uniapp 安卓ffmpeg音视频处理原生插件

目　录

使用方法

介绍

安卓ffmpeg音视频处理原生插件，支持音视频转换，音视频剪切，添加字幕，添加水印等操作，支持自定义指令，取消指令执行等，功能比较全面

联系作者

关注微信公众号可联系作者

插件地址

https://ext.dcloud.net.cn/plugin?id=16523

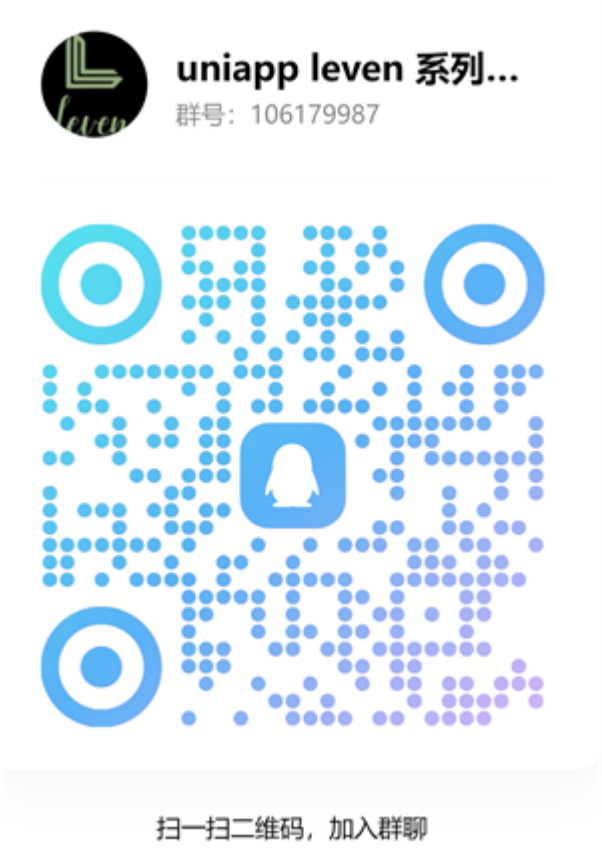
插件申请权限

1. android.permission.INTERNET
2. android.permission.ACCESS_NETWORK_STATE
3. android.permission.READ_EXTERNAL_STORAGE
4. android.permission.WRITE_EXTERNAL_STORAGE

示例文件和演示程序下载

使用方法

加QQ群下载，群号：106179987



公共操作用法

插件引入

在需要使用插件的页面加载以下代码

```
const commonModule = uni.requireNativePlugin("leven-ffmpeg-CommonModule");
```

页面内容

```
<template>
  <view>
    <uni-card title="操作文件">
      <view class="audio-file-box">
        <uni-easyinput v-model="videoPath" placeholder="视频文件路径"></uni-easyinpu
        <button style="margin-left: 10px;" type="primary" @click="selectVideoFile'
      </view>
      <view style="margin-bottom: 5px;"><text style="color: #999999; font-size: 14
      <view style="margin-bottom: 5px;"><text style="color: #999999; font-size: 14
```

使用方法

```
      <view><text style="color: #999999; font-size: 14px;">自定义指令内容：{{command
    </uni-card>
    <uni-card title="公共接口">
      <button type="primary" @click="filePermission">申请文件权限</button>
      <button type="primary" @click="customCommands">自定义指令(以视频翻转为例)</butto
      <button type="primary" @click="cancelCommands">取消执行命令</button>
      <button type="primary" @click="logStr = ''">清空日志</button>
    </uni-card>
    <uni-card class="uni-card-box" title="日志">
      <view><text style="font-size: 14px; flex-wrap: wrap;">{{logStr}}</text></vie
    </uni-card>
  </view>
</template>

<script>
  //需要引入媒体选择的插件，插件地址：https://ext.dcloud.net.cn/plugin?id=16808
  const mediaPickerModule = uni.requireNativePlugin("leven-mediaPicker-MediaPicke
  const commonModule = uni.requireNativePlugin("leven-ffmpeg-CommonModule");
  export default {
    data() {
      return {
        // 视频文件
        videoPath: "",
        //转换后文件所在目录
        dirPath: "/storage/emulated/0/leven-ffmpeg/",
        // 自定义指令内容
        command: "",
        logStr: "",
      }
    },
    methods: {
      // 选择视频文件
      selectVideoFile() {
        mediaPickerModule.video({
          suffix: [".mp4"],
          maxCount: 1
        }, res => {
          this.writeLog(JSON.stringify(res))
          if (res.code == 0 && res.data && res.data.files && Array.isArray(res.dat
            this.videoPath = res.data.files[0].path;
            // this.command = "ffmpeg -y -i " + this.videoPath + " -vf hflip -c:v
            this.command =
              `ffmpeg -y -i ${this.videoPath} -vf "color=color=#2B2D30:size=1920x1
            console.log(this.command)
          }
        })
      },
      // 申请文件权限
      filePermission() {
        commonModule.filePermission(res => {
```

```
            this.writeLog(JSON.stringify(res))
            this.command =
              `ffmpeg -y -i /storage/emulated/0/Movies/QQ/Video_1714987697800.mp4 -v
          })
        },
        // 自定义指令
        customCommands() {
          commonModule.customCommands({
            cmd: this.command
          }, res => {
            this.writeLog(JSON.stringify(res))
          })
        },
        // 取消执行命令
        cancelCommands() {
          commonModule.cancelCommands(res => {
            this.writeLog(JSON.stringify(res))
          })
        },
        // 写日志
        writeLog(str) {
          console.log(str);
          let logStr = uni.$lv.date.format(null, "yyyy-mm-dd hh:MM:ss") + " " + str
          this.logStr = logStr + this.logStr;
        },
      }
    }
</script>

<style>
  .audio-file-box {
    flex-direction: row;
    margin-bottom: 5px;
  }
</style>
```

音频操作用法

插件引入

在需要使用插件的页面加载以下代码

```
const module = uni.requireNativePlugin("leven-ffmpeg-AudioModule");
```

