFER_EVENT:
            this.bytesWritten += bytes;
            if (this.totalBytes != -1) {
                int percent = (int) (this.bytesWritten * 100.0 / this.totalBytes);
                System.out.println(bytes + " bytes have been written at this time, upload progress: " +
                        percent + "%(" + this.bytesWritten + "/" + this.totalBytes + ")");
            } else {
                System.out.println(bytes + " bytes have been written at this time, upload sub total : " +
                        "(" + this.bytesWritten + ")");
            }
            break;
        // 文件全部上传成功事件通知
        case TRANSFER_COMPLETED_EVENT:
            this.succeed = true;
            if (videoId != null) {
                System.out.println("Succeed to upload videoId " + videoId + " , " + this.bytesWritten + " byte
```

```
s have been transferred in total.");
            }
            if (imageId != null) {
                System.out.println("Succeed to upload imageId " + imageId + " , " + this.bytesWritten + " byt
es have been transferred in total.");
            }
            break;
        // 文件上传失败事件通知
        case TRANSFER_FAILED_EVENT:
            if (videoId != null) {
                System.out.println("Failed to upload videoId " + videoId + " , " + this.bytesWritten + " bytes
have been transferred.");
            }
            if (imageId != null) {
                System.out.println("Failed to upload imageId " + imageId + " , " + this.bytesWritten + " bytes
have been transferred.");
            }
            break;

        default:
            break;
      }
    }

    public boolean isSucceed() {
      return succeed;
    }

    public void onVidReady(String videoId) {
      setVideoId(videoId);
    }

    public void onImageIdReady(String imageId) {
      setImageId(imageId);
    }

    public String getVideoId() {
      return videoId;
    }

    public void setVideoId(String videoId) {
```

```
        this.videoId = videoId;
    }

    public String getImageId() {
        return imageId;
    }

    public void setImageId(String imageId) {
        this.imageId = imageId;
    }
}
```

3.2. Python上传SDK

SDK介绍

简介

点播服务(VoD)基于对象存储(OSS)构建，开通VoD时会自动分配独立的系统Bucket，以存储各种媒体文件，包括上传的视频、音频、图片等源文件，以及转码后的输出文件、截图和封面等等，并作为点播加速域名的源站。使用VoD上传SDK能方便、快速实现媒体文件的上传。支持的文件格式参考媒体上传文件支持。

功能介绍

使用此上传SDK可实现以下功能：

主要功能

- 可上传各种媒体文件到点播：视频(含音频)、图片、辅助媒资(如水印、字幕文件)。
- 上传本地媒体文件到点播，默认使用分片上传，最大支持48.8TB的单个文件；暂不支持断点续传。
- 上传网络媒体文件到点播，最大支持48.8TB的单个文件，会先下载到本地临时目录再上传；暂不支持断点续传。
- 上传M3U8视频，同时提供解析M3U8索引文件得到分片地址列表的接口；也可自行指定分片文件地址。

其它功能

- 上传进度条功能，支持默认进度回调和自定义进度回调。
- 可指定上传脚本部署的ECS区域，如果和点播存储区域相同，则自动使用内网上传，上传更快且更省公网流量。
- 可指定点播中心（默认为上海）和存储区域，便于海外上传。
- 支持上传时设置元数据(标题等)，以及StorageLocation、UserData、转码模板等。

SDK安装

环境要求

- Python 2.7 及以上版本，可到Python官网下载合适的版本安装。
- pip，若尚未安装请参看pip官网安装。

安装SDK

1. 安装依赖包

```
pip install aliyun-python-sdk-core
pip install aliyun-python-sdk-vod
pip install oss2
```

> 说明
> - 如果使用的是 Python 3.x，请将`pip install aliyun-python-sdk-core`修改为`pip install aliyun-python-sdk-core-v3`。如果同时安装了不同版本，可使用`pip3`命令。
> - 如果安装时出现权限问题，请在命令前加sudo。

2. 下载SDK

   访问下载页服务端上传SDK下载，选择Python版本下载。解压后拷贝 `VodUploadSDK-Python` 目录下的 `voduploadsdk` 即可使用， `samples` 目录为示例代码。

更新SDK

若发现新的接口或已有接口新的功能在当前SDK没有，可更新到最新版。

1. 更新依赖包

