**P2P 会议模块Android开发手册 v1.0.5**

# 1. 概述

Android版的P2P会议模块是在Android版的Peergine中间件(JNI插件)的基础上封装出来的一个java类。Peergine中间件的功能强大，API接口复杂，需要经过较长时间的学习才能掌握。现在在Peergine中间件的基础上封装出一个P2P会议类，API接口大大简化，方便于开发者的使用。P2P会议类的源代码在pgLibConference\App\Android目录中。

Android版的Peergine中间件以两种方式提供：单独的APK和库工程。

1）单独的APK：Peergine中间件是一个单独的APK，单独安装，不需要合并到APP的APK。安装APP的APK的Android设备上，必须也要安装Peergine中间件的APK，才能正常使用。Peergine中间件的APK在pgLibConference\Plugin\APK目录中。

2）库工程：Peergine中间件的库工程。把此库工程导入到eclipse的workspace中（与APP的工程同一个workspace），然后在APP的工程中引用此库工程（如何引用库工程请在网络上找教材）。编译APP的工程时就会把库工程打包在一起，合成一个APK。库工程在pgLibConference\Plugin\AndroidLib目录中。

# 2. 调用API的顺序样例

// 创建P2P会议对象：

pgLibConference m\_Conf = new pgLibConference();

// 创建事件回调接口对象

private pgLibConference.OnEventListener m\_OnEvent = new pgLibConference.OnEventListener() {

@Override

public void event(String sAct, String sData, String sRender) {

// 处理事件的代码……

}

};

// 设置事件回调接口到P2P会议对象。

m\_Conf.SetEventListener(m\_OnEvent);

// 初始化P2P会议类对象

// 同时初始化成功会登录P2P服务器。

// 对于主席端，会创建会议并获得成员管理权限

// 对于成员端来说，是先登录P2P服务器，接着连接主席端。

m\_Conf.Initialize();

// 创建Peergine中间件窗口。

//此窗口可以作为视频预览窗口。

// 此窗口必须创建，即使APP不需要视频预览或视频播放，都必须创建。因为，P2P中间件需要经过这个窗口的消息队列投递消息到APP的界面里，才成完成回调事件的上报。（APP可以把此窗口隐藏起来，而且不加入到layout中）。

m\_Wnd = (SurfaceView)m\_Conf.WndCreate(0, 0, 320, 240);

// 其他业务功能使用操作。根据实际的需要调用具体的API。

VideoStart、VideoOpen 、VideoClose、VideoStop、AudioStart、AudioStop、MessageSend、

// 销毁Peergine中间件窗口。（如果此窗口已经添加到layout中，则需要先从layout删除）

m\_Conf.WndDestroy();

// 注销下线

m\_Conf.Stop();

// 清理P2P会议对象

m\_Conf.Clean ();

**注意：所有P2P会议类的API接口都必须在界面主线程里调用。**

# 3. com.peergine.android.conference.pgLibConference类：

## 1）公有函数声明（还有一些接口函数的增加，具体情况以代码为准）：

### SetExpire:设置主席端与成员端的心跳间隔。

/\*\*  
 \* 描述：设置主席端与成员端的心跳间隔。  
 \* 阻塞方式：非阻塞，立即返回  
 \* iExpire：[IN] 心跳间隔。  
 \*/  
**public void SetExpire(int iExpire)**

### SetEventListener: 设置消息接收回调接口

/\*\*

\* 描述：设置消息接收回调接口。

\* 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 void SetEventListener(OnEventListener eventListener)**

