

## PC 端 Polyv. Player. DotNet. SDK 文档

### ◇ 简要说明:

1. 播放和下载 sdk 依赖环境为 .net 4.5
  2. PolyvPlayerSDK.dll 是 c#/wpf 播放逻辑库,
  3. PolyvLocal.dll 为 c# 下载类库,
  4. PolyvCore.dll 为 c/c++ 的密钥下载库,
  5. 所有库文件在 lib 文件夹内。
  6. 依赖库: ./ 代表程序运行目录,
    - ./PolyvPlayerSDK.dll 播放逻辑核心库
    - ./PolyvCore.dll 密钥下载库
    - ./PolyvLocal.dll 视频下载库
    - ./libcurl.dll curl 网络库
    - ./ICSharpCode.SharpZipLib.dll 压缩/解压缩库
    - ./Newtonsoft.Json.dll json 解析库
    - ./SoundTouch.dll 音频处理库
- 以下为运行目录中 ffmpeg 目录下的 dll
- ./ffmpeg/avcodec-57.dll
  - ./ffmpeg/avdevice-57.dll
  - ./ffmpeg/avfilter-6.dll
  - ./ffmpeg/avformat-57.dll
  - ./ffmpeg/avutil-55.dll
  - ./ffmpeg/postproc-54.dll
  - ./ffmpeg/swresample-2.dll
  - ./ffmpeg/swscale-4.dll
  - ./ffmpeg/ffmpeg.exe
  - ./ffmpeg/ffplay.exe
  - ./ffmpeg/ffprobe.exe
  - ./ffmpeg/libiconv-2.dll
  - ./ffmpeg/libgcc\_s\_dw2-1.dll

### ◇ 播放模块接口释义

#### 一、用户信息初始化

##### 1、用户管理后台 API 信息初始化

int initAccountInfo(string userId, string secretKey, string readToken);

- 使用方法: 初始化播放器时调用该接口;
- 传递参数: userId, secretKey, readToken
- 返回值:
  - 0: 成功
  - 1: 失败, 有参数为空

注意: PolyvPlayerSDK.dll 播放库和 PolyvLocal.dll 下载库都有 initAccountInfo 接口, 使用播放或下载功能, 都需要调用 initAccountInfo 初始化参数;

##### 2、用户信息初始化

`int initPlayerInfo(string viewerIp, string viewerId, string viewerName);`

- 使用方法：在播放视频前调用该接口；
- 传递参数：**viewerIp**(用户 ip，不能为空)，**viewerId**(用户自定义 ip)，**viewerName**(用户昵称)
- 返回值：  
0：成功  
-1：失败，**viewerIp** 为空

## 二、加载视频源

### 1、加载视频接口

`int PlayVideo(string videoId, int bitRate, int playType, string directPath);`

- 使用方法：加载视频源时调用该接口；
- 传递参数：  
**videoId**(视频 vid)  
**bitRate**(码率)：0(在线：管理后台设置的码率/本地：下载的最高码率)，1(标清)，2(高清)，3(超清)  
**playType**(播放模式),0(在线)，1(本地)  
**directPath**(下载视频存放的目录 e.g: `string directPath = "D:/视频目录/";`)
- 返回值：  
3:播放超清，2:播放高清，1:播放标清，-1:参数错误,-2:流量超标，-3:账号过期,-4:视频信息获取失败,-5:没有对应码率,-6:网络异常；
- 说明：  
播放在线视频时：如果 **bitRate** 为 3，但是后台只有 1 和 2，会自动降级加载 2，以此类推如果只有 1 则会自动加载 1。返回值 1、2、3 表示加载的对应码率。

播放本地视频：首先在本地查找 **bitRate** 对应的视频，没有该码率视频则查找低码率的。返回值 1、2、3 表示加载的对应码率。(3 没有找 2，2 没有找 1，1 也没有返回-5)

## 三、播放控制

视频加载成功后可以进行暂停、停止、拖放等控制

- 1、**Play()** 播放
- 2、**Pause()** 暂停
- 3、**Stop()** 停止
- 4、**Close()** 关闭视频源
- 5、**seek** 跳转,改变 **Position** 值