```
pip install --upgrade aliyun-python-sdk-vod
pip install --upgrade oss2
```

2. 下载最新的Python版上传SDK，覆盖原有本地SDK文件。> `voduploadsdk` 目录下的 `ChangeLog.txt` 为发版记录，首行即为当前SDK的版本号和发布日期。

使用说明

SDK结构

`voduploadsdk目录`

- AliyunVodUtils.py文件
  - AliyunVodLog，上传SDK的日志类，基于logging实现。
  - AliyunVodUtils，上传SDK的工具类。
  - AliyunVodException，上传SDK的异常类，做统一的异常处理，外部捕获此异常即可。
- UploadVideoRequest.py文件

- UploadVideoRequest，上传视频的请求类，字段参考获取视频上传地址和凭证。

- UploadImageRequest.py文件
  - UploadImageRequest，上传图片的请求类，字段参考获取图片上传地址和凭证。
- UploadAttachedMediaRequest.py文件
  - UploadAttachedMediaRequest，上传辅助媒资的请求类，字段参考获取辅助媒资上传地址和凭证。
- AliyunVodUploader.py文件，主要是AliyunVodUploader类
  - uploadLocalVideo，上传本地视频的接口。
  - uploadWebVideo，上传网络视频的接口。
  - uploadLocalM3u8，上传本地m3u8视频。
  - uploadWebM3u8，上传网络m3u8视频。
  - uploadImage，上传本地或网络图片文件。
  - uploadAttachedMedia，上传本地或网络辅助媒资文件。
  - parseWebM3u8，解析网络m3u8文件的分片信息。
  - parseLocalM3u8，解析本地m3u8文件的分片信息。
  - setApiRegion，设置VoD的接入地址，默认为cn-shanghai(上海)、海外支持ap-southeast-1(新加坡)等区域。详情参考点播中心和访问域名。
  - setMultipartUpload，设置分片上传的阈值、分片大小。
  - uploadProgressCallback，上传进度回调函数，可重写。
  - setEnableCrc，上传时是否启用CRC校验，默认开启。
- ChangeLog.txt 版本发布记录，首行即为当前SDK的版本号和发布日期。

samples目录

- uploadVideo.py，上传视频的示例代码。
- uploadImage.py，上传图片的示例代码。

使用示例

以上传本地视频和网络视频文件为例：

```
# -*- coding: UTF-8 -*-
from voduploadsdk.AliyunVodUtils import *
from voduploadsdk.AliyunVodUploader import AliyunVodUploader
from voduploadsdk.UploadVideoRequest import UploadVideoRequest

# 测试上传本地视频
def testUploadLocalVideo(accessKeyId, accessKeySecret, filePath, storageLocation=None):
    try:
        uploader = AliyunVodUploader(accessKeyId, accessKeySecret)
        uploadVideoRequest = UploadVideoRequest(filePath, 'test upload local video')
        # 可以设置视频封面，如果是本地或网络图片可使用UploadImageRequest上传图片到点播，获取到ImageURL
        #uploadVideoRequest.setCoverURL('https://sample.com/sample.jpg')
        #uploadVideoRequest.setTags('tag1,tag2')
```

```
        #uploadVideoRequest.setTags('tag1,tag2')
        if storageLocation:
            uploadVideoRequest.setStorageLocation(storageLocation)
        videoId = uploader.uploadLocalVideo(uploadVideoRequest)
        print("file: %s, videoId: %s" % (uploadVideoRequest.filePath, videoId))

    except AliyunVodException as e:
        print(e)

# 测试上传网络视频
def testUploadWebVideo(accessKeyId, accessKeySecret, fileUrl, storageLocation=None):
    try:
        uploader = AliyunVodUploader(accessKeyId, accessKeySecret)
        uploadVideoRequest = UploadVideoRequest(fileUrl, 'test upload web video')
        uploadVideoRequest.setTags('tag1,tag2')
        if storageLocation:
            uploadVideoRequest.setStorageLocation(storageLocation)
        videoId = uploader.uploadWebVideo(uploadVideoRequest)
        print("file: %s, videoId: %s" % (uploadVideoRequest.filePath, videoId))

    except AliyunVodException as e:
        print(e)

