|  |
| --- |
| 深圳市穿透科技有限公司 |
| P2P会议模块使用说明书 |
| 开发手册 |
|  |
| **穿透科技** |
| **2019/8/15** |

|  |
| --- |
| [在此处键入文档摘要。摘要通常为文档内容的简短概括。在此处键入文档摘要。摘要通常为文档内容的简短概括。] |

# 产品介绍

P2P会议模块可实现一对一或多方同时进行音视视频互动。

## 产品特点

* 支持音视频通过P2P的方式传输。
* 支持多种视频分辨率。
* 支持H264，H265等视频传输格式。
* 支持PCM的音频传输格式。

## 应用场景

视频通话：可实现类似微信一样的一对一或多人视频通话。

* 视频会议：多方视频互动进行会议讨论和分享

## 产品规格

|  |  |  |
| --- | --- | --- |
|  | 规格项 | 规格说明 |
| 功能规格 | 回音消除 | 对讲时消除减轻音频回音，使音频更清晰 |
| 硬件加速 | 使用GPU等硬件来进行视频编解码、合成的加速 |
| H.264 | 应用最广的高性能视频编码器 |
| H.265 | 更高压缩比、更节省带宽的视频编码器 |
| 参数规格 | 最大采集分辨率 | 可设置，不建议超过1080P |
| 最大传输分辨率 | 可设置，不建议超过1080P |
| 最大码率 | 可设置，不建议过大 |
| 最大帧率 | 可设置，不建议超过30fps |

## 平台兼容性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 规格/平台 | | Android | iOS | Web |
| 功能规格 | 回音消除 | ✓ | ✓ | ✓ |
| 硬件加速 | ✓ | ✓ | – |
| H.264编码 | ✓ | ✓ | ✓ |
| H.265编码 | ✓ | ✓ | ✓ |

## 与P2P直播模块SDK的区别

直播模块的视频流向，采集端采集视频，可以被多个播放端播放，视频流是单向的。

采集端

播放端

播放端

播放端

（图1）直播模块SDK视频流向图

P2P会议模块视频流向，会议SDK视频流是双向的，两两个节点之间都可以相互打开视频。

主席端

成员端1

成员端2

成员端3

（图2）P2P会议模块SDK视频流向图

# 准备工作

## 准备工作

P2P会议模块SDK基于穿透科技Peergine中间件的节点对象、组对象和音视频对象。因此在音视频对讲前需要创建节点对象登录到P2P服务器（初始化），需要创建组对象（Start），并加入组（JoinRequest），同时要支持音视频需要初始化音视频对象（VideoStart，AudioStart），最后需要指定对应节点打开视频（VideoOpenRequest）。

P2P会议模块SDK通过一个总的回调上报事件，不同的事件使用sAction的参数表示。

## 前提条件

### ANDROID

请确保满足以下开发环境要求:

* Android SDK API Level Level ≥ 15
* Android Studio 3.0 或以上版本
* App 要求 Android 4.1 或以上设备

### IOS

请确保满足以下开发环境要求:

* Xcode 10.0+
* iOS 8.0+ 真机(iPhone 或 iPad)
* 请确保您的项目已设置有效的开发者签名

### WebHtml5

请确保满足以下开发环境要求:

* 操作系统： Microsoft Windows 7+
* 安装Chrome 75+浏览器
* 安装 [pgHtml5Bridge](http://peergine.com/#/download?active=7) 后台服务插件
* 确保部署环境为localhost或http协议

## 添加SDK

### ANDROID

* 下载  [Android SDK](http://peergine.com/#/download?active=4) ，解压并打开。
* 将文件夹 pgLibConference 作为模块(Module)导入你创建的工程。
* 将文件夹 pgPluginLib 作为模块(Module)导入你创建的工程。
* 在你的 Module 中设置添加对 pgLibConference, pgPluginLib 的依赖

现在你已经设置好了 Android Studio 开发环境，可以开始使用 Android SDK 了！

### IOS

* 下载 [IOS SDK](http://peergine.com/#/download?active=4)，解压并打开。
* 将文件夹 pgLibConference拷贝到创建的工程。
* 将文件夹 pgPluginLib 拷贝到创建的工程。
* 将文件夹pgPluginLib下pgDybLiveMulti\_device.framework添加到工程依赖
* 在BuildSetting的Other Link Flags参数中增加：-liconv -lz -lresolv链接选项。
* 禁止使用bitcode
* 工程中引用pgLibConference文件夹下的所有源文件。
* 添加一系列的系统framework

|  |
| --- |
| CoreVideo.framework |
| CoreAudioKit.framework |
| UIKit.framework |
| AdSupport.framework |
| CoreGraphics.framework |
| CoreMedia.framework |
| AVFoundation.framework |
| AudioToolbox.framework |

现在你已经设置好了 XCode 开发环境，可以开始使用 IOS SDK 了！

### Web HTML5

* 下载 [Web HTML5 SDK](http://peergine.com/#/download?active=4)，解压并打开。
* 拷贝并引用 dist/pgLibConference2.js
* 拷贝并引用 DemoConference2/pgHtml5Bridge.js

现在你已经设置好了 Web开发环境，可以开始使用Web HTML5 SDK 了！

## 初始化

### ANDROID

|  |
| --- |
| String sInitParam = "(P2PTryTime){3}(LogLevel0){1}(LogLevel1){1}(LogLevel2){1}(LogLevel3){1}(Debug){1}(SocketInitWnd){15}";  m\_Conf2 = new pgLibConference2();  m\_Conf2.SetEventListener(m\_OnEvent);  int iErr = m\_Conf2.Initialize(sUser, "", sSvrAddr, "", sInitParam, context);  if (iErr>PG\_ERR\_Normal) {  Log.d("Conference", "Init failed");  return iErr;  } |