页面内容

```
<template>
  <view>
    <uni-card title="操作文件">
      <view class="audio-file-box">
        <uni-easyinput v-model="audioPath" placeholder="音频文件路径"></uni-easyinpu
        <button style="margin-left: 10px;" type="primary" @click="selectAudioFile'
      </view>
      <view class="audio-file-box">
        <uni-easyinput v-model="videoPath" placeholder="视频文件路径"></uni-easyinpu
        <button style="margin-left: 10px;" type="primary" @click="selectVideoFile'
      </view>
      <view style="margin-bottom: 5px;"><text style="color: #999999; font-size: 14
      <view><text style="color: #999999; font-size: 14px;">转换后的文件路径：{{dirPat
    </uni-card>
    <uni-card title="音频操作">
      <button type="primary" @click="transformAudio">音频转码</button>
      <button type="primary" @click="cutAudio">音频剪切</button>
      <button type="primary" @click="concatAudio">音频拼接</button>
      <button type="primary" @click="mixAudio">混音</button>
      <button type="primary" @click="changeVolume">更改音频音量</button>
      <button type="primary" @click="extractAudio">从视频文件抽取音频</button>
      <button type="primary" @click="decodeAudio">音频解码PCM</button>
      <button type="primary" @click="encodeAudio">音频编码</button>
      <button type="primary" @click="audioFadeIn">音频淡入</button>
      <button type="primary" @click="audioFadeOut">音频淡出</button>
      <button type="primary" @click="audio2Amr">音频转amr</button>
      <button type="primary" @click="audioInfo">获取音频信息</button>
      <button type="primary" @click="logStr = ''">清空日志</button>
    </uni-card>
    <uni-card class="uni-card-box" title="日志">
      <view><text style="font-size: 14px; flex-wrap: wrap;">{{logStr}}</text></vie
    </uni-card>
  </view>
</template>

<script>
  //需要引入媒体选择的插件，插件地址：https://ext.dcloud.net.cn/plugin?id=16808
  const mediaPickerModule = uni.requireNativePlugin("leven-mediaPicker-MediaPicke
  const commonModule = uni.requireNativePlugin("leven-ffmpeg-CommonModule");
  const module = uni.requireNativePlugin("leven-ffmpeg-AudioModule");
  export default {
    data() {
      return {
        // 音频文件
        audioPath: "",
        // 视频文件
```

```
        videoPath: "",
        //转换后文件所在目录
        dirPath: "/storage/emulated/0/leven-ffmpeg/",
        // 当前选择的文件名称
        audioName: "",
        logStr: "",
      }
    },
    methods: {
      // 选择音频文件
      selectAudioFile() {
        mediaPickerModule.audio({
          suffix: [".mp3"],
          maxCount: 1
        }, res => {
          this.writeLog(JSON.stringify(res))
          if (res.code == 0 && res.data && res.data.files && Array.isArray(res.da
            this.audioPath = res.data.files[0].path;
            this.audioName = res.data.files[0].name.replace(".mp3", "");
          }
        })
      },
      // 选择音频文件
      selectVideoFile() {
        mediaPickerModule.video({
          suffix: [".mp4"],
          maxCount: 1
        }, res => {
          this.writeLog(JSON.stringify(res))
          if (res.code == 0 && res.data && res.data.files && Array.isArray(res.da
            this.videoPath = res.data.files[0].path;
          }
        })
      },
      // 申请文件权限
      filePermission() {
        commonModule.filePermission(res => {
          this.writeLog(JSON.stringify(res))
        })
      },
      // 音频转码
      transformAudio() {
        module.transformAudio({
          //源文件
          src: this.audioPath,
          // 目标文件
          target: this.dirPath + "转码后的音频.aac"
        }, res => {
          this.writeLog(JSON.stringify(res))
        })
```

```
    },
    // 音频剪切
    cutAudio() {
      module.cutAudio({
        //类型，1.根据时长剪切，2.根据结束时间剪切
        type: 2,
        //源文件
        src: this.audioPath,
        // 目标文件
        target: this.dirPath + "剪切后的音频.mp3",
        //开始剪切时间
        startTime: 60,
        //时长（type=1时有效）
        duration: 60,
        //剪切结束时间（type=2时有效）
        endTime: 120
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 音频拼接
    concatAudio() {
      module.concatAudio({
        //源文件
        src: this.audioPath,
        //拼接文件
        append: path,
        // 目标文件
        target: this.dirPath + "拼接后的音频.mp3"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 音频混合
    mixAudio() {
      module.mixAudio({
        //源文件
        src: this.audioPath,
        //混合文件
        mix: path,
        // 目标文件
        target: this.dirPath + "混音后的音频.mp3"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 更改音频音量
    changeVolume() {
      module.changeVolume({
        //类型,1.固定音量（单位DB），2.音量的倍数
```

```
        type: 2,
        //源文件
        src: this.audioPath,
        //音量值
        // value: 20,
        //音量值（不定音量需要传整形，音量的倍数可以是浮点型）
        value: 1.5,
        // 目标文件
        target: this.dirPath + "更改音频音量后的音频.mp3"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 抽取音频
    extractAudio() {
      module.extractAudio({
        //源文件
        src: this.videoPath,
        // 目标文件(只能是aac格式)
        target: this.dirPath + "抽取音频后的音频.aac"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 音频解码
    decodeAudio() {
      module.decodeAudio({
        //源文件
        src: this.audioPath,
        // 目标文件(只能是pcm格式)
        target: this.dirPath + "解码后的音频.pcm",
        //采样率
        sampleRate: 44100,
        // 声道:1.单声道，2.立体声道
        channel: 2
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 音频编码
    encodeAudio() {
      module.encodeAudio({
        //源文件(只能是pcm格式)
        src: this.dirPath + "解码后的音频.pcm",
        // 目标文件
        target: this.dirPath + "编码后的音频.wav",
        //采样率
        sampleRate: 44100,
        // 声道:1.单声道，2.立体声道
        channel: 2
```

```
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 音频淡入
    audioFadeIn() {
      module.audioFadeIn({
        //源文件
        src: this.audioPath,
        // 目标文件
        target: this.dirPath + "音频淡入后的音频.mp3",
        //淡入时长，单位：秒
        duration: 5
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 音频淡出
    audioFadeOut() {
      module.audioFadeOut({
        //源文件
        src: this.audioPath,
        // 目标文件
        target: this.dirPath + "音频淡出后的音频.mp3",
        //淡出时长，单位：秒
        duration: 10
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 音频转amr
    audio2Amr() {
      module.audio2Amr({
        //源文件
        src: this.audioPath,
        // 目标文件
        target: this.dirPath + "音频转amr后的音频.amr"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 获取音频信息
    audioInfo() {
      module.audioInfo({
        //源文件
        src: this.audioPath
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
```

```
      // 写日志
      writeLog(str) {
        console.log(str);
        let logStr = uni.$lv.date.format(null, "yyyy-mm-dd hh:MM:ss") + " " + str
        this.logStr = logStr + this.logStr;
      }
    }
  }
</script>

<style>
  .audio-file-box {
    flex-direction: row;
    margin-bottom: 5px;
  }
</style>
```

