ALIBABA CLOUD

阿里云

物联网智能视频服务
应用开发指南

文档版本：20210622

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.iOS应用端接入

1.1. 工程配置

您可以使用视频边缘智能服务提供的应用端iOS SDK，实现视频播放的功能。

1.在podfile中添加引用源。

```
source 'https://github.com/aliyun/aliyun-specs.git'
```

2.添加库依赖。

```
pod 'IMSLinkVisualMedia_Base', '1.5.6'
```

3.执行pod update，则库安装完毕。

1.2. 视频播放器

播放器可用于RTMP，HLS流媒体播放

播放器说明

用于基于HLS的云端录像回放的播放，支持MPEG-TS和FMP4容器，AES-128加密方式。

功能	直播播放器	TF卡点播播放器	云端HLS播放器
视频播放	✓	✓	✓
音频播放	✓	✓	✓
暂停/恢复	x	✓	✓
跳至指定位置播放	X	✓	✓
总时长	x	✓	✓
当前播放进度	x	✓	✓
播放器状态变更通知	✓	✓	✓
静音	✓	✓	✓
变速播放	x	✓	✓
画面缩放模式设置	✓	✓	✓
播放器截图	✓	✓	✓
边播边录	✓	✓	✓
提供YUV数据	✓	✓	✓

功能	直播播放器	TF卡点播播放器	云端HLS播放器
提供SEI数据	✓	✓	✓

播放器错误列表

错误主码（PlayerMainError）	描述	子码（解析userInfo中subCode获取）	描述
ErrorSource	数据源相关错误	IMSLinkVisualPlayerErrorConnect	建立连接失败
		IMSLinkVisualPlayerErrorEncrypt	无效的解密密钥
		IMSLinkVisualPlayerErrorUrl	无效的播放地址
		IMSLinkVisualPlayerErrorDataSource	数据源错误或未设置
		IMSLinkVisualPlayerErrorGetURL	获取链接失败
		IMSLinkVisualPlayerErrorStop	关闭失败
		IMSLinkVisualPlayerErrorStart	启动失败
ErrorRender	渲染相关错误	IMSLinkVisualPlayerErrorDecode	解码错误
ErrorUnexpected	不符合预期错误	IMSLinkVisualPlayerErrorStream	接收数据流失败

1.3. RTMP播放器

本文介绍RTMP播放器的使用方法和支持的功能。

概述

播放器的运行、功能使用、代理回调的介绍，与RTMP的设置

使用指南

1. 引入框架。

```
//框架引入
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
```

2. 设置播放器日志。

```
//打开播放器日志 建议为IMSLinkVisualMediaLogInfo
[IMSLinkVisualPlayerViewController setLogLevel:IMSLinkVisualMediaLogInfo];
```

3. 创建播放器。

```
//创建播放器
IMSLinkVisualPlayerViewController *player = [[IMSLinkVisualPlayerViewController alloc] init];
//因为播放器为viewController，所以addChildViewController
[self addChildViewController:player];
//设置播放器frame
player.view.autoresizingMask = UIViewAutoresizingFlexibleWidth | UIViewAutoresizingFlexibleHeight;
player.view.frame = self.view.bounds;
//添加播放器view到指定位置
[self.view insertSubview:player.view atIndex:0];
//播放器代理
player.delegate = self;
```

4. 设置播放源。
   - 直播播放器

```
//设置参数 rtmpPath rtmp地址
//needEncrypt false 不需要加密，参数iv和key都传nil
/*
 注：直播切到手机后台后会自动调用stop方法，如果再次回到直播界面需要重新设置rtmpPath地址，并调用start方法
   rtmpPath具有时效性，如果rtmpPath已过期，需要重新获取rtmpPath地址
   TF卡点播和HLS播放不需要以上操作，请忽略
*/
[player setDataSource_Live:rtmpPath
        needEncrypt:needEncrypt
             iv:iv
            key:key];
```

