

uniapp 安卓VLC视频播放器UTS原生插件

目 录

使用说明	1
使用方法	1.1
更新日志	1.2
API	2
申请插件所需权限	2.1
组件	3
使用方法	3.1
初始化配置	3.2
组件事件	3.3
onPlayStatus	3.3.1
onError	3.3.2
组件方法	3.4
重新播放	3.4.1
暂停播放	3.4.2
继续播放	3.4.3
停止播放	3.4.4
设置视频播放比例	3.4.5
获取播放属性	3.4.6
设置音量	3.4.7
设置播放进度	3.4.8
设置播放速度	3.4.9
截图	3.4.10
开始录制视频	3.4.11
结束录制视频	3.4.12
是否正在播放	3.4.13
全屏切换	3.4.14

使用方法

介绍

安卓VLC视频播放器UTS原生插件支持播放本地视频，assets目录下视频文件，网络视频，直播等多种视频，该插件延迟较低，各种功能齐全欢迎使用，插件UTS开发支持uniapp和uniapp x

插件使用注意事项

1. 示例文件不包含插件，需要您在插件首页点击“试用”导入插件到项目中，插件导入后打自定义基座，运行项目的时候选择运行到基座即可
2. 组件只能在nvue/uvue页面中使用，不支持vue页面

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=21241>

示例文件下载

1. 插件首页下载（推荐）

通过“使用 HBuilderX 导入示例项目”可直接创建uniapp项目

2. 本地下载，[点击这里](#)即可下载

使用方法

演示程序下载

加群下载演示程序



uniapp leven 系列...

群号: 106179987



权限

1. android.permission.READ_EXTERNAL_STORAGE
2. android.permission.WRITE_EXTERNAL_STORAGE

API使用方法

在需要使用插件的页面加载以下代码

```
import * as module from "@uni_modules/leven-uts-vlcPlayer"
```

组件使用方法

使用方法

在需要使用插件的页面加载以下代码

```
<leven-uts-vlcPlayer ref="refVideoPlayer" style="flex:1; height: 300px;" :config='  
    </leven-uts-vlcPlayer>
```

具体的使用示例可参考示例文件

更新日志

2024-12-03

首次发布

申请插件所需权限

方法名

requestPermissions

用法

- 用法如下：

```
module.requestPermissions(res => {
  console.log(JSON.stringify(res))
});
```

- 参数说明

无

回调

- 示例

```
{
  "data": {
    "grantedList": [
      "android.permission.READ_EXTERNAL_STORAGE",
      "android.permission.WRITE_EXTERNAL_STORAGE"
    ]
  },
  "message": "权限被拒绝",
  "code": -1
}
```

- 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

参数名	参数类型	参数描述
data.grantedList	Array	申请的权限列表，被拒绝的时候会返回
code	Integer	返回类型，0.成功，其他：失败

使用方法

组件使用

在需要使用插件的页面加载以下代码

```
<leven-uts-vlcPlayer ref="refVideoPlayer" style="flex:1; height: 300px;" :config='  
</leven-uts-vlcPlayer>
```

组件只能在nvue/uvue页面中使用，不支持vue页面，config参数必传，具体请参考【[初始化配置](#)】

初始化配置

config

组件初始化配置，必传参数

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	视频播放地址或文件、文件路径，如果是内部存储文件需要传绝对路径，assets文件夹下的视频文件直接传文件名即可
sourceType	String	否	URL	视频播放源， URL：播放的是网络或直播视频， ASSETS：播放的是app内置assets文件夹下的视频文件， PATH：播放的是设备内部存储的视频文件
isAutoPlay	Boolean	否	true	是否自动播放

参数名	参数类型	是否必填	默认值	参数描述
scaleType	String	否	SURFACE_BEST_FIT	视频播放比例，可选值： SURFACE_4_3， SURFACE_16_9， SURFACE_BEST_FIT， SURFACE_FIT_SCREEN， SURFACE_FILL， SURFACE_ORIGINAL
options	Array[String]	否	无	初始化播放器模式，具体参数设置可自行搜索libvlc参数
imageDir	String	否	无	自定义截图保存文件夹，为空或不传该参数会默认保存到Pictures中
videoDir	String	否	无	自定义录制视频保存文件夹，为空或不传该参数会默认保存到Movies中
mediaOptions	Array[String]	否	无	媒体播放参数，具体参数设置可自行搜索libvlc media参数
useSurfaceView	Boolean	否	false	是否使用surfaceView播放视频，建议将参数设置为true，如果设置为false，播放器将以TextureView进行播放，重新播放时可能导致黑屏