### IOS

|  |
| --- |
| String sInitParam = @"(P2PTryTime){3}(LogLevel0){1}(LogLevel1){1}(LogLevel2){1}(LogLevel3){1}(Debug){1}(SocketInitWnd){15}";  if (isInputExternal) {  //sInitParam += "(VideoInExternal){1}";  }  m\_Conf2 = [[pgLibConference2 alloc] init];  [m\_Conf2 SetEventListener:(self)];  int iErr = [m\_Conf2 Initialize:sUser pass:@"" svraddr:sSvrAddr relayaddr:@"" param:sInitParam ext:context];  if (iErr > PG\_ERR\_Normal) {  OutString(@"Init failed");  return iErr;  } |

### Web HTML5

|  |
| --- |
| this.m\_sUser = sUser;  var sInitParam = "(P2PTryTime){3}(LogLevel0){1}(LogLevel1){1}(LogLevel2){1}(LogLevel3){1}(Debug){1}(SocketInitWnd){15}";    // eslint-disable-next-line no-undef  this.m\_Conf2 = Conf2;  this.m\_Conf2.SetEventListener(this);  var iErr = this.m\_Conf2.Initialize(sUser, "", sSvrAddr, "", sInitParam, Node);  if (iErr > PG\_ERR\_Normal) {  \_OutString("Conference", "Init failed");  return iErr;  } |

注意：初始化之后会有登录结果的事件回调

|  |
| --- |
| private OnEventListener m\_OnEvent = new OnEventListener() {  @Override  public void event(String sAct, String sData, final String sPeer,String sConfName,String sEventParam) {  // TODO Auto-generated method stub  // String sObjPeer = \_ObjPeerBuild(sPeer);  if (sAct.equals(EVENT\_VIDEO\_FRAME\_STAT)) {  EventVideoFrameStat(sAct, sData, sPeer,sConfName,sEventParam);  } else if (sAct.equals(EVENT\_LOGIN)) {  EventLogin(sAct, sData, sPeer,sConfName,sEventParam);  } else if (sAct.equals(EVENT\_LOGOUT)) {  EventLogout(sAct, sData, sPeer,sConfName,sEventParam);  } |

## 创建会议，初始化音视频

### ANDROID

|  |
| --- |
| int iErr = m\_Conf2.Start(sConfName, sChair);  if(iErr > PG\_ERR\_Normal){  showInfo("创建会议失败。 iErr = " + pgLibErr2Str(iErr) + " sConfName = " + sConfName);  return iErr;  }  iErr = m\_Conf2.VideoStart(sConfName,m\_sVideoParam,m\_sVideoParamLarge);  if(iErr > PG\_ERR\_Normal){  showAlert("初始化视频失败： iErr = " + pgLibErr2Str(iErr) + " sConfName = " + sConfName);  }  int iFlag = AUDIO\_SPEECH;  String sAudioParam = "(Flag){" + iFlag + "}";  iErr = m\_Conf2.AudioStart(sConfName,sAudioParam);  if(iErr > PG\_ERR\_Normal) {  showAlert("初始化音频失败： iErr = " + pgLibErr2Str(iErr) + " sConfName = " + sConfName);  } |

### IOS

|  |
| --- |
| int iErr = [m\_Conf2 Start:(sConfName) chair:sChair];  if (iErr > PG\_ERR\_Normal) {  String sOut = [NSString stringWithFormat:@"创建会议失败。 iErr = %@ sConfName = %@", pgLibErr2Str(iErr), sConfName];  showInfo(sOut);  return iErr;  }  iErr = [m\_Conf2 VideoStart:sConfName videoparam:m\_sVideoParam videoparam2:m\_sVideoParamLarge];  if (iErr > PG\_ERR\_Normal) {  String sOut = [NSString stringWithFormat:@"初始化视频失败。 iErr = %@ sConfName = %@", pgLibErr2Str(iErr), sConfName];  showAlert(sOut);  }  int iFlag = AUDIO\_SPEECH;  String sAudioParam = [NSString stringWithFormat:@"(Flag){%d}", iFlag];  iErr = [m\_Conf2 AudioStart:(sConfName) param:sAudioParam];  if (iErr > PG\_ERR\_Normal) {  String sOut = [NSString stringWithFormat:@"初始化音频失败。 iErr = %@ sConfName = %@", pgLibErr2Str(iErr), sConfName];  showAlert(sOut);  } |

### Web HTML5

|  |
| --- |
| var iErr = this.m\_Conf2.Start(sConfName, sChair);  if(iErr > PG\_ERR\_Normal){  showInfo("创建会议失败。 iErr = " + pgLibErr2Str(iErr) + " sConfName = " + sConfName);  return iErr;  }  iErr = this.m\_Conf2.VideoStart(sConfName,this.m\_sVideoParam,this.m\_sVideoParamLarge);  if(iErr > PG\_ERR\_Normal){  showAlert("初始化视频失败： iErr = " + pgLibErr2Str(iErr) + " sConfName = " + sConfName);  }  var iFlag = AUDIO\_SPEECH;  var sAudioParam = "(Flag){" + iFlag + "}";  iErr = this.m\_Conf2.AudioStart(sConfName,sAudioParam);  if(iErr > PG\_ERR\_Normal) {  showAlert("初始化音频失败： iErr = " + pgLibErr2Str(iErr) + " sConfName = " + sConfName);  } |

# 快速开始

## 打开预览

### ANDROID

|  |
| --- |
| iErr = m\_Conf2.PreviewStart(m\_sPrewParam);  if(iErr > PG\_ERR\_Normal){  return iErr;  } |

### IOS

|  |
| --- |
| iErr = [m\_Conf2 PreviewStart:m\_sPrewParam];  if (iErr > PG\_ERR\_Normal) {  return iErr;  } |

