蔚领云游戏 Android_SDK 对接文档

修订记录

时间	版本	修改内容	修改人
2021/06/04	1.0	初稿完成	孙贤武
2021/06/25	1.1	热更新 Sdk 所用权限修改	孙贤武
2021/06/30	1.2.5	热更新 Sdk 子进程初始化加载框架的方法的添加（1.3.2）	孙贤武

2021/07/07	1.2.6	补丁更新插件方案（1.7.2，1.8.2,1.8.3） 设置陀螺仪参数（1.3.17） 获取设备解码方式（1.3.18）	孙贤武
2021/07/12	1.2.7	由于阿里框架升级，之前修复 anr 的方法失效，所以所有进程都要 init（）	孙贤武
2021/12/30	2.1.2	新增方法： 判断设备是否支持 H265 解码（1.3.20） 自定义切换运营商（1.3.21） udp 连接判断 udp 是否连接成功（1.3.22）	沙宗超

一、概述

本文档面向安卓开发者。

本文档用于指导开发者快速接入蔚领云游戏 SDK （热更新版）。

Sdk 对外接口类：WLCGConfig

Sdk 错误码类：WLErrorCode

Sdk 常量类：

WLCGGameConstant（按键配置等的常量）

WLEventConstants(发送的广播配置常量)

注意:

（一）Sdk 必须先调用 init 方法进行蔚领插件的安装,再调用后续的方法

建议调用 init(android.app.Application hostApplication,

java.lang.String hostUrl,

java.lang.String tenantKey,

java.lang.String params,

WLPluginInstallListener listener)

（二）初始化/更新差插件成功与否都会通过 LocalBroadcastManager 发送广播（建议使用回调处理）

```
@Override
public void onReceive(Context context, Intent intent) {
    if (TextUtils.equals(intent.getAction(), WLEventConstants.ACTION)) {
        Bundle bundle = intent.getExtras();
        int eventType = bundle.getInt(WLEventConstants.TYPE,  defaultValue: 0);
        int eventCode = bundle.getInt(WLEventConstants.CODE,  defaultValue: 0);
        String eventInfo = bundle.getString(WLEventConstants.MSG,  defaultValue: "");

        switch (eventType) {
            case WLEventConstants.TYPE_INSTALL:
                //安装
                break;
            case WLEventConstants.TYPE_UPDATE:
                //更新
                break;
        }
        String info = "";
        switch (eventCode) {
            case WLEventConstants.CODE_INSTALL_SUCCESS:
                info = "安装成功";
                break;
            case WLEventConstants.CODE_INSTALL_FAIL:
                info = "安装失败";
                break;
            case WLEventConstants.CODE_UPDATE_SKIPED:
                info = "跳过此次更新";
                break;
            case WLEventConstants.CODE_UPDATE_SUCCESS:
                info = "更新成功";
                break;
            case WLEventConstants.CODE_UPDATE_FAIL:
                info = "更新失败";
                break;
            case WLEventConstants.CODE_UPDATE_RESET_FAIL:
                info = "回滚基线版本失败";
                break;
            case WLEventConstants.CODE_UPDATE_RESET_SUCCESS:
                info = "回滚基线版本成功";
                break;
        }
        String logInfo = "eventType:" + eventType + ",eventCode:" + eventCode + ",result =" + info;
        LogUtils.i(TAG,  msg: "onReceive " + logInfo);
        LogUtils.printJson(TAG, eventInfo);
```

二、接入流程

1.1. 导入资源包

Androidstudio 导入

1、导入蔚领提供的 aar 格式的 SDK 文件：

wl_game_sdk-V*****.aar;

2、在工程 App 对应 **build.gradle** 配置脚本 **dependencies** 段中添加 SDK 库依赖：

Implementation (name: 'wl_game_sdk-V*****', ext : 'aar')

需要添加

epositories {

latDir {

```
            dirs 'libs'
        }
    }
```

3、第三方依赖包

```
implementation 'com.alibaba:fastjson:1.2.67'
implementation 'com.google.protobuf:protobuf-java:3.5.1'
implementation 'com.google.protobuf:protoc:3.5.1'
implementation 'com.squareup.okhttp3:okhttp:3.8.0'
implementation 'com.squareup.okio:okio:1.13.0'
```

以上第三方依赖务必引入到工程中

4、so 库文件，目前仅支持 arm64-v8a、armeabi、armeabi -v7a 库文件

```
ndk {
        abiFilters "arm64-v8a" ,"armeabi-v7a","armeabi"
}
```

***** arm64-v8a 需要 Android 版本 23 及以上版本才可以使用
armeabi 及 armeabi-v7a 需要 Android 版本 16 以上才可以使用

1.2. 修改配置文件

1.2.1. 权限说明

权限	用途说明
<uses-ermissionandroid:name="android.permission.ACCESS_NETWORK_STATE" />	获取设备当前网络状态
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>	获取设备当前网络状态
<uses-permission android:name="android.permission.INTERNET"/>	App 联网权限

1.2.2. 运行环境配置

SDK 可以运行于 Android4.1（API Level 16）及以上版本

<uses-sdk android:minSdkVersion="16" android:targetSdkVersion="28" />

如果开发者声明 targetSdkVersion 到 API 23 以上，请确保调用本 SDK 的任何接口前，已经申请到了 SDK 要求的所有权限。否则 SDK 可能无法正常工作

1.2.3. 混淆配置

参见附录四

1.2.4. 设置设备横屏

启动游戏 activity 需要设置成横屏

android:screenOrientation=“landscape”

1.3. 功能接入

1.3.1. 初始化 sdk(***必须执行***)

在 Application 中进行热更新 SDK 的初始化（所有进程），会进行插件的安装

WLCGConfig.init(String hostUrl,Application application,String tenantKey,String parameters,WLPluginInstallListener listener);