初始化配置

参数名	参数类型	是否必填	默认值	参数描述
decoderEnabled	Boolean	否	false	是否使用硬解码
decoderForce	Boolean	否	false	是否强制使用硬解码

组件事件

onPlayStatus

介绍

播放器播放状态事件

用法

```
<leven-uts-vlcPlayer ref="refVideoPlayer" style="flex:1; height: 300px;" :config='
</leven-uts-vlcPlayer>
```

回调示例

```
onPlayStatus(e) {
  console.log(e.detail)
}
```

回调内容

```
{
  "status": "Playing",
  "time": 0,
  "length": 144410
}
```

回调说明

参数名	参数类型	参数描述
time	Integer	当前播放的时长，单位：毫秒
length	Integer	视频总时长，单位：毫秒
status	String	播放状态，可参考下方说明

onPlayStatus

status播放状态说明：

Playing：播放中

Buffering：正在缓存

EncounteredError：播放错误

MediaChanged：媒体改变

EndReached：到达播放列表的结尾，播放结束

ESAdded：添加了播放通道

ESDeleted：播放通道被删除

ESSelected：播放通道被选择

LengthChanged：总时长更新

Opening：播放器打开成功

PausableChanged：暂停状态被改变

Paused：播放暂停

PositionChanged：播放位置更新

RecordChanged：录制更新

SeekableChanged：搜索标记改变

Stopped：播放停止

TimeChanged：播放时长更新

Vout：输出的视频数量改变

onError

介绍

错误事件

用法

```
<leven-uts-vlcPlayer ref="refVideoPlayer" style="flex:1; height: 300px;" :config='
</leven-uts-vlcPlayer>
```

回调示例

```
onError(e) {
    console.log(e.detail)
}
```

回调内容

```
{
  "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

组件方法

重新播放

方法名

play

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.play(this.config, res => {
    this.writeLog("开始播放:" + 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	视频播放地址或文件、文件路径，如果是内部存储文件需要传绝对路径，assets文件夹下的视频文件直接传文件名即可
sourceType	String	否	URL	视频播放源，URL：播放的是网络或直播视频，ASSETS：播放的是app内置assets文件夹下的视频文件，PATH：播放的是设备内部存储的视频文件

参数名	参数类型	是否必填	默认值	参数描述
scaleType	String	否	SURFACE_BEST_FIT	视频播放比例， 可选值： SURFACE_4_3， SURFACE_16_9， SURFACE_BEST_FIT， SURFACE_FIT_SCREEN， SURFACE_FILL， SURFACE_ORIGINAL
options	Array[String]	否	无	初始化播放器模式，具体参数设置可自行搜索libvlc参数
mediaOptions	Array[String]	否	无	媒体播放参数，具体参数设置可自行搜索libvlc media参数
useSurfaceView	Boolean	否	false	是否使用surfaceView播放视频，建议将参数设置为true，如果设置为false，播放器将以TextureView进行播放，重新播放时可能导致黑屏
decoderEnabled	Boolean	否	false	是否使用硬解码
decoderForce	Boolean	否	false	是否强制使用硬解码

回调

重新播放

- 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

暂停播放

方法名

pause

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.pause(res => {
    this.writeLog("暂停播放:" + 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	数据对象
code	Integer	返回类型，0.成功，其他：失败

继续播放

方法名

resume

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.resume(res => {
    this.writeLog("继续播放:" + 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	数据对象
code	Integer	返回类型，0.成功，其他：失败

停止播放

方法名

stop

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.stop(res => {
    this.writeLog("停止播放:" + 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	数据对象
code	Integer	返回类型，0.成功，其他：失败

设置视频播放比例

方法名

setVideoScale

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.setVideoScale({
    scaleType: "SURFACE_16_9"
  }, res => {
    this.writeLog("开始播放:" + 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
scaleType	String	是	无	视频播放比例， 可选值： SURFACE_4_3， SURFACE_16_9 , SURFACE_BEST_FIT， SURFACE_FIT_SCREEN， SURFACE_FILL， SURFACE_ORIGINAL

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

获取播放属性

方法名

getPlayerData

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {  
  this.$refs.refVideoPlayer.getPlayerData(res => {  
    this.writeLog("获取播放器数据:" + JSON.stringify(res))  
  });  
}
```