## 四．播放回调

播放模块提供相应回调函数，播放结束、播放异常、缓冲开始/结束、**Seek** 开始/结束、视频加载中、加载完成

`public event RoutedEventHandler BufferingEnded;`

缓冲结束

`public event RoutedEventHandler BufferingStarted;`

缓冲开始

public event RoutedEventHandler SeekingStarted;  
Seek 开始

public event RoutedEventHandler SeekingEnded;  
Seek 结束

public event RoutedEventHandler MediaEnded;  
播放结束

public event EventHandler<ExceptionRoutedEventArgs> MediaFailed;  
播放失败或播放中出现异常返回值为 `ex = "BadDeviceId calling waveOutOpen"` 表示音频设备异常

public event RoutedEventHandler MediaOpened;  
视频加载完成

public event EventHandler<MediaOpeningRoutedEventArgs> MediaOpening;  
视频正在加载

## 五、音量控制

### 1、静音属性 IsMuted

public bool IsMuted;  
默认静音属性为 false, IsMuted = false;

### 2、音量大小控制属性 Volume，修改 Volume 值即可改变音量大小

public const double DefaultVolume = 1.0d;  
public const double MaxVolume = 1.0d;  
public const double MinVolume = 0.0d;  
默认音量为最大值，Volume = DefaultVolume;

## 六、倍速控制

倍速属性 SpeedRatio，修改 SpeedRatio 值即可改变播放速度

public const double DefaultSpeedRatio = 1.0d;  
public const double MinSpeedRatio = 0.0d;  
public const double MaxSpeedRatio = 2.0d;  
默认倍速为 1.0d(正常)，SpeedRatio = Constants.DefaultSpeedRatio;

## 七、跑马灯

跑马灯功能只在视频播放的时候有效

### 1. 初始化跑马灯接口

```
public void InitScrollText(string content, Single fontSize, Color fontColor);
```

传递参数:

content: 跑马灯显示的内容

fontSize: 字体大小

fontColor: 字体颜色

2. 开始显示跑马灯接口

```
public void StartDrawScrollText();
```

3. 停止显示跑马灯接口

```
public void StopDrawScrollText();
```

## 八、播放其他参数

1、string MediaFormat 媒体格式

2、Duration NaturalDuration 视频总时长

3、bool CanPause 可以暂停

4、bool IsLiveStream 是否直播流

5、bool IsSeekable 是否可以 seek

6、bool IsPlaying 是否正在播放

7、bool HasMediaEnded 是否播放结束

8、bool IsBuffering 是否正在缓冲

9、double BufferingProgress 缓冲进度 0-1 百分比

10、int BufferCacheLength 缓冲区最大值

11、double DownloadProgress 下载进度 0-1 百分比

12、int DownloadCacheLength 下载缓存最大值

13、bool IsOpening 是否正在打开视频

14、bool IsOpen 是否已经打开视频

15、Position 当前播放时间

16、endPosition 播放结束时间

17、播放状态 MediaState: Play,Close,Pause,Stop,Manual

18、StayInVideoTime 当前视频播放时间, 包含视频缓冲时间

19、RealPlayVideoTime 当前视频播放时间, 不包含视频缓冲时间

## ✧ 下载模块接口释义

### 1、用户管理后台 API 信息初始化

```
int initAccountInfo(string userId, string secretKey, string readToken);
```

➤ 使用方法: 初始化播放器时调用该接口;

➤ 传递参数: userId, secretKey, readToken

➤ 返回值:

0: 成功

-1: 失败, 有参数为空

### 2、下载接口

`int downloadVideo(string videoid, int bitRate, string directPath);`

➤ 使用方法：下载视频时调用该接口

➤ 传递参数：

videoid(视频 vid)

bitRate(码率)，1(标清)，2(高清)，3(超清)，如果指定的码率没有，则往低码率下载(3 没有找 2，2 没有找 1，1 也没有返回-5)

directPath(下载视频存放的目录 e.g: `string directPath = "{用户指定目录}/";`)

请注意用户指定目录后面需带斜杠"/"，同步用户指定目录需要用"/"作为分隔符

➤ 返回值：

0:成功，1:暂停，2:删除，-1:参数错误,-2:流量超标，-3:账号过期,-4:视频信息获取失败,-5:没有对应码率,-6:key 下载失败，-7: MP4 下载失败，-8:m3u8 下载失败-9:ts 下载失败；