   - TF卡点播播放器

```
//设置参数 rtmpPath rtmp地址
//needEncrypt false 不需要加密，参数iv和key都传nil
[player setDataSource_Vod:rtmpPath
        needEncrypt:needEncrypt
             iv:iv
            key:key];
```

5. 设置并使用播放器功能。
   - 开始播放

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
// 开始播放
[self.player start];
```

   - 静音播放器

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
//播放器静音，对讲不播放接收到的声音通过mute实现
self.player.mute = true;
```

   - 停止播放

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
// 停止播放，结束播放器
[self.player stop];
```

- 暂停播放（仅录播）

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
// 暂停播放
[self.player pause];
```

- 恢复播放（仅录播）

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
// 恢复播放
[self.player restore];
```

- 倍速播放（仅录播）

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
// 倍速播放
self.player.playSpeed = PlaySpeed_NORMAL;
```

- 视频截图

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
//获取视频截图
UIImage *image = [self.player videoSnapshot];
```

- 视频截图

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
//获取视频截图
UIImage *image = [self.player videoSnapshot];
```

- 边录边播 （不支持hlsPlayer）

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
- (IBAction)recordVideoButtonClick:(UIButton *)sender {
   sender.selected = !sender.selected;
   NSString *tmpPath = [NSTemporaryDirectory() stringByAppendingPathComponent:@"tmp.mp4"];
   if (sender.selected) {
      [self.player startRecordVideoWithfilePath:tmpPath];
   } else {
      [self.player stopRecordVideo];
   }
}
```

- 设置播放器缓存

```
#import <IMSLinkVisualMedia/IMSLinkVisualMedia.h>
/**
 //设置播放器缓存（缓存越大，延迟越高），一帧缓存延迟40ms，默认为5帧
 @param buffer 缓存大小 0 -- 16
 */
[self.player setDisplyBuffer:10];
```

- 设置播放器渲染模式

```
/// 播放器渲染模式
typedef NS_ENUM(NSUInteger, IMSLinkVisualPlayerDisplayMode){
  /// 客户对YUV数据不可见，完全由SDK处理和渲染，默认使用这个模式
  IMSLinkVisualPlayerDisplayMode_SDK,
  /// 允许客户对YUV数据进行二次加工，然后再还给SDK渲染
  IMSLinkVisualPlayerDisplayMode_Client_SDK,
  /// YUV数据完全交由客户处理和渲染
  IMSLinkVisualPlayerDisplayMode_Client
};
/// 设置播放器渲染模式，默认 IMSLinkVisualPlayerDisplayMode_SDK 模式
@property (assign, nonatomic) IMSLinkVisualPlayerDisplayMode lvDisplayMode;
/**
 直播、点播的共用代理接口
 视频帧YUV数据回调，同步接口，帧数据的内存和SDK是共用的，如外部需要缓存数据，必须把数据拷贝出去
 @see IMSLinkVisualPlayerViewController
 @param frame 视频帧YUV数据
 */
- (void)linkVisual:(IMSLinkVisualPlayerViewController *_Nullable)player withFrame:(IMSPlayerVideoF
rame *_Nullable)frame;
```

6. 使用播放器。
   - 直播播放器

```
  //设置参数 rtmpPath rtmp地址
  //needEncrypt false 不需要加密，参数iv和key都传nil
  /*
   注：直播切到手机后台后会自动调用stop方法，如果再次回到直播界面需要重新设置rtmpPath地址，并调
用start方法
     rtmpPath具有时效性，如果rtmpPath已过期，需要重新获取rtmpPath地址
     TF卡点播和HLS播放不需要以上操作，请忽略
  */
  [player setDataSource_Live:rtmpPath
          needEncrypt:needEncrypt
                iv:iv
               key:key];
```

   - TF卡点播播放器

