

YJOpenSDK-Android 接入指南

目录

1. 获取App SDK	-----
2. YJOpenSDK架构及调用流程	-----
• 2.1 整体架构	-----
• 2.2 基本调用流程	-----
3. 集成流程	-----
4. 基础功能	-----
• 4.1 初始化	-----
• 4.2 API 网络请求	-----
• 4.3 账户及登录	-----
• 4.3.1 获取验证码	-----
• 4.3.2 检查验证码是否正确	-----
• 4.3.3 注册	-----
• 4.3.4 验证码登录	-----
• 4.3.5 密码登录	-----
• 4.3.6 账号退出	-----
• 4.3.6 验证码枚举类说明	-----
• 4.3.7 退出登录	-----
• 4.3.8 监听token过期	-----
5. 绑定	-----
• 5.1 配网前检测（所有类型配网前均可使用）	-----
• 5.1.1 接口定义	-----
• 5.1.2 代码示例	-----
• 5.2 WIFI-AP绑定	-----
• 5.2.1 配网流程	-----
• 5.2.2 接口定义	-----
• 5.2.3 代码示例	-----
• 5.3 4G/有线绑定	-----
• 5.3.1 接口定义	-----
• 5.3.2 代码示例	-----
• 5.4 蓝牙绑定	-----

• 5.4.1 配网流程	-----
• 5.4.2 接口定义	-----
• 5.4.2 代码示例	-----
• 5.5 绑定步骤枚举	-----
6. 媒体播放	-----
• 播放器基本接口	-----
• 播放状态回调	-----
• 6.1 直播	-----
• 6.2 云录像	-----
• 6.3 卡录像	-----
7. 云端API	-----
• 7.1 云端接口请求头	-----
• 7.2 云端接口	-----
• 7.2.1 用户设备列表	-----
• 7.2.2 物模型下行指令	-----
• 7.2.3 设备能力级获取	-----
• 7.2.4 单设备一天录像批量查询接口	-----
• 7.2.5 查询要展示的告警日期	-----
• 7.2.6 查询事件列表	-----
8. SDK发布说明	-----
• 8.1 名词解释	-----
• 8.2 功能介绍	-----
9. 隐私声明	-----
• 9.1 收集个人信息说明	-----
• 9.2 权限说明	-----
SDK版本更新说明	-----