- 参数说明

无

回调

- 示例

```
{  
  "data": {  
    "time": 2501,  
    "length": 144410,  
    "scaleType": "SURFACE_BEST_FIT",  
    "url": "http://alvideo.ippzone.com/zyvd/98/90/b753-55fe-11e9-b0d8-00163e0c",  
    "isPlaying": true,  
    "volume": 100,  
    "vlcVersion": "3.0.9 Vetinari"  
  },  
  "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
data.time	Integer	当前播放的时长，单位：毫秒
data.length	Integer	视频总时长，单位：毫秒
data.scaleType	String	视频播放比例
data.url	String	视频地址
data.isPlaying	Boolean	是否正在播放
data.volume	Integer	视频音量
data.vlcVersion	String	vlc版本号
code	Integer	返回类型，0.成功，其他：失败

设置音量

方法名

setVolume

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.setVolume({
    volume: 10,
  }, res => {
    this.writeLog("设置音量:" + 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
volume	Integer	是	无	音量值

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

设置音量

参数名	参数类型	参数描述
code	Integer	返回类型，0.成功，其他：失败

设置播放进度

方法名

setTime

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.setTime({
    time: 15000,
  }, res => {
    this.writeLog("设置播放进度:" + 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
time	Integer	是	无	播放进度，单位：毫秒

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

设置播放进度

参数名	参数类型	参数描述
code	Integer	返回类型，0.成功，其他：失败

设置播放速度

方法名

setRate

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.setRate({
    rate: 2.5,
  }, res => {
    this.writeLog("设置播放进度:" + 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
rate	float	是	无	播放速度，1：正常播放

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

设置播放速度

参数名	参数类型	参数描述
code	Integer	返回类型，0.成功，其他：失败

截图

方法名

takeSnapshot

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.takeSnapshot(res => {
    this.writeLog("截图:" + JSON.stringify(res))
  });
}
```

- 参数说明

无

回调

- 示例

```
{
  "data": {
    "path": "/storage/emulated/0/Pictures/LevenVlc/1733220514974.jpg"
  },
  "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
data.path	String	图片保存路径
code	Integer	返回类型，0.成功，其他：失败

开始录制视频

方法名

startRecording

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.startRecording(res => {
    this.writeLog("开始录制视频:" + 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	数据对象
code	Integer	返回类型，0.成功，其他：失败

结束录制视频

方法名

stopRecording

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.stopRecording(res => {
    this.writeLog("结束录制视频:" + JSON.stringify(res))
  });
}
```

- 参数说明

无

回调

- 示例

```
{
  "data": {
    "path": "/storage/emulated/0/Movies/LevenVlc/1733220524315.mp4"
  },
  "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
data.path	String	视频文件保存路径
code	Integer	返回类型，0.成功，其他：失败

是否正在播放

方法名

isPlaying

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.isPlaying(res => {
    this.writeLog("是否正在播放:" + JSON.stringify(res))
  });
}
```

- 参数说明

无

回调

- 示例

```
{
  "data": {
    "isPlaying": true
  },
  "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
data.isPlaying	Boolean	是否正在播放
code	Integer	返回类型，0.成功，其他：失败

全屏切换

方法名

changeScreenDirection

用法

- 用法如下：

```
if (this.$refs.refVideoPlayer) {
  this.$refs.refVideoPlayer.changeScreenDirection(res => {
    this.writeLog("全屏:" + JSON.stringify(res))

    //10秒后退出全屏
    setTimeout(() => {
      this.$refs.refVideoPlayer.changeScreenDirection(res1 => {
        this.writeLog("退出全屏:" + JSON.stringify(res))
      })
    }, 10000)
  });
}
```

- 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

全屏切换

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