```
  //设置参数 rtmpPath rtmp地址
  //needEncrypt false 不需要加密，参数iv和key都传nil
  [player setDataSource_Vod:rtmpPath
          needEncrypt:needEncrypt
                iv:iv
               key:key];
```

7. 查看播放器状态。

```
/// 播放器状态
typedef NS_ENUM(NSUInteger,IMSLinkVisualPlayerState){
  /// 空闲
  IMSLinkVisualPlayerStateIdle = 0,
  /// 缓冲中
  IMSLinkVisualPlayerStateBuffering = 2,
  /// 开始播放
  IMSLinkVisualPlayerStateStartPlay = 4,
  /// 暂停播放
  IMSLinkVisualPlayerStatePausePlay = 8,
  /// 切换到后台
  IMSLinkVisualPlayerStateBackground = 16,
};
```

8. 监听回调结果。

```
@protocol IMSLinkVisualDelegate <NSObject>
/*------------------直播、点播共用代理接口---------------------*/
/**
 直播、点播 共用接口
 连接成功
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualConnect:(IMSLinkVisualPlayerViewController *_Nullable)player;
/**
 直播、点播 共用接口
 准备完成进入播放状态
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualReady:(IMSLinkVisualPlayerViewController *_Nullable)player;
/**
 直播、点播 共用接口
 播放成功停止（正常停止）
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualStop:(IMSLinkVisualPlayerViewController *_Nullable)player;
/**
 直播、点播 共用接口
 连接错误回调 (所有直播点播错误，都通过这个代理回调)
 @see IMSLinkVisualPlayerViewController
 @param error 错误信息 IMSLinkVisualPlayerError枚举 对应 error.code
 */
- (void)linkVisual:(IMSLinkVisualPlayerViewController *_Nullable)player errorOccurred:(NSError *_Nullable)error;
/**
 直播、点播 共用接口
 视频帧YUV数据回调，同步接口，帧数据的内存和SDK是共用的，如外部需要缓存数据，必须把数据拷贝出去
 @see IMSLinkVisualPlayerViewController
 @param frame 视频帧YUV数据
 */
- (void)linkVisual:(IMSLinkVisualPlayerViewController *_Nullable)player withFrame:(IMSPlayerVideoFrame *_Nullable)frame;
/**
```

```
/
直播、点播 共用接口
初次获取画面与分辨率发生变化时回调
注：会触发停止录像
EVENT_RESOLUTION_CHANGE
@see IMSLinkVisualPlayerViewController
@param width 宽
@param height 高
*/
- (void)linkVisualResolutionChange:(IMSLinkVisualPlayerViewController *_Nullable)player width:(NSInteger)width height:(NSInteger)height;
/*---------------------点播独占代理接口----------------------*/
/**
 点播
 seek成功完成
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualReplaySeekReady:(IMSLinkVisualPlayerViewController *_Nullable)player;
/**
 点播
 恢复播放成功完成
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualRestore:(IMSLinkVisualPlayerViewController *_Nullable)player;
/**
 点播
 暂停播放成功完成
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualPause:(IMSLinkVisualPlayerViewController *_Nullable)player;
/**
 点播
 播放到结尾的回调
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualPlayEnd:(IMSLinkVisualPlayerViewController *_Nullable)player;
/**
 点播
 当前时间回调
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualReplay:(IMSLinkVisualPlayerViewController *_Nullable)player
        currentTime:(NSInteger)currentTime;
/// SEI回调接口
/// @param player  播放器
/// @param data  回调二进制数据 data.length为数据长度
/// @param timeStamp  时间戳
- (void)linkVisualReplay:(IMSLinkVisualPlayerViewController *_Nullable)player
           buffer:(NSData*_Nullable)data
        timeStamp:(NSInteger)timeStamp;
@end
```

1.4. HLS播放器

介绍HLS播放器的使用的展示。

1. 创建云端HLS播放器。