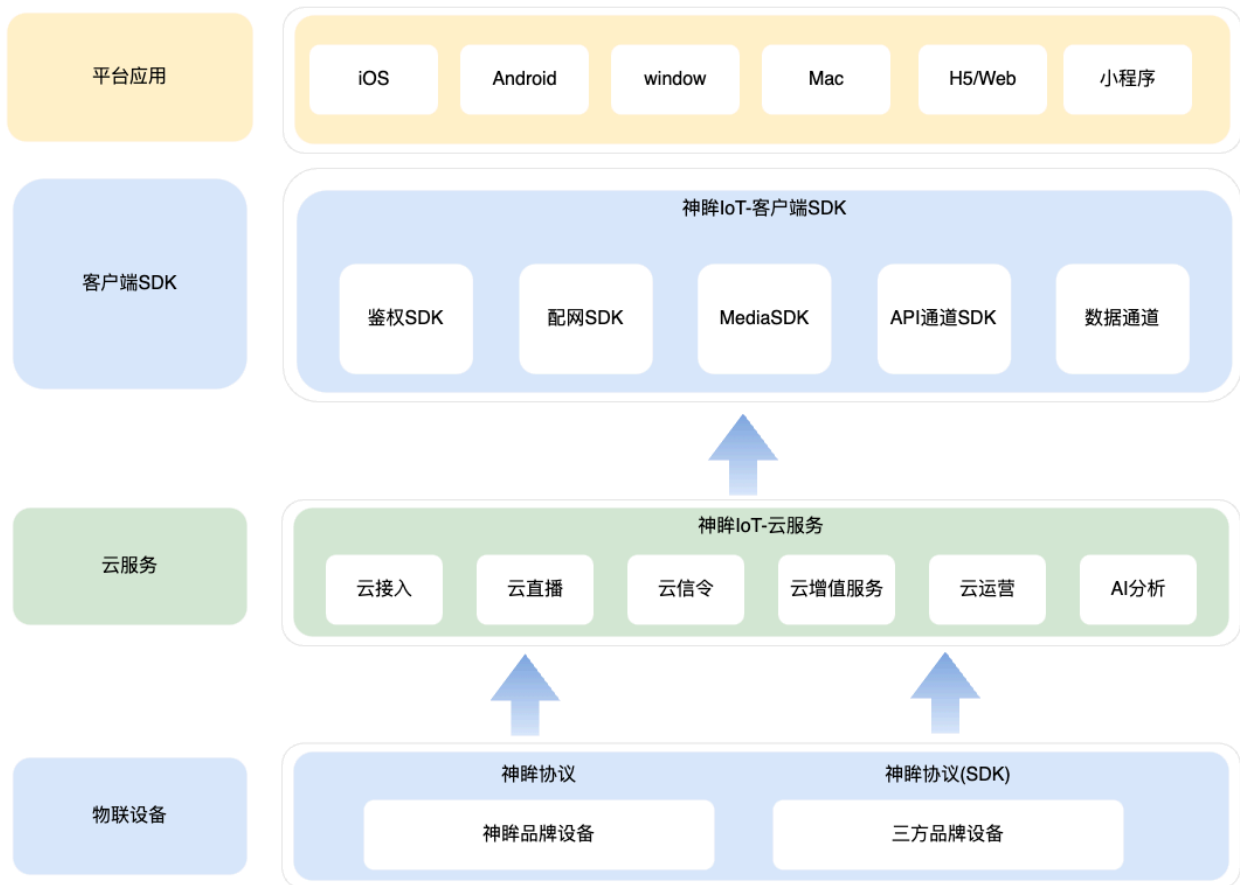
1. 获取App SDK

神眸App SDK是神眸针对视频物联场景所提供的应用端SDK。基于该SDK各项配置项开发包，可以实现用户账号、流媒体服务、设备控制、配网开发等功能。

关于App SDK的申请和权限获取可联系您的客户经理。

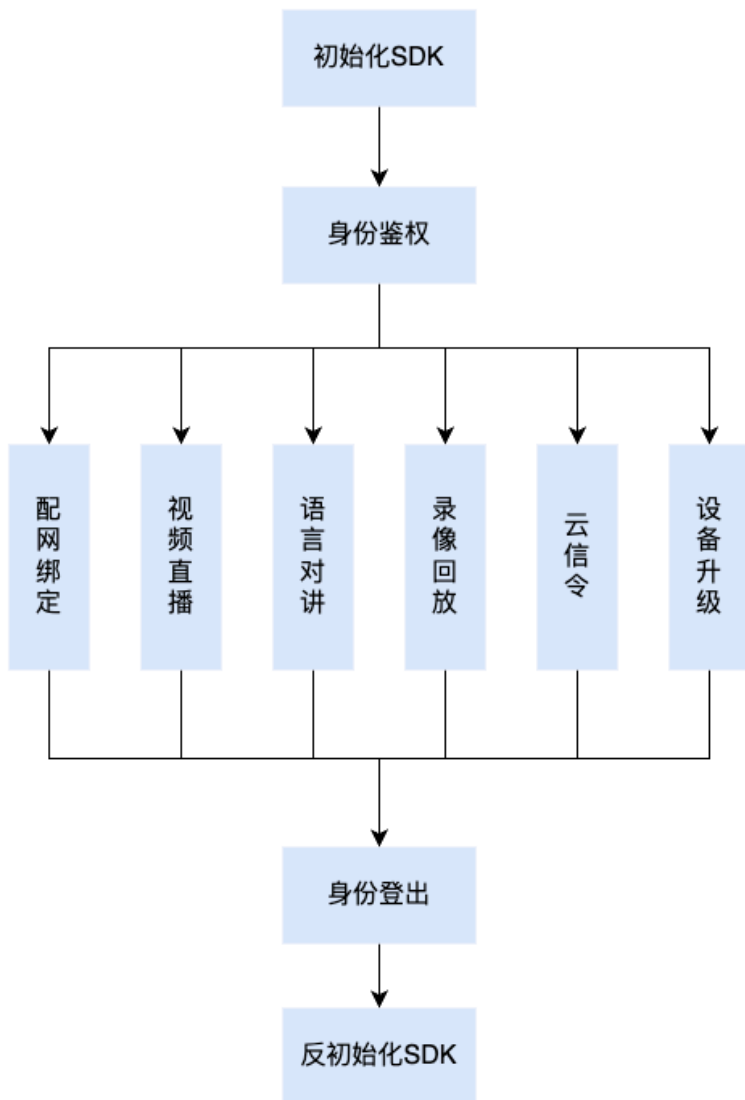
2. YJOpenSDK架构及调用流程

2.1 整体架构



该篇文档主要描述Android端OpenSDK接入指南

2.2 基本调用流程



3. 集成流程



复制代码

```
implementation("com.superacme.android:opensdk:0.5.0-SNAPSHOT")
```

4. 基础功能

4.1 初始化



复制代码

```
val config = Configuration.Builder(this)
    .gatewayConfig(
```

```

        gatewayConfig = GatewayConfig.Builder()
            .appKey("xxxxx")
            .appSecret("xxxxx")
            .header("appKey", "xxxxx")
            .header("appVersion", "0.0.1")
            .header("clientType", "android")
            .header("utdid", "xxxx")
            .header("utdName", "xxxxx")
            .header("timeZone", "8")
            .header("Accept-Language", "zh-CN")
            .build()
    )
    .logStrategy(object : LogStrategy {
        override fun log(priority: Int, tag: String, message:
String) {
            Log.println(priority, "Demo-$tag", message)
        }
    })
    .region("China")
    .debugMode(true)
    .params("timeZone", 8)
    .params("countryCode", "CN")
    .params("Accept-Language", "zh-CN")
    .build()

```

参数解释：

appKey：开发者在神眸平台申请的应用凭证

appSecret： app的密钥（需要保护好，防止泄露）

clientType：客户端的类型。（比如android、ios）

utdid：设备序列号