参数	类型	说明
hostUrl	String	配置的请求地址
hostApplication	Application	宿主的 Application

tenantKey	String	配置的租户 key
params	String	配置的对应的对应的加密的参数串
listener	WLPluginInstallListener	安装结果的回调，见 1.8.1

1.3.2. 开启 SDK debug 模式

说明：开发模式可以开通 debug 模式，方便查看日志，正式上线需要关闭， 默认关闭

方法：WLCGConfig.openDebug(true);

1.3.3 启动游戏

说明：启动游戏

方法：

WLCGConfig.startGame(Activity activity, FrameLayout container,int connectType, String gameData, WLCGListener mGameListener);

参数	类型	说明
activity	Activity	-当前启动游戏的 activity
container	FrameLayout	-展示游戏画面的容器 layout
connectType	int	-连接方式 0：TCP，1：UDP
gameData	String	-启动游戏的参数信息(启动游戏请求 pass 后台返回的 sdkMsg 中的所有数据,需参考 PAAS 调度文档)
listener	WLCGListener	-游戏监听回调

1.3.4 退出游戏

说明:该方法在结束游戏成功后，会执行 initSurfaceView 方法传递过来的 activity 的 finish 方法。

方法：通知容器结束游戏

```
WLCGConfig.exitGame();
```

1.3.5 游戏挂起 --（暂停接收音视频流）

说明：游戏挂起--当应用切换到后台时--只有游戏启动成功后可以切换到后台，如果在游戏还没有启动成功切换到后台，游戏结束，起后游戏长时间无操作，或应用被系统清理，服务端将结束本次游戏会话

方法：

```
WLCGConfig.onPause();
```

1.3.6 游戏恢复 --（重新开始接收音视频流）

说明：游戏恢复--应用从后台切换到前台时，重启发送音视频流

方法：

```
WLCGConfig.onResume();
```

1.3.7 向游戏传递数据

说明：向游戏传递字符串数据（游戏里需要输入账号密码等方法时使用）

方法：

```
WLCGConfig.sendMSGToGame(String message);
```

参数	类型	说明
message	String	向游戏传递字符数据

1.3.8 设置接收数据的时间间隔

说明：设置接收数据的时间间隔

方法：

```
WLCGConfig.setReceiveDateTime(Context context,int time);
```

参数	类型	说明
context	Context	上下文
time	int	设置接收数据的时间间隔,秒

1.3.9 自定义设置卡顿时间

说明：自定义设置上报网络是否出现卡顿的时间间隔

方法：

```
WLCGConfig.setAVLagThreshold(int time);
```

参数	类型	说明
time	int	设置卡顿时间,毫秒（默认 200 毫秒）

1.3.10 重连游戏

说明:该方法是在游戏过程中返回 code 为 6075/6041 时可以执行该方法重连游戏，如连接失败，就需要重新请求调度启动游戏。

方法:

```
WLCGConfig.reconnectServer();
```

1.3.11 上行通道，向游戏发送指定数据

说明：上行通道，向游戏发送指定数据

方法:

```
WLCGConfig.sendDataToGame(byte[] data,int length);
```

**** SD 内部会有数据重传功能，有概率出现重复数据情况，调用方需要处理一下重复数据 ****

参数	类型	说明
data	byte[]	向游戏发送的数据
length	int	数据长度

1.3.12 设置是否开启自动重连

说明：当在游戏中网络断开或者报 6075 状态码时，可以自动重连游戏，当重连失败后，抛出重连失败消息

方法:

```
WLCGConfig.openAutoReconnectServer(boolean isOpen);
```

参数	类型	说明

isOpen	boolean	true:开启重连，false：关闭自动重连，默认 true

1.3.13 获取测速节点列表

说明：获取测速节点列表

方法：

```
WLCGConfig.getNodeList(ResutCallBackListener listener);
```

参数	类型	说明

listener	ResutCallBackListener	回调接口，返回测速结果
		new ResutCallBackListener() {
		@Override
		public void succes(String lines) {
		//lines 为返回的测速结果
		返回结果格式：
		{
		"bandwidth": 6.62,
		"nodeResult": [
		{
		"lineId": "0",
		"nodeId": "1001806",
		"nodeName": "上海测试一区",
		"pingResult": 40,
		"provide": "0",
		"type": "x86"
		},
		{
		"lineId": "3",
		"nodeId": "1001806",
		"nodeName": "上海测试一区",
		"pingResult": 78,
		"provide": "0",
		"type": "x86"
		},
		]
		}
		}

		@Override
		public void error(String s) {
		//获取测速结果失败
		}
		})

1.3.14 获取 bizData

说明：获取 bizData--(请求 paas 后台 dispatch 方法所需数据)

方法：

```
String bizData = WLCGConfig.getBizData();
```

1.3.15 获取 extData 数据

说明：获取 extData 数据--(请求 paas 后台 dispatch 方法所需数据)

方法：

```
String extData = WLCGConfig.getExtData();
```

1.3.16 获取码率配置数据

说明：获取码率配置数据

方法：

```
WLCGConfig.getBitrateConfig(ResutCallBackListener listener);
```

参数	类型	说明
listener	ResutCallBackListener	请求结果回调

1.3.17 启用 AAC_LD 模式

说明：启用 AAC_LD 模式(需要在执行启动游戏方法前执行)

```
WLCGConfig.enableLowDelayAudio(boolean isOpen);
```

参数	类型	说明
isOpen	boolean	启用开关：true：开启，false：关闭，默认开启

1.3.18 游戏保活接口

说明：游戏保活接口（弹出广告或者其他一些耗时操作时不想游戏断开，可以使用该接口，保证客户端一直有指令传给游戏，防止出现长时间无操作踢出，每执行一次该接口，会向游戏发送一次模拟事件指令（不影响游戏的正常事件指令））