```
//创建播放器
IMSLinkVisualPlayerViewController *player = [[IMSLinkVisualPlayerViewController alloc] init];
//因为播放器为viewController，所以addChildViewController
[self addChildViewController:player];
//设置播放器frame
player.view.autoresizingMask = UIViewAutoresizingFlexibleWidth | UIViewAutoresizingFlexibleHeight;
player.view.frame = self.view.bounds;
//添加播放器view到指定位置
[self.view insertSubview:player.view atIndex:0];
//播放器代理
player.delegate = self;
```

2. 设置云端HLS。

```
//设置参数 HLSPath http***.m3u8 seekTime的单位：秒
[player setDataSource_HLS:hlsPath
        seekTime:seekTime];
```

3. 运行播放器。请参见RTMP播放器。

1.5. 语音对讲

1.5.1. 概述

iOS应用端SDK提供语音对讲功能，实现应用端和IPC设备之间，端到端的单向或双向的实时对讲。

对讲类型

语音对讲的类型及其注意事项如下：

类型	说明	注意事项
单向对讲	应用端采集并发送音频数据到设备端进行播放。	App端采集音频期间手机保持静音。
双向对讲	应用端和设备端同时采集音频和播放音频。	设备端必须支持声学回声消除AEC（Acoustic Echo Cancellation），否则不建议使用该功能。

音频类型

语音对讲支持G711a、G711u以及AAC编码方式，三种音频类型详细信息为：

> **说明**
>
> 选择编码方式前，请确认您的设备端IPC设备是否支持。

- 采样率：支持8 kHz和16 kHz。
- 支持编码。
- 支持解码。

1.5.2. 操作步骤

本文介绍在iOS应用端实现语音对讲功能的操作步骤。

前提条件

- 已创建产品和设备，具体操作，请参见设备接入。
- 已完成工程配置，具体操作，请参见工程配置。

操作步骤

下面为您介绍完成一次语音对讲的操作步骤。操作步骤中接口和代理的详细信息，请参见接口说明和代理方法。

1. 创建语音对讲实例。

```
IMSLinkVisualPlayerViewController * player = [IMSLinkVisualPlayerViewController new];
```

2. 设置对接参数，即MIC采集参数。

```
IMSLinkVisualAudioParams *intercomEncodeParams = [[IMSLinkVisualAudioParams alloc] init];
  intercomEncodeParams.sampleRate = 8000;
  intercomEncodeParams.channel = 1;
  intercomEncodeParams.bitsPerSample = 16;
  intercomEncodeParams.format = IMSLinkVisualAudioFormatG711a;
  player.intercomEncodeParams = intercomEncodeParams;
```

3. 设置代理回调。

```
player.intercomDelegate = self;
```

4. 启动对接。

```
[player startIntercom:IMSLinkVisualIntercomAudioModeIntercom];
```

5. 在采集回调里发送对讲数据。

```
- (void)linkVisualIntercom:(IMSLinkVisualPlayerViewController * _Nullable)player recordData:(NSData * _Nullable)data {
  ///直接发送
  [player sendAudioData:data];
}
```

6. 当需要停止语音对讲时，调用以下方法，停止对讲。

```
[player stopIntercom];
```

1.5.3. 接口说明

本文介绍语音对讲接口的详细信息。

创建实例

```
IMSLinkVisualPlayerViewController * player = [IMSLinkVisualPlayerViewController new]; //创建语音对讲实例
 player.intercomEncodeParams = intercomEncodeParams; //设置对讲模式和音频参数
```

设置对讲数据源

```
/**
 设置对讲业务数据源
 @param rtmpPath：rtmp地址
 @param needEncrypt：是否加密，对讲都为加密，此处设置为true
 @param iv：解密向量，16 Byte，如需base64转码请自行查阅文档
 @param key：解密密钥，16 Byte，如需base64转码请自行查阅文档
 NSData* iv = [[NSData alloc] initWithBase64EncodedString:ivString options:NSDataBase64DecodingIgnoreU
nknownCharacters];
 @return 是否成功设置数据源
 */