视频操作用法

插件引入

在需要使用插件的页面加载以下代码

```
const module = uni.requireNativePlugin("leven-ffmpeg-VideoModule");
```

页面内容

```
<template>
  <view>
    <uni-card title="操作文件">
      <view class="audio-file-box">
        <uni-easyinput v-model="waterPath" placeholder="水印文件路径"></uni-easyinpu
        <button style="margin-left: 10px;" type="primary" @click="selectImageFile'
      </view>
      <view class="audio-file-box">
        <uni-easyinput v-model="videoPath" placeholder="视频文件路径"></uni-easyinpu
        <button style="margin-left: 10px;" type="primary" @click="selectVideoFile'
      </view>
      <view style="margin-bottom: 5px;"><text style="color: #999999; font-size: 14
      <view><text style="color: #999999; font-size: 14px;">转换后的文件路径：{{dirPat
    </uni-card>
```

使用方法

```
    <uni-card title="视频操作">
      <button type="primary" @click="transformVideo">视频转码</button>
      <button type="primary" @click="extractVideo">从视频文件抽取视频（无声音）</butto
      <button type="primary" @click="mixAudioVideo">音视频合成</button>
      <button type="primary" @click="cutVideo">视频剪切</button>
      <button type="primary" @click="concatVideo">视频拼接</button>
      <button type="primary" @click="screenShot">视频封面</button>
      <button type="primary" @click="frame2Image">视频某一帧的图片</button>
      <button type="primary" @click="video2Image">视频转图片</button>
      <button type="primary" @click="waterMarkImage">添加水印图片</button>
      <button type="primary" @click="video2Gif">视频转成Gif</button>
      <!-- <button type="primary" @click="screenRecord">屏幕录制</button> -->
      <button type="primary" @click="image2Video">图片合成视频</button>
      <button type="primary" @click="multiVideo">多画面拼接</button>
      <button type="primary" @click="reverseVideo">视频反序倒播</button>
      <button type="primary" @click="denoiseVideo">视频降噪</button>
      <button type="primary" @click="picInPicVideo">画中画</button>
      <button type="primary" @click="videoScaleMultiple">视频倍数缩放</button>
      <button type="primary" @click="videoScaleWidthHeight">视频宽高缩放</button>
      <button type="primary" @click="videoSpeed">视频倍速播放</button>
      <button type="primary" @click="video2Yuv">视频转yuv</button>
      <button type="primary" @click="yuv2H264">YUV转H264</button>
      <button type="primary" @click="video2HLS">视频切片</button>
      <button type="primary" @click="hls2Video">切片合成视频</button>
      <button type="primary" @click="videoParams">参数调节(亮度，对比度，饱和度)</butt
      <button type="primary" @click="videoRotation">视频旋转</button>
      <button type="primary" @click="videoFlip">视频翻转</button>
      <button type="primary" @click="videoInfo">获取视频信息</button>
      <button type="primary" @click="compress">视频压缩</button>
      <button type="primary" @click="rtmpRegister">注册推流服务</button>
      <button type="primary" @click="rtmpStart">开始推流</button>
      <button type="primary" @click="rtmpClose">关闭推流</button>
      <button type="primary" @click="logStr = ''">清空日志</button>
    </uni-card>
    <uni-card class="uni-card-box" title="日志">
      <view><text style="font-size: 14px; flex-wrap: wrap;">{{logStr}}</text></vie
    </uni-card>
  </view>
</template>

<script>
  //需要引入媒体选择的插件，插件地址：https://ext.dcloud.net.cn/plugin?id=16808
  const mediaPickerModule = uni.requireNativePlugin("leven-mediaPicker-MediaPicke
  const commonModule = uni.requireNativePlugin("leven-ffmpeg-CommonModule");
  const module = uni.requireNativePlugin("leven-ffmpeg-VideoModule");
  export default {
    data() {
      return {
        // 水印文件
        waterPath: "",
```

```
        // 视频文件
        videoPath: "",
        //转换后文件所在目录
        dirPath: "/storage/emulated/0/leven-ffmpeg/",
        // 当前选择的文件名称
        audioName: "",
        logStr: "",
      }
    },
    methods: {
      // 选择水印文件
      selectImageFile() {
        mediaPickerModule.image({
          suffix: [".png"],
          maxCount: 1
        }, res => {
          this.writeLog(JSON.stringify(res))
          if (res.code == 0 && res.data && res.data.files && Array.isArray(res.da
            this.waterPath = res.data.files[0].path;
          }
        })
      },
      // 选择视频文件
      selectVideoFile() {
        mediaPickerModule.video({
          suffix: [".mp4"],
          maxCount: 1
        }, res => {
          this.writeLog(JSON.stringify(res))
          if (res.code == 0 && res.data && res.data.files && Array.isArray(res.da
            this.videoPath = res.data.files[0].path;
          }
        })
      },
      // 视频转码
      transformVideo() {
        module.transformVideo({
          //源文件
          src: this.videoPath,
          // 目标文件
          target: this.dirPath + "转码后的视频.mp4"
        }, res => {
          this.writeLog(JSON.stringify(res))
        })
      },
      // 抽取视频
      extractVideo() {
        module.extractVideo({
          //源文件
          src: this.videoPath,
```

```
        // 目标文件
        target: this.dirPath + "抽取后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 音视频合成
    mixAudioVideo() {
      module.mixAudioVideo({
        //视频源文件
        videoSrc: this.dirPath + "抽取后的视频.mp4",
        //音频源文件
        audioSrc: this.dirPath + "抽取音频后的音频.aac",
        // 目标文件
        target: this.dirPath + "音视频合成后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频剪切
    cutVideo() {
      module.cutVideo({
        //源文件
        src: this.videoPath,
        // 目标文件
        target: this.dirPath + "剪切后的视频.mp4",
        // 剪切开始时间（单位：秒）
        startTime: 0,
        // 剪切时长（单位：秒）
        duration: 5,
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频拼接
    concatVideo() {
      module.concatVideo({
        //源文件
        src: this.videoPath,
        // 拼接的视频
        append: this.videoPath,
        // 目标文件
        target: this.dirPath + "拼接后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频封面
    screenShot() {
      module.screenShot({
```

```
        //源文件
        src: this.videoPath,
        // 目标文件
        target: this.dirPath + "视频的截图.jpg",
        // 截图宽度，可以不传，默认：0（按视频宽度）
        width: 0,
        // 截图高度，可以不传，默认：0（按视频高度）
        height: 0
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频某一帧的图片
    frame2Image() {
      module.frame2Image({
        //源文件
        src: this.videoPath,