方法：

```
WLCGConfig.keepAliveForGame();
```

1.3.19 是否开启游戏调度串重复使用校验

说明：游戏调度串重复使用校验（防止某些手机启动游戏后放后台应用被杀掉，点击应用重新启动该页面，导致游戏启动异常或者游戏数据丢失）

方法：

```
WLCGConfig.openVerificationForLastGameData(boolean isOpen);
```

参数	类型	说明
isOpen	boolean	是否开启，默认关闭

1.3.20 判断当前设备是否支持 H265

说明：判断当前设备是否支持 H265，如果支持 H265 返回 21，如果不支持返回 18，获取到的数值可直接赋值给 dispatch 接口的 codecType 字段

方法：

```
WLCGConfig.getMediaCodecType();
```

1.3.21 自定义选择运营商接口

说明：自定义选择运营商（单线模式可使用，游戏过程中可切换）

方法：

```
WLCGConfig.customOperatorForType(String type);
```

参数	类型	说明
type	String	游戏返回状态码 6122，返回的可切换运营商

1.3.22 获取当前连接状态是否为 UDP 连接

说明：获取当前连接状态是否为 UDP 连接，true：udp 连接成功

方法：

```
WLCGConfig.isUDPConnected();
```

1.3.23 执行解绑操作

说明：解绑，例如解绑 SDK 初始化时注册的广播等， 必须在调用 init 方法的类里，在结束时调用该方法

方法：

```
WLCGConfig.unRegisterEvent();
```

1.4. _游戏参数设置

1.4.1 设置切换游戏码率

说明：该方法是用于在游戏中可按照定义切换视频传输码率，返回当前码率名称

方法：

```
String resolution =WLCGConfig.setBitrateByLevel(int level);
```

参数	类型	说明
level	int	1：流畅（1.5M 即 1500） 2：标清（2.5M 即 2500） 3：高清（4.0M 即 4000） 4：蓝光（6.5M 即 6500） -1：顺序切换下一档

1.4.2 设置游戏画面尺寸

说明：设置游戏画面尺寸 ，启动游戏后可切换设置

方法：

```
WLCGConfig.getInstance().setVideoScreen(nt type);
```

参数	类型	说明

type	int	type = 0 全屏显示 type = 1 16 : 9 type = 2 4 : 3

1.4.3 自定义设置游戏码率

说明：自定义设置游戏码率，1.4.1 方法 码率值是固定的，该方法可灵活设置

方法：

```
WLCGConfig.setBitrate(int bitrate);
```

参数	类型	说明
bitrate	int	具体码率值

1.4.4 自动降码率

说明：根据当前网络情况自动调整游戏码率（UDP 模式）

（游戏启动后设置生效）

方法：

```
WLinkConfig.autoBitrateAdjust(int minBit);
```

参数	类型	说明
minBit	int	码率最低可以降到多少，minBit = 0 : 关闭自动降码率功能，默认开启状态，默认降低码率值为 1500.（建议在设置码率时设置一个自动降码率的下限值）

1.4.5 重传开关

说明：视频数据重传开关（UDP 模式）（游戏启动后设置生效）

方法：

```
WLinkConfig.switchDataRetransmission(boolean isOpen);
```

参数	类型	说明
isOpen	boolean	数据重传开关，默认开启状态

1.4.6 前向纠错开关

说明：前向纠错开关（UDP 模式）（游戏启动后设置生效）

方法：

```
WLinkConfig.switchForwardErrorCorrection(boolean isOpen);
```

参数	类型	说明
isOpen	boolean	前向纠错开关，默认开启

1.4.7 游戏内设置帧率

说明：游戏内设置帧率，游戏启动后设置有效。

方法：

```
WLCGConfig.setFps(int fps);
```

参数	类型	说明
fps	int	设置的帧率（只支持 30fps/60fps，其他值暂不支持）

1.5. 按键操作

1.5.1 手柄按键事件

说明：该方法同时适用于虚拟手柄和物理手柄的按键定义

方法：

WLCGConfig.OnGamePadButton(int userIndex,int keycode,int action);

参数	类型	参数名称	说明
userIndex	int	手柄 index	代表第 index 个手柄
keycode	int	手柄按键键值	请参照 附录二：手柄按键映射
action	int	按键状态	WLCGGameConstant.CUSTOM_KEY_DOWN(按键按下) WLCGGameConstant.CUSTOM_KEY_UP (按键抬起)

1.5.2 手柄摇杆事件

说明：该方法同时适用于虚拟手柄和物理手柄的按键定义

方法：

WLCGConfig.onGamePadAxis(int user_index,int type,int xValue,int yValue)

参数	类型	参数名称	说明
userIndex	int	手柄 index	代表第 index 个手柄

type	int	手柄摇杆类型标识	WLCGGameConstant.AXIS_LXLY (左摇杆标示) WLCGGameConstant.AXIS_RXRY (右摇杆标示) WLCGGameConstant.AXIS_RT (RT 标示) WLCGGameConstant.AXIS_LT (LT 标示) WLCGGameConstant.AXIS_HAT (十字键标示)
xValue	int	X 轴	摇杆 X 轴取值范围 （-32767~32767） LT 取值范围 （0-255） 十字键 HAT_X （-1--1）
yValue	int	Y 轴	摇杆 Y 轴取值范围 （-32767~32767） RT 取值范围 （0-255） 十字键 HAT_Y （-1--1）

1.5.3 鼠标事件

说明：该方法同时适用于虚拟手柄和物理手柄的按键定义

方法：

```
WLCGConfig.onCustomMouseEvent(int keycode,int action,int xValue,int yValue);
```

