ALIBABA CLOUD

阿里云

视频点播

最佳实践

文档版本：20220506

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.点播多码率自适应配置

多码率自适应即指将指定的音视频文件流统一打包生成一个自适应码流文件，该自适应码流文件包含不同音视频文件流的码率、分辨率等信息，播放器则根据网络带宽环境自动选择最适合当前带宽环境的码流播放。

> **说明** 目前使用比较广的多码率自适应输出格式为HLS和DASH，视频点播目前仅支持HLS多码率自适应生成。更多信息，请参见多码率自适应。

简介

视频点播服务目前支持视频打包模板和字幕打包模板，用于生成可切换不同分辨率和不同语言字幕的多码率自适应文件。支持通过控制台和API的方式创建多码率自适应模板。

相比较传统的单码率播放，自适应码率文件能够让播放器可根据终端网络带宽环境，选择最适合当前带宽环境和终端设备的码流播放，从而提升播放体验。视频打包模板和普通转码模板的区别，如下表所示：

区别项	视频打包模板	普通转码模板
参数配置	需要配置打包参数（打包类型、宽带阈值）。	无此参数配置。
字幕添加	• 控制台方式：可设置外挂字幕。 • API方式：可设置压制字幕及外挂字幕。	只能通过API的方式设置压制字幕。
播放效果	可根据客户当前宽带环境自动选择码率进行播放。	只能播放指定的码流。

注意事项

- 视频打包模板不支持使用标准加密，如果要使用标准加密，请使用普通转码模板。
- 字幕文件必须和视频源文件存储在同一个区域，同一个OSS Bucket中。
- 字幕打包模板不能单独创建使用，必须与视频打包模板一同使用。
- 字幕打包模板中，字幕文件仅只支持VTT格式。通过API方式配置时需注意，字幕参数覆盖时，Language参数仅用于检索需要替换的字幕文件，而Language本身不会被替换，不存在的字幕语言不能替换。
- 转码服务均为收费项，具体请参见收费标准。

控制台方式配置

1. 创建多码率自适应的转码模板组。

   > **说明** 如果只需要使用多码率打包模板，则不需要创建普通转码模板，或删除所有普通转码模板，以免产生多余的转码费用。

   i. 登录视频点播控制台，在点播控制台左侧的导航栏选择**配置管理 > 媒体处理配置 > 转码模板组**。

   ii. 在转码模板组页签，单击**添加转码模板组**，进入添加转码模板页面。

   iii. 设置模板组名称。

iv. 在**视频打包模板**下方，单击**添加模板**，配置视频打包模板参数。

- 其中**基本参数**的**封装格式**固定为hls，**视频打包参数**的配置说明如下：
  - **打包类型**：固定为**HLS打包**。
  - **带宽阈值**：提供给播放器根据当前网络带宽环境判断需要选择播放的码流，单位：bps。建议使用推荐设置。
- **基本参数**、**视频参数**、**音频参数**、**高级参数**、**条件转码参数**的配置与普通转码模板的配置类似，请参见普通转码模板设置。
- 根据实际需求，通过**添加模板**，来创建多个不同码率、分辨率、清晰度的视频打包模板。

v. 在**字幕打包模板**下方，单击**添加模板**，单击字幕打包列表下的**添加**，配置字幕打包模板。

根据实际需求，通过**添加**，来添加多个不同语言类型的字幕，每种类型仅支持一个字幕。

字幕模板各参数说明如下：

参数名称	描述
打包类型	现视频点播只支持HLS打包，因此固定为**HLS打包**。
语言类型	字幕的语言类型，例如：**cn(中文)**、**ja(日文)**、**en-US(英文)**等。
字幕流名称	字幕在播放器上显示的名称，仅支持中文、英文、数字、中划线（-）、下划线（_）。 示例：test
字幕存储地址	字幕文件的存储地址，为完整的OSS地址。字幕文件仅支持存储在视频点播分配的系统Bucket上，不支持CDN域名地址和HTTPS地址。 字幕文件可通过调用CreateUploadAttachedMedia接口上传到视频点播分配的系统Bucket上。 示例：http://example-bucket-****.oss-cn-shanghai.aliyuncs.com/subtitles/4dba87c2-a787-42cd-8328-2369aeb8****-cn.vtt
操作	支持**删除**操作。