        // 目标文件
        target: this.dirPath + "视频某一帧的图片的截图.jpg",
        // 某一帧的时间，格式：hh:mm:ss.xxx
        time: "00:00:05.123"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频转图片
    video2Image() {
      module.video2Image({
        //源文件
        src: this.videoPath,
        // 目标文件夹
        target: this.dirPath + "video2Image/",
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 添加水印图片
    waterMarkImage() {
      module.waterMarkImage({
        //源文件
        src: this.videoPath,
        //水印文件
        water: this.waterPath,
        // 目标文件夹
        target: this.dirPath + "添加水印后的视频.mp4",
        // 水印位置,1.左上角（默认），2.右上角，3.右下角，4.左下角
        position: 2
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
```

```
    },
    // 视频转gif
    video2Gif() {
      module.video2Gif({
        //源文件
        src: this.videoPath,
        // 目标文件
        target: this.dirPath + "视频转gif后的图片.gif",
        //开始时间，单位：秒
        startTime: 0,
        // 时长，单位：秒
        duration: 5
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 图片转视频
    image2Video() {
      module.image2Video({
        //源文件(图片列表的文件夹)
        src: this.dirPath + "video2Image/",
        // 目标文件
        target: this.dirPath + "图片转视频后的视频.mp4",
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 多画面拼接
    multiVideo() {
      module.multiVideo({
        //源文件
        src: this.videoPath,
        //要拼接的文件
        append: this.videoPath,
        // 目标文件
        target: this.dirPath + "多画面拼接后的视频.mp4",
        // 布局方向,1.横向拼接，2.垂直拼接
        direction: 1
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频反序倒播
    reverseVideo() {
      module.reverseVideo({
        //源文件
        src: this.videoPath,
        // 目标文件
        target: this.dirPath + "视频反序倒播后的视频.mp4"
      }, res => {
```

```
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频降噪
    denoiseVideo() {
      module.denoiseVideo({
        //源文件
        src: this.videoPath,
        // 目标文件
        target: this.dirPath + "视频降噪后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 画中画
    picInPicVideo() {
      module.picInPicVideo({
        //源文件
        src: this.videoPath,
        //画中画文件
        append: this.videoPath,
        // 画中画宽度
        width: 300,
        // 画中画高度
        height: 200,
        // 画中画位置,1.左上角(默认),2.右上角,3.右下角,4.左下角
        position: 2,
        // 目标文件
        target: this.dirPath + "画中画后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频倍数缩放
    videoScaleMultiple() {
      module.videoScaleMultiple({
        //源文件
        src: this.videoPath,
        // 类型,1.缩小,2.放大
        type: 1,
        // 缩放的倍数
        value: 2,
        // 目标文件
        target: this.dirPath + "视频倍数缩放后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频宽高缩放
    videoScaleWidthHeight() {
```

```
      module.videoScaleWidthHeight({
        //源文件
        src: this.videoPath,
        // 宽度，-1.高度等比例缩放
        width: 200,
        // 高度,-1.宽度等比例缩放
        height: 300,
        // 目标文件
        target: this.dirPath + "视频宽高缩放后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频倍数播放
    videoSpeed() {
      module.videoSpeed({
        //源文件
        src: this.videoPath,
        // 播放速度，取值范围：[0.25,4] 小于1快速播放，大于1慢速播放
        speed: 2,
        // 目标文件
        target: this.dirPath + "视频倍数播放后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频转YUV
    video2Yuv() {
      module.video2Yuv({
        //源文件
        src: this.videoPath,
        // 目标文件
        target: this.dirPath + "视频转YUV后的文件.yuv"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // YUV转H264
    yuv2H264() {
      module.yuv2H264({
        //源文件
        src: this.dirPath + "视频转YUV后的文件.yuv",
        // 转换后的宽度，默认：720
        width: 720,
        // 转换后的高度，默认：1280
        height: 1280,
        // 目标文件
        target: this.dirPath + "YUV转H264后的文件.h264"
      }, res => {
        this.writeLog(JSON.stringify(res))
```

```
        })
      },
      // 视频切片
      video2HLS() {
        module.video2HLS({
          //源文件
          src: this.videoPath,
          // 切片时长
          splitTime: 5,
          // 目标文件
          target: this.dirPath + "hls/切片文件.m3u8"
        }, res => {
          this.writeLog(JSON.stringify(res))
        })
      },
      // 切片合成视频
      hls2Video() {
        module.hls2Video({
          //切片文件
          src: this.dirPath + "hls/切片文件.m3u8",
          // 目标文件
          target: this.dirPath + "切片合成的视频后的视频.mp4"
        }, res => {
          this.writeLog(JSON.stringify(res))
        })
      },
      // 视频参数调节
      videoParams() {
        module.videoParams({
          //源文件
          src: this.videoPath,
          // 亮度，有效值[-1.0,1.0]，默认：0
          bright: 0.1,
          //对比度，有效值[-2.0,2.0]，默认：0
          contrast: 0.9,
          // 饱和度,有效值[0,3.0]，默认：1
          saturation: 3,
          // 目标文件
          target: this.dirPath + "视频参数调节后的视频.mp4"
        }, res => {
          this.writeLog(JSON.stringify(res))
        })
      },
      // 视频旋转
      videoRotation() {
        module.videoRotation({
          //源文件
          src: this.videoPath,
          // 旋转类型，1.顺时针旋转画面90度，2.逆时针旋转画面90度，3.顺时针旋转画面90度再水平
          transpose: 1,
```

```
        // 目标文件
        target: this.dirPath + "视频旋转后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频翻转
    videoFlip() {
      module.videoFlip({
        //源文件
        src: this.videoPath,
        // 翻转类型，1.水平翻转，2.垂直翻转
        type: 1,
        // 目标文件
        target: this.dirPath + "视频翻转后的视频.mp4"
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 获取视频信息
    videoInfo() {
      module.videoInfo({
        //源文件
        src: this.videoPath
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 视频压缩
    compress() {
      module.compress({
        //源文件
        src: this.videoPath,
        // 目标文件
        target: this.dirPath + "压缩后的视频.mp4",
        //帧率，默认原视频帧率
        fps: 30,
        // 码率，从0~500，越大码率越低，一般18~32效果较好，默认：30
        rate: 30,
        //分辨率，原视频分辨率
        size: [720, 1080]
      }, res => {
        this.writeLog(JSON.stringify(res))
      })
    },
    // 注册推流服务
    rtmpRegister() {
      module.rtmpRegister({
        url: "rtmp://195964.push.tlivecloud.com/live/leven?txSecret=cce20f82e2a7
      }, res => {
```