#### 执行测试代码 ####
accessKeyId = '<AccessKeyId>'
accessKeySecret = '<AccessKeySecret>'

localFilePath = '/opt/video/sample.mp4'
testUploadLocalVideo(accessKeyId, accessKeySecret, localFilePath)

fileUrl = 'http://sample.oss.aliyuncs.com/video/sample.mp4'
#testUploadWebVideo(accessKeyId, accessKeySecret, fileUrl)
```

更多示例代码请参考解压后VodUploadSDK-Python目录下的samples目录。

常见问题

- 如何实现内网上传？ 在点播控制台存储管理查看 媒资存储区域，将上传脚本部署在对应区域的ECS上，然后在初始化AliyunVodUploader类实例时指定 `ecsRegionId` 参数为对应区域，如：

```
uploader = AliyunVodUploader(accessKeyId, accessKeySecret, 'cn-shanghai')
```

- 如何获知上传进度？ AliyunVodUploader类的 `uploadProgressCallback` 接口接收上传进度回调，可获取上传的媒体ID(视频ID)、已上传大小、总大小等信息，可重写此函数实现自己的上传进度回调。
- 如何关闭日志打印？ AliyunVodUtils.py文件的AliyunVodLog类， `VOD_PRINT_INFO_LOG_SWITCH` 为日志打印开关，值更改为0即可关闭日志打印。
- 上传是同步还是异步？ 本SDK上传时都为同步，会阻塞相应进程或线程，如要实现异步，建议使用单独线程上传。
- 支持断点续传吗？ 暂时不支持断点续传，重新上传时会创建新的视频(或图片等)信息；后续版本会支持，时间待定。
- 支持Windows下执行吗？支持Python3吗？ 本SDK支持Linux/Unix/Mac/Windows等多种平台；支持Python2.7或以上版本，包括Python3，但请安装相应版本的扩展。

3.3. PHP上传SDK

SDK介绍

简介

点播服务(VoD)基于对象存储(OSS)构建，开通VoD时会自动分配独立的系统Bucket，以存储各种媒体文件，包括上传的视频、音频、图片等源文件，以及转码后的输出文件、截图和封面等等，并作为点播加速域名的源站。使用VoD上传SDK能方便、快速实现媒体文件的上传。支持的文件格式参考 媒体上传文件支持。

功能介绍

使用此上传SDK可实现以下功能：

主要功能

- 可上传各种媒体文件到点播：视频(含音频)、图片、辅助媒资(如水印、字幕文件)。
- 上传本地媒体文件到点播，默认使用分片上传，最大支持48.8TB的单个文件；暂不支持断点续传。
- 上传网络媒体文件到点播，最大支持48.8TB的单个文件，会先下载到本地临时目录再上传；暂不支持断点续传。
- 上传M3U8视频，同时提供解析M3U8索引文件得到分片地址列表的接口；也可自行指定分片文件地址。

其它功能

- 上传进度条功能，支持默认进度回调和自定义进度回调。
- 可指定上传脚本部署的ECS区域，如果和点播存储区域相同，则自动使用内网上传，上传更快且更省公网流量。
- 可指定点播中心（默认为上海）和存储区域，便于海外上传。
- 支持上传时设置元数据(标题等)，以及StorageLocation、UserData、转码模板等。

SDK安装

环境要求

- PHP 5.3+，可通过 `php -v` 命令查看当前的PHP版本。若未安装，请到 PHP官网 下载合适的版本安装。
- 确认已安装并启用相应扩展：php-mbstring、php_curl
- 支持 Linux/Unix/Mac/Windows 多平台

安装SDK

访问下载页 服务端上传SDK下载，选择PHP版本下载。解压后拷贝 `VodUploadSDK-PHP` 目录下的 `voduploadsdk` 即可使用， `samples` 目录为示例代码。

更新SDK

若发现新的接口或已有接口新的功能在当前SDK没有，可更新到最新版： 下载最新的PHP版上传SDK，解压后覆盖现有文件。

> 说明 `voduploadsdk` 目录下的 `ChangeLog.txt` 为发版记录，首行即为当前SDK的版本号和发布日期。

使用说明

SDK结构

结构图

上传SDK下的文档结构（ `VodUploadSDK-PHP` 目录）如下：