vi. 视频打包模板及字幕打包模板创建完成后，单击**保存**。

2. 发起多码率转码。

   i. 在视频点播控制台，选择**媒资库 > 音/视频**。

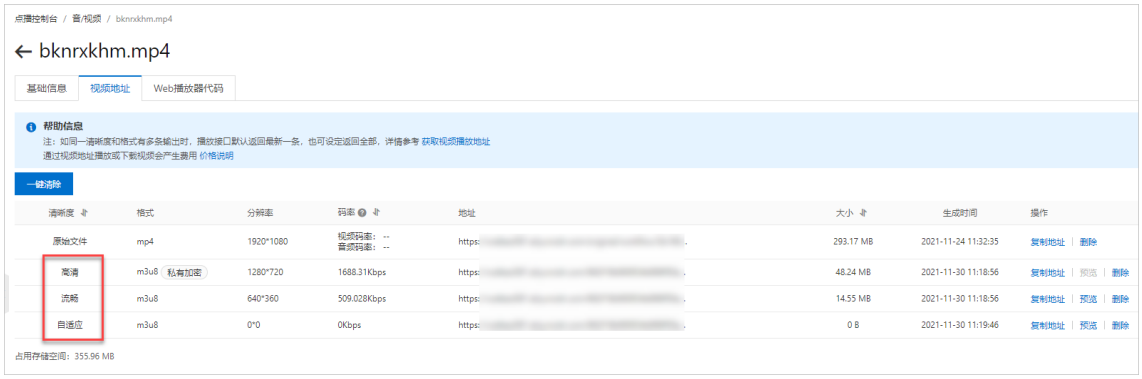
   ii. 在**音/视频**页面，单击目标视频的**媒体处理**。

   iii. 处理类型选择**用转码模板组处理**，转码模板组选择第一步所创建的转码模板组，单击**确定**，发起媒体处理。

3. 效果展示。

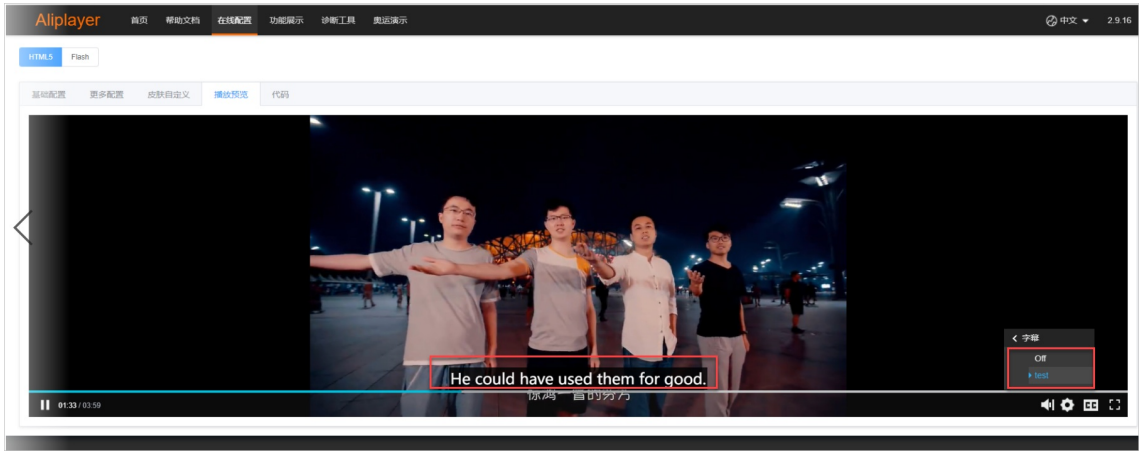
   - 多码率打包后码流展示：（2路转码流，1路自适应流）

     在点播控制台，选择**媒资库 > 音/视频**，单击目标视频（上一步经过媒体处理的视频）的**管理**，选择**视频地址**页签。



- 播放效果展示：（使用上图中自适应的地址播放）

  在Aliplayer播放器的**基础配置**页，输入播放地址（使用上图中视频地址页中自适应的地址），在**播放预览**页查看播放效果。



API方式配置

1. 创建用于多码率自适应的转码模板组。

   本示例代码中列举了普通转码模板配置、多码率打包模板配置、字幕打包模板配置，可根据实际情况使用。

   > **说明** 如果只需要使用多码率打包模板，则不需要创建普通转码模板，或删除所有普通转码模板，以免产生多余的转码费用。

```
    /**
     * 以下为调用示例
     */
    public static void main(String[] args) throws ClientException {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeyS
ecret>");
        AddTranscodeTemplateGroupResponse response = new AddTranscodeTemplateGroupRespo
nse();
        try {
            response = addTranscodeTemplateGroup(client);
            System.out.println("TranscodeTemplateGroupId = " + response.getTranscodeTem
plateGroupId());
        } catch (Exception e) {
            System.out.println("ErrorMessage = " + e.getLocalizedMessage());
```

```
        }
        System.out.println("RequestId = " + response.getRequestId());
    }
    /**
     * 添加转码模板组配置
     */
    public static AddTranscodeTemplateGroupResponse addTranscodeTemplateGroup(DefaultAcsClient client) throws Exception {
        AddTranscodeTemplateGroupRequest request = new AddTranscodeTemplateGroupRequest();
        request.setName("grouptest2");
        JSONArray transcodeTemplateList = new JSONArray();
        //普通转码模板配置
        transcodeTemplateList.add(buildNormalTranscodeConfig());
        //多码率打包转码模板配置
        transcodeTemplateList.add(buildVideoPackageConfig());
        //字幕打包配置
        transcodeTemplateList.add(buildSubtitlePackageConfig());
        request.setTranscodeTemplateList(transcodeTemplateList.toJSONString());
        System.out.println("request = " + JSONObject.toJSONString(request));
        return client.getAcsResponse(request);
    }
    /**
     * 构建需要添加的转码模板配置数据
     *
     * @return
     */
    public static JSONObject buildNormalTranscodeConfig() {
        JSONObject transcodeTemplate = new JSONObject();
        //模板类型< Normal：普通转码模板；VideoPackage：视频流打包模板；SubtitlePackage：字幕打包模板>
        transcodeTemplate.put("Type","Normal");
        //模板名称
        transcodeTemplate.put("TemplateName", "普通转码模板mp4");
        //清晰度
        transcodeTemplate.put("Definition", "HD");
        //视频流转码配置
        JSONObject video = new JSONObject();
        video.put("Width", 1280);
        //video.put("Height", 720);
        video.put("Bitrate", 1500);
        video.put("Fps", 25);
        //video.put("Remove", false);
        video.put("Codec", "H.264");
        video.put("Gop", "250");
        video.put("LongShortMode", false);
        transcodeTemplate.put("Video", video);
        //音频流转码配置
        JSONObject audio = new JSONObject();
        audio.put("Codec", "AAC");
        audio.put("Bitrate", "64");
        audio.put("Channels", "2");
        audio.put("Samplerate", "32000");
        transcodeTemplate.put("Audio", audio);
        //封装容器
```

```
        //封装容器
        JSONObject container = new JSONObject();
        container.put("Format", "mp4");
        transcodeTemplate.put("Container", container);
        //字幕替换设置
        JSONObject subtitleSetting = new JSONObject();
        JSONArray subtitleList = new JSONArray();
        JSONObject subtitle = new JSONObject();
        //字幕文件的OSS地址(不支持https地址)
        subtitle.put("SubtitleUrl", "http://outin-8db8d2****3e1c9256.oss-cn-shanghai.al
iyuncs.com/subtitle/3215879C9F724A43BC84C63BE2AA19AF****.srt");
        //字幕内容的编码格式，取值：auto(自动检测)、UTF-8、GBK、BIG5
        subtitle.put("CharEncode", "UTF-8");
        subtitleList.add(subtitle);
        transcodeTemplate.put("SubtitleList", subtitleList);
        return transcodeTemplate;
    }
    /**
     * 视频打包配置
     *
     * @return
     */
    private static JSONObject buildVideoPackageConfig() {
        JSONObject transcodeTemplate = new JSONObject();
        //模板类型< Normal：普通转码模板；VideoPackage：视频流打包模板；SubtitlePackage：字幕
打包模板>
        transcodeTemplate.put("Type","VideoPackage");
        //模板名称
        transcodeTemplate.put("TemplateName", "HLS流畅打包");
        //清晰度
        transcodeTemplate.put("Definition", "LD");
        //视频流转码配置
        JSONObject video = new JSONObject();
        video.put("Width", 1280);
        //video.put("Height", 720);
        video.put("Bitrate", 1500);
        video.put("Fps", 25);
        //video.put("Remove", false);
        video.put("Codec", "H.264");
        video.put("Gop", "250");
        video.put("LongShortMode", false);
        transcodeTemplate.put("Video", video);
        //音频流转码配置
        JSONObject audio = new JSONObject();
        audio.put("Codec", "AAC");
        audio.put("Bitrate", "64");
        audio.put("Channels", "2");
        audio.put("Samplerate", "32000");
        transcodeTemplate.put("Audio", audio);
        //封装容器
        JSONObject container = new JSONObject();
        container.put("Format", "m3u8");
        transcodeTemplate.put("Container", container);
        //封装容器设置为m3u8，必须设置MuxConfig
        JSONObject muxConfig = new JSONObject();
```

```
        JSONObject segment = new JSONObject();
        segment.put("Duration", "10");//单位是秒
        muxConfig.put("Segment", segment);
        transcodeTemplate.put("MuxConfig",muxConfig);
        //打包配置
        JSONObject packageSetting = new JSONObject();
        //打包类型：取值：HLSPackage (HLS自适应码率打包)
        packageSetting.put("PackageType","HLSPackage");
        JSONObject packageConfig = new JSONObject();
        packageConfig.put("BandWidth","500000");
        packageSetting.put("PackageConfig",packageConfig);
        transcodeTemplate.put("PackageSetting",packageSetting);
        return transcodeTemplate;
    }
    /**
     * 字幕打包配置
     *
     * @return
     */
    private static JSONObject buildSubtitlePackageConfig() {
        JSONObject transcodeTemplate = new JSONObject();
        //模板类型< Normal：普通转码模板；VideoPackage：视频流打包模板；SubtitlePackage：字幕
打包模板>
        transcodeTemplate.put("Type","SubtitlePackage");
        //模板名称
        transcodeTemplate.put("TemplateName", "多字幕打包");
        //清晰度
        transcodeTemplate.put("Definition", "HD");
        //视频流转码配置
        JSONObject video = new JSONObject();
        video.put("Width", 1280);
        //video.put("Height", 720);
        video.put("Bitrate", 1500);
        video.put("Fps", 25);
        //video.put("Remove", false);
        video.put("Codec", "H.264");
        video.put("Gop", "250");
        video.put("LongShortMode", false);
        transcodeTemplate.put("Video", video);
        //音频流转码配置
        JSONObject audio = new JSONObject();
        audio.put("Codec", "AAC");
        audio.put("Bitrate", "64");
        audio.put("Channels", "2");
        audio.put("Samplerate", "32000");
        transcodeTemplate.put("Audio", audio);
        //封装容器
        JSONObject container = new JSONObject();
        container.put("Format", "m3u8");
        transcodeTemplate.put("Container", container);
        //封装容器设置为m3u8，必须设置MuxConfig
        JSONObject muxConfig = new JSONObject();
        JSONObject segment = new JSONObject();
        segment.put("Duration", "10");//单位是秒
```

```
        muxConfig.put("Segment", segment);
        transcodeTemplate.put("MuxConfig",muxConfig);
        /*
        字幕文件OSS地址(不支持https地址、不支持纯CDN域名加速地址)
        说明：字幕文件必须和视频源文件在同一个区域(例如：cn-shanghai)，并且在同一个Bucket当中
         */
        //字幕打包模板
        JSONObject subtitlePackageConfig = new JSONObject();
        subtitlePackageConfig.put("Type","SubtitlePackage");
        //字幕打包配置
        JSONObject subtitlePackageSetting = new JSONObject();
        //打包类型：取值：HLSPackage (HLS自适应码率打包)
        subtitlePackageSetting.put("PackageType","HLSPackage");
        JSONArray subtitleExtractConfigs = new JSONArray();
        //字幕1
        JSONObject subtitleExtractConfig = new JSONObject();
        JSONArray subtitleUrlList = new JSONArray();
        subtitleUrlList.add("http://outin-bfefbb9****e1c7426.oss-cn-shanghai.aliyuncs.com/subtitle/260447BA31D24F9E9E7752BF73F1319B****.vtt");
        subtitleExtractConfig.put("SubtitleUrlList",subtitleUrlList);
        subtitleExtractConfig.put("Language","cn");
        subtitleExtractConfig.put("Format","vtt");
        subtitleExtractConfig.put("Name","中文-test");
        //字幕2
        JSONObject subtitleExtractConfig2 = new JSONObject();
        JSONArray subtitleUrlList2 = new JSONArray();
        subtitleUrlList2.add("http://outin-bfefbb9****3e1c7426.oss-cn-shanghai.aliyuncs.com/subtitle/661C67325E0543F0BB8CA7AAB756E6D8****.vtt");
        subtitleExtractConfig2.put("SubtitleUrlList",subtitleUrlList2);
        subtitleExtractConfig2.put("Language","en-US");
        subtitleExtractConfig2.put("Format","vtt");
        subtitleExtractConfig2.put("Name","英文-test");
        subtitleExtractConfigs.add(subtitleExtractConfig);
        subtitleExtractConfigs.add(subtitleExtractConfig2);
        subtitlePackageSetting.put("SubtitleExtractConfigList",subtitleExtractConfigs);
        transcodeTemplate.put("PackageSetting",subtitlePackageSetting);
        return transcodeTemplate;
    }
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret) throws ClientException {
        // 点播服务接入区域
        String regionId = "cn-shanghai";
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKeySecret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }
```

2. 发起多码率转码。

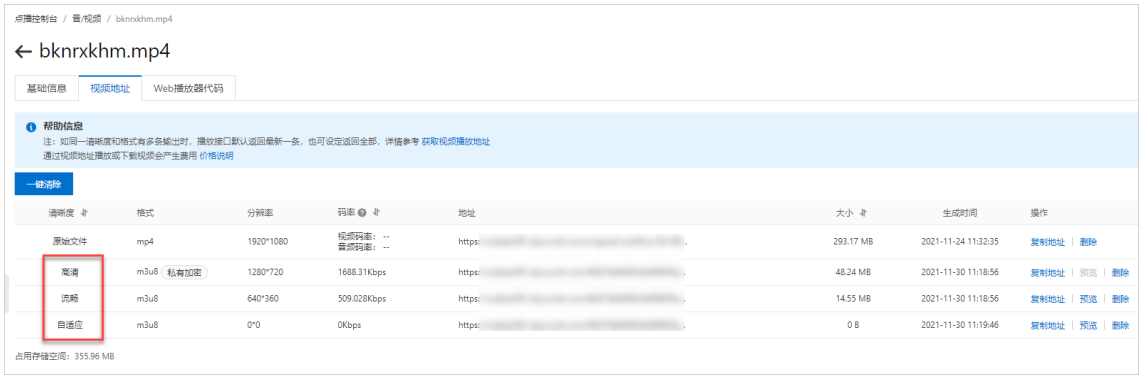
```
     public static void main(String[] args) {
       //regionId根据服务接入地址填写，例：接入服务在上海，则填cn-shanghai
       String regionId = "cn-shanghai";
        DefaultProfile profile = DefaultProfile.getProfile(regionId, "<accessKeyId>", "
```

```
<accessSecret>");
        IAcsClient client = new DefaultAcsClient(profile);
        String videoId = "76913816d****8e57e8c2952";
        String templateId = "4733b3a5****ae36ac22d34";
        try {
            SubmitTranscodeJobsResponse response = submitTranscodeJobs(client,videoId,t
emplateId);
        } catch (ServerException e) {
            e.printStackTrace();
        } catch (ClientException e) {
            System.out.println("ErrCode:" + e.getErrCode());
            System.out.println("ErrMsg:" + e.getErrMsg());
            System.out.println("RequestId:" + e.getRequestId());
        }
    }
    /**
     * 提起转码作业
     * 传入视频ID和模板组ID
     *
     * @param client
     * @param videoId
     * @param templateGroupId
     * @return
     * @throws Exception
     */
    public static SubmitTranscodeJobsResponse submitTranscodeJobs(DefaultAcsClient clie
nt, String videoId, String templateGroupId) throws Exception {
        SubmitTranscodeJobsRequest request = new SubmitTranscodeJobsRequest();
        //需要转码的视频ID
        request.setVideoId(videoId);
        //转码模板组ID
        request.setTemplateGroupId(templateGroupId);
        //设置转码优先级(默认级别为6，数字越大，级别越高)，范围是[1-10]
        request.setPriority("8");
        JSONObject overrideParams = buildOverrideParams();
        //覆盖参数
        request.setOverrideParams(overrideParams.toJSONString());
        return client.getAcsResponse(request);
    }
    //以下为打包字幕替换参数构建示例
    public static JSONObject buildOverrideParams() {
        JSONObject overrideParams = new JSONObject();
        //打包字幕替换设置
        JSONObject packageSubtitleSetting = new JSONObject();
        JSONArray packageSubtitleList = new JSONArray();
        JSONObject packageSubtitle = new JSONObject();
      //需要替换的打包模板组中字幕模板的ID
        packageSubtitle.put("SubtitlePackageTemplateId", "69fa6ee58****e8492c76168****"
);
      //Language参数仅用于检索需要替换的字幕文件，而Language本身不会被替换，不存在的字幕语言不能
替换。
        packageSubtitle.put("Language", "cn");
        //字幕文件的OSS地址(不支持https地址)，字幕文件需要与视频在同一个存储区域。
        packageSubtitle.put("SubtitleUrl", "http://outin-bfefbb9****3e1c7426.oss-cn-sha
nghai.aliyuncs.com/subtitle/043956117D0C475EAB0CE8C4E7294221****.vtt");
```

```
        packageSubtitleList.add(packageSubtitle);
        packageSubtitleSetting.put("PackageSubtitleList", packageSubtitleList);
        overrideParams.put("PackageSubtitleSetting", packageSubtitleSetting);
        return overrideParams;
    }
```