使用方法

```
            this.writeLog(JSON.stringify(res))
          })
        },
        // 开始推流
        rtmpStart() {
          module.rtmpStart({
            // path: this.dirPath + "压缩后的视频.mp4"
            path: "/storage/emulated/0/Android/data/test.leven.uniplugin.com/video/1
          }, res => {
            this.writeLog(JSON.stringify(res))
          })
        },
        // 关闭推流
        rtmpClose() {
          module.rtmpClose(res => {
            this.writeLog(JSON.stringify(res))
          })
        },
        // 写日志
        writeLog(str) {
          console.log(str);
          let logStr = uni.$lv.date.format(null, "yyyy-mm-dd hh:MM:ss") + " " + str
          this.logStr = logStr + this.logStr;
        }
      }
    }
</script>

<style>
  .audio-file-box {
    flex-direction: row;
    margin-bottom: 5px;
  }
</style>
```

更新日志

2024-07-2 `v1.1.3`

- [优化] 视频添加水印新增参数 `padding` 水印图片距离视频边距

2024-3-12 `v1.1.2`

- [优化] 修复amr文件无法转换的问题

2024-2-26 `v1.1.1`

- 去除不必要的插件引入

2024-02-20 `v1.1.0`

- 新增接口【视频压缩】
- 新增接口【注册推流服务】
- 新增接口【开始推流】
- 新增接口【关闭推流】

2024-01-29

首次发布

执行回调

回调示例

```
{
    "data": {
        "status": "start"
    },
    "message": "",
    "code": 0
}

{
    "data": {
        "status": "running",
        "progress": 1,
        "transformDuration": 1045938
    },
    "message": "",
    "code": 0
}

{
    "data": {
        "status": "cancel"
    },
    "message": "",
    "code": 0
}

{
    "data": {
        "status": "complete"
    },
    "message": "",
    "code": 0
}
```

回调说明

参数名	参数类型	参数描述
message	String	消息提示

执行回调

参数名	参数类型	参数描述
data	Object	数据对象
data.status	String	处理状态，start：开始处理，running：处理中，complete：处理完成，cancel：取消处理
data.progress	Float	处理进度，此参数只能作为参考，***部分接口需自己处理，公共方法无此参数返回***
data.transformDuration	Integer	当前处理的时长
data.duration	Integer	源文件的时长，***公共方法无此参数返回***
code	Integer	返回类型，0.成功，其他：失败

申请文件权限

方法名

`filePermission`

用法

- 用法如下：

```
module.filePermission(res => {
  console.log(res)
});
```

- 参数说明

  无

回调

- 示例

```
{
  "data": {},
  "message": "",
  "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

自定义指令

方法名

`customCommands`

用法

- 用法如下：

```
commonModule.customCommands({
  cmd: "ffmpeg -y -i /storage/emulated/0/DCIM/Camera/VID_20240101_092924.mp4 -
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
cmd	String	是	无	要执行的命令

回调

请参考回调说明

取消执行命令

方法名

`cancelCommands`

用法

- 用法如下：

```
commonModule.cancelCommands(res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
cmd	String	是	无	要执行的命令

回调

- 示例

```
{
  "data": {},
  "message": "",
  "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	数据对象
code	Integer	返回类型，0.成功，其他：失败

音频转码

方法名

`transformAudio`

用法

- 用法如下：

```
module.transformAudio({
  //源文件
  src: this.audioPath,
  // 目标文件
  target: this.dirPath + "转码后的音频.aac"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件

回调

请参考回调说明

音频剪切

方法名

`cutAudio`

用法

- 用法如下：

```
module.cutAudio({
  //类型，1.根据时长剪切，2.根据结束时间剪切
  type: 2,
  //源文件
  src: this.audioPath,
  // 目标文件
  target: this.dirPath + "剪切后的音频.mp3",
  //开始剪切时间
  startTime: 60,
  //时长（type=1时有效）
  duration: 60,
  //剪切结束时间（type=2时有效）
  endTime: 120
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
type	Integer	是	无	剪切类型，1.根据时长剪切，2.根据结束时间剪切
src	String	是	无	源文件
target	String	是	无	目标文件
startTime	Integer	否	无	开始剪切时间，单位：秒（type=2时有效）