➤ 说明：

① 下载方式为同步下载，下载的视频位置：{用户指定目录}/video/{视频 vid 命名}/

② 相同 vid 各个码率对应的视频都在同一目录下；

③ 非加密视频时后缀为.MP4 的文件；

④ 加密视频由后缀为.ts .key .m3u8 三种文件组成

⑤ 下载视频存放位置命名规则：

`string mp4Path = "./video/"+videoid+"/" + videopool_id + "_" + bitRate + ".mp4";`

`string m3u8Path = "./video/"+videoid+"/" + videopool_id + "_" + bitRate + ".m3u8";`

`string keyPath = "./video/"+videoid+"/" + videopool_id + "_" + bitRate + ".key";`

`string tsPath = "./video/"+videoid+"/" + videopool_id + "_" + bitRate + "_" + index + ".ts";`

⑥ 变量释义/举例：

videoid: 68040382d0870df0ae53c796419359fe\_6

Videopool\_id: videoid 去掉后面两位 68040382d0870df0ae53c796419359fe

bitRate : 码率(1-标清 2-高清 3-超清)

Index: ts 序号 从 0 开始

### 3、暂停下载接口

`public void pauseDownload();`

➤ 说明：暂停时 **abort** 变量为 **true**，下载接口 `downloadVideo` 返回 1

### 4、删除文件接口

`public int deleteVideo(string videoid, int bitRate, string directPath);`

➤ 参数：

videoid(视频 vid)

bitRate(码率)，1(标清)，2(高清)，3(超清)

directPath(下载视频存放的目录 e.g: `string directPath = "D:/视频目录/";`)

➤ 说明：下载过程中删除视频时，isDelete 变量为 **true**，下载接口 `downloadVideo` 返回 2

## 5、下载重要参数

public bool abort = false;//暂停, 暂停正在下载视频为 true  
public bool isDelete = false;//删除,删除正在下载视频为 true

## 6、下载进度回调事件

Demo 调用示范: Media.OnDownloadProgress += DownloadProgressEvent;

public delegate void OnDownloadProgressHandler(string videoid, long  
receivedBytes,long totalBytes);  
public event OnDownloadProgressHandler OnDownloadProgress;  
返回参数说明: videoid(视频 vid), totalBytes(总大小)receivedBytes(已接收的文件  
大小)

## 7、服务器没有指定码率时, 下载最高码率触发事件

Demo 调用示范: Media.OnCurrentBitRate+= OnCurrentBitRateEvent;

public delegate void OnCurrentBitRateHandler (string videoid,int inputBitRate,int  
realBitRate);  
public event OnCurrentBitRateHandler OnCurrentBitRate;  
返回参数: videoid(视频 vid), inputBitRate(输入的码率),realBitRate(实际下载的码  
率)

## 8、删除视频回调事件

//删除回调

public delegate void OnDeleteInfoHandler(bool status,string videoid, int bitRate,  
string msg);  
public event OnDeleteInfoHandler OnDeleteInfo;  
返回参数: status(删除状态, true:删除成功,false:删除失败), videoid(视频 vid),  
bitRate(码率), msg(删除提示)

## 9、获取所选码率的视频大小

public long getFileSize(string videoid, int bitRate);

- 返回值: getFileSize 接口返回将要下载的对应该码率视频的字节数, 若返回-1  
表示 videojson 请求失败或者该视频异常。
- 参数:  
videoid(视频 vid)  
bitRate(码率), 1(标清), 2(高清), 3(超清)

# ◇ 网络状况和日志记录

## 1、网络异常检测

.net 本身提供网络异常回调，无需 sdk 提供

```
NetworkChange.NetworkAvailabilityChanged += new  
NetworkAvailabilityChangedEventHandler(networkchanged);
```

```
public void networkchanged(object sender, NetworkAvailabilityEventArgs e)  
{  
    if (!e.IsAvailable) tipsLabel.Text = "网络中断";  
    else tipsLabel.Text = "网络恢复";  
}
```

## 2、日志系统

Sdk 提供本地日志功能，播放和下载均会打印相应日志

利用 SDK 的 mylog 对象可以在上层使用日志打印功能

```
mylog.Info("您希望打印的重要信息");  
mylog.Debug("您希望打印的调试信息");  
mylog.Warning("您希望打印的警告信息");  
mylog.Error("您希望打印的异常信息");
```