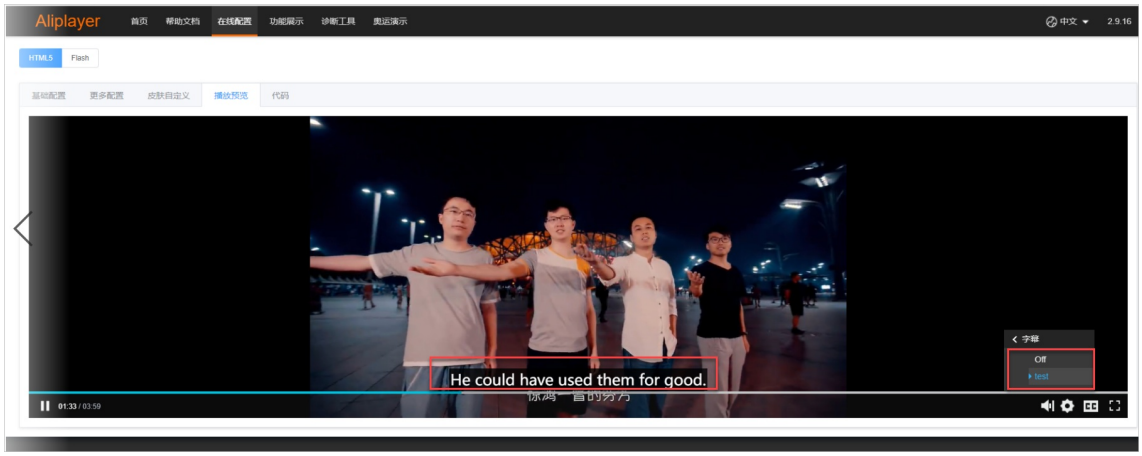
3. 效果展示。
   - 多码率打包后码流展示：（2路转码流，1路自适应流）

     在点播控制台，选择**媒资库 > 音/视频**，单击目标视频（上一步经过媒体处理的视频）的**管理**，选择**视频地址**页签。



   - 播放效果展示：（使用上图中自适应的地址播放）

     在Aliplayer播放器的**基础配置**页，输入播放地址（使用上图中视频地址页中自适应的地址），在**播放预览**页查看播放效果。



2.设置视频封面

视频上传时，若指定了封面则会使用指定的图片作为封面，如果不指定，则会将一张视频的截图作为封面，上传完成后也可以对封面进行更新。

背景信息

点播上传的每一个视频都设置了封面图片，并提供了多种设置和修改视频封面的方法。如上传时指定了封面，则会使用指定的图片作为封面；如果不指定封面，点播会默认进行视频截图，并将一张视频截图作为封面。视频上传完成之后也可以对封面进行更新。

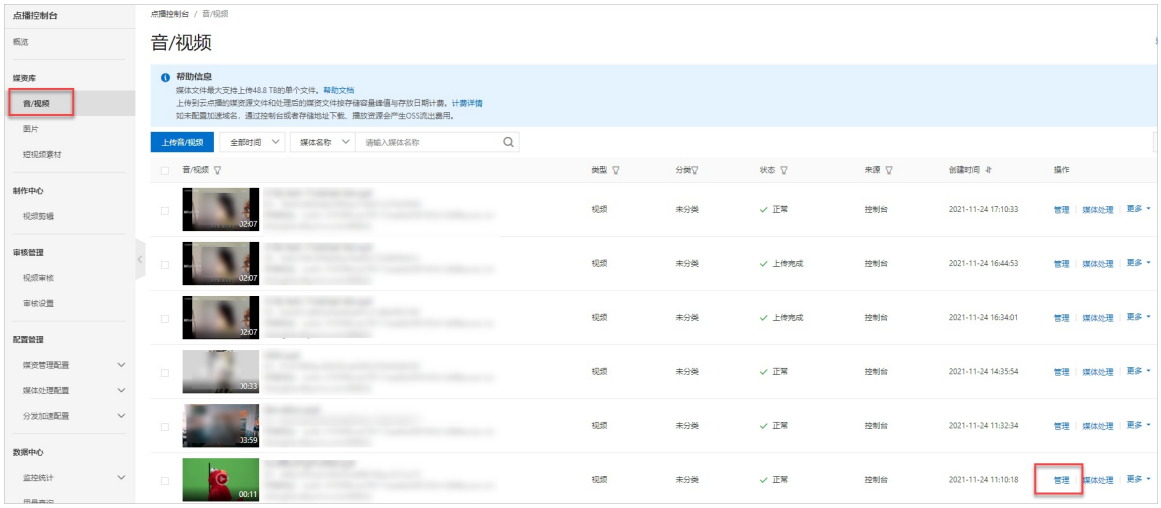
准备工作

- 您已经注册了阿里云账号并完成账号实名认证。注册地址请参见阿里云官网。注册指引请参见注册阿里云账号。实名认证指引请参见个人实名认证或企业实名认证。
- 调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。
- 如果使用智能封面，需要先申请开通智能封面服务。
- 如果服务地域已绑定域名，域名需要配置证书后，才可以设置封面，配置域名证书的具体操作，请参见HTTPS安全加速设置。如果服务地域没有绑定域名，可以直接设置封面。

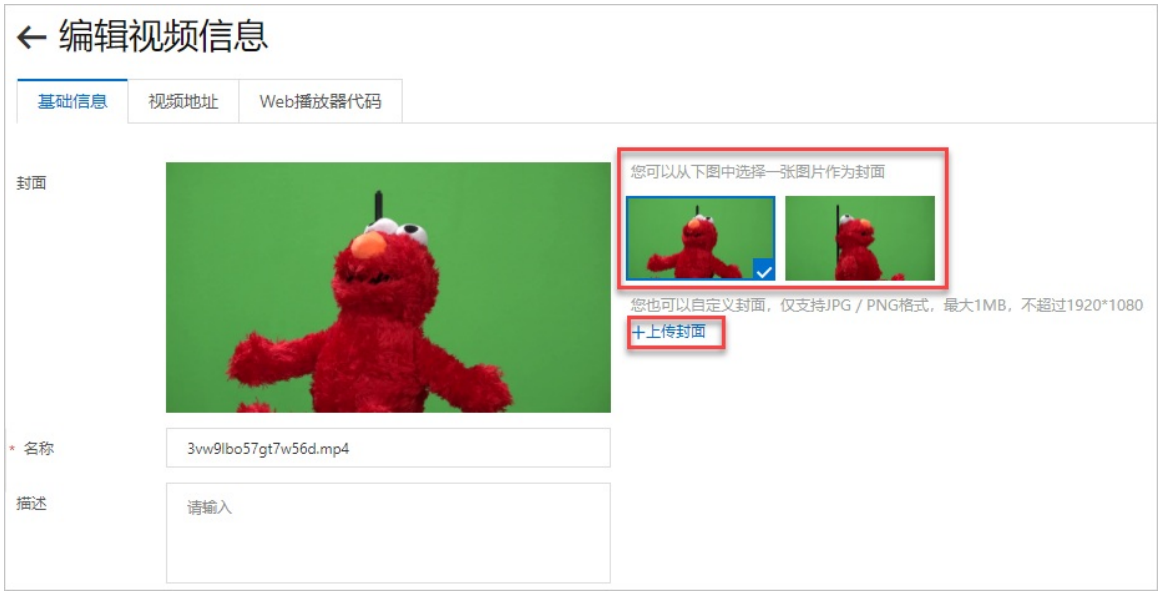
控制台设置封面

1. 登录视频点播控制台。
2. 在点播控制台左侧导航栏的**媒资库**区域，单击**音/视频**。
3. 选择需要修改封面的视频，单击**管理**。

> **说明**
>
> 视频上传成功后，会使用一张默认的视频截图作为封面截图。



4. 单击**编辑视频信息**，在页面可以从自动截取的多张图片里面选一张作为封面图。也可单击**上传封面**选择一张本地图片设置为封面图。



> 说明 上传的图片仅支持JPG、PNG格式，最大1MB，像素不超过1920Px × 1080Px。

使用API/SDK设置封面

视频点播提供了多种视频上传方式，可以在上传时指定封面图，如果不指定，视频上传完成后，点播会默认进行封面截图。可以设置封面的接口如下：

- 调用获取音视频上传地址和凭证接口时，可以传入CoverURL（自定义视频封面URL地址）参数指定封面图。
- 调用URL拉取上传接口时，在UploadMetadata里面传入CoverURL（自定义视频封面URL地址）参数指定封面图。
- 如果文件已经存储于OSS，可调用注册媒资信息接口注册媒资信息到点播，在注册媒资的时候可以在RegisterMetadata里面传入CoverURL（自定义视频封面URL地址）参数指定封面图。
- 如果视频已经上传完成，可以通过修改单个音视频信息和批量修改音视频信息接口传入CoverURL（自定义视频封面URL地址）参数指定封面图。

使用智能封面

智能封面服务支持对视频内容进行分析和理解，提取最能代表视频内容的多张截图作为封面备选图；也支持将视频内容的关键画面进行抽取，自动合成Gif作为动图封面。可访问视频AI-视网膜开始体验，正式使用需要到智能封面申请开通服务。效果如下：

3.如何选择转码类型

在面对不同行业用户丰富的转码场景需求时，视频点播针对多种业务场景提供了适应多场景化的转码处理方案，实现将用户定制化的场景需求进行抽象提取，最终以适用于其他点播用户的同样或类似的业务场景需求。

前提条件

- 您已经开通了视频点播服务。开通步骤请参见开通视频点播服务。
- 添加转码模板组，请参见普通转码模板设置。

名词解释

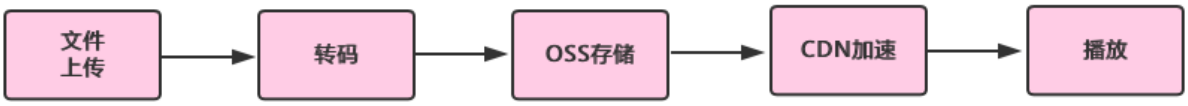
- 文件上传：统一指文件上传到点播，上传来源可以是上传SDK、直播录制、短视频SDK、OpenApi获取凭证后手动上传等。
- 转码处理：统一指针对上传文件（包括视频、音频等）按照指定的转码参数进行内容处理。
- 云剪辑：对已经上传到点播的视频进行在线剪辑，如拼接、截取等一系列操作。
- AI处理：对已经上传到点播的视频进行AI处理，如智能审核、内容分析（标签分析、语音文字识别等）、智能首图、新闻拆条等处理。
- CDN加速：指对内容进行全网分发，加快内容访问速度，提高用户体验。
- 不同规格视频：主要是指视频的分辨率、码率等编码参数不同的转码输出视频，这些不同规格视频可以适应不同的网络带宽环境。
- 转码后分发：视频上传完成后，源片经转码处理后，输出不同规格视频再经CDN加速分发，供终端播放。
- 不转码即分发：视频上传完成后，源片可经CDN加速播放，但不会触发任何转码处理流程。