参数	类型	参数名称	说明
Keycode	int	鼠标 code	WLCGGameConstant.MOUSE_LBUTTON 鼠标左键 WLCGGameConstant.MOUSE_RBUTTON 鼠标右键 WLCGGameConstant.MOUSE_MBUTTON 鼠标中键 WLCGGameConstant.MOUSE_MWHEELUP 鼠标滚轮-上 WLCGGameConstant.MOUSE_MWHEELDOWN 鼠标滚轮-下
action	int	按键状态	WLCGGameConstant.CUSTOM_KEY_DOWN 按下 WLCGGameConstant.CUSTOM_KEY_UP 抬起 WLCGGameConstant.CUSTOM_KEY_MOVE 移动
xValue	int	鼠标当前 x 坐标值	鼠标当前 x 坐标值，取值为 MotionEvent.getRawX()

yValue	int	鼠标当前 y 坐标值	鼠标当前 y 坐标值，取直为 MotionEvent.getRawY()

1.5.4 触屏事件

说明：触屏事件

方法：

WLCGConfig.onCustomTouchEvent(MotionEvent event);

参数	类型	参数名称	说明
event	MotionEvent	触屏 MotionEvent 事件	触屏 MotionEvent 事件

1.5.5 键盘按键事件

说明：该方法同时适用于虚拟键盘和物理键盘的事件

方法：

WLCGConfig. onKeyBoardEvent(int keyCode,int action)

参数	类型	参数名称	说明
keycode	int	键盘按键键值	请参照 附录三：键盘按键映射
action	int	按键状态	WLCGGameConstant.CUSTOM_KEY_DOWN(按 键按下) WLCGGameConstant.CUSTOM_KEY_UP (按键抬起)

1.5.6 默认 OnkeyDown 事件

说明：默认 OnKeyDown 事件，实体手柄按键映射转换内部处理

方法：

WLCGConfig.defaultOnKeyDown(int keyCode,KeyEvent event);

参数	类型	参数名称	说明
keycode	int	按键键值	
event	KeyEvent	key 事件	

1.5.7 默认 OnKeyUp 事件

说明：默认 OnKeyDown 事件，实体手柄按键映射转换内部处理

方法：

WLCGConfig.defaultOnKeyDown(int keyCode,KeyEvent event);

参数	类型	参数名称	说明
keycode	int	按键键值	
event	KeyEvent	key 事件	

1.5.8 默认 OnGenericMotionEvent 事件

说明：默认 OnGenericMotionEvent 事件，内部处理手柄摇杆事件

方法：

WLCGConfig.defaultOnGenericMotionEvent(MotionEvent event);

参数	类型	参数名称	说明
event	MotionEvent	MotionEvent 事件	

1.6. sdk 版本获取

1.6.1. sdk 版本获取

获取蔚领热更新 sdk 版本名：WLCGConfig.getSDKVersion();

获取蔚领热更新 sdk 版本号：WLCGConfig.getSDKVersionCode();

1.6.2. Game sdk 版本获取

获取蔚领 GAME 插件 sdk 版本名:WLCGConfig.getGamePluginSDKVersion()

获取蔚领插件 sdk 版本号，插件安装失败时返回 -1：WLCGConfig.getGamePluginSDKVersionCode()

1.7. 插件更新

1.7.1 完整插件包更新模式

说明：更新插件版本调用以下方法

方法：

WLCGConfig.updatePlugin(update, listener)

参数	类型	说明
update	WLPluginUpdate	插件更新信息组成的对象
listener	WLPluginUpdateListener	更新结果回调

WLPluginUpdate 属性解析

属性	类型	说明
pluginName	String	必填，插件名字，目前是：com.welinkpass.hotfix.sdk
updateType	int	必填，更新类型， 安装：WLPluginUpdate.UPDATE_TYPE_NEED_UPDATE， 回滚：WLPluginUpdate.UPDATE_TYPE_RESET，回滚到基线版本（集成到当前 app 中的 sdk 版本）
versionCode	long	【升级插件】必填，插件的版本号，蔚领提供插件时提供
channelId	String	【升级插件】选填
versionName	String	选填，插件版本号
updateNote	String	选填，插件的更新日志
pluginFileSize	int	选填，插件完整包的大小
pluginPath	String	【升级插件】必填，插件完整包的地址，本地下载好的插件完整包位置
pluginMD5	String	【升级插件】必填，插件完整包的 MD5 值，蔚领提供插件时提供

1.7.2 插件补丁包更新模式

说明：补丁包更新模式（补丁包包体小）

方法：

WLCGConfig.updatePlugin(update, listener)

参数	类型	说明
update	WLPatchPluginUpdate	插件更新信息组成的对象
listener	WLPluginUpdateListener	更新结果回调

WLPatchPluginUpdate 属性解析

属性	类型	说明
pluginName	String	必填，插件名字，目前是：com.welinkpass.hotfix.sdk
updateType	int	必填，更新类型， 安装：WLPluginUpdate.UPDATE_TYPE_NEED_UPDATE， 回滚：WLPluginUpdate.UPDATE_TYPE_RESET，回滚到基线版本（集成到当前 app 中的 sdk 版本）
versionCode	long	【升级插件】必填，插件的版本号，蔚领提供插件时提供
channelId	String	【升级插件】选填
versionName	String	选填，插件版本号
updateNote	String	选填，插件的更新日志
patchFileSize	int	选填，插件补丁包大小
patchPath	String	【升级插件】必填，插件补丁包的地址，本地下载好的插件补丁包位置
patchMD5	String	【升级插件】必填，插件补丁包的 MD5 值，蔚领提供

1.8. 类的属性解析

1.8.1. WLPluginInstallResult

插件安装回调类

参数	类型	说明
pluginName	String	插件名字，目前是：com.welinkpass.hotfix.sdk
installResultCode	int	安装成功：WLEventConstants.CODE_INSTALL_SUCCESS 安装失败：WLEventConstants.CODE_INSTALL_FAIL
currentStep	String	当前步骤
timeInfo	String	每个安装步骤的时间信息
installTime	String	总的安装时间

repeatCount	String	操作的次数
currentVersion	String	插件当前版本
errorCode	String	安装失败的错误码，参见附录五 WLErrorCode
extraMsg	String	额外信息，安装成功失败都会有相应的信息