```
VodUploadSDK-PHP
└──────voduploadsdk
|   |   ChangeLog.txt
|   |   Autoloader.php
|   |
|   └──────uploader
|   |   |   UploadVideoRequest.php
|   |   |   UploadImageRequest.php
|   |   |   UploadAttachedMediaRequest.php
|   |   |
|   |   |   AliyunVodUploader.php
|   |   |   AliyunVodUtils.php
|   |
|   └──────aliyun-php-sdk-core
|   └──────aliyun-php-sdk-oss
|   └──────aliyun-php-sdk-vod
|
└──────samples
    |   uploadVideo.php
    |   uploadImage.php
    |   uploadAttachedMedia.php
```

辅助文件

- ChangeLog.txt 版本发布记录，首行即为当前SDK的版本号和发布日期。
- Autoloader.php 类文件加载器，使用SDK时只需require此文件即可。

主体代码

- UploadVideoRequest.php
  - UploadVideoRequest，上传视频的请求类，字段参考获取视频上传地址和凭证。
- UploadImageRequest.php
  - UploadImageRequest，上传图片的请求类，字段参考获取图片上传地址和凭证。
- UploadAttachedMediaRequest.php
  - UploadAttachedMediaRequest，上传辅助媒资的请求类，字段参考获取辅助媒资上传地址和凭证。
- AliyunVodUploader.php，主要是AliyunVodUploader类
  - __construct，构造函数，可设置上传的AccessKey和点播接入区域。
  - uploadLocalVideo，上传本地视频的接口。
  - uploadWebVideo，上传网络视频的接口。

- uploadLocalImage，上传本地图片。
- uploadWebImage，上传网络图片。
- uploadLocalAttachedMedia，上传本地辅助媒资文件。
- uploadWebAttachedMedia，上传网络辅助媒资文件。
- uploadLocalM3u8，上传本地m3u8视频。
- uploadWebM3u8，上传网络m3u8视频。
- parseM3u8File，解析m3u8索引文件得到分片地址列表。
- setEcsRegionId，设置上传脚本部署的ECS区域(如有)，如与点播存储同一区域会自动启用内网上传。
- setEnableSSL，是否启用SSL(网络请求使用HTTPS)，默认不启用，以避免相关扩展未安装或配置异常时无法使用。
- uploadProgressCallback，上传进度回调函数，可重写。

- AliyunVodUtils.php
  - AliyunVodUtils，工具类，提供截取字符串、获取扩展名、获取文件名等静态函数。
  - AliyunVodLog，实现简单打印的日志类，$logSwitch为日志开关。
  - AliyunVodDownloader，实现下载网络文件。
  - AliyunVodReportUpload，实现上传进度汇报。
  - AliyunVodError，定义错误码。

依赖包

- aliyun-php-sdk-core，上传SDK依赖的基础类，封装了阿里云API签名和HTTP请求等。
- aliyun-php-sdk-vod，点播的服务端接口SDK，封装了点播API的请求。
- aliyun-php-sdk-oss，上传SDK依赖的OSS类，封装了OSS上传等操作。

samples

- uploadVideo.php，上传视频的示例代码。
- uploadImage.php，上传图片的示例代码。
- uploadAttachedMedia.php，上传辅助媒资的示例代码。

使用示例

以上传本地视频和网络视频文件为例：

```
<?php
/**
 * Created by Aliyun ApsaraVideo VoD.
 * User: https://www.aliyun.com/product/vod
 * API document: https://www.alibabacloud.com/help/zh/doc-detail/55407.htm
 */

require_once dirname(__DIR__) . DIRECTORY_SEPARATOR . 'voduploadsdk' . DIRECTORY_SEPARATOR . 'Autoloader.php';