普通转码

异步处理、延迟播放：通常视频上传到点播会先经过转码处理，生成不同规格的视频以适应不同的网络带宽环境及多终端处理，然后经过CDN加速分发，最终供终端播放。

使用方法：**异步处理、延迟播放**这类场景，用户只需要在转码设置进行配置，创建转码模板组并设置为默认模板组，然后上传视频即可，后续流程自动完成。

该场景的视频的处理流程如下图所示：



不转码即分发

快速分发：短视频不转码、实时播放

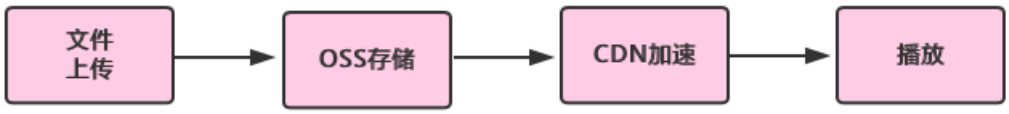
经由短视频SDK录制并上传的视频，这类视频的编码信息实际已经基本符合在网络带宽传输环境且能被各个终端兼容处理，因此用户可选择对这类视频无需再经由转码处理，而是直接通过CDN加速分发，不仅可快速响应播放请求，也可节约用户的转码成本。

> **说明** 目前仅以下格式支持不转码直接播放：MP4、FLV、M3U8、MP3、WEBM。

使用方法：如果没有该模板设置，用户只需在转码设置进行配置，将不转码即分发模板组激活并设置为默认模板组，然后上传视频即可。

> **说明** 用户开通点播服务，点播服务会自动提供该处理场景的模板，并设置成默认使用模板。

该场景的视频处理流程如下图所示：



4.直播转点播

直播转点播（直转点）是将直播流同步录制为点播视频，并支持媒资管理、媒体处理（转码及内容审核、智能首图等AI处理）、内容制作（云剪辑）、CDN分发加速等一系列操作，可配置工作流自动处理，也可通过API/SDK灵活触发。

准备工作

- 开通视频点播服务，请参见开通指引。
- 开通视频直播服务，请参见开通指引。
- 添加直转点录制配置，请参见录制存储至VOD。

上述准备工作完成后，即可开始进行接入。

> **说明** 下文中仅存储、仅合成模板组需联系点播进行激活。

名词解释

直转点，结合视频点播的转码、云剪辑、AI处理、事件通知等功能，可适应多种业务场景。

名词解释：

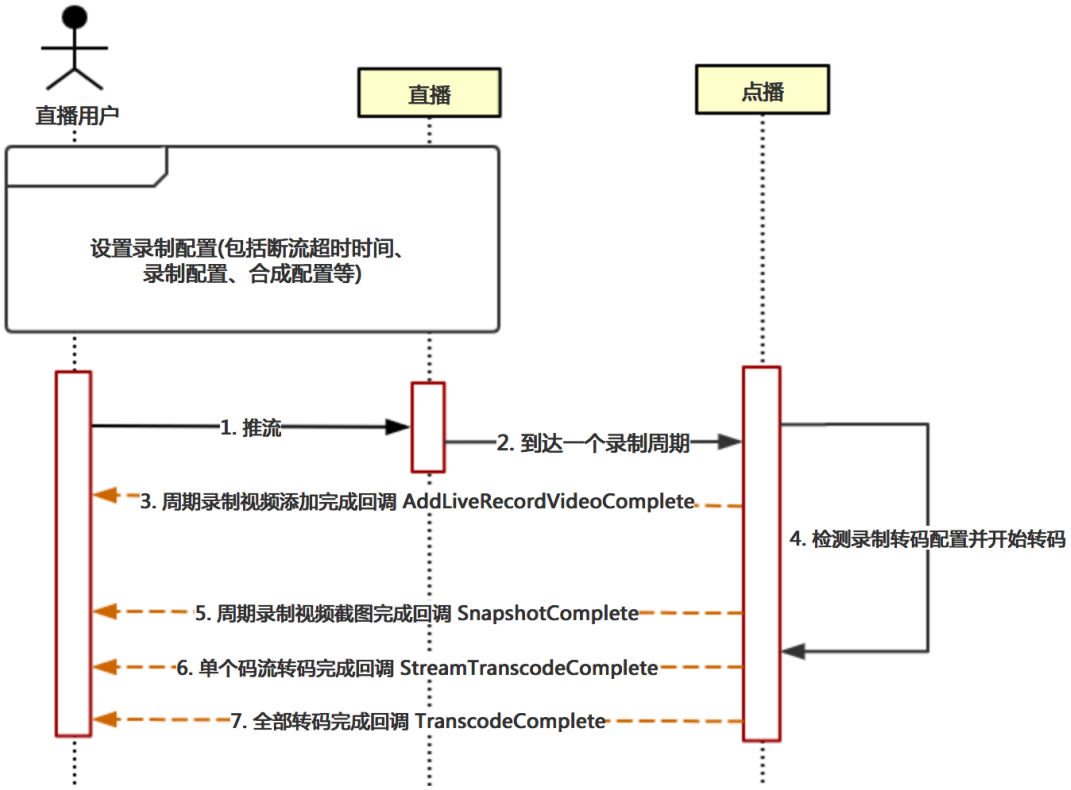
- 录制转码模板组：直播录制到点播同时，点播会使用该模板组对视频进行转码操作。
- 合成转码模板组：多个录制视频进行自动合成时，点播会使用该模板组对视频进行合成+转码操作。
- 仅存储：对直播内容进行录制后，不进行任何后续操作。
- 仅合成：对直播内容进行合成后，不进行任何后续操作。
- 直播录制周期：直播录制到点播的周期，如一场直播3个小时，如果需要在直播过程中就可提供已录制内容的点播服务，则可设置录制周期为1小时，即直播1小时过后，点播可提供前一小时内容的点播观看服务。

实践1

直播录制 + 自动转码 + CDN加速

直播录制后快速将录制文件进行转码和CDN加速，供用户进行点播播放，适用于大部分直播场景（不需要对内容进行二次加工）

流程如下：



1. 客户进行直播推流
2. 推流达到一个录制周期，则会自动将录制文件添加到点播系统。
3. 点播记录完成后，会生成点播系统的唯一视频ID，并将该视频信息回调给用户，即AddLiveRecordVideoComplete通知，并附带直播相关的DomainName、AppName、StreamName信息。客户收到回调后，需记录该视频信息，并以该VideoId作为索引进行后续视频状态更新。
4. 点播系统检测用户录制配置中的录制转码组ID（该转码组中含有具体码流转码任务），进行对应的转码操作。
5. 截图完成、单个码流转码完成、全部码流转码完成时，会给用户进行回调（回调顺序无时序），用户需根据回调信息中的VideoId进行视频状态更新。转码完成后，即可进行后续的播放操作（转码回调信息中含播放地址或后续通过GetPlayInfo接口根据VideoId进行播放地址获取，该播放地址已经经过CDN加速）。

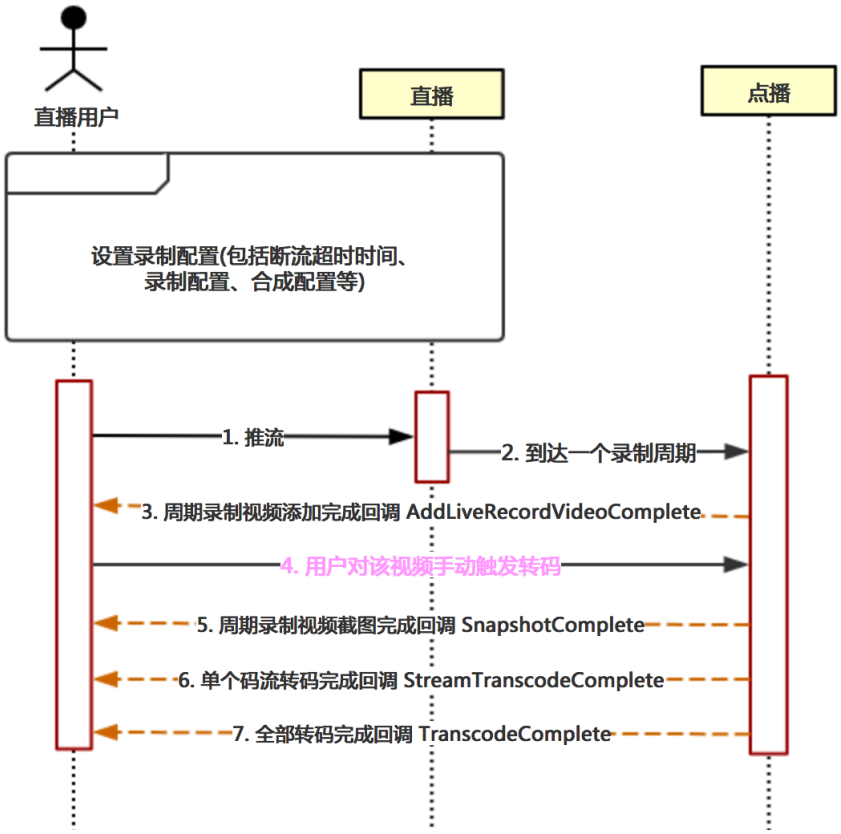
实践2

直播录制 + 仅存储到点播 + 手动发起转码 + CDN加速

部分用户希望将直播录制的视频仅先存储到点播，且先不进行后续的转码操作，则可在新建录制配置时，将录制转码组选择为仅存储模板组（仅存储模板组需联系点播进行激活）。如果后续希望对视频进行转码，则可进行手动触发转码操作。同时，可配合点播云剪辑功能进行使用，效果更佳。

适用场景：直播完成后，需要对内容进行二次加工，如体育赛事、游戏直播剪辑等，后续由用户主动发起转码和CDN加速全流程（转码完成后点播会自动对输出文件进行CDN加速）

流程如下：



1. 客户进行直播推流。
2. 推流达到一个录制周期，则会自动将录制文件添加到点播系统。
3. 点播记录完成后，会生成点播系统的唯一视频ID，并将该视频信息回调给用户，即AddLiveRecordVideoComplete通知，并附带直播相关的DomainName、AppName、StreamName信息。客户收到回调后，需记录该视频信息，并以该VideoId作为索引进行后续视频状态更新。
4. 点播系统检测用户录制配置中的录制转码组ID（此时为仅存储模板组），则点播系统不进行后续转码操作。
5. 用户对该视频手动触发转码操作（调用转码任务API），在此之前可进行云剪辑等操作。
6. 截图完成、单个码流转码完成、全部码流转码完成时，会给用户进行回调（回调顺序无时序），用户需根据回调信息中的VideoId进行视频状态更新。转码完成后，即可进行后续的播放操作（转码回调信息中含播放地址或后续通过GetPlayInfo接口根据VideoId进行播放地址获取，该播放地址已经经过CDN加速）。