### Web HTML5

|  |
| --- |
| iErr = this.m\_Conf2.PreviewStart(this.m\_sPrewParam,this.mPreviewEl);  if(iErr > PG\_ERR\_Normal){  return iErr;  } |

## 请求加入会议

成员端调用请求加入会议

|  |
| --- |
| iErr = m\_Conf2.JoinRequest(sConfName);  if (iErr > PG\_ERR\_Normal) {  showInfo("发送加入会议请求失败： iErr = " + pgLibErr2Str(iErr) + " sConfName = " + sConfName);  TimerReJoinRequest(sConfName,sChair);  } |

请求加入会议之后主席端会收到请求加入会议事件

|  |
| --- |
| else if (sAct.equals(EVENT\_JOIN\_REQUEST)) {  EventJoinRequest(sAct, sData, sPeer, sConfName,sEventParam);  } |

需要在请求加入会议事件产生后将节点加入会议

|  |
| --- |
| private void EventJoinRequest(String sAct, String sData, String sPeer,String sConfName, String sEventPara) {  // TODO: 2016/11/7 sPeer请求加入会议 MemberAdd表示把他加入会议  showInfo(sPeer + "请求加入会议->同意");  if(verifyPeer(sConfName,sPeer)){  m\_Conf2.MemberAdd(sConfName,sPeer);  }  } |

主席节点将成员节点加入会议之后，会产生加入会议的事件

|  |
| --- |
| } else if (sAct.equals(EVENT\_JOIN)) {  EventJoin(sAct, sData, sPeer,sConfName,sEventParam);  } |

## 打开会议节点视频

请求打开会议中某节点视频

|  |
| --- |
| if(peer.pView == null){  //从中间件内部 pgLibView 分配一个SurfaceView,只有这样创建的才有效  peer.pView = pgLibView.Get(sConfName + sPeer);  }  if(peer.pLayout == null){  //申请桌面的linear  peer.pLayout=m\_LayoutMange.Alloc(sConfName + " : " + sPeer);  if(peer.pLayout == null) {  showAlert("无法申请到LenearLayout，VideoOpenRequest ： " + sConfName + " sPeer = " + sPeer);  return PG\_ERR\_System;  }  }  int iErr = m\_Conf2.VideoOpenRequest(sConfName,sPeer, iStreamMode,peer.pView,"");  if (iErr > PG\_ERR\_Normal) {  showInfo("失败");  return iErr;  }  peer.pLayout.removeAllViews();  if(peer.pView.getParent() != null){  ((LinearLayout)(peer.pView.getParent())).removeAllViews();  }  peer.pLayout.addView(peer.pView); |

如果该节点以经加入会议并初始化了音视频，会产生回调事件

|  |
| --- |
| else if (sAct.equals(EVENT\_VIDEO\_REQUEST)) {  EventVideoOpenRequest(sAct, sData, sPeer,sConfName,sEventParam);  } |

在回调事件中接收视频请求

|  |
| --- |
| private void EventVideoOpenRequest(String sAct, String sData, String sPeer, String sConfName, String sEventPara) {  //收到视频请求  showInfo(sPeer + " 请求视频连线->同意");  //// TODO: 2016/11/7 在这之后回复  //调用  if(sEventPara.equals("0")){  pgVideoOpenResponse(sConfName,sPeer);  }  }  …  /\*\*  \* 打开视频时完成窗口和相关数据的改变  \*  \* @param sConfName 对象ID  \* @param sPeer 对象ID  \* @return ErrCode 错误码  \*/  private int pgVideoOpenResponse(String sConfName,String sPeer){  ConferencePeer peer = conferencePeerList.\_Add(sConfName,sPeer);  if(peer == null){  showAlert("申请内存失败" + sConfName + " sPeer = " + sPeer);  return PG\_ERR\_System;  }  if(peer.pView == null){  //从中间件内部 pgLibView 分配一个SurfaceView,只有这样创建的才有效  peer.pView = pgLibView.Get(sConfName + sPeer);  }  if(peer.pLayout == null){  //申请桌面的linearlayout  peer.pLayout=m\_LayoutMange.Alloc(sConfName + " : " + sPeer);  if(peer.pLayout == null) {  showAlert("无法申请到LenearLayout，VideoOpenRequest ： " + sConfName + " sPeer = " + sPeer);  return PG\_ERR\_System;  }  }  int iErr = m\_Conf2.VideoOpenResponse(sConfName,sPeer,PG\_ERR\_Normal, iStreamMode,peer.pView,"");  if (iErr > PG\_ERR\_Normal) {  showInfo("失败");  return iErr;  }  peer.pLayout.removeAllViews();  if(peer.pView.getParent() != null){  ((LinearLayout)(peer.pView.getParent())).removeAllViews();  }  peer.pLayout.addView(peer.pView);  return iErr;  } |

打开的对端视频会在传入的SurfaceView中显示，将SurfaceView添加到桌面的Layout就能显示播放对讲视频。

# 进阶教程

## 外部采集视频

当使用的视频源是网络（RTSP、RTMP、SIP等）视频源时，或者摄像头特殊默认视频采集不能生效的情况下，中间件支持外部采集视频的方法。（P2P会议SDK只支持安卓）

|  |
| --- |
| sInitParam += "(VideoInExternal){1}";  …  external = new VideoAudioInputExternal(m\_Conf2.GetNode(),mPreviewLayout,iMode,context);  external.VideoInputExternalEnable(); |

具体查看文件

pgPluginLibExter\src\main\java\com\peergine\plugin\exter\ VideoAudioInputExternal.java