date_default_timezone_set('PRC');
```

```
// 测试上传本地视频
function testUploadLocalVideo($accessKeyId, $accessKeySecret, $filePath)
{
    try {
        $uploader = new AliyunVodUploader($accessKeyId, $accessKeySecret);
        $uploadVideoRequest = new UploadVideoRequest($filePath, 'testUploadLocalVideo via PHP-SDK');
        //$uploadVideoRequest->setCateId(1);
        //$uploadVideoRequest->setCoverURL("http://xxxx.jpg");
        //$uploadVideoRequest->setTags('test1,test2');
        //$uploadVideoRequest->setStorageLocation('outin-xx.oss-cn-beijing.aliyuncs.com');
        //$uploadVideoRequest->setTemplateGroupId('6ae347b0140181ad371d197ebe289326');
        $userData = array(
            "MessageCallback"=>array("CallbackURL"=>"https://demo.sample.com/ProcessMessageCallbac
k"),
            "Extend"=>array("localId"=>"xxx", "test"=>"www")
        );
        $uploadVideoRequest->setUserData(json_encode($userData));
        $res = $uploader->uploadLocalVideo($uploadVideoRequest);
        print_r($res);
    } catch (Exception $e) {
        printf("testUploadLocalVideo Failed, ErrorMessage: %s\n Location: %s %s\n Trace: %s\n",
            $e->getMessage(), $e->getFile(), $e->getLine(), $e->getTraceAsString());
    }
}

// 测试上传网络视频
function testUploadWebVideo($accessKeyId, $accessKeySecret, $fileURL)
{
    try {
        $uploader = new AliyunVodUploader($accessKeyId, $accessKeySecret);
        $uploadVideoRequest = new UploadVideoRequest($fileURL, 'testUploadWebVideo via PHP-SDK');
        $res = $uploader->uploadWebVideo($uploadVideoRequest);
        print_r($res);
    } catch (Exception $e) {
        printf("testUploadWebVideo Failed, ErrorMessage: %s\n Location: %s %s\n Trace: %s\n",
            $e->getMessage(), $e->getFile(), $e->getLine(), $e->getTraceAsString());
    }
}

// 测试上传本地m3u8视频
```

```
// 测试上传本地m3u8视频
function testUploadLocalM3u8($accessKeyId, $accessKeySecret, $m3u8FilePath)
{
    try {
        $uploader = new AliyunVodUploader($accessKeyId, $accessKeySecret);
        $uploadVideoRequest = new UploadVideoRequest($m3u8FilePath, 'testUploadLocalM3u8 via PHP-SDK');
        // 调用接口解析m3u8的分片地址列表，如果解析结果不准确，请自行拼接地址列表(默认分片文件和m3u8文件位于同一目录)
        $sliceFiles = $uploader->parseM3u8File($m3u8FilePath);
        //print_r($sliceFiles);
        $res = $uploader->uploadLocalM3u8($uploadVideoRequest, $sliceFiles);
        print_r($res);
    } catch (Exception $e) {
        printf("testUploadLocalM3u8 Failed, ErrorMessage: %s\n Location: %s %s\n Trace: %s\n",
            $e->getMessage(), $e->getFile(), $e->getLine(), $e->getTraceAsString());
    }
}

// 测试上传网络m3u8视频
function testUploadWebM3u8($accessKeyId, $accessKeySecret, $m3u8FileUrl)
{
    try {
        $uploader = new AliyunVodUploader($accessKeyId, $accessKeySecret);
        $uploadVideoRequest = new UploadVideoRequest($m3u8FileUrl, 'testUploadWebM3u8 via PHP-SDK');
        // 调用接口解析m3u8的分片地址列表，如果解析结果不准确，请自行拼接地址列表(默认分片文件和m3u8文件位于同一目录)
        $sliceFileUrls = $uploader->parseM3u8File($m3u8FileUrl);
        //print_r($sliceFileUrls);
        $res = $uploader->uploadWebM3u8($uploadVideoRequest, $sliceFileUrls);
        print_r($res);
    } catch (Exception $e) {
        printf("testUploadWebM3u8 Failed, ErrorMessage: %s\n Location: %s %s\n Trace: %s\n",
            $e->getMessage(), $e->getFile(), $e->getLine(), $e->getTraceAsString());
    }
}


#### 执行测试代码 ####
$accessKeyId = '<AccessKeyId>';
```

```
$accessKeySecret = '<AccessKeySecret>';



//$localFilePath = 'C:\test\sample.mp4';
$localFilePath = '/opt/video/sample.mp4';
//testUploadLocalVideo($accessKeyId, $accessKeySecret, $localFilePath);