### GetNode: 获取自身的pgLibJNINode 类

/\*\*

\* 描述：获取自身的pgLibJNINode 类

\* 阻塞方式：非阻塞，立即返回。

\* 返回值：自身的pgLibJNINode 类

\*/

**public pgLibJNINode GetNode()**

### GetSelfPeer: 获取自身的P2P节点名

/\*\*

\* 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()**

### Initialize: 对象初始化函数：

/\*\*

\* 描述：P2P会议对象初始化函数

\* 阻塞方式：非阻塞，立即返回。

\* sName：[IN] 会议名称

\* sChair：[IN] 主席端设备ID

\* sUser：[IN] 登录用户名，自身的设备ID

\* sPass：[IN] 登录密码

\* sSvrAddr：[IN] 登录服务器地址和端口，格式：x.x.x.x:x

\* 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。

\* sVideoParam：[IN] 视频参数，格式为：(Code){3}(Mode){2}(Rate){40}(LCode){3}(LMode){2}

\* (LRate){40}(CameraNo){0}(Portrait){1}(BitRate){400}

\* 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

\* LCode: 不同流视频压缩编码类型：1为MJPEG、2为VP8、3为H264。

\* LMode: 不同流视频图像的分辨率（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

\* LFrmRate: 不同流视频的帧间间隔（毫秒）。例如40毫秒的帧率为：1000/40 = 25 fps

\* InputExternal: 使能视频的外部采集

\* OutputExternal: 使能视频的外部输出。输出解码后的视频帧，比如格式为RGB24

\* OutputExtCmp: 使能视频的外部采集。输出未解码的视频帧，比如H264

\* 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。

\* Rotate: 采集图像的角度。

\* BitRate: 视频压缩后的码率。单位为 Kbps

\*

\* oCtx： Android程序的上下文对象

\* 返回值：true 成功， false 失败

\*/

**public boolean Initialize(String sName,String sChair, String sUser, String sPass, String sSvrAddr, String sRelayAddr, String sVideoParam, Context oCtx)**

### Initialize: 对象初始化函数：

/\*\*

\* 描述：P2P会议对象初始化函数

\* 阻塞方式：非阻塞，立即返回。

\* sUser：[IN] 登录用户名，自身的设备ID

\* sPass：[IN] 登录密码

\* sSvrAddr：[IN] 登录服务器地址和端口，格式：x.x.x.x:x

\* 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。

\* sVideoParam：[IN] 视频参数，格式为：(Code){3}(Mode){2}(Rate){40}(LCode){3}(LMode){2}

\* (LRate){40}(CameraNo){0}(Portrait){1}(BitRate){400}

\* 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

\* LCode: 不同流视频压缩编码类型：1为MJPEG、2为VP8、3为H264。

\* LMode: 不同流视频图像的分辨率（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

\* LFrmRate: 不同流视频的帧间间隔（毫秒）。例如40毫秒的帧率为：1000/40 = 25 fps

\* VideoInExternal: 使能视频的外部采集

\* 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。

\* Rotate: 采集图像的角度。

\* BitRate: 视频压缩后的码率。单位为 Kbps

\*

\* oCtx： Android程序的上下文对象

\* 返回值：true 成功， false 失败

\* 与上面初始化函数的区别：上面的初始化函数在初始化登陆之后，同时初始化了会议。

\*/

**public boolean Initialize(String sUser, String sPass, String sSvrAddr, String sRelayAddr, String sVideoParam, Context oCtx)**

### Clean: 对象清理函数

/\*\*

\* 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()**

### WndCreate: 播放窗口创建函数

/\*\*

\* 描述：创建播放窗口对象

\* 阻塞方式：阻塞

\* iX：[IN] 窗口水平位置

\* iY：[IN] 窗口垂直位置

\* iW：[IN] 窗口宽度

\* iH：[IN] 窗口高度

\* 返回值：SurfaceView对象，可加入到程序主View中

\*/

**public SurfaceView WndCreate(int iX, int iY, int iW, int iH)**

### WndDestroy: 播放窗口销毁函数

/\*\*

\* 描述：销毁播放窗口对象

\* 阻塞方式：阻塞

\* 返回值：true 成功 false 失败

\*/

**public void WndDestroy()**

### MemberAdd: 添加成员

/\*\*

\* 描述：添加成员（主席端）

\* 阻塞方式：非阻塞，立即返回

\* sMember：[IN] 成员名

\* 返回值： true 操作成功，false 操作失败

\*/

**public boolean MemberAdd(String sMember)**

### MemberDel: 删除成员

/\*\*

\* 描述：删除成员（主席端）

\* sMember：[IN] 成员名

\* 阻塞方式：非阻塞，立即返回

\*/

**public void MemberDel(String sMember)**

### Join: 请求加入会议

/\*\*

\* 描述：请求加入会议（成员端）

\* 阻塞方式：非阻塞，立即返回

\* 返回值： true 操作成功，false 操作失败

\*/

**public boolean Join()**

### Leave: 离开会议

/\*\*

\* 描述：离开会议

\* 阻塞方式：非阻塞，立即返回

\*/

**public void Leave()**

### VideoStart: 初始化视频设置

/\*\*

\* 描述：初始化视频设置

\* 成员：iFlag[in] 参考[1）静态成员定义：](#_1）静态成员定义：)