外部采集重点需要实现

pgDevVideoIn.OnCallback中的回调函数，以及调用

pgDevVideoIn.CaptureProcExt 函数将视频帧输入中间件。

## 录像录音

|  |
| --- |
| Date currentTime = new Date();  SimpleDateFormat formatter = new SimpleDateFormat("yyyyMMddHHmmss");  String sDate = formatter.format(currentTime);  sBothPath = GetSdcardDir() + "/test/record" + sDate + ".avi";  int iErr = m\_Conf2.RecordStart(sConfName,sPeer,iStreamMode,sBothPath, PG\_RECORD\_NORMAL);  if(iErr!=0){  showInfo("录像失败。 已经关闭 Err = " + iErr);  } |

参数录像模式的定义

|  |
| --- |
| /\*\*  \* 录制对端的视频和音频  \*/  public static final int PG\_RECORD\_NORMAL = 0;  /\*\*  \* 录制对端的视频  \*/  public static final int PG\_RECORD\_ONLYVIDEO = 1;  /\*\*  \* 录制对端的音频  \*/  public static final int PG\_RECORD\_ONLYAUDIO = 2;  /\*\*  \* 录制对端的视频，需要在外部调用其他录制音频的API（如RecordAudioBothStart）配合使用才能录制。  \*/  public static final int PG\_RECORD\_ONLYVIDEO\_HASAUDIO = 3;  /\*\*  \* 录制对端的音频，需要在外部调用其他录制视频的API配合使用才能录制。  \*/  public static final int PG\_RECORD\_ONLYAUDIO\_HASVIDEO = 4; |

## 功能扩展

P2P会议SDK是在Peergine中间的基础上封装实现的，只封装了会议的业务逻辑。如果客户有其他SDK中未提供的功能需求，可联系穿透科技技术人员协商获取扩展函数。

扩展函数示例：

|  |
| --- |
| //  public int RecordAudioBothStart(String sAviPath) {  pgLibJNINode Node = m\_Conf2.GetNode();  if (Node != null) {  if (Node.ObjectAdd("\_vTemp", "PG\_CLASS\_Audio", "", 0)) {  String sData = "(Path){" + Node.omlEncode(sAviPath) + "}(Action){1}(MicNo){65535}(SpeakerNo){65535}(HasVideo){1}";  /\*String sData = "(Path){" + Node.omlEncode(sAviPath) + "}(Action){1}(MicNo){1}(SpeakerNo){65535}(HasVideo){1}";\*/  int iErr = Node.ObjectRequest("\_vTemp", 38, sData, "");  Log.d("pgLiveCapture", "RecordAudioBothStart, iErr=" + iErr);  Node.ObjectDelete("\_vTemp");  return iErr;  }  }  return 1;  } |