utdName：设备型号

timeZone：设备所在时区

Accept-Language： app 所支持的语言

logStrategy：是统一的sdk 输出日志

4.2 API 网络请求

api 网络请求采用Retrofit 调用形式，例如获取设备列表

```
@POST("/user/api/v2/allGroupDeviceList")
suspend fun getDeviceList(): NetResult<HomeModel>
```

调用形式如下：

```
AcmeGateWay.createApi(HomeApi::class.java).getDeviceList()
```

4.3 账户及登录

4.3.1 获取验证码

```
/**
 * 获取验证码
 */
fun getSMSCode(
    phoneNum: String? = null,
    phoneArea: String? = null,
    lang: String?,
    country: String?,
    requestType: RequestSMSCodeType,
    email: String? = null,
    callback: ((Boolean, String?) -> Unit)? = null
)
```

4.3.2 检查验证码是否正确

```
/**
 * 检查验证码是否正确
 */
fun checkSMSCode(
    code: String,
    requestType: String,
    phoneCode: String?,
    phoneAreaCode: String?,
    email: String?,
    callback: ((Boolean, String?) -> Unit)? = null
)
```

4.3.3 注册

```
fun register(  
    phoneNum: String?,  
    email: String?,  
    smsCode: String,  
    pwdMD5: String,  
    requestType: String,  
    phoneArea: String?,  
    callback: ((Boolean, String) -> Unit)? = null  
)
```

复制代码

4.3.4 验证码登录

```
/**  
 * callback (loginResult, showFirstGivenMemberTip, errorMsg)  
 */  
fun loginWithSMSCode(  
    code: String,  
    phoneCode: String?,  
    phoneAreaCode: String?,  
    email: String?,  
    callback: ((Boolean, Boolean, String?) -> Unit)? = null  
)
```