- (BOOL)setDataSource_Intercom:(NSString *_Nullable)rtmpPath
        needEncrypt:(BOOL)needEncrypt
            iv:(NSData *_Nullable)iv
           key:(NSData *_Nullable)key;
```

启动对讲

```
/**
 开始语音对讲
 @param mode即对讲音源模式，可设置为：
0：播放直播音频（播放器获取播放器画面与声音时，支持此模式）
1：播放对讲音频
2：不使用播放器播放（使用者只接收数据，不播放或自行播放）
 */
- (void)startIntercom:(IMSLinkVisualIntercomAudioMode)mode;
```

发送本地MIC数据到设备

```
/**
 发送设备端接收的录音数据，格式为脉冲编码调制PCM（Pulse Code Modulation），
iOS SDK会转码成intercomEncodeParams指定的格式
 @param data:语音数据，语音数据由代理方法中回调的音频数据
 */
- (void)sendAudioData:(NSData *_Nullable)data;
```

停止对讲

```
/**
 结束语音对讲
 */
- (void)stopIntercom;
```

1.5.4. 代理方法

本文介绍开发iOS应用端语音对讲功能时，相关的代理方法。

连接服务器

```
/**
 语音对讲成功连接服务器
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualIntercomConnect:(IMSLinkVisualPlayerViewController *_Nullable)player;
```

准备完成

```
/**
 语音对讲准备完成
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualIntercomReady:(IMSLinkVisualPlayerViewController *_Nullable)player;
```

接收设备端音频参数

```
/**
 接收到设备端的音频参数
 @see IMSLinkVisualPlayerViewController
 @see IMSLinkVisualAudioParams
 @param params 音频参数
 */
- (void)linkVisualIntercom:(IMSLinkVisualPlayerViewController *_Nullable)player
         audioParams:(IMSLinkVisualAudioParams *_Nullable)params;
```

接收设备端音频数据

```
/**
 接收到设备端的音频数据
 @see IMSLinkVisualPlayerViewController
 @param data 音频数据
 */
- (void)linkVisualIntercom:(IMSLinkVisualPlayerViewController *_Nullable)player
          audioData:(NSData *_Nullable)data;
```

手机端的录音音频数据

```
/**
 手机端的录音音频数据
 @see IMSLinkVisualPlayerViewController
 @param data 音频数据
 */
- (void)linkVisualIntercom:(IMSLinkVisualPlayerViewController *_Nullable)player
        recordData:(NSData *_Nullable)data;
```

停止语音对讲

```
/**
 停止语音对讲
 @see IMSLinkVisualPlayerViewController
 */
- (void)linkVisualIntercomStop:(IMSLinkVisualPlayerViewController *_Nullable)player;
```

错误回调