实践3

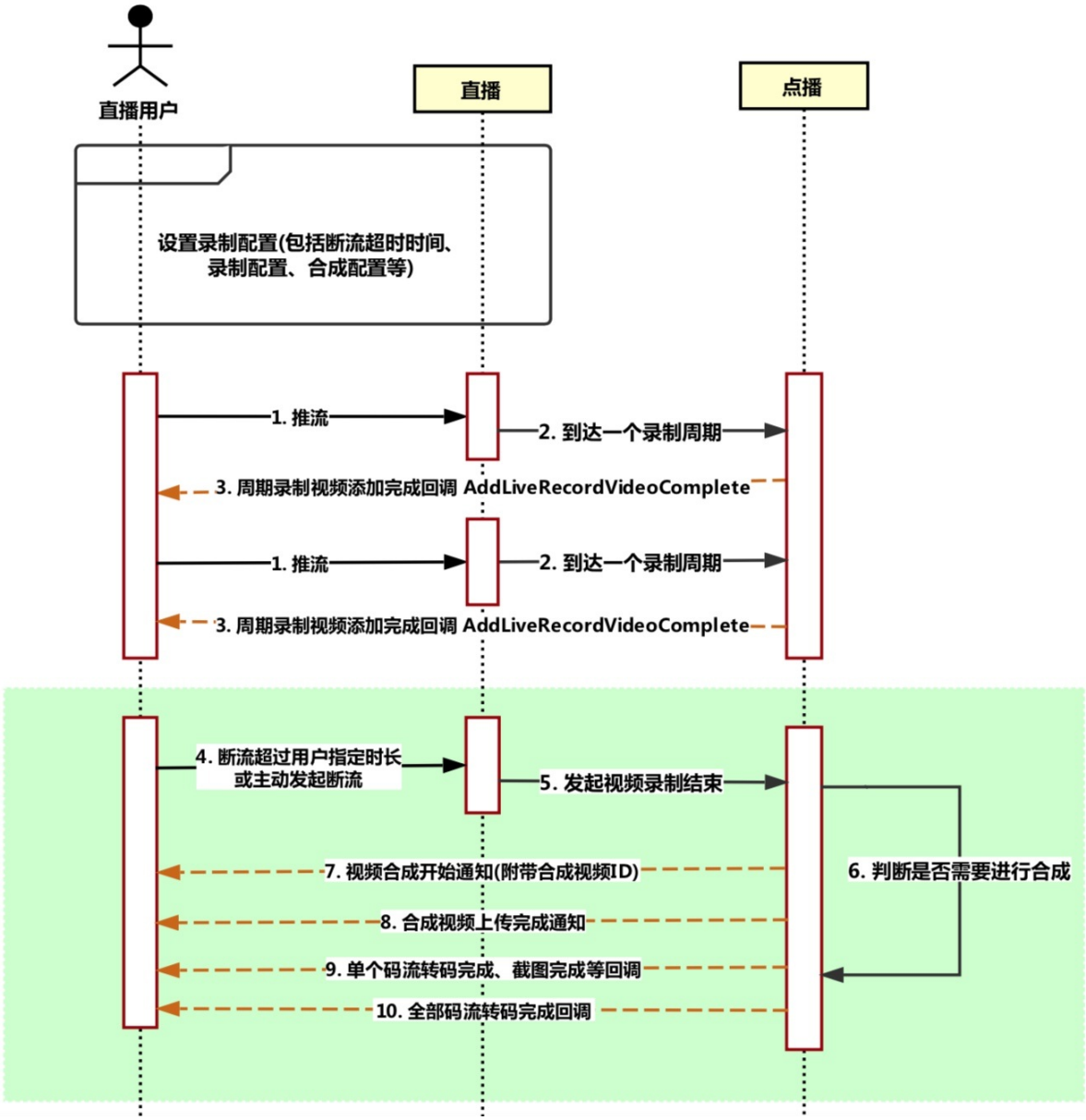
直播录制 + 多周期视频自动合并

部分用户希望将自己录制周期生成的多个文件（如录制周期为20分钟，一次直播1个小时，则会生成三个视频）进行视频合成，再将合成后的视频进行处理，点播提供了自动合成的功能。可在新建直播录制配置时，将自动合成开关打开，并且配置进行合成时所使用的合成转码模板组（也可配置为仅合成，后续由用户触发转码，与前两节所述区别相同）。点播会在用户断流超过指定时间（可由直播进行配置断流超时时间）后，进行本次直播的视频自动合成并根据转码配置进行后续操作。

合成 + 自动转码

适用场景：一场直播完成后，需要对所有录制周期内的分段进行自动合并，同时发起转码等全流程。如体育赛事、教育多节授课合并等。

流程如下：



1. 客户进行直播推流。
2. 推流达到一个录制周期，则会自动将录制文件添加到点播系统。
3. 点播记录完成后，会生成点播系统的唯一视频ID，并将该视频信息回调给用户，即AddLiveRecordVideoComplete通知，并附带直播相关的DomainName、AppName、StreamName信息。客户收到回调后，需记录该视频信息，并以该VideoId作为索引进行后续视频状态更新。
4. 客户断流超时或主动触发断流。
5. 点播收到直播发送的本次直播结束消息。
6. 点播系统检测用户录制配置中的合成配置，判断是否需要发起自动合成。如果需要，则按照录制配置中的合成转码组进行合成和转码。
7. 视频开始合成，点播会生成一个合成后视频的唯一视频ID，并将该视频信息回调给用户，即

LiveRecordVideoComposeStart通知，并附带直播相关的DomainName、AppName、StreamName信息。客户收到回调后，需记录该视频信息，并以该VideoId作为索引进行后续视频状态更新。

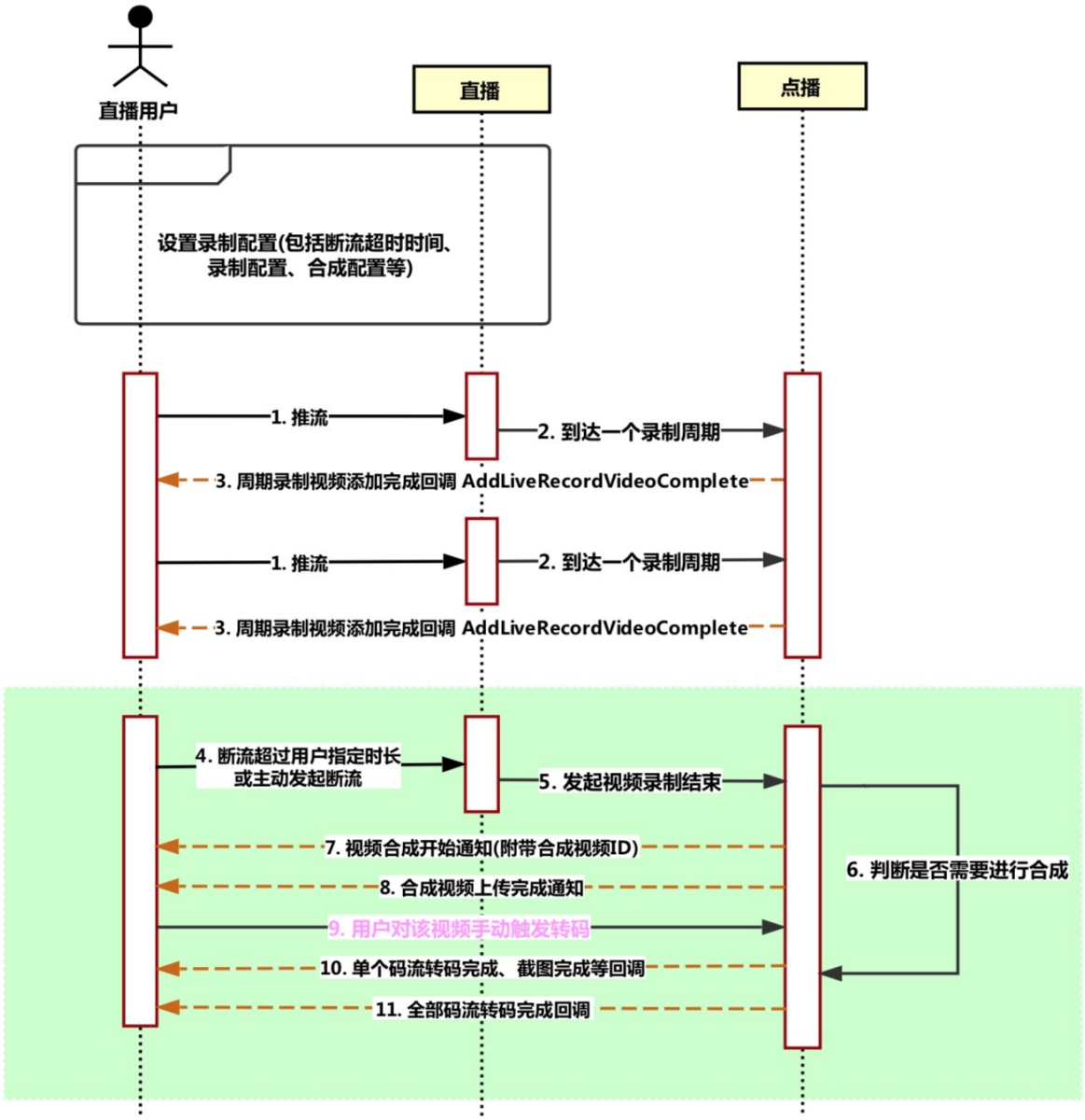
8. 视频源文件合成完成后，点播会将该状态回调给用户，即FileUploadComplete通知。
9. 截图完成、单个码流转码完成、全部码流转码完成时，会给用户进行回调（回调顺序无时序），用户需根据回调信息中的VideoId进行视频状态更新。转码完成后，即可进行后续的播放操作（转码回调信息中含播放地址或后续通过GetPlayInfo接口根据VideoId进行播放地址获取，该播放地址已经经过CDN加速）。

合成 + 手动发起转码

可在新建直播录制配置时，将自动合成开关打开，并且配置进行合成时所使用的合成转码模板组（本情况配置成仅合成，后续由用户触发转码）。

适用场景：一场直播完成后，需要对所有录制周期内的分段进行自动合并，合并后文件不做任何处理，由用户进行二次加工，如云剪辑等，后续再进行手动转码，如视频内嵌广告、体育赛事空挡部分内容剪切等。

流程如下：



1. 客户进行直播推流。
2. 推流达到一个录制周期，则会自动将录制文件添加到点播系统。
3. 点播记录完成后，会生成点播系统的唯一视频ID，并将该视频信息回调给用户，即AddLiveRecordVideoComplete通知，并附带直播相关的DomainName、AppName、StreamName信息。客户收到回调后，需记录该视频信息，并以该VideoId作为索引进行后续视频状态更新。
4. 客户断流超时或主动触发断流。
5. 点播收到直播发送的本次直播结束消息。
6. 点播系统检测用户录制配置中的合成配置，判断是否需要发起自动合成。如果需要，则按照录制配置中的合成转码组进行合成和转码，由于本场景配置的仅合成，则点播系统不会自动发起转码。
7. 视频开始合成，点播会生成一个合成后视频的唯一视频ID，并将该视频信息回调给用户，即LiveRecordVideoComposeStart通知，并附带直播相关的DomainName、AppName、StreamName信息。客户收到回调后，需记录该视频信息，并以该VideoId作为索引进行后续视频状态更新。
8. 视频源文件合成完成后，点播会将该状态回调给用户，即FileUploadComplete通知。此时，代表本次录制合成的源文件已经处于正常就绪状态，用户可对该视频进行后续的转码触发等操作。
9. 用户对该视频手动触发转码操作（调用转码任务API），在此之前可进行云剪辑等操作。
10. 截图完成、单个码流转码完成、全部码流转码完成时，会给用户进行回调（回调顺序无时序），用户需根据回调信息中的VideoId进行视频状态更新。转码完成后，即可进行后续的播放操作（转码回调信息中含播放地址或后续通过GetPlayInfo接口根据VideoId进行播放地址获取，该播放地址已经经过CDN加速）。