参数名	参数类型	是否必填	默认值	参数描述
endTime	Integer	否	无	剪切结束时间，单位：秒（type=2时有效）
duration	Integer	否	无	剪切时长，单位：秒（type=1时有效）

回调

请参考回调说明

音频拼接

方法名

`concatAudio`

用法

- 用法如下：

```
module.concatAudio({
  //源文件
  src: this.audioPath,
  //拼接文件
  append: path,
  // 目标文件
  target: this.dirPath + "拼接后的音频.mp3"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
append	String	是	无	拼接文件

回调

请参考回调说明

混音

方法名

`mixAudio`

用法

- 用法如下：

```
module.mixAudio({
  //源文件
  src: this.audioPath,
  //混合文件
  mix: path,
  // 目标文件
  target: this.dirPath + "混音后的音频.mp3"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
mix	String	是	无	混音文件

回调

请参考回调说明

更改音频音量

方法名

`changeVolume`

用法

- 用法如下：

```
module.changeVolume({
  //类型,1.固定音量（单位DB），2.音量的倍数
  type: 2,
  //源文件
  src: this.audioPath,
  //音量值
  // value: 20,
  //音量值（不定音量需要传整形，音量的倍数可以是浮点型）
  value: 1.5,
  // 目标文件
  target: this.dirPath + "更改音频音量后的音频.mp3"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
type	Integer	是	无	类型,1.固定音量（单位DB），2.音量的倍数
src	String	是	无	源文件
target	String	是	无	目标文件
value	Integer/Float	是	无	音量值（固定音量需要传整形，音量的倍数可以是浮点型）

回调

更改音频音量

请参考回调说明

从视频文件抽取音频

方法名

`extractAudio`

用法

- 用法如下：

```
module.extractAudio({
  //源文件
  src: this.videoPath,
  // 目标文件(只能是aac格式)
  target: this.dirPath + "抽取音频后的音频.aac"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件

回调

请参考回调说明

音频解码PCM

方法名

`extractAudio`

用法

- 用法如下：

```
module.decodeAudio({
  //源文件
  src: this.audioPath,
  // 目标文件(只能是pcm格式)
  target: this.dirPath + "解码后的音频.pcm",
  //采样率
  sampleRate: 44100,
  // 声道:1.单声道，2.立体声道
  channel: 2
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件，**只能是pcm格式**
sampleRate	Integer	否	44100	采样率
channel	Integer	否	2	声道:1.单声道，2.立体声道

回调

请参考回调说明

音频编码

方法名

`encodeAudio`

用法

- 用法如下：

```
module.encodeAudio({
  //源文件(只能是pcm格式)
  src: this.dirPath + "解码后的音频.pcm",
  // 目标文件
  target: this.dirPath + "编码后的音频.wav",
  //采样率
  sampleRate: 44100,
  // 声道:1.单声道，2.立体声道
  channel: 2
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件，**只能是pcm格式**
target	String	是	无	目标文件
sampleRate	Integer	否	44100	采样率
channel	Integer	否	2	声道:1.单声道，2.立体声道

回调

请参考回调说明

音频淡入

方法名

`audioFadeIn`

用法

- 用法如下：

```
module.audioFadeIn({
  //源文件
  src: this.audioPath,
  // 目标文件
  target: this.dirPath + "音频淡入后的音频.mp3",
  //淡入时长，单位：秒
  duration: 5
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
duration	Integer	是	无	淡入时长，单位：秒

回调

请参考回调说明

音频淡出

方法名

`audioFadeOut`

用法

- 用法如下：

```
module.audioFadeOut({
  //源文件
  src: this.audioPath,
  // 目标文件
  target: this.dirPath + "音频淡出后的音频.mp3",
  //淡出时长，单位：秒
  duration: 10
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
duration	Integer	是	无	淡出时长，单位：秒

回调

请参考回调说明

音频转amr

方法名

`audioFadeOut`

用法

- 用法如下：

```
module.audio2Amr({
  //源文件
  src: this.audioPath,
  // 目标文件
  target: this.dirPath + "音频转amr后的音频.amr"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件

回调

请参考回调说明

获取音频信息

方法名

`audioInfo`

用法

- 用法如下：

```
module.audioInfo({
  //源文件
  src: this.audioPath
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件

回调

- 示例

```
{
    "data": {
        "sampleRate": 44100,
        "bitRate": 320000,
        "channels": 2,
        "duration": 240354
    },
    "message": "",
    "code": 0
}
```

- 回调说明：

获取音频信息

参数名	参数类型	参数描述
message	String	消息提示
data.sampleRate	Integer	采样率
data.bitRate	Integer	比特率
data.channels	Integer	声道数
data.duration	Integer	时长，单位：毫秒
code	Integer	返回类型，0.成功，其他：失败

视频转码

方法名

`transformVideo`

用法

- 用法如下：

```
module.transformVideo({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "转码后的视频.avi"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件

回调

请参考回调说明

从视频文件抽取视频（无声音）

方法名

`extractVideo`

用法

- 用法如下：

```
module.extractVideo({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "抽取后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件

回调

请参考回调说明

音视频合成

方法名

`mixAudioVideo`

用法

- 用法如下：

```
module.mixAudioVideo({
  //视频源文件
  videoSrc: this.dirPath + "抽取后的视频.mp4",
  //音频源文件
  audioSrc: this.dirPath + "抽取音频后的音频.aac",
  // 目标文件
  target: this.dirPath + "音视频合成后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
videoSrc	String	是	无	视频源文件
audioSrc	String	是	无	音频源文件
target	String	是	无	目标文件

回调

请参考回调说明

视频剪切

方法名

`cutVideo`

用法

- 用法如下：

```
module.cutVideo({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "剪切后的视频.mp4",
  // 剪切开始时间（单位：秒）
  startTime: 0,
  // 剪切时长（单位：秒）
  duration: 5,
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
startTime	Integer	是	无	剪切开始时间（单位：秒）
duration	Integer	是	无	剪切时长（单位：秒）