$webFileURL = 'http://vod-test1.cn-shanghai.aliyuncs.com/b55b904bc612463b812990b7c8cc95c8/daa30
814c0c340cf8199926f78aa5c0e-a0bc05ba62c3e95cc672e88b828148c9-ld.mp4?auth_key=1608774986-0-0-c
56acd302bea0c331370d8ed686502fe';
testUploadWebVideo($accessKeyId, $accessKeySecret, $webFileURL);

$localM3u8FilePath = '/opt/video/m3u8/sample.m3u8';
//testUploadLocalM3u8($accessKeyId, $accessKeySecret, $localM3u8FilePath);

$webM3u8FileURL = 'http://vod-test1.cn-shanghai.aliyuncs.com/b55b904bc612463b812990b7c8cc95c8/d
aa30814c0c340cf8199926f78aa5c0e-195a25af366b5edae324c47e99a03f04-ld.m3u8?auth_key=1608775606
-0-0-9fb038deaecd009dadd86721c5855629';
//testUploadWebM3u8($accessKeyId, $accessKeySecret, $webM3u8FileURL);
```

更多示例代码请参考解压后 `VodUploadSDK-PHP` 目录下的 `samples` 目录。

常见问题

- 如何实现内网上传？ 在点播控制台存储管理查看 媒资存储区域，将上传脚本部署在对应区域的ECS上，然后在初始化AliyunVodUploader类实例后，调用 `setEcsRegionId` 接口设置对应区域，如：cn-shanghai等。
- 如何获知上传进度？ AliyunVodUploader类的 `uploadProgressCallback` 接口接收上传进度回调，可获取上传的媒体ID(视频ID)、已上传大小、总大小等信息，可重写此函数实现自己的上传进度回调。
- 如何关闭日志打印？ AliyunVodUtils.php文件的AliyunVodLog类，$logSwitch为日志打印开关，更改为false即可关闭日志打印。
- 上传是同步还是异步？ 本SDK上传时都为同步，会阻塞相应进程或线程，如要实现异步，建议使用单独线程上传。
- 支持断点续传吗？ 暂时不支持断点续传，重新上传时会创建新的视频(或图片等)；后续版本会支持，时间待定。
- 支持Windows下执行吗？ 本SDK支持Linux/Unix/Mac/Windows等多种平台，但都需要安装PHP及相应扩展(如php-mbstring等)。
- 遇到连接不上怎么处理？ 请先确认连接外网是否有问题，可 `ping vod.cn-shanghai.aliyuncs.com` 看看是否能连接阿里云点播服务器，如连接有问题可确认您的网络配置，如是否允许连接外网。其次，确认是否开启了SSL，如果使用了HTTPS请求，可能相应扩展(如php_openssl)没有正确安装，或配置有问题。

3.4. C_C++上传SDK

SDK介绍

简介

点播服务(VoD)基于对象存储(OSS)构建，开通VoD时会自动分配独立的系统Bucket，以存储各种媒体文件，包括上传的视频、音频、图片等源文件，以及转码后的输出文件、截图和封面等等，并作为点播加速域名的源站。使用VoD上传SDK能方便、快速实现媒体文件的上传。支持的文件格式参考 媒体上传文件支持。

功能介绍

使用此上传SDK可实现以下功能：

主要功能

- 可上传各种媒体文件到点播：视频(含音频)、图片、辅助媒资(如水印、字幕文件)。
- 上传本地媒体文件到点播，默认使用分片上传，最大支持48.8TB的单个文件；暂不支持断点续传。
- 上传网络媒体文件到点播，最大支持48.8TB的单个文件，会先下载到本地临时目录再上传；暂不支持断点续传。
- 上传M3U8视频，同时提供解析M3U8索引文件得到分片地址列表的接口；也可自行指定分片文件地址。

其它功能

- 上传进度条功能，支持默认进度回调和自定义进度回调。
- 可指定上传脚本部署的ECS区域，如果和点播存储区域相同，则自动使用内网上传，上传更快且更省公网流量。
- 可指定点播中心和存储区域，便于海外上传。
- 支持上传时设置元数据(标题等)，以及StorageLocation、UserData、转码模板等。

SDK安装

环境要求

- 点播C/C++上传SDK与C/C++服务端SDK统一，具体环境准备、SDK安装及初始化等可参考C/C++SDK使用指南。

安装SDK

访问下载页 服务端上传SDK下载SDK下载，选择C/C++版本下载,并按照上述方式进行安装。

更新SDK

若发现新的接口或已有接口新的功能在当前SDK没有，可更新到最新版：下载最新的C/C++版上传SDK，解压后覆盖现有文件。

> 说明 解压后 `aliyun-c-sdk-vod` 目录下的 `ChangeLog.txt` 为发版记录，首行即为当前SDK的版本号和发布日期。

使用说明

SDK结构

辅助文件

- ChangeLog.txt 版本发布记录，首行即为当前SDK的版本号和发布日期。

头文件

- upload.h
  - 上传功能头文件，包含基础上传结构体、上传参数以及上传函数等。

基础结构体

- CreateUploadVideoRequest
  - CreateUploadVideoRequest，上传视频的请求类，字段参考获取视频上传地址和凭证。
- CreateUploadImageRequest
  - CreateUploadImageRequest，上传图片的请求类，字段参考获取图片上传地址和凭证。
- CreateUploadAttachedMediaRequest
  - CreateUploadAttachedMediaRequest，上传辅助媒资的请求类，字段参考获取辅助媒资上传地址和凭证。
- UploadOptions

- UploadOptions，上传参数结构体，包含如下参数：
  - void (*uploadProgressCallback) (int64_t, int64_t)，用于实现自定义上传进度回调，如果不设置则使用默认，设置为NULL则不进行回调。
  - ecsRegionId，设置上传脚本部署的ECS区域(如有)，如与点播存储同一区域会自动启用内网上传。
  - multipartUploadLimit，分片上传阈值，单位字节，默认为10MB(只对视频上传有效)。
  - multipartUploadOnceSize，分片大小，单位字节，默认为10MB(只对视频上传有效)。
  - tmpDir 本地临时存储地址，只对网络上传方式生效。

上传函数

- uploadLocalVideo，上传本地视频的接口。
- uploadWebVideo，上传网络视频的接口。
- uploadLocalImage，上传本地图片。
- uploadWebImage，上传网络图片。
- uploadLocalAttachedMedia，上传本地辅助媒资文件。
- uploadWebAttachedMedia，上传网络辅助媒资文件。
- uploadLocalM3u8，上传本地m3u8视频。
- uploadWebM3u8，上传网络m3u8视频。

samples

- UploadVideo.cpp，上传视频的示例代码。
- UploadImage.cpp，上传图片的示例代码。
- UploadAttachedMedia.cpp，上传辅助媒资的示例代码。

使用示例

以上传本地视频和网络视频文件为例：