复制代码

4.3.5 密码登录

```
fun login(userName: String, password: String, loginCallback:  
ILoginCallback)
```

复制代码

4.3.6 账号退出

```
fun logout(callback: ILogoutCallback)
```

复制代码

4.3.6 验证码枚举类说明

复制代码

```
/**
 * 验证码根据传入的字符串区分用途，注册：signup,忘记密码：forgetpass,终端绑定
验证：macbindcheck,
 * 用户绑定/换绑第一步：userchangebindstepone, 用户绑定/换绑第二步：
userchangebindsteptwo,用户通过验证码登录：login
 */
enum class RequestSMSCodeType(val str: String) {
    LOGIN("login"),
    SIGN_UP("signup"),
    FORGETPASS("forgetpass"),
    RESET_PWD("resetpwd"),
    MACBINDCHECK("macbindcheck"),
    USERCHANGEBINDSTEPONE("userchangebindstepone"),
    USERCHANGEBINDSTEPTWO("userchangebindsteptwo"),
}
```

4.3.7 退出登录

复制代码

```
fun logout(callback: ILogoutCallback)
```

调用实例：

1. 获取验证码getSMSCode。
2. 检查验证码是否正确checkSMSCode
3. UI界面设置密码
4. 调用注册接口register

4.3.8 监听token过期

监听token 过期，需要跳转到登录界面，重新登录，参考以下代码

复制代码

```
SessionManager.setRefreshTokenInvalidListener(object :
RefreshTokenInvalidListener {
```

```

        override fun onRefreshTokenInvalided() { //需要跳转到登录界面，可能会返回多次，需要应用开发者对这种情况下进行处理
            startActivity(Intent(this@SampleApplication,
                LoginActivity::class.java).apply {
                    this.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
                })
        }
    })
}

```

5. 绑定

5.1 配网前检测（所有类型配网前均可使用）

5.1.1 接口定义


 复制代码

```

/**
 * 配网前检测
 * 配网前需要对设备序列号进行校验，主要校验包括：合法性、是否已被绑定、产品
 * 基础信息、支持的配网方式、等
 *
 * @param deviceName
 * @param callBack
 */
fun getPrepareDeviceInfo(
    deviceName: String,
    callBack: (info: String?, error: Exception?) -> Unit?
)

```

5.1.2 代码示例

 复制代码

```

override suspend fun getDeviceInfo(deviceName: String): DeviceInfoV0? {
    return suspendCancellableCoroutine {
        connectApi.getPrepareDeviceInfo(deviceName = deviceName,
            callBack = { info, error ->
                Logger.i(TAG, "getBindBeforeInfo: info : $info")
                Logger.i(TAG, "getBindBeforeInfo: error : $error")
                if (info != null) {

```

```

        val bindBeforePO = Gson().fromJson(info,
BindBeforePO::class.java)
        Logger.i(TAG, "bindBeforePO: info : $bindBeforePO")
        it.resume(parseDeviceInfo(bindBeforePO))
    } else {
        it.resume(null)
    }
}
})
}
}

```

5.2 WIFI-AP绑定

5.2.1 配网流程



5.2.2 接口定义

```

/**
 * 配网前，进行设备信息配置
 */
fun configBindDevice(
    productKey: String, deviceName: String,
    callBack: (info: String?, error: Exception?) -> Unit?
)

/**
 * AP热点配网绑定
 * 正式进入配网绑定流程，需传入配网的WiFi相关信息，可通过bindStep监听绑定
流程进行到的节点；
 * 该流程会有两个回调：
 * a. 配网过程回调：在配网流程中将配网进度节点回调；
 * b. 最终配网结果回调：设备添加绑定最终结果回调；
 */
fun bindDevice(
    wifiSSID: String,
    wifiPW: String,
    bindStep: (step: BindDeviceStep) -> Unit?,