\* 阻塞方式：非阻塞，立即返回

\* 返回值： true 操作成功，false 操作失败

\*/

**public boolean VideoStart(int iFlag)**

### VideoStop: 停止播放和采集视频

/\*\*

\* 描述：停止播放和采集视频

\* 阻塞方式：非阻塞，立即返回

\*/

**public void VideoStop()**

### VideoOpen: 打开某一成员的视频

/\*\*

\* 描述：打开某一成员的视频

\* 阻塞方式：非阻塞，立即返回

\* 返回值： true 操作成功，false 操作失败

\* sPeer:成员节点名

\* iW: 窗口宽度

\* iH: 窗口高度

\*/

### VideoOpenL: 以不同流打开某一成员的视频（请求端有效）

/\*\*

\* 描述：以不同流打开某一成员的视频（请求端有效）

\* 阻塞方式：非阻塞，立即返回

\* 返回值： true 操作成功，false 操作失败

\* sPeer:成员节点名

\* iW: 窗口宽度

\* iH: 窗口高度

\*/

### VideoReject: 拒绝打开某一成员的视频

/\*\*

\* 描述：打开某一成员的视频

\* 阻塞方式：非阻塞，立即返回

\* 返回值： true 操作成功，false 操作失败

\* sPeer:成员节点名

\*/

**public SurfaceView VideoReject(String sPeer)**

### VideoClose: 关闭某一成员视频

/\*\*

\* 描述：关闭某一成员视频

\* 阻塞方式：非阻塞，立即返回

\*/

**public void VideoClose(String sPeer)**

### VideoGetView: 获取已打开成员视频的View

**public SurfaceView VideoGetView(String sPeer)**

### VideoControl: 控制成员的视频流

/\*\*

\* 描述：控制成员的视频流

\* 阻塞方式：非阻塞，立即返回

\*/

### VideoSource: 摄像头切换

/\*\*

\* 描述：摄像头切换

\* 阻塞方式：非阻塞，立即返回

\* iCameraNo：摄像头编号

\*/

**public boolean VideoControl(String sPeer, boolean bEnable)**

### AudioStart: 开始播放或采集音频

/\*\*

\* 描述：开始播放或采集音频

\* 阻塞方式：非阻塞，立即返回

\* 返回值： true 操作成功，false 操作失败

\*/

**public boolean AudioStart()**

### AudioStop: 停止播放或采集音频

/\*\*

\* 描述：停止播放或采集音频

\* 阻塞方式：非阻塞，立即返回

\*/

**public void AudioStop()**

**AudioSpeech: 控制某个节点是否能播放本节点的音频。**

/\*\*

\* 描述：控制某个节点是否能播放本节点的音频。

\* 阻塞方式：非阻塞，立即返回

\* sPeer：节点名

\* bEnable: true接收 ，false不接收

\* 返回值： true 操作成功，false 操作失败

\*/

**public boolean Speech(String sPeer, boolean bEnable)**

### MessageSend: 给指定节点发送消息

/\*\*

\* 描述：给指定节点发送消息

\* 阻塞方式：非阻塞，立即返回

\* sMsg：[IN] 消息内容

\* sPeer：[IN]节点名称

\* 返回值： true 操作成功，false 操作失败

\*/

**public boolean MessageSend(String sData, String sPeer)**

### NotifySend: 给其他所有成员节点发送消息

/\*\*

\* 描述：给其他所有成员节点节点发送消息

\* 阻塞方式：非阻塞，立即返回

\* sMsg：[IN] 消息内容

\* 返回值： true 操作成功，false 操作失败

\*/

**public boolean NotifySend(String sData)**

**CameraSwitch: 摄像头控制**

/\*\*

\* 描述：摄像头控制。

\* 阻塞方式：非阻塞，立即返回

\* 返回值： true 操作成功，false 操作失败

\*/

**Public boolean CameraSwitch (boolean bEnable)**

### SvrRequest: 给服务器发送消息

/\*\*

\* 描述：给服务器发送消息。

\* 阻塞方式：非阻塞，立即返回

\* 返回值： true 操作成功，false 操作失败

\*/

**Public boolean** **SvrRequest(String sData)**

### ServerNetMode: 指定客户端与P2P服务器的连网方式

/\*\*