1.8.2. WLPluginUpdateResult

插件更新回调类

参数	类型	说明
pluginName	String	插件名字，目前是：com.welinkpass.hotfix.sdk
updateResultCode	int	更新成功：WLEventConstants.CODE_UPDATE_SUCCESS 更新失败：WLEventConstants.CODE_UPDATE_FAIL 补丁模式更新成功：WLEventConstants.CODE_UPDATE_WITHPATCH_SUCCESS 补丁模式更新失败：WLEventConstants.CODE_UPDATE_WITHPATCH_FAIL 跳过更新：WLEventConstants.CODE_UPDATE_SKIPED 回滚成功：WLEventConstants.CODE_UPDATE_RESET_SUCCESS 回滚失败：WLEventConstants.CODE_UPDATE_RESET_FAIL
fromVersionName	String	当前版本名
toVersionName	String	目标版本名
fromVersionCode	String	当前版本号
toVersionCode	String	目标版本号
updateNote	String	更新日志
updateTime	String	更新耗时 ms
errorCode	String	更新失败的错误码，参见附录五 WLErrorCode
extraMsg	String	额外信息，更新成功失败都会有相应的信息

channel	String	渠道
pluginPath	String	完整插件包地址，仅在完整插件包更新模式有值
applyPatchInfo	WLApplyPatchInfo	补丁模式更新的额外信息，仅在补丁更新模式有值

1.8.3. WLApplyPatchInfo

补丁模式更新的额外信息

参数	类型	说明
patchPath	String	补丁地址
patchFileTime	String	合成新插件耗时 ms
oldPluginMD5	String	旧插件的 MD5
patchMD5	String	补丁的 MD5
newPluginMD5	String	合成的新插件的 MD5

1.9. WLCGListener 接口方法

1.9.1. 启动游戏或游戏过程中异常消息

说明：启动游戏失败消息（包括 连接游戏服务器失败，服务器断开等失败的

方法：

```
startGameError(int code ， String error);
```

参数	类型	说明
code	int	状态码--见附录一
error	String	返回的异常消息提示--见附录一

1.9.2. 连接服务器或游戏过程中消息返回

说明：连接服务器成功或游戏过程中消息返回

方法：

startGameInfo(int code ， String msg);

参数	类型	说明
code	int	状态码--见附录一
error	String	返回的连接服务器成功或游戏过程中消息提示--见附录一

1.9.3. 游戏启动成功收到游戏画面

说明：接收到游戏画面，可以隐藏展示 loading 图和进度条

方法：

startGameScreen();

1.9.4. 显示菜单消息

说明：显示菜单按钮（游戏手柄 快捷键 L1+SELECT 可以呼出菜单按钮）

可以自定义菜单栏，并实现 虚拟手柄的 隐藏开关，调整游戏的码流

方法：

showConfigView();

1.9.5. 显示游戏统计数据

说明：显示游戏中统计数据

方法：

showGameStatisticsData(String data);

参数	类型	说明
data	String	包括：带宽，帧率，码率，so 返回提示，解码时间，延迟 码率：玩家切换后的码率

返回数据示例：

```
{
    "bandWidth":596805,    //当前接收视频流带宽 bps
    "bitrate":4990,        //玩家设置码率 bps
    "decodecTime":12,      //每帧解码时间 毫秒
    "errorMsg":"",         //底层返回提示
    "fps":60,              //当前游戏帧率
    "netWorkDelay":16,     //带宽延迟 毫秒
    "nowTime":1591496273308//当前时间 毫秒
    "serverFps":16,            //服务端统计 FPS
    "packetLossCont":16,       //一秒内最大连续丢包数
    "packetLossRate":6.666,    //一秒内丢包率，百分比
    "packetLossTime":16,       //一秒内最大连续丢包时长
}
```

1.9.6. 游戏内透传数据

说明：该方法用于接收从游戏内发出的数据，适用场景包含游戏内发起支付等指令

方法：

onGameData(byte [] date);

参数	类型	说明
data	byte []	游戏返回的游戏字节数据

1.9.7. 手柄震动反馈

说明：游戏内震动消息反馈

方法：

OnGamePadVibration(int userIndex,int leftMotorSpeed,int rightMotorSpeed);

参数	类型	说明
userIndex	int	代表第 index 个手柄
leftMotorSpeed	int	左马达震动强度
rightMotorSpeed	int	右马达震动强度

1.9.8. 通知服务端指针样式

说明：当游戏启动成功后，传输第一帧画面时发送该消息，或者鼠标指针样式或状态发生改变的时候发送该消息

方法：

onCursorData(int isShow , String data , int size);

参数	类型	参数名称	说明
isShow	int	隐藏或显示指针	0 = 隐藏 1 = 显示 2 = 触控笔或触控屏接入导致不显示（隐藏状态）
data	Stirng	指针数据	指针样式的 Base64 格式数据，一个 ICO 格式数据
size	int	数据大小	

1.9.9. 通知服务端光标位置

说明：当检测到服务端光标位置与上次传递的位置不同时（即游戏自己重置了光标位置）发送该消息

*** 返回的光标位置是基于服务端游戏启动时的分辨率的相对位置，前端显示时需要进行坐标转换处理

方法：

onCursorPos(int x , int y);

参数	类型	参数说明	说明
x	int	x 坐标	
y	int	y 坐标	

附录一：游戏返回状态码

startGameError(int code,String message)（回调方法）

code	message
6017	当前环境所有可用节点测速失败（所有全部超时）
6018	当前环境无可用测速节点
6041	连接游戏服务器失败,connect 方法连接失败返回
6043	连接游戏服务器超时
6068	当前系统版本过低，无法正常启动游戏 （Android 版本低于 4.1 时的消息）
6075	游戏服务器连接断开，可尝试重连