```

```

        result: (failStep: BindDeviceStep?, result: String?, error:
Exception?) -> Unit?
    )

    /**
     * 获取设备热点名称
     * 在完成AP热点配网配置后，可获得设备热点名称，
     * 如果未完成，此处会返回null
     */
    fun deviceHotspotName(): String?

    /**
     * 终止配网
     * 在发起AP热点绑定后，可以终止配网
     * 但如果设备侧已经在绑定过程中，是无法真正终止配网，设备仍可能完成绑定流转
     */
    fun stopBindDevice()

    /**
     * 清理缓存
     * 配网结束后清理缓存，快速释放
     */
    fun clearUp()

```

5.2.3 代码示例

∨
复制代码

```

    override suspend fun configBindDevice(productKey: String,
deviceName: String): String {
        return suspendCancellableCoroutine {
            connectApi.configBindDevice(
                productKey = productKey,
                deviceName = deviceName,
                callBack = { info, error ->
                    it.resume(connectApi.deviceHotspotName() ?: "")
                })
        }
    }

    override suspend fun bindDevice(wifiSSID: String, wifiPW: String):
Boolean {
        return suspendCancellableCoroutine {
            connectApi.bindDevice(wifiSSID = wifiSSID, wifiPW = wifiPW,
bindStep = { step ->

```

```

        Logger.i(TAG, "WifiApConnectRepoImpl.bindDevice: step : $step")
    }, result = { failStep, result, error ->
        Logger.i(TAG, "WifiApConnectRepoImpl.bindDevice: failStep : $failStep")
        Logger.i(TAG, "WifiApConnectRepoImpl.bindDevice: result : $result")
        Logger.i(TAG, "WifiApConnectRepoImpl.bindDevice: error : $error")

        if (result != null) {
            it.resume(true)
        }
        if (error != null) {
            it.resume(false)
        }
    })
}

override fun stopBindDevice() {
    connectApi.stopBindDevice()
}

override fun clearUp() {
    connectApi.clearUp()
}
}

```

5.3 4G/有线绑定

对于有线设备，可以通过4G网络进行设备配网绑定

5.3.1 接口定义

▽
复制代码

```

/**
 * 4G网络绑定
 * 正式进入配网绑定流程，可通过bindStep监听绑定流程进行到的节点；
 * 最终配网结果回调：设备添加绑定最终结果回调；
 */
fun bindDevice(
    productKey: String,
    deviceName: String,
    result: (result: String?, error: Exception?) -> Unit?
)

```

```

/**
 * 终止配网
 * 但如果设备侧已经在绑定过程中，是无法真正终止配网，设备仍可能完成绑定流转
 */
fun stopBindDevice()

/**
 * 清理缓存
 * 配网结束后清理缓存，快速释放
 */
fun clearUp()

```

5.3.2 代码示例

 复制代码

```

    override suspend fun bindDevice(productKey: String, deviceName:
String): Boolean {
        return suspendCancellableCoroutine {
            connectApi.bindDevice(
                productKey = productKey,
                deviceName = deviceName,
                result = { result, error ->
                    Logger.i(TAG, "CellularConnectRepoImpl.bindDevice:
result : $result")
                    Logger.i(TAG, "CellularConnectRepoImpl.bindDevice:
error : $error")
                    if (result != null) {
                        it.resume(true)
                    }
                    if (error != null) {
                        it.resume(false)
                    }
                })
        }
    }

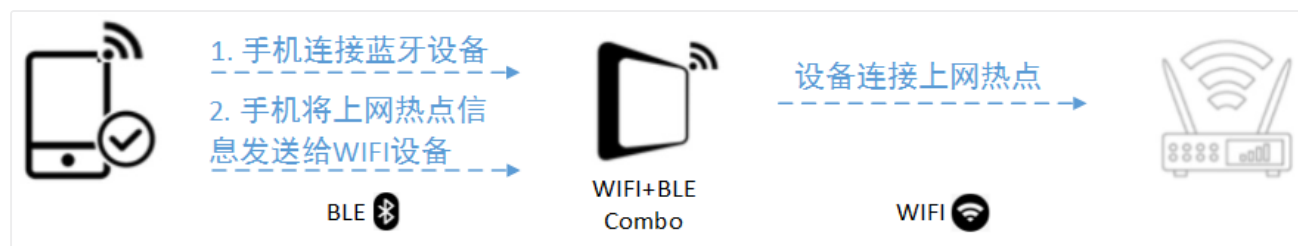
    override fun stopBindDevice() {
        connectApi.stopBindDevice()
    }

    override fun clearUp() {
        connectApi.clearUp()
    }