```
/**
 语音对讲错误回调
 @see IMSLinkVisualPlayerViewController
 @see IMSLinkVisualIntercomError
 @param error 错误 IMSLinkVisualIntercomError枚举 对应 error.code
 */
- (void)linkVisualIntercom:(IMSLinkVisualPlayerViewController *_Nullable)player
      errorOccurred:(NSError *_Nullable)error;
```

1.5.5. 错误列表

本文介绍iOS应用端语音对讲功能的错误列表。

错误枚举	描述	解决方法
IMSLinkVisualIntercomErrorGetURL	获取链接失败。	请确认以下设置项然后重试： • 确认填充参数是否正确。 • 确认平台设置是否正确。 • 请确认是否已开启对讲的物模型。
IMSLinkVisualIntercomErrorInUse	对讲正在使用。	设备正在与他人对讲，请等待或关闭他人对讲，然后重试。
IMSLinkVisualIntercomErrorStart	启动失败。	请重新启动。

错误枚举	描述	解决方法
IMSLinkVisualIntercomErrorConnect	建立连接失败。	请重新启动。
IMSLinkVisualIntercomErrorParams	语音对讲参数错误。	请确认以下设置项然后重试： • 确认对讲参数是否正确。 • 确认设备是否支持语音对讲。
IMSLinkVisualIntercomErrorStream	接收参数错误。	请确认网络或数据是否异常，然后重新启动。
IMSLinkVisualIntercomErrorDecode	解码错误。	请确认设备端是否按照要求发送了数据，然后重新启动。
IMSLinkVisualIntercomErrorRecorder	录音机运行失败。	对讲录音被占用，请尝试关闭其它对讲后，重新启动对讲。
IMSLinkVisualIntercomErrorSendData	发送对讲数据失败。	对讲状态异常，请尝试关闭后重新启动对讲。
IMSLinkVisualIntercomErrorStop	关闭对讲失败。	该报错可忽略，如有需要，您可重新发起对讲。

2.Web应用端接入

2.1. 播放器概述

视频边缘智能服务提供Web播放器，用于在PC端进行视频直播和录像文件播放，本文介绍Web播放器的基本信息。

播放器兼容性

- 支持的音频编码协议：AAC
- 支持的视频编码协议：H264
- 支持的浏览器：Chrome、Firefox、Edge、Safari和UC浏览器
- 支持的播放协议：
  - 直播播放器：HTTP-FLV和HLS

    > **说明**
    >
    > HTTP-FLV延时时间约3s，HLS延时时间为5~10s。

  - 云端录像点播播放器：HLS
  - 本地录像点播播放器：HTTP-FLV

播放器功能

Web播放器提供如下功能，相关功能实现方法，请参考Web端播放器开发。

- 视频播放
- 音频播放
- 暂停或恢复播放
- 从指定位置开始播放
- 获取播放总时长
- 获取当前播放进度
- 播放器状态变更通知
- 循环播放
- 异常重试播放

多窗口直播

Web播放器在直播场景下，通过配置多个直播窗口，可实现多窗口直播。

多窗口直播时，对您现场的网络状况、浏览器网络下载能力、播放器运行CPU要求较高。使用前，建议参考以下内容进行调试和配置：

- 使用测速软件，验证现场网络条件是否满足多窗口播放要求。

  播放1080p的摄像头采集的视频时，建议每路至少预留4 Mbit/s的下载带宽。对于清晰度更高的视频，需要更大的下载速度。

- 当窗口数大于等于4时，建议采用辅码流或低码流进行直播播放观看。
- 在多窗口播放出现某个窗口无法播放时，先验证单窗口是否能够正常播放。

  解决方法，请参考多窗口直播场景下，出现某个窗口无法播放，应该如何处理？
- 在多窗口播放出现某个窗口播放卡顿时，先检测单窗口是否流畅播放。

  解决方法，请参考多窗口直播场景下，出现某个窗口播放卡顿时，应该如何处理？

2.2. 播放器开发

Web端播放器支持HLS和HTTP-FLV播放协议，本文介绍Web端播放器的开发方法。

视频边缘智能服务LinkVisual Web端播放器，基于Video.js框架，集合LinkVisual视频业务开发，为Web端播放器的视频直播和录像文件播放提供解决方案。支持播放HTTP-FLV协议和HLS协议的视频。关于Web端播放器的详细信息，请参考播放器概述。

步骤一：在HTML代码中引入CDN资源

CDN资源如下：

```
<link href="//g.alicdn.com/code/lib/video.js/7.9.7/video-js.min.css" rel="stylesheet">
<script src="//g.alicdn.com/code/lib/video.js/7.9.7/video.min.js"></script>
<script src="//g.alicdn.com/linkplatform/videojs-plugins/0.0.3??videojs-tech-flv.js,videojs-plugin-hls-compat.js"></script>
```

步骤二：配置JavaScript文件

JavaScript文件包括注册插件到Video.js，配置Video.js提供的方法以及配置LinkVisual提供的方法。