\* 描述：指定客户端与P2P服务器的连网方式。

\* 在手机系统上使用P2P时，如果手机休眠，则网络切换到“只使用Relay转发)”方式连接，

\* 可增强手机APP在休眠状态下的在线能力。

\* 阻塞方式：非阻塞，立即返回

\* iMode：[IN] 连网模式。0为自动选择，1为只用P2P穿透，2为只用Relay转发。

\* 返回值： true 操作成功，false 操作失败

\*/

**public boolean ServerNetMode(int iMode)**

### TimerOutAdd: 将TimerOut接口添加到超时处理列表中

/\*\*  
 \* 描述:将TimerOut接口添加到超时处理列表中。  
 \* 阻塞方式：非阻塞  
 \* @param timerOut  
 \*/  
**public void TimerOutAdd(TimerOut timerOut)**

### TimerOutDel：将TimerOut接口从超时处理列表中删除

/\*\*  
 \* 描述:将TimerOut接口从超时处理列表中删除  
 \* 阻塞方式：非阻塞  
 \* @param timerOut  
 \*/  
**public void TimerOutDel(TimerOut timerOut)**

### TimerStart：开启一个定时器

/\*\*  
 \* 描述：开启一个定时器  
 \* 阻塞方式：非阻塞  
 \* @param sParam : 超时处理接口收到的参数  
 \* @param iTimeout :超时时间  
 \* @param bRepeat : 是否循环  
 \* @return : 定时器实例ID  
 \*/  
**public int TimerStart(String sParam, int iTimeout, boolean bRepeat)**

### TimerStop ：关闭一个定时器

/\*\*  
 \* 描述：关闭一个定时器  
 \* 阻塞方式：非阻塞  
 \* @param iTimerID : 定时器实例ID  
 \*/

**public void TimerStop(int iTimerID)**

## 2）事件回调接口：

### OnEventListener.event: 事件上报接口函数

/\*\*

\* 描述：上报事件给应用层。

\*

\* 参数：

\* sAct：上报的事件操作（Action），有效的Action值如下：

\* VideoStatus：视频传输播放状态上报，

\* sData参数的格式为：”bitrate=码率(bps)&frmrate=帧率&frmplay=帧号”

\* Message：接收到对端发送的消息，sData参数为消息内容，sPeer参数为对端的节点ID。

\* Login：本端登录到P2P服务器返回，sData参数返回登录的错误码。

\* Logout：本端从P2P服务器注销。

\* AskJoin：请求加入会议。

\* sPeer：节点名

\* AskLeave:请求离开会议。

\* sPeer：节点名

\* Join：已经加入会议。

\* Leave：已经离开会议。

\* VideoSync:视频类同步消息。

\* sPeer：节点名

\* VideoSyncL:不同流视频类同步消息。

\* sPeer：节点名

\* VideoOpen:请求打开视频。

\* sPeer：节点名

\* VideoLost：已经得不到sPeer的视频流了。

\* sPeer：节点名

\* VideoClose：已经关闭视频。

\* sPeer：节点名

\* VideoJoin:打开视频结果通知。

\* ChairmanSync:主席同步通知。

\* ChairmanOffline:主席端不在线通知。

\* PeerSync:主席同步通知。

\* PeerOffline:主席端不在线通知。

\* Notify：收到其他节点群发的消息。

\* sData消息的参数。

\* SvrNotify：收到P2P服务器下推的消息。

\* sData消息的参数。

\*

\* VideoFrameStat：视频帧统计信息上报。

\* sData参数的格式为：”total=总帧数&drop=丢弃的帧数”

\*

\* sData：上报事件的参数，参考每个sAct的定义。

\*

\* sPeer：进行通信的对端P2P节点ID。

\*

\* 返回值：无

\*/

**public void OnEventListener.event(String sAct, String sData, String sPeer)**

### event回调函数实现样例（伪码）

public void event(String sAct, String sData, String sPeer) {

if (sAct.equals("VideoStatus")) {

// 视频播放状态上报

// sData参数的格式为：”bitrate=码率(bps)&frmrate=帧率&frmplay=帧号”