```

5.4 蓝牙绑定

5.4.1 配网流程



5.4.2 接口定义

复制代码

```
/**
 * 发起蓝牙扫描
 *
 * @param callBack 会多次回调扫描到的设备
 */
fun scanDevice(callBack: (scanDeviceInfoList:
List<ScanDeviceInfo>?, error: Exception?) -> Unit?)

/**
 * 停止蓝牙扫描
 */
fun stopScan()

/**
 * 发起蓝牙配网绑定
 * 正式进入配网绑定流程，需传入配网的WiFi相关信息，可通过bindStep监听绑定
流程进行到的节点；
 * 该流程会有两个回调：
 * a. 配网过程回调：在配网流程中将配网进度节点回调；
 * b. 最终配网结果回调：设备添加绑定最终结果回调；
 */
fun bindDevice(
    scanDeviceInfo: ScanDeviceInfo,
    wifiSSID: String,
    wifiPW: String,
    bindStep: (step: BindDeviceStep) -> Unit?,
    result: (failStep: BindDeviceStep?, result: String?, error:
Exception?) -> Unit?
)

/**
 * 终止配网
 * 在发起蓝牙绑定后，可以终止配网
```

```

    * 但如果设备侧已经在绑定过程中，是无法真正终止配网，设备仍可能完成绑定流转
    */
fun stopBind()

/**
 * 清理缓存
 * 配网结束后清理缓存，快速释放
 */
fun clearUp()

```

5.4.2 代码示例

 复制代码

```

    override suspend fun scanDeviceList(callBack: (scanDeviceInfoList:
List<ScanDeviceInfo>?, error: Exception?) -> Unit?) {
        connectApi.scanDevice(callBack)
    }

    override fun stopScanDeviceList() {
        connectApi.stopScan()
    }

    override suspend fun bindDevice(
        scanDeviceInfo: ScanDeviceInfo,
        wifiSsid: String,
        wifiPwd: String
    ): Boolean {
        return suspendCancellableCoroutine {
            connectApi.bindDevice(
                scanDeviceInfo = scanDeviceInfo,
                wifiSSID = wifiSsid,
                wifiPW = wifiPwd,
                bindStep = { step -> },
                result = { failStep, result, error ->
                    if (result != null) {
                        it.resume(true)
                    }
                    if (error != null) {
                        it.resume(false)
                    }
                }
            )
        }
    }
}

```



```

    override fun stopBindDevice() {
        connectApi.stopBind()
    }

    override fun clearUp() {
        connectApi.clearUp()
    }

```

5.5 绑定步骤枚举

复制代码

```

enum class BindDeviceStep {
    DEFAULT,
    READY_BIND,
    CONNECTED_DEVICE_HOT,
    SEND_WIFI_INFO,
    DEVICE_ONLINE,
    DEVICE_BIND
}

```

6. 媒体播放

播放器基本接口

复制代码

```

public interface IRMPPlayer {
    //设置渲染控件
    void setVideoView(RMPVideoView view);

    //设置播放状态回调
    void setPlayerListener(RMPlayerListener listener);

    //开始播放
    int start();

    //停止播放
    int stop();

    /**
     * 获取播放统计, 如fps, 码流等
     * @param stat output object
     * @return true if get valid stat, false otherwise
     */
}

```

```

    boolean getStats(RMPlayerStatistics stat);

    //截图
    int snapshot(String file);

    //开始录制mp4
    int startFileRecording(String file);

    //停止录制mp4
    int stopFileRecording();


    //静音
    int muteRemoteAudio(boolean mute);

    //获取录制时长
    long getFileRecordingDuration();

    //释放播放器
    void release();
}