```
function getStreamUrl() {
 return Promise.resolve('...'); // 获取播放地址返回一个promise
}

function getSourceType(url) {
 let type = '';
 if (/\.flv/.test(url)) {
  type = 'video/x-flv';
 } else if (/\.m3u8/.test(url)) {
  type = 'application/x-mpegURL';
 } else if(/\.mp4/.test(url)) {
  type = 'video/mp4';
 }
 return type;
}

function createPlayer(cb) {
 var videoEl = document.getElementById('videoEl');
 var player = videojs(videoEl, {  //配置Video.js提供的方法
  autoplay: false,
  muted: true,
  controls: true,
  preload: 'none',
  loop: false,
  techOrder: ['flv', 'html5'],
```

```
  flv: {
   reconnInterval: 5000,
   reconnTimes: 10,
   getStreamUrl,
   mediaDataSource: {
    cors: true,
    withCredentials: true
   },
   flvConfig: {
    lazyLoadMaxDuration: 24 * 60 * 60,
   },
  },
  html5: {
   hls: {
    cacheEncryptionKeys: true,
   },
  },
 }, cb);

//LinkVisual相关功能

 player.on('error', (e) => {
  console.error('videojs error: ', e);
 });
 player.on('flvError', (e) => {
  console.error('flv error: ', e.errorInfo);
 });

 var wrappedPlayer = {
  play: function(loadSource) {
   if (loadSource) {
    if (!player.paused()) {
     player.pause();
    }
    player.reset();
    player.removeClass('vjs-paused');
    player.addClass('vjs-waiting');
    return getStreamUrl().then((url) => {
     var source = {
      src: url,
      type: getSourceType(url)
     };
     player.src(source);
     player.autoplay(true);
     player.load();
    });
   }

   return player.play();
  },

  pause: function() {
   return player.pause();
  }
```

```
 }

 return wrappedPlayer;
}

//注册插件到Video.js
if (!videojs.getTech('flv')) {
 videojs.registerTech('flv', videojsTechFlv);
}
if (!videojs.getPlugin('hlsCompat')) {
 videojs.registerPlugin('hlsCompat', videojsPluginHlsCompat);
}

const player = createPlayer(function () {
 player.play(true);
});
```

接口说明

- Video.js框架的参数说明如下表：

> **说明**
>
> Video.js框架是用于在网页上嵌入视频播放器的开源JS库。获取Video.js提供的其他配置播放器方法，请参考Video.js。

方法	是否可选	描述
autoplay	可选	播放器初始化之后，是否自动播放： ○ true：自动播放。 ○ false（默认）：手动播放。
muted	可选	是否静音： ○ true：静音。 ○ false（默认）：不静音。 **注意** 如果视频加载后就要立马播放，需要设置为true，防止被浏览器阻止播放视频。

方法	是否可选	描述
controls	可选	是否配置控制条： ○ true（默认）：配置。 ○ false：不配置。 若设置为false，则只能通过调用接口控制视频播放。
preload	可选	是否预加载视频。可配置为如下值： ○ auto：自动预加载。 ○ metadata：加载元数据信息，例如视频的长度，播放画面的尺寸等。 ○ none：不预加载，开始播放时才开始下载视频。
loop	可选	是否循环播放： ○ true：循环播放。 ○ false（默认）：不循环播放。
techOrder	必选	Video.js的默认播放顺序。本示例中，配置为 `['flv', 'html5']` 。
HLS相关配置		
cacheEncryptionKeys	必选	是否缓存密钥： ○ true：缓存 ○ false：不缓存 若设置为false，则每次获取ts文件前再获取一次密钥。

- LinkVisual提供的接口：
  - 播放视频

    ```
    play() => Promise<undefined> | undefined
    ```

  - 暂停播放视频

    ```
    pause() => Player
    ```

- 设置播放源

  如果不传入参数，则使用获取播放源功能。