6076	游戏服务器连接断开，建议不尝试重连操作
6095	解码失败
6096	初始化方法失败，请重新初始化
6097	启动游戏失败
6098	启动游戏容器失败
6100	启动参数缺失
6101	启动游戏参数缺失
6102	启动游戏参数校验错误
6103	启动游戏参数不完整
6104	启动参数解析失败，请检查参数
6106	游戏重连失败
6108	初始化参数校验失败，请检查并重新初始化
6114	初始化失败，未拿到节点信息，请检查网络
6115	初始化失败，请求节点失败（前后台时间不一致）
6116	初始化失败，获取节点失败
6120	添加调度串校验功能，（解决手机放后台应用被杀掉，点击应用重新启动该页面 导致某些游戏数据不可用）
6121	上行通道发送数据过大（不可超过 1.9M）
1001	业务需求踢人状态码（可带踢人消息）
1002	业务需求踢人状态码，不带消息
1012	游戏启动成功 30 秒内就异常退出

1013	游戏启动成功超过 30 秒，异常退出
1017	游戏未启动成功就正常退出　　启动失败
1018	游戏启动成功 30 秒内就正常退出　　启动失败
1019	游戏正常退出... （--玩家在游戏内点击了退出游戏操作/或游戏 crash）
1010	用户已重新连接或已使用新设备连接
1011	游戏异常退出...
1020	用户太长时间未操作了（长时间没操作，超时退出） （---目前默认长时间无操作时间是 10 分钟）
1030	游戏启动超过加载时间了 （游戏启动时间过长 3 分钟）
1040	游戏长时间黑屏 （游戏卡住，可能服务器运行环境出问题了）
1080	服务端适配配置不正确 （环境部署错误，生产环境不应出现）
1090	服务端 vos 版本配置错误 （环境部署错误，生产环境不应出现）
1100	服务端模块严重系统错误 （服务端 Windows 跑飞了）
1110	用户端未通过有效性认证 （服务端返回）
1111	用户已连接，但未在 5 秒内接收到用户 token
1112	GS 尚未分配（可能此容器上次 session 已清理）
1113	用户端有效性认证超时 （前端在一定时间内没有收到游戏端反馈）

7012	启动游戏失败，游戏找不到（EXE 文件未找到，3.7.1.2 添加）
7103	服务端游戏服务已终止 （--当服务端游戏正在结束的过程中，重连服务）
7104	服务器内部编码错误（服务器上硬件显卡故障）

startGameInfo(int code,String message)（回调方法）

code	message
6042	连接游戏服务器成功...,connect 方法连接成功返回
6066	长时间无操作提示 ---{"noInputTotal":"300","noInputTime":"30"}， noInputTime （无操作时间） noInputTotal（设置的无操作时间）
6077	游戏输入框点击获取焦点，弹出输入框键盘 ----游戏需要支持（玩家点击游戏里面的输入框，可通知前端呼出键盘，用于输入）
6079	URL 地址　向前传递 URL，用于打开浏览器 ----游戏需要支持（游戏里面点击链接地址，把链接地址发送到前端）
6080	token 校验通过
6082	当前设备不支持 H265,自动切换为 H264 解码
6090	首帧解码时间--毫秒
6091	音频丢包个数
6092	卡顿时间---毫秒

6093	出现丢帧
6094	首个音频帧收到
6099	其他上报消息
6105	游戏断开，正在重连。。。
6107	游戏重连成功
6110	声音卡顿时间
6111	服务端启动游戏的时间
6112	服务端游戏分辨率，如 1280x720
6117	客户端收到的视频帧的真实分辨率，如 1280x720
6122	单线模式，返回可选运营商 ["LT","DX","YD"]

附录二：手柄按键映射

KeyEvent.KEYCODE_DPAD_UP	游戏手柄 上键
KeyEvent.KEYCODE_DPAD_DOWN	游戏手柄 下键
KeyEvent.KEYCODE_DPAD_LEFT	游戏手柄 左键

KeyEvent.KEYCODE_DPAD_RIGHT	游戏手柄 右键
KeyEvent.KEYCODE_BUTTON_START	游戏手柄 START 键
KeyEvent.KEYCODE_BUTTON_SELECT	游戏手柄 SELECT 键
KeyEvent.KEYCODE_BUTTON_THUMBL	游戏手柄 左摇杆
KeyEvent.KEYCODE_BUTTON_THUMBR	游戏手柄 右摇杆
KeyEvent.KEYCODE_BUTTON_L1	游戏手柄 L1 键
KeyEvent.KEYCODE_BUTTON_R1	游戏手柄 R1 键
KeyEvent.KEYCODE_BUTTON_L2	游戏手柄 L2 键
KeyEvent.KEYCODE_BUTTON_R2	游戏手柄 R2 键
KeyEvent.KEYCODE_BUTTON_A	游戏手柄 A 键
KeyEvent.KEYCODE_BUTTON_B	游戏手柄 B 键
KeyEvent.KEYCODE_BUTTON_X	游戏手柄 X 键
KeyEvent.KEYCODE_BUTTON_Y	游戏手柄 Y 键

以上按键都是 Android KeyEvent 中定义的常量

附录三：键盘按键映射

KeyEvent.KEYCODE_ESCAPE	ESC
KeyEvent.KEYCODE_0	0
KeyEvent.KEYCODE_1	1
KeyEvent.KEYCODE_2	2