```

播放状态回调

 复制代码

```

public interface RMPlayerListener {
    /**
     * 播放错误回调，如拉流超时等
     * @param type {@link RMPlayerErrorType}
     * @param code {@link RMPlayerErrorCode}
     * @param desc
     */
    void OnError(@RMPlayerErrorType int type, @RMPlayerErrorCode int
code, String desc);

    /**
     *
     * @param state {@link RMPlayerState}
     * @param extra play speed
     */
    void OnPlayerStateChange(@RMPlayerState int state, int extra);

    /**
     *
     */
}

```

```

    * @param state {@link RMPlayerTalkState}
    */
    void OnTalkStateChange(@RMPlayerTalkState int state);
    void OnPlaybackSpeedUpdate(int speed);
    void OnSeekComplete(boolean success);

    /**
     *
     * @param state {@link RMPlayerBufferState}
     * @param buffer_duration
     */
    void OnBufferStateUpdate(@RMPlayerBufferState int state, long
buffer_duration);

    //player state callback
    void OnFirstFrameRendered(long elapse_ms);
    void OnVideoSizeChanged(int width, int height);

    //snapshot callback

    /**
     *
     * @param file
     * @param result {@link RMPlayerSnapshotResult}
     * @param desc
     */
    void OnSnapshotResult(String file, @RMPlayerSnapshotResult int
result, String desc);
    //recording state callback
    void OnFileRecordingStart(String file);

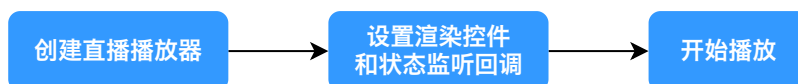
    /**
     *
     * @param file
     * @param code RMPlayerRecordingError
     * @param desc
     */
    void OnFileRecordingError(String file, @RMPlayerRecordingError int
code, String desc);
    void OnFileRecordingFinish(String file);

    void OnVodPlayProgress(long millis);
    void OnVodPlayComplete();
}

```

6.1 直播

播放流程



示例代码

⌵ 复制代码

```
IRMPLivePlayer livePlayer = new
RMPNetPlayerBuilder(getApplicationContext())
    .setDeviceInfo(deviceName, productKey)
    .createLivePlayer();

livePlayer.setPlayerListener(this);
livePlayer.setVideoView(renderView);
livePlayer.start();

//停止播放
livePlayer.stop();

//重新开始播放
livePlayer.start();

//销毁播放器
livePlayer.release();
```

6.2 云录像

播放流程



示例代码

复制代码

```
IRMPVodPlayer vodPlayer = new
RMPNetPlayerBuilder(getApplicationContext())
    .setDeviceInfo(deviceName, productKey)
    .createCloudVodPlayer();

vodPlayer.setPlayerListener(this);
vodPlayer.setVideoView(renderView);
vodPlayer.setCloudSource(url, meta, IRMPVodPlayer.VOD_MODE_All);
vodPlayer.start();

//暂停播放
vodPlayer.pause();

//继续播放
vodPlayer.resume();

//播放新的云存录像
vodPlayer.stop();
vodPlayer.setCloudSource(url2, meta2, IRMPVodPlayer.VOD_MODE_All);
vodPlayer.start();
```

```
//销毁播放器  
vodPlayer.release()
```

6.3 卡录像



代码

```
IRMPVodPlayer vodPlayer = new  
RMPNetPlayerBuilder(getApplicationContext())  
    .setDeviceInfo(deviceName, productKey)  
    .createVodPlayer();  
  
vodPlayer.setPlayerListener(this);  
vodPlayer.setVideoView(renderView);  
//设置播放时间范围  
//startSec, nowSec: unix时间, 单位秒  
//seekSec: 相对startSec的偏移时间  
vodPlayer.setDeviceSource(startSec, nowSec, seekSec);  
vodPlayer.start();  
  
//暂停播放  
vodPlayer.pause();  
//继续播放  
vodPlayer.resume();
```

复制代码

```
//播放新的时间范围
vodPlayer.stop();
vodPlayer.setDeviceSource(startSec2, nowSec2, seekSec2);
vodPlayer.start();

//销毁播放器
vodPlayer.release()
```