5.点播资源迁移

视频点播支持将第三方平台的视频资源迁移到视频点播，也支持阿里云账号间的视频资源迁移。本文介绍上述资源的迁移方法。

迁移目标及方法

视频点播支持以下视频资源的迁移：

- 第三方资源：将存储于个人网站、云端的视频等数据资源迁移到视频点播。
- 跨阿里云账号资源：将视频点播OSS Bucket中的视频迁移到另一个账号中的OSS Bucket中。

资源迁移的方法、工具及流程如下：

迁移目标	迁移方法	迁移工具	迁移流程
第三方资源	公网下载公网上传	• 视频点播上传SDK（推荐） • 视频点播服务端API	 1. 准备待迁移资源下载地址。 2. 搭建上传服务（上传SDK或服务端API）。 3. 上传视频资源。 4. （可选）整理资源关系。
跨阿里云账号资源	通过ECS内网上传 **注意** 适用本方法的前提条件如下： • 用户的视频源文件存储在OSS中。 • 用户的点播账号和OSS账号为同一阿里云账号。 • 用户的ECS与视频源文件在一个地域。	• 视频点播上传SDK（推荐） • 视频点播服务端API	 1. 获取视频资源地址并修改为OSS内网地址。 2. 搭建上传服务（上传SDK或服务端API）。 3. 上传视频资源。

迁移目标	迁移方法	迁移工具	迁移流程
跨阿里云账号资源	公网上传	• 视频点播上传SDK（推荐） • 视频点播服务端API	1. 从视频点播获取视频资源地址。 2. 搭建上传服务（上传SDK或服务端API）。 3. 上传视频资源。

准备工作

- 您已经注册了阿里云账号并完成账号实名认证。注册地址请参见阿里云官网。注册指引请参见注册阿里云账号。实名认证指引请参见个人实名认证或企业实名认证。
- 调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

> **说明** 如果使用RAM用户完成迁移，请先创建RAM用户再获取AccessKey。详细步骤请参见RAM子账号访问。

迁移第三方资源

第三方资源指存储于个人网站、云端的视频资源。请参考以下步骤将第三方资源迁移到阿里云视频点播：

1. 准备需要迁移的资源，用户可以根据自身的数据处理习惯保存迁移文件下载地址。

   > **说明** 用户需要准备所有迁移文件的下载地址，需要注意下载地址如果存在鉴权，尽量保证鉴权时间够长，避免下载时文件下载地址失效。

2. 搭建上传服务。

   代码示例请参考上传服务搭建及代码示例。推荐使用视频点播上传SDK完成迁移。

   > **说明** 需要整理迁移前后资源关系的用户，上传程序应当记录源文件地址和上传后VideoId的关联，如：在上传时打印日志形式记录、上传时源文件地址写入视频媒资信息记录等，具体如何记录依赖于用户场景。如果使用URL批量拉取，该接口同步返回信息中有对应上传源文件地址，有需求可以合理使用。

3. 执行步骤2搭建的上传代码，上传视频至视频点播。
4. （可选）整理点播资源关系。

   顺利完成数据迁移，如有整理资源的需求，请根据上传时记录的迁移源文件地址和上传到点播后VideoId之间的对应关系进行整理。

迁移跨账号资源（公网上传）

如果您希望在更多地区复制视频点播解决方案的场景，可将现有视频点播存储的内容迁移到不同的帐户。使用公网上传的迁移步骤如下：

> **说明** 本文以调用接口获取源文件地址为例介绍迁移步骤。您也可以通过视频点播控制台获取源文件地址，详细操作请参见媒资数据导出。

1. 调用点播搜索媒体信息接口筛选出要迁移的视频VideoId。
2. 将步骤1获取的VideoId作为入参，调用获取源文件信息接口获取所有需要迁移视频的源文件地址并保存。

   > **说明** 如果用户需要使用转码流作为迁移源文件下载地址，可使用点播服务，更多操作，请参见媒资管理。

3. 搭建上传服务。

   代码示例请参考上传服务搭建及代码示例。推荐使用视频点播上传SDK完成迁移。

4. 执行步骤3搭建的上传代码，上传视频至视频点播。

迁移跨账号资源（内网上传）

如果您希望在更多地区复制视频点播解决方案的场景，可将现有视频点播存储在OSS的内容迁移到不同的帐户。迁移过程中可使用内网上传，但使用内网上传必须满足以下前提条件：

- 用户的视频源文件存储在OSS中。
- 用户的点播账号和OSS账号为同一阿里云账号。
- 用户的ECS与视频源文件在一个地域。

如上述条件均满足，可参考以下步骤完成迁移：

1. 调用视频点播服务端获取源文件信息接口获取文件OSS地址。

   > **说明** 将请求参数 `OutputType` 的值设为 `oss` 。

2. 将获取到的OSS地址修改为内网地址，修改方法为在OSS地域后增加 `-internal` 。以下是修改前后的地址示例：

获取的OSS回源地址	修改后的内网地址
outin-67870fd5b29****98a3900163e1c35d5.oss-cn-shanghai.aliyuncs.com/customerTrans/2a13b91506f9158f****7317f4a9d4c9/30f24681-1718d5c6237-**4bd.mp4	outin-67870fd5b29****98a3900163e1c35d5.oss-cn-shanghai-internal.aliyuncs.com/customerTrans/2a13b91506f9158f****7317f4a9d4c9/30f24681-1718d5c6237-**4bd.mp4

   更多关于通过内网访问OSS资源的信息，请参考通过OSS内网地址访问OSS资源。

3. 搭建上传服务。

   > **说明** 针对跨阿里云账号迁移的场景，如需通过内网上传，建议您将上传服务部署在与点播OSS Bucket（中国内地默认上海）同地域的ECS。按照此方式部署后，使用上传SDK上传时指定参数 `regionId` 为上述地域，上传过程会自动走内网。

   代码示例请参考上传服务搭建及代码示例。推荐使用视频点播上传SDK完成迁移。

4. 执行步骤3搭建的上传代码，上传视频至视频点播。

上传服务搭建及代码示例

无论是第三方资源还是阿里云账号间资源迁移，都需要下载原视频再通过上传工具上传到视频点播。视频点播提供上传SDK及服务端接口两种上传工具。您的上传服务可通过上传SDK或服务端接口实现。使用上传SDK为同步上传，更具实效性，故推荐使用。使用服务端接口为异步上传，实时性相对较差。

搭建上传服务步骤

1. 集成上传SDK或服务端SDK。

   根据您使用的语言，集成上传SDK请参考上传SDK，集成服务端SDK请参考服务端SDK。

2. 编写上传服务代码。如何编写请参考下述代码示例。

上传SDK代码示例（推荐）

以下代码仅以Java上传SDK为例展示。更多语言的上传SDK及操作示例，请参考上传SDK。

```
import com.aliyun.vod.upload.impl.UploadVideoImpl;
import com.aliyun.vod.upload.req.UploadStreamRequest;
import com.aliyun.vod.upload.resp.UploadStreamResponse;
import java.io.*;
import java.net.URL;
/**
 * 使用上传SDK进行视频文件上传
 */
public class UploadStreamDemo {
    /**
     * 流式上传接口
     *
     * @param accessKeyId
     * @param accessKeySecret
     * @param title
     * @param fileName
     * @param inputStream
     */
    private static void testUploadStream(String accessKeyId, String accessKeySecret, String
title, String fileName, InputStream inputStream) {
        UploadStreamRequest request = new UploadStreamRequest(accessKeyId, accessKeySecret,
title, fileName, inputStream);
        /* 自定义消息回调设置，参数说明请参见基本数据类型 */
        //request.setUserData(""{\"Extend\":{\"test\":\"www\",\"localId\":\"xxxx\"},\"Messa
geCallback\":{\"CallbackURL\":\"http://example.aliyundoc.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)，如http://****.example.com/image_01.jpg*/
        //request.setCoverURL("<Your CoverURL>");
        /* 模板组ID(可选) */
        //request.setTemplateGroupId("8c4792cbc8694e****fd5330e56a33d");
        /* 工作流ID(可选) */
        //request.setWorkflowId("d4430d07361f****1339577859b0177b");
        /* 存储区域(可选) */
        //request.setStorageLocation("outin-20170323****266-5sejdln9o.oss-cn-shanghai.aliyu
ncs.com");
        /* 点播服务接入点 */
```

```
        request.setApiRegionId("cn-shanghai");
        /* ECS部署区域*/
        // request.setEcsRegionId("cn-shanghai");
        UploadVideoImpl uploader = new UploadVideoImpl();
        UploadStreamResponse response = uploader.uploadStream(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");
        }
    }
    public static void main(String[] args) {
        /**
         * 用户可自行添加url数据源,并传入视频媒资信息，上传视频资源
         */
        InputStream inputStream = null;
        //您的视频地址。如http://example.aliyundoc.com/video/****.mp4
        String url = "<Your File URL>";
        try {
            inputStream = new URL(url).openStream();
        } catch (IOException e) {
            e.printStackTrace();
        }
        //以下参数重的AccessKey ID, AccessKey Secret提前准备好的AccessKey信息。<Your Video Title>为视频标题。<Your Video with File Extension>为含文件扩展名的视频，如video-1.mp4。
        testUploadStream("<AccessKey Id>", "<AccessKey Secret>", "<Your Video Title>", "<Your Video with File Extension>", inputStream);
    }
}
```

服务端API代码示例（不推荐）

视频点播的URL拉取上传接口支持将迁移资源文件下载地址逐个传入，执行完成上传任务提交也可以根据个人需求在上传时传入视频的媒资信息，如：标题、分类、描述等。但上传过程为异步，故一般不推荐使用。