回调

请参考回调说明

视频拼接

方法名

`concatVideo`

用法

- 用法如下：

```
module.concatVideo({
  //源文件
  src: this.videoPath,
  // 拼接的视频
  append: this.videoPath,
  // 目标文件
  target: this.dirPath + "拼接后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
append	Integer	是	无	拼接的视频

回调

请参考回调说明

视频封面

方法名

`screenShot`

用法

- 用法如下：

```
module.screenShot({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "视频的截图.jpg",
  // 截图宽度，可以不传，默认：0（按视频宽度）
  width: 0,
  // 截图高度，可以不传，默认：0（按视频高度）
  height: 0
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
width	Integer	否	无	截图宽度，可以不传，默认：0（按视频宽度）
height	Integer	否	无	截图高度，可以不传，默认：0（按视频高度）

回调

请参考回调说明

视频某一帧的图片

方法名

`frame2Image`

用法

- 用法如下：

```
module.frame2Image({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "视频某一帧的图片的截图.jpg",
  // 某一帧的时间，格式：hh:mm:ss.xxx
  time: "00:00:05.123"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
time	Integer	是	无	某一帧的时间，格式：hh:mm.xxx

回调

请参考回调说明

视频转图片

方法名

`video2Image`

用法

- 用法如下：

```
module.video2Image({
  //源文件
  src: this.videoPath,
  // 目标文件夹
  target: this.dirPath + "video2Image/",
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件夹

回调

请参考回调说明

添加水印图片

方法名

`waterMarkImage`

用法

- 用法如下：

```
module.waterMarkImage({
  //源文件
  src: this.videoPath,
  //水印文件
  water: this.waterPath,
  // 目标文件
  target: this.dirPath + "添加水印后的视频.mp4",
  // 水印位置,1.左上角（默认），2.右上角，3.右下角，4.左下角
  position: 2,
  //水印距离视频边距，默认：40
  padding: 100
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
water	String	是	无	水印文件
position	Integer	否	1	水印位置,1.左上角（默认），2.右上角，3.右下角，4.左下角
padding `v1.1.3`	Integer	否	40	水印距离视频边距

回调

请参考回调说明

视频转成Gif

方法名

`video2Gif`

用法

- 用法如下：

```
module.video2Gif({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "视频转gif后的图片.gif",
  //开始时间，单位：秒
  startTime: 0,
  // 时长，单位：秒
  duration: 5
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
startTime	Integer	是	无	开始时间，单位：秒
duration	Integer	是	无	时长

回调

请参考回调说明

图片合成视频

方法名

`image2Video`

用法

- 用法如下：

```
module.image2Video({
  //源文件(图片列表的文件夹)
  src: this.dirPath + "video2Image/",
  // 目标文件
  target: this.dirPath + "图片转视频后的视频.mp4",
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件(图片列表的文件夹)
target	String	是	无	目标文件

回调

请参考回调说明

多画面拼接

方法名

`multiVideo`

用法

- 用法如下：

```
module.multiVideo({
  //源文件
  src: this.videoPath,
  //要拼接的文件
  append: this.videoPath,
  // 目标文件
  target: this.dirPath + "多画面拼接后的视频.mp4",
  // 布局方向,1.横向拼接，2.垂直拼接
  direction: 1
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
append	String	是	无	要拼接的文件
direction	Integer	是	无	布局方向,1.横向拼接，2.垂直拼接

回调

请参考回调说明

视频反序倒播

方法名

`reverseVideo`

用法

- 用法如下：

```
module.reverseVideo({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "视频反序倒播后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件

回调

请参考回调说明

视频降噪

方法名

`denoiseVideo`

用法

- 用法如下：

```
module.denoiseVideo({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "视频降噪后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件

回调

请参考回调说明

画中画

方法名

`picInPicVideo`

用法

- 用法如下：

```
module.picInPicVideo({
  //源文件
  src: this.videoPath,
  //画中画文件
  append: this.videoPath,
  // 画中画宽度
  width: 300,
  // 画中画高度
  height: 200,
  // 画中画位置,1.左上角（默认），2.右上角，3.右下角，4.左下角
  position: 2,
  // 目标文件
  target: this.dirPath + "画中画后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
append	String	是	无	画中画文件
width	Integer	是	无	画中画宽度
height	Integer	是	无	画中画高度
position	Integer	否	1	画中画位置,1.左上角（默认），2.右上角，3.右下角，4.左下角

回调

请参考回调说明

视频倍数缩放

方法名

`videoScaleMultiple`

用法

- 用法如下：

```
module.videoScaleMultiple({
  //源文件
  src: this.videoPath,
  // 类型，1.缩小，2.放大
  type: 1,
  // 缩放的倍数
  value: 2,
  // 目标文件
  target: this.dirPath + "视频倍数缩放后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
type	Integer	是	无	类型，1.缩小，2.放大
value	Integer	是	无	缩放的倍数

回调

请参考回调说明

视频宽高缩放

方法名

`videoScaleWidthHeight`

用法

- 用法如下：

```
module.videoScaleWidthHeight({
  //源文件
  src: this.videoPath,
  // 宽度，-1.高度等比例缩放
  width: 200,
  // 高度,-1.宽度等比例缩放
  height: 300,
  // 目标文件
  target: this.dirPath + "视频宽高缩放后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
width	Integer	是	无	宽度，-1.高度等比例缩放
height	Integer	是	无	高度,-1.宽度等比例缩放

回调

请参考回调说明

视频倍速播放

方法名

`videoSpeed`

用法

- 用法如下：

```
module.videoSpeed({
  //源文件
  src: this.videoPath,
  // 播放速度，取值范围：[0.25,4] 小于1快速播放，大于1慢速播放
  speed: 2,
  // 目标文件
  target: this.dirPath + "视频倍数播放后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
speed	Float	是	无	播放速度，取值范围：[0.25,4] 小于1快速播放，大于1慢速播放

回调

请参考回调说明

视频转yuv

方法名

`video2Yuv`

用法

- 用法如下：

```
module.video2Yuv({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "视频转YUV后的文件.yuv"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件

回调

请参考回调说明

YUV转H264

方法名

`yuv2H264`

用法

- 用法如下：