}

else if (sAct.equals("Message")) {

// 接收消息

}

else if (sAct.equals("Login")) {

// 登录返回，sData里是返回的错误码

}

else if (sAct.equals("Logout")) {

// 已经注销

}

else if (sAct.equals("ChairmanSync")) {

// Chairman 正在线上

}

else if(sAct.equals("AskJoin")) {

//请求加入会议

}

else if(sAct.equals("AskLeave")) {

//请求离开会议

}

else if(sAct.equals("Join")){

//加入会议通知

}

else if(sAct.equals("Leave")){

//离开会议通知

}

else if (sAct=("VideoSync")) {

//视频类同步消息

}

else if (sAct=("VideoSyncL")) {

//不同流视频类同步消息

}

else if (sAct.equals("VideoOpen")) {

//sPeer请求打开视频

}

else if(sAct.equals("VideoLost")){

//sPeer的视频流接收不到了

}

else if(sAct.equals("VideoClose")){

//sPeer已经挂断视频

}

else if (sAct .equals( "VideoJoin")) {

//打开视频结果通知

}

else if(sAct.equals("ChairmanSync")){

//主席同步通知

}

else if (sAct .equals( " ChairmanOffline")) {

//主席不在线通知

}

else if(sAct.equals("PeerSync")){

//sPeer同步通知

}

else if (sAct .equals( " PeerOffline")) {

//sPeer不在线通知

}

else if (sAct.equals("SvrNotify")) {

// 接收到服务器下推的消息

// sData消息的参数。

}

else if (sAct.equals("Notify")) {

// 接收到某节点推送的消息

// sData消息的参数。

}

else if (sAct.equals("VideoFrameStat")) {

// 视频帧统计信息上报。

// sData参数的格式为：”total=总帧数&drop=丢弃的帧数”

}

}

## 3）超时回调接口：

### TimerOut. TimerProc

/\*\*

\*描述：定时器超时后将在TimerProc函数中处理，参数sParam是TimerStart的sParam参数。

\*

\*

\*/

**pgLibConference.TimerOut timerOut =new pgLibConference.TimerOut() {  
 @Override  
 public void TimerProc(String sParam) {**

**}**

**}**

# 4. com.peergine.android.Conference.pgVideoPutMode类：

## 1）静态成员定义：

**public static final int Normal = 0;** //video 正常

**public static final int OnlyInput = 1;**// video 只输入

**public static final int OnlyOutput = 2;**// Video 只输出

# 5. 修改记录

|  |  |  |
| --- | --- | --- |
| 序号 | 修改内容（最新的细节和说明在注释中，以代码为准） | 版本号 |
| 1 | 添加对两种视频流的支持，API为VideoOpenL 通知为VideoSync。  添加拒绝打开视频的API：VideoReject。  添加摄像头切换API：VideoSource。  修改API:Speech->AudioSpeech。  修改了一些API的文档说明。  修复了一些bug。 | 1 |
| 2 | 1.添加视频外部采集接口参数VideoInExternal  2.视频流角度调整参数。  3.添加API给服务器发送扩展消息  4.添加事件PeerSync表示本端与对端建立通道，可以利用MessageSend进行通信。  5.添加事件PeerOffline表示对端已经离线。  6.添加事件VideoLost用来报告视频已经丢失  7.内部添加成员端对主席端的心跳。以加快将网络异常报告给应用程序。  8.内部添加视频连接时的心跳，以检测异常重启时对端不能及时获取视频丢失的消息。  进行了若干优化。修复了若干BUG。 | 4 |
| 3 | 1. 上报PeerOffline，Leave ，VideoClose等消息取消在Sdk中清理数据的操作。效果：如果外部不进行VideoClose，关于视频的连接数据将保持。 2. 优化心跳。 3. 修复一些崩溃的bug。 4. 增加一个简易的定时器接口以及以系列函数。 5. 修改Sdk结构使得可以在Sdk初始化后直接调用VideoStart，AudioStart。在Join后任意位置调用VideoOpen。之前是Join后才能调用VideoStart和AudioStart。VideoSync后才能VideoOpen。 | 13 |
| 4 |  |  |