```
void testCallback(int64_t consumed_bytes, int64_t total_bytes)
{
   printf("total :%ld, %ld\n", consumed_bytes, total_bytes);
}

// 测试上传本地视频
VodApiResponse testUploadLocalVideo(VodCredential authInfo) {
   CreateUploadVideoRequest request;
   request.fileName = "testLocal.mp4";
   request.title = "testUploadLocalVideo";
 request.cateId = "1";
   request.coverURL = "http://xxxx.jpg";
   request.tags = "test1,test2";
   request.templateGroupId = "6ae347b0140181ad371d197ebe289326";
   requests.torageLocation. = "outin-xx.oss-cn-beijing.aliyuncs.com";
   Json::Value userData;
```

```
    Json::Value callbackUrl;
    callbackUrl["CallbackURL"] = "https://demo.sample.com/ProcessMessageCallback";
    userData["MessageCallback"] = callbackUrl;
    Json::Value extend;
    extend["localId"] = "xxx";
    extend["test"] = "www";
    userData["Extend"] = extend;
    request.userData = userData.toStyledString();
    UploadOptions uploadOptions;
    //设置上传脚本部署的ECS区域(如有)，如与点播存储同一区域会自动启用内网上传
    //uploadOptions.ecsRegionId = "cn-shanghai";
    //设置自定义回调函数,否则设置为默认,设置为NULL不回调
    //uploadOptions.uploadProgressCallback = testCallback;
    //uploadOptions.multipartUploadLimit = 20*1024*1024;//设置分片上传阈值
    //uploadOptions.multipartUploadOnceSize = 10*1024*1024;//设置分片上传分片大小
    VodApiResponse result = uploadLocalVideo(authInfo, request, "./test.mp4", uploadOptions);
    return result;
}