```
module.yuv2H264({
  //源文件
  src: this.dirPath + "视频转YUV后的文件.yuv",
  // 转换后的宽度，默认：720
  width: 720,
  // 转换后的高度，默认：1280
  height: 1280,
  // 目标文件
  target: this.dirPath + "YUV转H264后的文件.h264"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
width	Integer	否	720	转换后的宽度，默认：720
height	Integer	否	1280	转换后的高度，默认：1280

回调

请参考回调说明

视频切片

方法名

`video2HLS`

用法

- 用法如下：

```
module.video2HLS({
  //源文件
  src: this.videoPath,
  // 切片时长
  splitTime: 5,
  // 目标文件
  target: this.dirPath + "hls/切片文件.m3u8"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件
target	String	是	无	目标文件
splitTime	Integer	是	无	切片时长，单位：秒

回调

请参考回调说明

切片合成视频

方法名

`hls2Video`

用法

- 用法如下：

```
module.hls2Video({
  //切片文件
  src: this.dirPath + "hls/切片文件.m3u8",
  // 目标文件
  target: this.dirPath + "切片合成的视频后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	切片文件
target	String	是	无	目标文件

回调

请参考回调说明

参数调节

方法名

`videoParams`

用法

- 用法如下：

```
module.videoParams({
  //源文件
  src: this.videoPath,
  // 亮度，有效值[-1.0,1.0]，默认：0
  bright: 0.1,
  //对比度，有效值[-2.0,2.0]，默认：0
  contrast: 0.9,
  // 饱和度,有效值[0,3.0]，默认：1
  saturation: 3,
  // 目标文件
  target: this.dirPath + "视频参数调节后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	切片文件
target	String	是	无	目标文件
bright	Float	否	0	亮度，有效值[-1.0,1.0]，默认：0
contrast	Float	否	0	对比度，有效值[-2.0,2.0]，默认：0
saturation	Float	否	1	饱和度,有效值[0,3.0]，默认：1

回调

请参考回调说明

视频旋转

方法名

`videoRotation`

用法

- 用法如下：

```
module.videoRotation({
  //源文件
  src: this.videoPath,
  // 旋转类型，1.顺时针旋转画面90度，2.逆时针旋转画面90度，3.顺时针旋转画面90度再水平翻转，
  transpose: 1,
  // 目标文件
  target: this.dirPath + "视频旋转后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	切片文件
target	String	是	无	目标文件
transpose	Integer	是	无	旋转类型，1.顺时针旋转画面90度，2.逆时针旋转画面90度，3.顺时针旋转画面90度再水平翻转， 0.逆时针旋转画面90度水平翻转

回调

视频旋转

请参考回调说明

视频翻转

方法名

`videoFlip`

用法

- 用法如下：

```
module.videoFlip({
  //源文件
  src: this.videoPath,
  // 翻转类型，1.水平翻转，2.垂直翻转
  type: 1,
  // 目标文件
  target: this.dirPath + "视频翻转后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	切片文件
target	String	是	无	目标文件
type	Integer	是	无	翻转类型，1.水平翻转，2.垂直翻转

回调

请参考回调说明

获取视频信息

方法名

`videoInfo`

用法

- 用法如下：

```
module.videoInfo({
  //源文件
  src: this.videoPath
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	源文件

回调

- 示例

```
{
    "data": {
        "height": 1080,
        "fps": 29,
        "duration": 8454,
        "width": 2336,
        "sampleRate": 48000,
        "bitRate": 19796006,
        "channels": 2
    },
    "message": "",
    "code": 0
}
```

- 回调说明：

获取视频信息

参数名	参数类型	参数描述
message	String	消息提示
data.width	Integer	视频宽度
data.height	Integer	视频高度
data.fps	Integer	视频帧率
data.sampleRate	Integer	采样率
data.bitRate	Integer	比特率
data.channels	Integer	声道数
data.duration	Integer	时长，单位：毫秒
code	Integer	返回类型，0.成功，其他：失败

视频压缩 v1.1.0

方法名

`compress` `v1.1.0`

用法

- 用法如下：

```
module.compress({
  //源文件
  src: this.videoPath,
  // 目标文件
  target: this.dirPath + "压缩后的视频.mp4",
  //帧率，默认原视频帧率
  fps: 30,
  // 码率，从0～500，越大码率越低，一般18～32效果较好，默认：30
  rate: 30,
  //分辨率，原视频分辨率
  size: [720, 1080]
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
src	String	是	无	视频文件
target	String	是	无	目标文件
fps	Integer	否	原视频帧率	帧率
rate	Integer	否	30	码率
size	Integer[]	否	原视频分辨率	分辨率

回调

请参考回调说明

注册推流服务 v1.1.0

方法名

`rtmpRegister` `v1.1.0`

用法

- 用法如下：

```
module.rtmpRegister({
  url: "rtmp://195964.push.tlivecloud.com/live/aa?txSecret=7dcbecaa3eff10bea87
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
url	String	是	无	推流地址

回调

- 示例

```
{
  "data": {
    "status": "onConnecting"
  },
  "message": "",
  "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	对象数据

参数名	参数类型	参数描述
data.status	Integer	推流服务器连接状态，onConnecting：连接中，onConnect：连接成功，onPusher：推流中，onStop：停止推流
code	Integer	返回类型，0.成功，其他：失败

开始推流 v1.0.0

方法名

`rtmpStart` `v1.1.0`

用法

- 用法如下：

```
module.rtmpStart({
  path: this.dirPath + "压缩后的视频.mp4"
}, res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明

参数名	参数类型	是否必填	默认值	参数描述
path	String	是	无	推流文件路径

回调

- 示例

```
{
  "data": {},
  "message": "",
  "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	对象数据
code	Integer	返回类型，0.成功，其他：失败

关闭推流 v1.1.0

方法名

`rtmpClose` `v1.1.0`

用法

- 用法如下：

```
module.rtmpClose(res => {
  this.writeLog(JSON.stringify(res))
})
```

- 参数说明
  无

回调

- 示例

```
{
  "data": {},
  "message": "",
  "code": 0
}
```

- 回调说明：

参数名	参数类型	参数描述
message	String	消息提示
data	Object	对象数据
code	Integer	返回类型，0.成功，其他：失败