示例代码如下：

```
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.vod.model.v20170321.UploadMediaByURLRequest;
import com.aliyuncs.vod.model.v20170321.UploadMediaByURLResponse;
import java.net.URLEncoder;
/**
 * URL异步批量拉取
 * 使用前建议阅读该接口文档，了解其优势及缺陷
 * 请参见URL拉取上传
```

```
 */
public class UploadMediaByURLDemo {
    public static DefaultAcsClient initVodClient(String accessKeyId, String accessKeySecret
) throws ClientException {
        String regionId = "cn-shanghai";  // 点播服务接入区域
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKey
Secret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }
    /**
     * URL批量拉取上传
     * @param client 发送请求客户端
     * @return UploadMediaByURLResponse URL批量拉取上传响应数据
     * @throws Exception
     */
    public static UploadMediaByURLResponse uploadMediaByURL(DefaultAcsClient client,String
url) throws Exception{
        UploadMediaByURLRequest request = new UploadMediaByURLRequest();
        //String url = "http://xxxx.mp4";
        String encodeUrl = URLEncoder.encode(url, "UTF-8");
        request.setUploadURLs(encodeUrl);
        //上传视频元数据信息，为JSON字符串
        JSONObject uploadMetadata = new JSONObject();
        uploadMetadata.put("SourceUrl", encodeUrl);
        uploadMetadata.put("Title", "upload by url sample");
        JSONArray uploadMetadataList = new JSONArray();
        uploadMetadataList.add(uploadMetadata);
        request.setUploadMetadatas(uploadMetadataList.toJSONString());
        JSONObject userData = new JSONObject();
    //回调信息设置，点播控制台勾选视频上传能完成事件，视频成功上传后会发送通知消息到该URL
        JSONObject messageCallback = new JSONObject();
        messageCallback.put("CallbackURL", "<Your Callback URL>");//用户接受回调消息的URL， 如
http://example.com/callback****
        messageCallback.put("CallbackType", "http");
        userData.put("MessageCallback", messageCallback.toJSONString());
        JSONObject extend = new JSONObject();//此处写入的消息会写入回调消息中，用于数据透传
        extend.put("MyId", "user-defined-id");
        userData.put("Extend", extend.toJSONString());
        request.setUserData(userData.toJSONString());
        return client.getAcsResponse(request);
    }
    // 请求示例
    public static void main(String[] argv) throws ClientException {
        DefaultAcsClient client = initVodClient("<Your AccessKeyId>", "<Your AccessKeySecre
t>");
        UploadMediaByURLResponse response = new UploadMediaByURLResponse();
        try {
    //此处用户可修改成读取，事前准备的迁移资源文本
            response = uploadMediaByURL(client,"<Your File URL>");//文件中的上传地址,如http:/
/example.aliyundoc.com.****.mp4
            System.out.print("UploadJobs = " + JSON.toJSONString(response.getUploadJobs())
+ "\n");
        } catch (Exception e) {
```

```
            System.out.print("ErrorMessage = " + e.getLocalizedMessage());
        }
        System.out.print("RequestId = " + response.getRequestId() + "\n");
    }
}
```

6.使用阿里云播放器实现全屏秒播

您可以通过本文了解如何使用阿里云播放器实现全屏秒播。

背景信息

使用阿里云播放器实现全屏秒播可以通过完成以下两个步骤实现。

1. 首帧图和播放的首帧画面一致。
2. 播放器预加载。

> 说明
> - 目前阿里云播放器只支持通过Android端和iOS端实现全屏秒播。
> - 使用以上两种方式在WIFI环境下可以做到平均300毫秒左右的起播速度。

首帧图和播放的首帧画面一致

1. 集成阿里云播放器SDK。
   - 集成Android播放器SDK。具体操作请参见快速集成。
   - 集成iOS端放器SDK。具体操作请参见快速集成。
2. 在视频点播控制台上设置视频封面。具体操作请参见设置视频封面。
3. 选择阿里云播放器播放视频。

   显示视频的时候先显示封面，然后再播放视频。如果保证封面图和首帧画面一致，则用户感觉不到是封面的存在，这样就造成了视频秒开极快的假象。

> 说明
> - 在快速滑动的时候仅仅只请求封面图。
> - 在用户滑动到一半的时候，展现预先下载的一个封面图。

播放器预加载

播放器SDK提供预加载功能，是对本地缓存（边播边缓存）功能的升级，通过设置视频缓存的内存占用大小，更能提升视频的起播速度。

使用说明

- 目前支持MP4、MP3、FLV、HLS（单码率视频流）等单个媒体文件的加载。
- 仅支持UrlSource播放方式播放视频的预加载，暂不支持VidAuth、VidSts方式播放视频的预加载。

功能实现

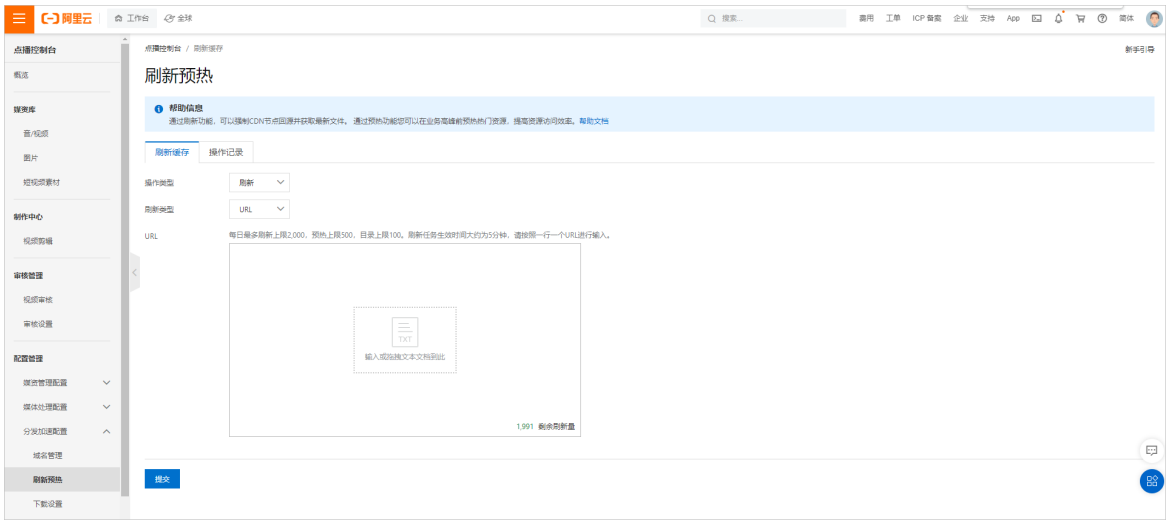
- Android播放器具体操作请参见进阶功能。
- iOS播放器具体操作请参见进阶功能。

可选：启动刷新预热

启动刷新预热可以帮助播放器实现预加载功能，更快速的实现全屏秒播。

1. 登录视频点播控制台。

2. 在点播控制台左侧的导航栏选择**配置管理**。
3. 单击**分发加速配置 > 刷新预热**，进入刷新预热页面。
4. 在操作类型中选择**预热**，填写需要预热的视频URL。



说明 由于刷新预热有一定的回源流量成本，可根据视频热度进行预热，将视频主动预热在阿里云各个节点上。

7.通过计算资源用量选择计费方式

您可以根据不同的用量选择不同的计费方式，可以让您节省大量的成本。本文为您介绍如何计算用量以及选择计费方式。

案例

小王是一名创业者，带领团队自主研发了一个在线教育平台。他希望把视频存储在阿里云上，视频存量约2000个，占用存储空间近2 T，每月预计新增视频200个，并新增存储约200 G，课程视频的时长集中在25~35分钟，并且按照不同课程进行分类管理。为了保障各端的观看效果，计划为用户提供标清480P和高清720P两种清晰度。目前已有用户500人左右，每日平均视频观看次数2000次，在移动端和PC端观看次数比例大致为3:1。小王想要了解自己视频资源用量情况以及如何选择计费形式，性价比才能最优。

案例分析

从案例背景中可以看出小王需要用到的点播服务功能包括：视频上传与存储、视频转码。其中，视频播放产生的下行流量消耗与视频的观看次数、视频时长、码率大小有关。在计算用量前假设：

- 视频时长：根据课程视频的时长特点（集中在25~35分钟），视频时长按平均值30分钟计算。
- 清晰度分布：根据移动端和PC端的观看特点，移动端默认采用标清480P的清晰度，PC端默认采用高清720P的清晰度（忽略用户自行切换清晰度的情况）。
- 存储大小：案例中包含存量、增量两类视频。当进行转码操作后，原视频转码产生两种清晰度视频，因此，实际存储占用按3倍于原始视频大小进行计算。（忽略输出视频分辨率和码率对视频体积的影响，如果转码后的视频分辨率和码率高于原视频，则实际体积会变大。）

如何计算资源用量？

本案例计算得出的用量和费用都为预估值。

从案例中可以看出小王每年的资源用量消费（暂不考虑业务快速增长的情况）如下：

项目	公式	用量
存储用量	存储用量：存量视频+增量视频-免费存储	• 存量视频：2048×3=6144 GB • 增量视频：200×3×12=7200 GB 那么，年平均存储用量：6144+7200=13344 GB。
流量用量	流量用量：（观看次数×视频时长×视频码率大小）/8（byte）	• 移动端标清流量：1500（次数）×0.0009（标清码率：Gbps）×3（平均时长：分钟）×60（分/秒）/8（byte）=303.75GB/天 • PC端高清流量：500（次数）×0.0015（高清码率：Gbps）×30（平均时长：分钟）×60（分/秒）/8（byte）=168.75 GB/天 那么，年平均流量用量：（303.75+168.75）×365=172462.5 GB。
转码时长	转码时长：由转码输出视频的规格决定	• 存量标清视频：2000个×30分钟=60000分钟 • 存量高清视频：2000个×30分钟=60000分钟 • 增量标清视频：200个×30分钟×12=72000分钟 • 增量高清视频：200个×30分钟×12=72000分钟

如何选择计费方式？

本文将通过按量付费和预付费资源包两种计费方式，分别为小王计算平均每年的费用。

- 单价的变动请以阿里云官网发布的数据为准，详情请参见：视频点播价格详情页。
- 资源包价格及购买，请单击资源包。

计费方式	计费公式	用量费用
按量付费	存储费用：存储单价x存储用量	• 存储单价：0.12元/GB/月 • 存储量：13344 GB/年 那么，年平均存储费用：0.12元/GB/月×13344 GB×12月=**19215.36**元/年。
	流量费用：流量单价x流量用量	• 流量单价：0.24元/GB • 流量用量：172462.5 GB 那么，年平均流量费用：0.24元/GB×172462.5 GB=**41391**元/年。

	转码费用：转码单价x转码时长	• 转码单价
		  ○ 标清（LD）（640×480）：0.0217元/分钟
		  ○ 高清（SD）（1280×720）：0.0326元/分钟
		• 转码时长
		  ○ 高清视频：60000+72000=132000分钟
		  ○ 标清视频：60000+72000=132000分钟
		那么，年平均转码费用：0.0217元/分钟x132000分+0.0326元/分钟x132000分=**7167.6**元/年。

预付费资源包	存储费用：存储包平均每年的费用	• 存储用量：13344 GB=14 TB • 存储包规格：建议小王选2个5 TB/年和4个1 T/年的存储包，共计13950元。
	流量费用：流量包平均每年的费用	• 流量用量：172462.5 GB=169 TB/年 • 流量包规格：建议小王选1个200 T/年的流量包，共计21800元。

	转码费用：转码资源包支持H.264的LD、SD、HD三种转码规格的抵扣，资源包时长指H.264LD的可抵扣时长，LD、SD、HD按1:1.5:3的比例抵扣。	• 转码时长：
		  ○ 高清视频：132000分钟/年
		  ○ 标清视频：132000分钟/年
		那么，年平均转码时长：132000×1.5+132000=33万分钟
		• 转码包规格：建议小王选1个50万分钟/年的转码时长包，共计5425元。