KeyEvent.KEYCODE_3	3
KeyEvent.KEYCODE_4	4
KeyEvent.KEYCODE_5	5
KeyEvent.KEYCODE_6	6
KeyEvent.KEYCODE_7	7
KeyEvent.KEYCODE_8	8
KeyEvent.KEYCODE_9	9
KeyEvent.KEYCODE_MINUS	-_
KeyEvent.KEYCODE_EQUALS	+=
KeyEvent.KEYCODE_DEL	DEL
KeyEvent.KEYCODE_TAB	TAB
KeyEvent.KEYCODE_LEFT_BRACKET	{[
KeyEvent.KEYCODE_RIGHT_BRACKET	]}
KeyEvent.KEYCODE_ENTER	ENTER
KeyEvent.KEYCODE_CTRL_LEFT	左 CTRL
KeyEvent.KEYCODE_SEMICOLON	；：
KeyEvent.KEYCODE_APOSTROPHE	‘ ”
KeyEvent.KEYCODE_GRAVE	~ ·
KeyEvent.KEYCODE_SHIFT_LEFT	左 SHIFT
KeyEvent.KEYCODE_BACKSLASH	\
KeyEvent.KEYCODE_COMMA	,

KeyEvent.KEYCODE_PERIOD	.
KeyEvent.KEYCODE_SLASH	/?
KeyEvent.KEYCODE_SHIFT_RIGHT	右 SHIFT
KeyEvent.KEYCODE_ALT_LEFT	左 ALT
KeyEvent.KEYCODE_SPACE	空格
KeyEvent.KEYCODE_CAPS_LOCK	CAPS_LOCK
KeyEvent.KEYCODE_F1	F1
KeyEvent.KEYCODE_F2	F2
KeyEvent.KEYCODE_F3	F3
KeyEvent.KEYCODE_F4	F4
KeyEvent.KEYCODE_F5	F5
KeyEvent.KEYCODE_F6	F6
KeyEvent.KEYCODE_F7	F7
KeyEvent.KEYCODE_F8	F8
KeyEvent.KEYCODE_F9	F9
KeyEvent.KEYCODE_F10	F10
KeyEvent.KEYCODE_F11	F11
KeyEvent.KEYCODE_F12	F12
KeyEvent.KEYCODE_NUM_LOCK	Num_lock
KeyEvent.KEYCODE_NUMPAD_7	7
KeyEvent.KEYCODE_NUMPAD_8	8

KeyEvent.KEYCODE_NUMPAD_9	9
KeyEvent.KEYCODE_NUMPAD_SUBTRACT	-
KeyEvent.KEYCODE_NUMPAD_4	4
KeyEvent.KEYCODE_NUMPAD_5	5
KeyEvent.KEYCODE_NUMPAD_6	6
KeyEvent.KEYCODE_NUMPAD_ADD	+
KeyEvent.KEYCODE_NUMPAD_0	0
KeyEvent.KEYCODE_NUMPAD_1	1
KeyEvent.KEYCODE_NUMPAD_2	2
KeyEvent.KEYCODE_NUMPAD_3	3
KeyEvent.KEYCODE_NUMPAD_DOT	.
KeyEvent.KEYCODE_NUMPAD_ENTER	小键盘 确认键
KeyEvent.KEYCODE_CTRL_RIGHT	右 CTRL
KeyEvent.KEYCODE_ALT_RIGHT	右 ALT
KeyEvent.KEYCODE_MOVE_HOME	HOME
KeyEvent.KEYCODE_MOVE_END	END
KeyEvent.KEYCODE_FORWARD_DEL	FORWARD_DEL
KeyEvent.KEYCODE_INSERT	INSTER
KeyEvent.KEYCODE_DPAD_UP	上
KeyEvent.KEYCODE_DPAD_LEFT	左
KeyEvent.KEYCODE_DPAD_RIGHT	右

KeyEvent.KEYCODE_DPAD_DOWN	下
KeyEvent.KEYCODE_PAGE_DOWN	PAGR_DOWN
KeyEvent.KEYCODE_PAGE_UP	PAGE_UP
KeyEvent.KEYCODE_MEDIA_PAUSE	PAUSE
KeyEvent.KEYCODE_A	A
KeyEvent.KEYCODE_B	B
KeyEvent.KEYCODE_C	C
KeyEvent.KEYCODE_D	D
KeyEvent.KEYCODE_E	E
KeyEvent.KEYCODE_F	F
KeyEvent.KEYCODE_G	G
KeyEvent.KEYCODE_H	H
KeyEvent.KEYCODE_I	I
KeyEvent.KEYCODE_J	J
KeyEvent.KEYCODE_K	K
KeyEvent.KEYCODE_L	L
KeyEvent.KEYCODE_M	M
KeyEvent.KEYCODE_N	N
KeyEvent.KEYCODE_O	O
KeyEvent.KEYCODE_P	P
KeyEvent.KEYCODE_Q	Q

KeyEvent.KEYCODE_R	R
KeyEvent.KEYCODE_S	S
KeyEvent.KEYCODE_T	T
KeyEvent.KEYCODE_U	U
KeyEvent.KEYCODE_V	V
KeyEvent.KEYCODE_W	W
KeyEvent.KEYCODE_X	X
KeyEvent.KEYCODE_Y	Y
KeyEvent.KEYCODE_Z	Z
KeyEvent.KEYCODE_NUMPAD_MULTIPLY	*
KeyEvent.KEYCODE_NUMPAD_SUBTRACT	-
KeyEvent.KEYCODE_NUMPAD_DIVIDE	/
KeyEvent.KEYCODE_NUM_LOCK	NUM_LOCK