// 测试上传网络视频
VodApiResponse testUploadWebVideo(VodCredential authInfo) {
    CreateUploadVideoRequest request;
    request.fileName = "testWeb.mp4";
    request.title = "testUploadWebVideo";
    UploadOptions uploadOptions;
    //设置上传脚本部署的ECS区域(如有)，如与点播存储同一区域会自动启用内网上传
    //uploadOptions.ecsRegionId = "cn-shanghai";
    //设置自定义回调函数,否则设置为默认,设置为NULL不回调
    //uploadOptions.uploadProgressCallback = testCallback;
    //uploadOptions.multipartUploadLimit = 20*1024*1024;//设置分片上传阈值
    //uploadOptions.multipartUploadOnceSize = 10*1024*1024;//设置分片上传分片大小
    //设置下载临时目录,默认为/tmp/
    //uploadOptions.tmpDir = "/tmp/";
    VodApiResponse result = uploadWebVideo(authInfo, request, "<Your Download Url>", uploadOption
s);
    return result;
}

// 测试上传本地m3u8视频
VodApiResponse testUploadLocalM3u8(VodCredential authInfo) {
    CreateUploadVideoRequest request;
```

```
    request.fileName = "testLocal.m3u8";
    request.title = "testUploadLocalM3u8";
    list<string> tsList;
    //注意:如果不添加则会自动解析
    //tsList.push_back("/tmp/1.ts");
    UploadOptions uploadOptions;
    VodApiResponse result = uploadLocalM3u8(authInfo, request, "./test.m3u8", tsList, uploadOptions);
    return result;
}

// 测试上传网络m3u8视频
VodApiResponse testUploadWebM3u8(VodCredential authInfo) {
    CreateUploadVideoRequest request;
    request.fileName = "testWeb.m3u8";
    request.title = "testUploadWebM3u8";
    list<string> tsList;
    //注意:如果不添加则会自动解析
    //tsList.push_back("<Ts1 Download Url>");
    //tsList.push_back("<Ts2 Download Url>");
    UploadOptions uploadOptions;
    VodApiResponse result = uploadWebM3u8(authInfo, request, "<Your M3u8 Download Url>", tsList, up
loadOptions);
    return result;
}


#### 调用测试代码 ####
VodCredential initVodClient(std::string accessKeyId, std::string accessKeySecret) {
    VodCredential authInfo;
    authInfo.accessKeyId = accessKeyId;
    authInfo.accessKeySecret = accessKeySecret;
    authInfo.regionId = "cn-shanghai";
    return authInfo;
}

int main(int argc, char * argv[]) {
    VodCredential authInfo = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecret>");
    VodApiResponse response;

    response = testUploadLocalVideo(authInfo);
    //response = testUploadWebVideo(authInfo);
```

```
    //response = testUploadLocalM3u8(authInfo);
    //response = testUploadWebM3u8(authInfo);
    printf("httpCode: %d, result: %s\n", response.httpCode, response.result.c_str());
}
```

更多示例代码请参考解压后 `aliyun-c-sdk-vod` 目录下的 `samples` 目录。

常见问题

- 如何实现内网上传？ 在点播控制台存储管理查看 媒资存储区域，将上传脚本部署在对应区域的ECS上，然后在初始化AliyunVodUploader类实例后，调用 `setEcsRegionId` 接口未对应区域，如：cn-shanghai 等。
- 如何获知上传进度？ UploadOptions中的 `uploadProgressCallback` 函数指针指向回调函数地址，可获取已上传大小、总大小等信息，可重新赋值该成员实现自己的上传进度回调。
- 上传是同步还是异步？ 本SDK上传时都为同步，会阻塞相应进程或线程，如要实现异步，建议使用单独线程上传。
- 支持断点续传吗？ 暂时不支持断点续传，重新上传时会创建新的视频(或图片等)；后续版本会支持，时间待定。
- 遇到连接不上怎么处理？ 请先确认连接外网是否有问题，可 `ping vod.cn-shanghai.aliyuncs.com` 看看是否能连接阿里云点播服务器，如连接有问题可确认您的网络配置，如是否允许连接外网