则两种计费方式的年平均费分别为：

- 按量付费：19215.36+41391+7167.6=**67773.96**元
- 预付费资源包：13950+21800+5425=**41175**元

通过以上两种计费方式得出的费用对比，可以看出：

- 选择预付费资源包的计费方式将为您节省大量成本，累计购买的资源包越多，享受的优惠越多。
- 此外，您也可以考虑组合预付费资源包和按量付费的方式计费，以实现最大的资源使用率和最低的成本。

8.点播试看

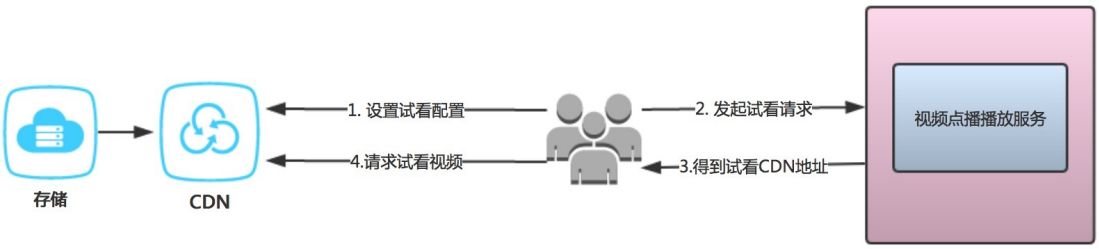
阿里云视频点播提供完整的试看解决方案。本文介绍如何开启试看功能并提供获取试看地址的最佳实践。

功能简介

试看指用户在观看视频或者音频等内容时，只能观看指定时间的内容，通常用于付费业务场景。阿里云视频点播提供完整的试看解决方案。您可以通过视频点播播放服务（调用服务端获取音视频播放地址接口）设置试看时长后获取试看地址或者自行拼接含有试看信息的试看地址（URL鉴权地址）。

> **注意** 如果设置的试看时长超过原视频总时长，用户可凭借点播返回的试看地址播放完整视频。

阿里云视频点播试看功能基于阿里云CDN加速实现。试看的基本原理是：播放端携带含有指定试看时长信息的试看地址（CDN加速地址）访问服务，云端对试看地址鉴权。鉴权通过则返回指定的文件内容，否则拒绝访问并返回403。下图展示了点播试看的实现流程：



1. 点播用户配置CDN加速域名并开启试看功能。

   > **注意** 域名配置和开启试看功能是使用试看功能的前提。详细操作请参见开启试看功能。

2. 点播用户的播放端向视频点播发起试看请求。
3. 视频点播根据域名配置及试看请求中的试看时长配置生成试看地址。

   > **说明** 点播支持调用服务端接口生成试看地址，也支持用户自行拼接试看地址。详情请参见调用接口获取试看地址或手动拼接试看地址。

4. 点播用户的播放端通过播放地址向CDN请求播放视频（试看）。

使用限制

- 目前试看功能支持的文件格式为MP4、HLS。其中，MP4视频的Meta信息必须在文件头部，不支持Meta信息在尾部的MP4视频。通过视频点播服务转码封装格式为MP4的时候，会将Meta信息放置在文件头部。
- 试看时间与关键帧存在依赖（点播转码输出文件默认10秒一个关键帧），因此短视频不建议使用试看，长视频试看时间建议至少设置为30秒。

  > **说明** 关键帧时长可通过转码模板修改。修改方法请参见普通转码模板设置。

- HLS文件试看精度为ts分段时长，具体可能存在误差，试看时长采用最大化原则。以10秒一个ts为标准，如果试看时长设置为15秒，实际返回数据为20秒。

开启试看功能

无论您调用点播接口获取试看地址还是手动拼接试看地址，都必须提前配置域名并开启试看功能。详细配置及操作指引如下：

1. 配置CDN加速域名。操作指引请参见添加加速域名。

   > 注意 只有配置CDN加速域名的域名才支持试看功能。

2. 为域名开启URL鉴权，开启过程中打开试看功能。操作指引请参见URL鉴权。

   > 注意 如果域名没有开启试看，请求点播时不能携带试看参数，返回的地址不能访问。因此，在开启URL鉴权功能时必须同时打开**支持试看**按钮。

   开启试看功能控制台界面截图参考如下：

   > 说明 如需手动拼接试看链接，需要传入privateKey参数用于计算鉴权值。其取值为控制台操作时生成或输入的**主Key**或**备Key**值。请根据需求记录上述值为后续使用做准备。

3. 为域名开启Range回源和拖拽播放。操作指引请参见配置Range回源及拖拽播放。

调用接口获取试看地址

视频点播提供获取视频播放地址的接口。接口详情请参见获取音视频播放地址。用户可集成服务端SDK，通过SDK调用该接口获取试看地址。具体步骤如下：

> 注意 调用接口前请确保已开启试看功能。操作指引请参见开启试看功能。

1. 根据业务需求集成服务端SDK。集成步骤请参考服务端SDK文档。
2. 通过SDK调用获取音视频播放地址接口。调用接口时设置请求参数PlayConfig中的PreviewTime指定试看时长。服务端会根据试看时长设置返回试看地址。PlayConfig参数描述请参见PlayConfig。

获取试看地址示例代码（Java）

> **说明**
> - 更多语言示例代码请参见OpenAPI。
> - 调用服务端接口需要使用AccessKey完成身份验证，请提前获取AccessKey。获取方法请参见获取AccessKey。

```
import com.alibaba.fastjson.JSONObject;
import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.profile.DefaultProfile;
import com.aliyuncs.exceptions.ClientException;
import com.aliyuncs.vod.model.v20170321.GetPlayInfoRequest;
import com.aliyuncs.vod.model.v20170321.GetPlayInfoResponse;
/**
 * @date 2021/12/30
 */
public class VodPreviewTest {
    public static void main(String[] args) throws ClientException {
        //请根据点播服务接入区域填写，更多信息，请参见点播地域标识。
        String regionId = "cn-shanghai";
        //提前获取的账号AccessKey。
        String accessKeyId = "<your accessKeyId>";
        //提前获取的账号AccessKey Secret。
        String accessKeySecret = "<your accessKeySecret>";
        //视频ID。示例：533606af570e4db4961248d0978b****。通过控制台上传的视频，可登录点播控制台，选择媒资库 > 音/视频查看视频ID。通过CreateUploadVideo接口上传的视频，视频ID为返回参数VideoId的值。
        String videoId = "<your videoId>";
        DefaultAcsClient client = InitVodClient(regionId, accessKeyId, accessKeySecret);
        GetPlayInfoResponse response = null;
        try {
            response = getPlayInfo(client, videoId);
        } catch (Exception e) {
            e.printStackTrace();
        }
        System.out.println("response = " + JSONObject.toJSONString(response));
    }
    /**
     * 初始化Client
     *
     * @param regionId
     * @param accessKeyId
     * @param accessKeySecret
     * @return
     * @throws ClientException
     */
    public static DefaultAcsClient InitVodClient(String regionId, String accessKeyId, String accessKeySecret) throws ClientException {
        DefaultProfile profile = DefaultProfile.getProfile(regionId, accessKeyId, accessKey
```

```
Secret);
        DefaultAcsClient client = new DefaultAcsClient(profile);
        return client;
    }
    /**
     * 获取视频播放地址
     *
     * @param client
     * @param videoId
     * @return
     * @throws Exception
     */
    public static GetPlayInfoResponse getPlayInfo(DefaultAcsClient client, String videoId)
throws Exception {
        GetPlayInfoRequest request = new GetPlayInfoRequest();
        request.setVideoId(videoId);
        //设置过期时间，单位秒，不设置，默认3600s
        request.setAuthTimeout(3600L);
        request.setFormats("mp4");
        JSONObject playConfig = new JSONObject();
        //视频点播试看时长，单位为秒。最小值1
        playConfig.put("PreviewTime", "30");
        request.setPlayConfig(playConfig.toJSONString());
        return client.getAcsResponse(request);
    }
}
```

手动拼接试看地址

视频点播支持手动拼接带试看信息的URL鉴权地址。操作步骤如下：

> **注意** 拼接试看地址前请确保已开启试看功能。操作指引请参见开启试看功能。

1. 手动拼接加入试看参数的鉴权URL地址。与完整观看时URL鉴权地址拼接不同的是，拼接试看地址在计算md5hash时要带上试看时长参数 `previewTime` ，即在原来URL鉴权md5hash计算方式的基础上，加入试看时长的计算。

完整视频地址md5hash计算	试看地址md5hash计算
`MD5(uri-timestamp-rand-uid-PrivateKey)`	`MD5(uri-timestamp-rand-uid-PrivateKey-previewTime)`

> **说明** md5hash计算的参数描述及手动拼接鉴权URL的方法请参考URL鉴权。如果需要看完整视频，不设置试看参数即可。

2. 在拼接完成的鉴权URL末尾加上 `&end=` ， `&end=` 后添加试看参数 `previewTime` ，生成完整的试看地址。

手动拼接Jar包需要的依赖（Java）

```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.4</version>
</dependency>
```

手动拼接试看地址示例代码（Java）

```
    import java.util.UUID;
    import java.net.URL;
    import org.apache.commons.lang3.StringUtils;
    private String generateRand() {
        return UUID.randomUUID().toString().replaceAll("-", "");
    }
    private String md5(String str) {
        try {
            MessageDigest md5 = MessageDigest.getInstance("MD5");
            md5.update(str.getBytes("UTF-8"));
            return bytesToHex(md5.digest());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public String genAuthKey(String object, String privateKey, Long expireTime, Long previe
wTime) {
        String rand = "0";
        String uid = "0";
        if (StringUtils.isBlank(privateKey)) {
            return "";
        }
        rand = generateRand();
        long timestamp = System.currentTimeMillis() / 1000 + (expireTime == null ? 0 : expi
reTime);
        String authStr = timestamp + "-" + rand + "-" + uid;
        String md5Str = object + "-" + authStr + "-" + privateKey;
        if(previewTime!=0)
            md5Str = md5Str + "-" + previewTime;
        String auth_key = authStr + "-" + this.md5(md5Str);
        return auth_key;
    }
    public void previewTest() throws Exception {
        try {
            String key = "<Your PrivateKey>";//控制台配置的主Key或备Key。如何获取请参见开启试看功
能。
            String fileUrl = "<Your File URL>";//文件地址，示例：http://example.aliyundoc.com
/test/bee21427ca3346848835c1bd786054c5-19bd8528c1d51576cd726cf86471ca0****.mp4
            URL url = new URL(fileUrl);
            String file = url.getFile();
            Long previewTime = 120L;//试看时长。
            Long expireTime = 1800L;
            String auth_key =genAuthKey(file, key, expireTime, previewTime);
            fileUrl = fileUrl + "?auth_key=" + auth_key;
            if(previewTime != 0)
                fileUrl = fileUrl + "&end=" + previewTime;
            System.out.println(fileUrl);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```