以上按键都是 Android KeyEvent 中定义的常量

附录四：混淆配置

```
#蔚领 sdk 混淆

-keep class com.welinkpass.gamesdk.constants.WLCGGameConstant{*;}

-keep class com.welinkpass.gamesdk.constants.WLEventConstants{*;}

-keep class com.welinkpass.gamesdk.constants.WLErrorCode{*;}
```

```
-keep class com.welinkpass.gamesdk.entity.**{*;}

-keep class com.welinkpass.gamesdk.listener.WLPluginInstallListener{*;}

-keep class com.welinkpass.gamesdk.listener.WLPluginUpdateListener{*;}

-keep class com.welinkpass.gamesdk.view.**{*;}

-keep class com.welinkpass.gamesdk.WLCGConfig{*;}

-keep class com.welinkpass.gamesdk.gamecontror.**{*;}

-keep class com.welinkpass.gamesdk.utils.WLProcessUtils{*;}

-keep class com.welinkpass.bridge.**{*;}

#阿里动态加载混淆

-keep class com.aliott.agileplugin.AgileHostRuntime {*;}

-keep class com.aliott.agileplugin.proxy.** {*;}

-keep class com.aliott.agileplugin.component.**{*;}

-keep class com.aliott.agileplugin.redirect.**{*;}

-keep class com.aliott.agileplugin.runtime.** {*;}

-keep class com.aliott.agileplugin.utils.ServiceChecker {*;}

-keep class com.aliott.agileplugin.proxy.PluginProxyActivity {*;}

-keep class com.aliott.agileplugin.bridge.AgilePluginBridge {*;}

-keep class com.aliott.agileplugin.AgilePluginManager {*;}

-keep class com.aliott.agileplugin.AgilePlugin {*;}

#dontnote 指定不去输出打印该类产生的错误或遗漏

-dontnote com.aliott.agileplugin.**
```

```
-dontnote com.welinkpass.bridge.**

-dontwarn com.aliott.agileplugin.**
```

第三方库混淆

```
#fastjson

-dontwarn com.alibaba.fastjson.**

-keep class com.alibaba.fastjson.** { *; }

-keepattributes Signature

-keepattributes *Annotation*

# OkHttp3

-dontwarn com.squareup.okhttp3.**

-keep class com.squareup.okhttp3.** { *;}

# Okio

-dontwarn com.squareup.**

-dontwarn okio.**

-keep public class org.codehaus.* { *; }

-keep public class java.nio.* { *; }
```

附录五：WLErrorCode 错误码

ERROR_UNKNOW	安装/更新 插件失败,未知错误
ERROR_INSTALL_GET_GAMESERVICE	安装插件，获取操作类出错

ERROR_INIT_FAIL	阿里注册插件出错 具体看 extraMsg
ERROR_CAN_NOT_FIND_PLUGIN	安装插件时，配置的插件未找到， 可能没有加载插件信息
ERROR_INSTALL_APK_FAIL	安装插件，installPluginApk 时复制本地插件出错 可能是复制插件 apk 文件出错
ERROR_INSTALL_SO_LIB_FAIL	安装插件，加载插件中的 so 出错， 可能是复制 so 文件出错
ERROR_INSTALL_DEX_FAIL	安装插件，加载插件里的 dex 文件失败
ERROR_INSTALL_CONTEXT_FAIL	安装插件，加载插件的 context，出错
ERROR_INSTALL_PACKAGE_INFO_FAIL	安装插件，加载插件的 PackageInfo 出错
ERROR_INSTALL_APPLICATION_FAIL	安装插件，加载插件的 Application 出错
ERROR_INSTALL_REMOTE_FAIL	安装插件，加载远程插件出错
ERROR_INSTALL_LOADED_APK	安装插件，installLoadedApk 出错
ERROR_UPDATE_PREPARE_APK_FAIL	更新插件失败，新插件的文件不存在， 可能是复制插件 apk 文件失败
ERROR_UPDATE_PREPARE_SO_LIB_FAIL	更新插件失败，解压复制新插件 so 失败
ERROR_UPDATE_PREPARE_DEX_FAIL	更新插件失败：更新新插件里的 dex 文件失败
ERROR_UPDATE_PREPARE_PACKAGE_FAIL	更新插件失败 1.加载新插件里的 packageInfo 失败 2.更新的插件文件中的版本号 与更新信息中的版本号校验不一致

ERROR_UPDATE_BY_PLUGIN_VERSION_IS_LESS_CURRENT	更新插件失败： 当前插件版本号高于更新的版本号
ERROR_CHECK_UPDATE	请求服务器是否有新插件时出错
ERROR_CHECK_MD5	md5 校验不通过
ERROR_UPGRADE_VERSION	更新插件失败，新插件的版本号是空的
ERROR_SET_NEW_VERSION	更新插件失败，设置新的版本号失败
ERROR_DOWNLOAD_FILE	下载服务器新插件到本地时出错
ERROR_ACCESS_OSS	请求服务器下载新插件时出错
ERROR_UPDATE_DO_NOT_HAS_THIS_PLUGIN	更新插件失败，不包含此 plugin
ERROR_UPDATE_NEWPLUGINFILE_NOT_ACCESSABLE	更新插件失败，插件文件不可访问
ERROR_UPDATE_PLUGIN_NAME_NOT_MATCH	更新插件失败 插件名跟更新文件中的插件名不一致
ERROR_UPDATE_GET_PLUGINFILE_PLUGINNAME_FAIL	获取插件文件中的配置的插件名失败 插件不是蔚领 GAME 插件
ERROR_UPDATE_AVAILABLE_SPACE_NOT_ENOUGH	更新插件失败，sd 卡剩余空间不足
ERROR_UPDATE_OLDPLUGIN_NOT_ACCESSABLE	补丁更新模式失败，访问旧的插件文件失败
ERROR_UPDATE_PATCHFILE_NOT_ACCESSABL	补丁更新模式失败：补丁文件不可访问

E	
ERROR_UPDATE_PATCHFILE_CHECKMD5	补丁更新模式失败：补丁文件 MD5 校验失败
ERROR_UPDATE_GENERATE_FILE	补丁更新模式失败：合成新插件文件失败
ERROR_UPDATE_PATCH_UNKNOW	补丁更新模式失败：未知错误