SDK录像函数只能录制指定节点的音频数据，此扩展函数可以录制设备上的麦克风和扬声器上的音频，也就是对讲音频。

# API参考

|  |
| --- |
| 使用的新类是：pgLibConference2  新的 Demo是： DemoConference2  以下是API接口说明：以下接口说明仅供参考，实际API和详细说明以源码中注释说明为准。  以下函数原型使用的是安卓版本的，IOS和Web HTML5都封装了同名函数。  /\*\*  \* 获取当前版本  \* @return 版本号  \*/  public String Version()  /\*\*  \* 描述：设置消息接收回调接口。（JS版本没有这个类）  \* 阻塞方式：非阻塞，立即返回  \* eventListener：[IN] 实现了OnEventListner接口的对象，必须定义event函数。  \*/  public interface OnEventListener {  /\*\*  \* 上报事件回调 （）  \* @param sAct 上报事件名称 Action  \* @param sData 上报事件数据，默认为空  \* @param sPeer 上报事件对端节点, 默认为空  \* @param sGroup 上报事件相关会议名称，默认为空 \*\*\*-> 会议名称都使用参数名称 sConfName  \* @param sEventParam 上报事件额外参数,默认为空  \*  \*/  void event(String sAct, String sData, String sPeer,String sGroup,String sEventParam);  }  /\*\*  \* 描述：设置消息接收回调接口。  \* 阻塞方式：非阻塞，立即返回  \* @param eventListener ：[IN] 实现了OnEventListner接口的对象，必须定义event函数。 （JS版本为定义实现了event函数的对象，  \* 函数的参数和OnEventListener.event 一致）  \*/  public void SetEventListener(pgLibConference.OnEventListener eventListener)  /\*\*  \* 描述：P2P会议对象初始化函数  \* 阻塞方式：非阻塞，立即返回。  \* @param sUser ：[IN] 登录用户名，自身的设备ID  \* @param sPass ：[IN] 登录密码  \* @param sSvrAddr ：[IN] 登录服务器地址和端口，格式：x.x.x.x:x  \* @param sRelayAddr ：[IN] 转发服务器地址和端口，格式：x.x.x.x:x。  \* 如果传入空字符串，则使用登录服务器的IP地址加上443端口构成转发服务器地址。  \* @param sInitParam ：视频参数，格式为：(P2PTryTime){3}(SKTBufSize1){256}(SKTBufSize1){256}  \* VideoInExternal: 启用视频输入回调接口。0为禁用，1为启用。  \* VideoOutExternal: 启用视频解码后输出回调接口。0为禁用，1为启用。  \* VideoOutExtCmp: 启用视频解码前输出回调接口。0为禁用，1为启用。  \* AudioInExternal: 启用音频输入回调接口。0为禁用，1为启用。  \* AudioOutExternal: 启用音频解码后输出回调接口。0为禁用，1为启用。  \* SKTBufSize0 :优先级0的发送缓冲区长度（单位为K字节），传0则使用缺省值，缺省为64(K)  \* SKTBufSize1 :优先级1的发送缓冲区长度（单位为K字节），传0则使用缺省值，缺省为64(K)  \* SKTBufSize2 :优先级2的发送缓冲区长度（单位为K字节），传0则使用缺省值，缺省为512(K)  \* SKTBufSize3 :优先级3的发送缓冲区长度（单位为K字节），传0则使用缺省值，缺省为128(K)  \* Digest: 是否使用摘要方式传递密码。0为明文方式，1为摘要方式。（缺省为1）  \* P2PTryTime :P2P穿透尝试时间（单位为秒）。  \* (iP2PTryTime == 0)：使用缺省值，缺省值为6秒。  \* (iP2PTryTime > 0 && iP2PTryTime <= 3600)：超时值为所传的iP2PTryTime  \* (iP2PTryTime > 3600)：禁用P2P穿透，直接用转发。  \* LogLevel0: 是否开启Major级别的日志信息输出。1为开启，0为关闭，默认为1  \* LogLevel1: 是否开启General级别的日志信息输出。1为开启，0为关闭，默认为1  \* LogLevel2: 是否开启Suggestive级别的日志信息输出。1为开启，0为关闭，默认为0  \* LogLevel3: 是否开启Info级别的日志信息输出。1为开启，0为关闭，默认为0  \* Debug: 是否开启调试信息打印。1为开启，0为关闭，默认为0  \* //EncryptMsg: 是否加密传输(发送)消息。1为加密，0为不加密，默认为0  \* LoginDelayInterval: 尝试重新登录的退避时间的增长步进（秒）。有效范围1 ~ 300，默认10  \* LoginDelayMax: 尝试重新登录的退避时间的最大值（秒）。有效范围30 ~ 300，默认300  \* @param oCtx： Android程序的上下文对象 (JS 传入 Node对象)  \* @return 错误码 PG\_ERR\_\*  \*/  public int Initialize(String sUser, String sPass, String sSvrAddr,  String sRelayAddr, String sInitParam, Context oCtx)  /\*\*  \* 描述：P2P会议对象清理函数  \* 阻塞方式：非阻塞，立即返回。  \*/  public void Clean();  /\*\*  \* 描述：获取自身的P2P节点名  \* 阻塞方式：非阻塞，立即返回。  \* 返回值：自身的P2P节点名  \* 作用：扩展时利用此类，进行底层操作。  \*/  public pgLibJNINode GetNode();  /\*\*  \* 描述：获取自身的P2P节点名  \* 阻塞方式：非阻塞，立即返回。  \* 返回值：自身的P2P节点名  \*/  public String GetSelfPeer();  /\*\*  \* Scan the in the same lan.  \* @return 错误码 PG\_ERR\_\*  \*/  public int LanScanStart()  /\*\*  \* 描述：通过节点名与其他节点建立联系 （节点名在我们P2P网络的功能类似英特网的IP地址）  \* 阻塞方式：非阻塞。  \* @param sPeer 对端的节点名（用户名 ID）  \* @return 错误码 PG\_ERR\_\*  \*/  public int PeerAdd(String sPeer);  /\*\*  \* Sdk扩展运用之添加通信节点， 使用之后会产生PeerSync事件  \* 删除节点连接。（一般不用主动删除节点，因为如果没有通信，节点连接会自动老化。）  \* @param sPeer 对端的节点名（用户名）  \*/  public void PeerDelete(String sPeer);  /\*\*  \* 获取节点连接信息  \* @param sPeer 对端节点 名称  \* @param bReport 是否上报  \* @return 错误码 PG\_ERR\_\*  \*/  public int PeerGetInfo(String sPeer, boolean bReport);  /\*\*  \* 描述：给指定节点发送消息  \* 阻塞方式：非阻塞，立即返回  \*  \* @param sMsg [IN] 消息内容  \* @param sPeer [IN]节点名称  \* @return 错误码 PG\_ERR\_\*  \*/  public int MessageSend(String sPeer,String sMsg);  /\*\*  \* 描述：给指定节点发送RPC消息  \* 阻塞方式：非阻塞，立即返回  \*  \* @param sMsg ：[IN] 消息内容  \* @param sPeer ：[IN]节点名称  \* @param sSession :[IN]可以为空，发送成功后，  \* sPeer端会收到RpcRequest事件，可以在回调中调用setRpcResponse 设置回复消息。  \* 可以收到RpcResponse事件，sParam 为 事件参数 sEventParam = sParam+":"+错误码 0 表示正常成功  \* @return 错误码 PG\_ERR\_\*  \*/  public int RpcRequest(String sPeer,String sMsg, String sParam);  /\*\*  \* 描述：设置RPC回复消息;  \* 马上设置，不支持线程设置  \* @param sData ：[IN] 消息内容  \*  \*/  public void setRpcResponse(String sData);  /\*\*  \* 描述：创建播放窗口对象 （JS 没有）  \* 阻塞方式：阻塞  \*/  public SurfaceView PreviewCreate();  /\*\*  \* 描述：销毁播放窗口对象（JS 没有）  \* 阻塞方式：阻塞  \*/  public void PreviewDestroy();  /\*\*  \* 描述：开始预览  \* 阻塞方式：阻塞  \* @param sPrvwParam 预览视频参数  \* Code: 视频压缩编码类型：1为MJPEG、2为VP8、3为H264。  \* Mode: 视频图像的分辨率（尺寸），有效数值如下：  \* 0: 80x60, 1: 160x120, 2: 320x240, 3: 640x480,  \* 4: 800x600, 5: 1024x768, 6: 176x144, 7: 352x288,  \* 8: 704x576, 9: 854x480, 10: 1280x720, 11: 1920x1080  \* FrmRate: 视频的帧间间隔（毫秒）。例如40毫秒的帧率为：1000/40 = 25 fps  \* BitRate: 视频压缩后的码率。单位为 Kbps  \* CameraNo: 摄像头编号，CameraInfo.facing的值。  \* Portrait: 采集图像的方向。0为横屏，1为竖屏。  \* （@param sDivPrew JS特定参数，用来显示预览的div）  \* @return 返回值：SurfaceView对象，可加入到程序主View中  \*/  public SurfaceView PreviewStart(String sPrvwParam);  /\*\*  \* 描述：停止预览  \* 阻塞方式：阻塞  \*/  public void PreviewStop();  /\*\*  \* 描述：开始会议，初始化视音频等会议相关数据。  \* 阻塞方式：非阻塞  \*  \* @param sName 会议名称 \*\*\*-> 会议名称都使用参数名称 sConfName  \* @param sChair 主席端ID名称  \* @return 错误码 PG\_ERR\_\*  \*/  public int Start(String sName, String sChair)  /\*\*  \* 描述：停止会议，初始化视音频等会议相关数据。  \* 阻塞方式：非阻塞  \* @param sName 会议名称 \*\*\*-> 会议名称都使用参数名称 sConfName  \*  \*/  public void Stop(String sName);  /\*\*  \* 描述：在会议中发送广播消息  \* 阻塞方式：非阻塞，立即返回  \*  \* @param sName [IN] 会议名称  \* @param sData [IN] 消息内容  \* @return 错误码 PG\_ERR\_\*  \*/  public int NotifySend(String sName,String sData);  /\*\*  \* 描述：添加成员（主席端）  \* 阻塞方式：非阻塞，立即返回  \* @param sName [IN] 会议名称  \* @param sMember ：[IN] 成员名  \* @return 错误码 PG\_ERR\_\*  \*/  public int MemberAdd(String sName,String sMember);  /\*\*  \* 描述：删除成员（主席端）  \* @param sName [IN] 会议名称  \* @param sMember ：[IN] 成员名  \* 阻塞方式：非阻塞，立即返回  \*/  public void MemberDel(String sName,String sMember);  /\*\*  \* 描述：请求加入会议  \* 阻塞方式：非阻塞，立即返回  \* @param sName [IN] 会议名称  \* @param sChair 主席端ID名称  \* @return 错误码 PG\_ERR\_\*  \*/  public int JoinRequest(String sName, String sChair);  /\*\*  \* 描述：初始化视频设置  \*  \* 阻塞方式：非阻塞，立即返回  \* @param sName [in] 会议名称：  \* @param sVideoParamDef [in] 默认视频流参数,示例(Flag){0}(Code){3}(Mode){2}(Rate){40}  \* Flag ：0 初始化视频正常 ; 1 初始化视频只接收视频不发送视频 ；2 初始化视频只发送视频不接收视频.  \* Code: 视频压缩编码类型：1为MJPEG、2为VP8、3为H264。  \* Mode: 视频图像的分辨率（尺寸），有效数值如下：  \* 0: 80x60, 1: 160x120, 2: 320x240, 3: 640x480,  \* 4: 800x600, 5: 1024x768, 6: 176x144, 7: 352x288,  \* 8: 704x576, 9: 854x480, 10: 1280x720, 11: 1920x1080  \* FrmRate: 视频的帧间间隔（毫秒）。例如40毫秒的帧率为：1000/40 = 25 fps  \* BitRate: 视频压缩后的码率。单位为 Kbps  \* @param sVideoParamLarge [in] 视频流参数  \* @return true 操作成功，false 操作失败  \*/  public int VideoStart(String sName, String sVideoParamDef,String sVideoParamLarge);  /\*\*  \* 描述：停止播放  \* @param sName [IN] 会议名称  \* 阻塞方式：非阻塞，立即返回  \*/  public int VideoStop(String sName);  /\*\*  \* 描述：打开某一成员另外一条流的视频  \* @param sName 会议名称  \* @param sPeer 成员节点名  \* @param iMode 视频流选择 0：默认视频流 1：额外的视频流 \*\*\*-> 可以用参数名称 iStreamMode  \* @param oView pgLibView产生的View （JS 版本第4个参数是用来显示节点视频的窗口div）  \* @param sParam 额外的参数：(DevNo){1} （JS 版本第5个参数无效）  \* DevNo : 外部接口播放时 作为参数。  \* @return error code @link pgLibError.java  \*/  public int VideoOpenRequest(String sName,String sPeer,int iMode,SurfaceView oView, String sParam)  /\*\*  \* 描述：打开某一成员另外一条流的视频  \*  \* @param sName 会议名称  \* @param sPeer 成员节点名  \* @param iMode 视频流选择 0：默认视频流 1：额外的视频流 \*\*\*-> 可以用参数名称 iStreamMode  \* @param iErr 传入错误码 PG\_ERR\_Normal 正常，PG\_ERR\_Reject 拒绝  \* @param oView pgLibView产生的View （JS 版本第5个参数是用来显示节点视频的窗口div）  \* @param sParam 额外的参数：(DevNo){1} （JS 版本第6个参数无效）  \* DevNo : 外部接口播放时 作为参数。  \* @return error code @link pgLibError.java  \*/  public int VideoOpenResponse(String sName,String sPeer,int iMode,int iErr,SurfaceView oView, String sParam);  /\*\*  \* 描述：关闭某一成员视频  \* 阻塞方式：非阻塞，立即返回  \* @param sName 会议名称  \* @param sPeer 成员节点名  \* @param iMode 视频流选择 0：默认视频流 1：额外的视频流 \*\*\*-> 可以用参数名称 iStreamMode  \* @return error code @link pgLibError.java  \*/  public void VideoClose(String sName,String sPeer,int iMode);  /\*\*  \* 检查sConfName.Peer节点的视频打开状态 ,在某些突然掉线，或者对端没有响应的情况下。  \* @param sConfName 会议名称  \* @param sPeer 对端节点名称  \* @param iStreamMode 视频流  \* @return ErrCode  \* PG\_ERR\_System : 没有初始化  \* PG\_ERR\_BadParam ： 参数为空  \* PG\_ERR\_NoExist ： 找不到这个会议  \* PG\_ERR\_BadStatus ： 这个节点的视频流没有打开  \* 其他错误 ：发送请求失败  \*/  public int VideoCheckStatus(String sConfName,String sPeer,int iStreamMode)  /\*\*  \* 描述：控制成员的视频流  \* 阻塞方式：非阻塞，立即返回  \*  \* @param sName 会议名称  \* @param sPeer 成员节点名  \* @param iMode 视频流选择 0：默认视频流 1：额外的视频流 \*\*\*-> 可以用参数名称 iStreamMode  \* @param sPeer 节点ID或对象  \* @param bEnable 是否接收和发送视频流  \* @return error code @link pgLibError.java  \*/  public int VideoControl(String sName,String sPeer,int iMode, boolean bEnable);      /\*\*  \* 描述：抓拍 sObjPeer 节点的图片  \* 阻塞方式：非阻塞，立即返回  \*  \* @param sName 会议名称  \* @param sPeer 成员节点名  \* @param iMode 视频流选择 0：默认视频流 1：额外的视频流 \*\*\*-> 可以用参数名称 iStreamMode  \* @param sPath 路径  \* @return error code @link pgLibError.java  \*/  public int VideoCamera(String sName,String sPeer,int iMode, String sPath);    /\*\*  \* 描述：开始播放或采集音频  \* 阻塞方式：非阻塞，立即返回  \* @param sName 会议名称  \* @param sAudioParam 音频初始化参数：(Flag){0}  \* Flag ：0 初始化音频控制正常对讲；1 初始化音频控制自己静音 ；2 初始化音频控制静音其他成员； 3 初始化音频控制不接收音频也不发送音频。  \* @return error code @link pgLibError.java  \*/  public int AudioStart(String sName,String sAudioParam);  /\*\*  \* 描述：停止播放或采集音频  \* 阻塞方式：非阻塞，立即返回  \* @param sName 会议名称  \*/  public void AudioStop(String sName);  /\*\*  \* 启用或禁用音频输入的静音。  \* @param sName 会议名称  \* @param iValue 1为启用，0为禁用。  \* @return 错误码 @link PG\_ERR\_  \*/  public int AudioMuteInput(String sName,int iValue)  /\*\*  \* 启用或禁用音频输出的静音。  \* @param iValue 1为启用，0为禁用。  \* @return 错误码 @link PG\_ERR\_  \*/  public int AudioMuteOutput(String sName,int iValue)  /\*\*  \* 描述：控制某个节点是否能播放本节点的音频，本节点能否播放对方的音频  \* 阻塞方式：非阻塞，立即返回  \*  \* @param sName 会议名称  \* @param sPeer ：节点名  \* @param bSendEnable : true接收 ，false不接收  \* @param bRecvEnable 返回值： true 操作成功，false 操作失败  \*/  public boolean AudioSpeech(String sName,String sPeer, boolean bSendEnable, boolean bRecvEnable)  /\*\*  \* 开始录制视频，要求：视频通话正在进行。  \* @param sName 会议名称  \* @param sPeer 录制端ID，ID为本身则录制本端视频  \* @param iVideoMode 视频流选择 0：默认视频流 1：额外的视频流 \*\*\*-> 可以用参数名称 iStreamMode  \* @param sAviPath 视频保存路径  \* @param iMode 录制模式，0 同时录制视音频；1 只录制视频；2 只录制音频  \* @return 错误码 @link pgLibError  \*/  public int RecordStart(String sName,String sPeer, boolean iVideoMode, String sAviPath, int iMode);  /\*\*  \* 停止录制  \* @param sName 会议名称  \* @param sPeer 录制端ID，ID为本身则录制本端视频  \* @param iVideoMode 视频流选择 0：默认视频流 1：额外的视频流 \*\*\*-> 可以用参数名称 iStreamMode  \* @param iMode 录制模式，0 同时录制视音频；1 只录制视频；2 只录制音频  \*/  public void RecordStop(String sName,String sPeer, boolean iVideoMode,int iMode);  /\*\*  \* 添加文件传输通道  \* @param sTargePeer 对端ID  \* @param sSourcePeer 本端ID  \*/  public int FileAdd(String sTargePeer,String sSourcePeer);  /\*\*  \* 删除文件传输通道  \* @param sTargePeer 对端ID  \* @param sSourcePeer 本端ID  \*/  public int FileDelete(String sTargePeer,String sSourcePeer);  /\*\*  \* 上传文件请求  \* @param sTargePeer 对端ID  \* @param sSourcePeer 本端ID  \* @param sPath 本地文件路径  \* @param sPeerPath 建议对端文件保存路径  \* @return 错误码  \*/  public int FilePutRequest(String sTargePeer,String sSourcePeer, String sPath, String sPeerPath)  /\*\*  \* 下载文件请求  \* @param sTargePeer 对端ID  \* @param sSourcePeer 本端ID  \* @param sPath 本地文件保存路径  \* @param sPeerPath 对端文件路径  \* @return 错误码  \*/  public int FileGetRequest(String sTargePeer,String sSourcePeer, String sPath, String sPeerPath)  /\*\*  \* 接受文件传输请求  \* @param sTargePeer 对端ID  \* @param sSourcePeer 成员端或对端ID  \* @param sPath 文件路径（空为默认值，下载请求时下载文件路径，上传请求时上传文件保存路径）  \* @return 错误码  \*/  public int FileAccept(String sTargePeer,String sSourcePeer, String sPath)  /\*\*  \* 拒绝文件传输  \* @param sTargePeer 对端ID  \* @param sTargePeer 本端ID  \* @param iErrCode 错误码  \* @return 错误码  \*/  public int FileReject(String sTargePeer,String sSourcePeer, int iErrCode);  /\*\*  \* 取消文件传输  \* @param sTargePeer 对端ID  \* @param sTargePeer 本端ID  \* @return 错误码  \*/  public int FileCancel(String sTargePeer,String sSourcePeer);  //========  /\*\*  \* 登录事件  \*/  public static final String EVENT\_LOGIN = "Login";  /\*\*  \* 登出事件  \*/  public static final String EVENT\_LOGOUT = "Logout";  /\*\*  \* 因为其他设备使用同一个ID登录，被服务器踢出  \*/  public static final String EVENT\_KICK\_OUT = "KickOut";  /\*\*  \* 上报相对节点的信息。  \* sData 上报信息格式： peer=xxx&through=xxx&proxy=xxx&addrlcl=xxx&addrrmt=xxx&tunnellcl=xxx&tunnelrmt=xxx&privatermt=xxx  \*/  public static final String EVENT\_PEER\_INFO = "PeerInfo";  /\*\*  \* 节点同步  \*/  public static final String EVENT\_PEER\_SYNC = "PeerSync";  /\*\*  \* 节点离线消息  \*/  public static final String EVENT\_PEER\_OFFLINE = "PeerOffline";  /\*\*  \* 节点消息事件  \*/  public static final String EVENT\_MESSAGE = "Message";  /\*\*  \* 上报RpcRequest 消息；  \*/  public static final String EVENT\_RPC\_REQUEST = "RpcRequest";  /\*\*  \* 上报 RpcResponse 消息；;sEventParam 上报sParam + ":" + iErrCode  \*/  public static final String EVENT\_RPC\_RESPONSE = "RpcResponse";  /\*\*  \* 服务器下发消息事件  \*/  public static final String EVENT\_SVR\_NOTIFY = "SvrNotify";  /\*\*  \* 服务器回复消息错误事件  \*/  public static final String EVENT\_SVR\_REPLYR\_ERROR = "SvrReplyError";  /\*\*  \* 服务器回复消息事件  \*/  public static final String EVENT\_SVR\_RELAY = "SvrReply";  /\*\*  \* 上报局域网节点信息  \*/  public static final String EVENT\_LAN\_SCAN\_RESULT = "LanScanResult";  /\*\*  \* 成员端请求加入会议事件（主席端上报）  \*/  public static final String EVENT\_JOIN\_REQUEST = "JoinRequest";  /\*\*  \* 成员加入组事件  \*  \*/  public static final String EVENT\_JOIN = "Join";  /\*\*  \* 成员离开会议事件  \*/  public static final String EVENT\_LEAVE = "Leave";  /\*\*  \* 广播消息事件  \*/  public static final String EVENT\_NOTIFY = "Notify";  /\*\*  \* 视频丢失事件  \*/  public static final String EVENT\_VIDEO\_LOST = "VideoLost";  /\*\*  \* 视频通道同步事件 ;sEventParam 上报VideoMode  \*/  public static final String EVENT\_VIDEO\_SYNC = "VideoSync";  /\*\*  \* 请求视频通话 ;sEventParam 上报VideoMode  \*/  public static final String EVENT\_VIDEO\_REQUEST = "VideoRequest";  /\*\*  \* 请求视频通话结果上报事件 ;sEventParam 上报VideoMode  \*/  public static final String EVENT\_VIDEO\_RESPONSE = "VideoResponse";  /\*\*  \* 视频关闭事件 ;sEventParam 上报VideoMode  \*/  public static final String EVENT\_VIDEO\_CLOSE = "VideoClose";  /\*\*  \* 获取VideoStatus 消息回应上报信息  \* sData : 错误码  \* PG\_ERR\_System : 没有初始化  \* PG\_ERR\_BadParam ： 参数为空  \* PG\_ERR\_NoExist ： 找不到这个会议  \* PG\_ERR\_BadStatus ： 这个节点的视频流没有打开  \* 其他错误 ：系统错误  \* sEventParam ： StreamMode  \*/  public static final String EVENT\_VIDEO\_CHECK\_STATUS = "VideoCheckStatus";  /\*\*  \* 视频状态信息上报 , ;sEventParam 上报VideoMode  \* Peer：指定上报视频统计的节点。  \* Total：总发送的视频帧数  \* Drop：丢弃的视频帧数  \*/  public static final String EVENT\_VIDEO\_FRAME\_STAT = "VideoFrameStat";  /\*\*  \* 拍照结果事件 ;sEventParam 上报VideoMode  \*/  public static final String EVENT\_VIDEO\_CAMERA = "VideoCamera";  /\*\*  \* 视频录像结果事件  \*/  public static final String EVENT\_RECORD = "Record";  /\*\*  \* 音频通道同步事件  \*/  public static final String EVENT\_AUDIO\_SYNC = "AudioSync";  /\*\*  \* 文件上传请求 sObjPeer 为 ID  \*/  public static final String EVENT\_FILE\_PUT\_REQUEST = "FilePutRequest";  /\*\*  \* 文件下载请求 sObjPeer 为 成员ID。  \*/  public static final String EVENT\_FILE\_GET\_REQUEST = "FileGetRequest";  /\*\*  \* 文件传输进度  \*/  public static final String EVENT\_FILE\_PROGRESS = "FileProgress";  /\*\*  \* 文件传输结束  \*/  public static final String EVENT\_FILE\_FINISH = "FileFinish";  /\*\*  \* 文件传输中断  \*/  public static final String EVENT\_FILE\_ABORT = "FileAbort";  /\*\*  \* 文件传输被拒绝  \*/  public static final String EVENT\_FILE\_REJECT = "FileReject";  /\*\*  \* 文件传输请求被接受  \*/  public static final String EVENT\_FILE\_ACCEPT = "FileAccept"; |