  ```
  src(source?: { src: string, type: string }) => string | undefined
  ```

- 重置播放器

  调用后，会同时触发Video.js中的reset方法。

  ```
  reset() => void
  ```

- 加载src中的资源

  ```
  load() => void
  ```

- 设置从指定时间开始播放视频

  如果不传入该参数，则返回当前的播放时间。

  ```
  currentTime(seconds?: number) => number
  ```

- 自动播放

  需要与Video.js中 `autoplay` 参数保持一致，如果不传参数，则获取Video.js中 `autoplay` 的值。

  ```
  autoplay(value: boolean | string) => void
  ```

- 是否为暂停状态

  ```
  paused() => boolean
  ```

- 开启监听事件

  ```
  on(type: string, fn: Function) => Player
  ```

  监听事件包括如下内容：

  - canplay：加载的视频流达到了播放要求。
  - ended：视频播放结束后触发。
  - error：监听播放异常时触发。
  - flvError：监听播放FLV视频流触发的异常。
  - loadeddata：浏览器加载到了视频帧时触发。
  - play：播放视频时触发。
  - playing：视频正在播放时触发，由于网络或者其它情况导致视频播放卡顿，该事件会被反复触发。

- 移除监听事件

  ```
  off(type: string, fn: Function) => Player
  ```

- FLV相关配置

方法	是否可选	描述
reconnInterval	必选	直播重连间隔时长，单位为毫秒。例如设置为5000。
reconnTimes	必选	播放失败时，请求重连的次数。例如设置为10。
getStreamUrl: async()	必选	返回一个播放地址或者一个Promise最终返回播放地址。
cors	可选	是否支持跨域访问： ■ true（默认）：支持。 ■ false：不支持。
withCredentials	可选	是否在跨域请求时携带Cookie： ■ true：携带。 ■ false（默认）：不携带。

- FLV播放相关配置

方法	是否可选	描述
lazyLoadMaxDuration	可选	加载指定时间的视频流后停止拉流，默认为3*60s。

2.3. 常见问题

本文是LinkViusal Web播放器开发时的相关常见问题和解决方法。

Web播放器是否支持RTMP播放源？

不支持。RTMP浏览器播放依赖Flash插件，Flash插件在Chrome等浏览器上将全面禁止，浏览器将无法使用Flash播放RTMP播放源。

为什么FLV直播地址可以下载，但不能播放？

这是因为浏览器对于播放流的质量要求比较高，视频流中出现的脏数据会导致视频播放中断，播放器会重连播放，超过最大重连次数将不再重连，即无法播放。

为什么播放HLS播放源几分钟后无法播放？

请检查播放器初始化代码，是否设置了参数 `cacheEncryptionKeys` 为 `true` 。

如果未设置该参数，播放器每次获取TS文件前会先获取一遍密钥，几分钟后获取密钥的接口上的token失效，接口返回401错误，播放停止。

为什么要设置静音播放？

在非静音情况下，浏览器会阻止视频自动播放。

多窗口直播场景下，出现某个窗口无法播放，应该如何处理？

先验证单窗口是否能够正常播放，如果：

- 单窗口无法播放：解决单窗口播放问题，通常是设备未推流或者推流异常引起。
- 单窗口播放正常，多窗口同时播放时某个窗口异常：通过 `pending` HTTP请求，在浏览器调试模式下，查看是否浏览器资源受限。建议您减少多窗口的数量，达到更佳的播放效果。

多窗口直播场景下，出现某个窗口播放卡顿时，应该如何处理？

先检测视频源单窗口是否流畅播放，如果：

- 单窗口播放不流畅：可能是播放端网络资源不足，或者设备上行推流网络资源不足，请先解决单窗口问题。
- 单窗口播放流畅，多窗口同时播放时某个窗口卡顿：浏览器解码能力不足，或者浏览器网络下载能力不足引起，需要降低浏览器消耗，建议减少播放窗口数量，达到更佳的播放效果。