7. 云端API

7.1 云端接口请求头

请求头参数名称	参数值	是否必须
Content-Type	application/json	是
token	登录后返回	是

7.2 云端接口

云端接口请求方式均为post方式

7.2.1 用户设备列表

接口路径：/user/api/v1/deviceList

接口入参：

参数名称	类型	是否必须	备注
owned	int	否	绑定类型 0-分享的设 备， 1-自己绑定的设 备， 不传代表-所有设备
productKeyList	string[]	否	产品key数组

出参：

名称	类型	是否必须	备注
code	number	必须	code = 0成功
success	boolean	必须	
message	string	必须	
data	object []	必须	
id	string	必须	设备id
gmtCreate	string	必须	创建时间
gmtModified	string	必须	修改时间
productKey	string	非必须	自研设备所属产品的ProductKey
deviceName	string	非必须	自研设备序列号
iotId	string	必须	云端设备唯一标识
nickName	string	必须	设备昵称
picUrl	string	必须	设备图片地址
devType	string	非必须	设备类型
devModel	string	非必须	设备型号
status	string	非必须	自研设备状态。0（表示未激活）；1（表示在 （表示离线）；8（表示禁用）
groupId	string	非必须	分组id
groupName	string	非必须	分组名称
owned	string	必须	0- 分享 1-自己绑定
deviceSource	string	必须	设备来源分类
productSource	integer	必须	产品来源分类

7.2.2 物模型下行指令

接口路径：/operation/api/v1/unified/operation/down

接口入参：

名称	类型	是否必须	默认值	备注	其他信息
productKey	string	非必须		产品关键KEY	
deviceName	string	非必须		设备名称，和 productKey唯一	
deviceId	string	非必须		设备 id,productKey 与deviceName 组合,二选一	
method	string	必须		物模型方法： 属性操作 thing.service.pr operty.set，服 务使用 thing.service.\${ tsl.service.iden tifier}	
params	object	必须		jsonSchema支 持的数据类型	备 注: jsonSchem a支持的数据类 型
identifier	string	必须		功能点唯一标 识	
id	string	必须		调用端流水号	
version	string	非必须	1.0	默认1.0	
operatingMode	number	非必须	0	0是同步，1是 异步，默认同	

				步	
channelId	integer	非必须		通道号,多目设备需要传指定通道号,非多目设备此参数不传	

出参：

名称	类型	是否必须	默认值	备注	其他信息
code	integer	非必须			
message	string	非必须			
data	object	非必须			

7.2.3 设备能力级获取

接口路径：/operation/api/v1/unified/operation/tag/key/get

接口入参：

名称	类型	是否必须	默认值	备注	其他信息
deviceIds	string []	非必须		设备id列表，为空就查询用户所有的设备	item 类型: string
attrKey	string	必须		约定的标签编号，如能力级传Capabilities	固定值 Capabilities

出参：

名称	类型	是否必须	默认值	备注	其他信息
code	integer	非必须			
message	string	非必须			

data	object []	非必须			item 类型: object
deviceId	string	非必须			
attrKey	string	非必须		标签key	
attrValue	string	非必须		标签值	
gmtModified	integer	非必须		修改时间	

7.2.4 单设备一天录像批量查询接口

接口路径: /message/api/v1/event/alarm/video/file/tag

接口入参:

Headers:

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是	application/json	
Accept-Language		否		多语言, 中国zh-CN
timeZone		否		当前手机时区

Body:

名称	类型	是否必须	默认值	备注	其他信息
eventDay	string	必须		事件发生的当天yyyy-mm-DD形式 eventDay:如, 2022-10-01	
deviceId	integer	必须		设备ID集合	
alarmType	integer	非必须		告警类型。1 (表示移动侦	

				测)；2（表示声音侦测）；3（表示人形侦测）.....	
--	--	--	--	-----------------------------	--

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	非必须			
message	string	非必须			
data	object	非必须			
type	string	非必须		压缩类型,gzip	
encodedContent	string	非必须		压缩内容	

encodeContent，先用Base64解密，再用gzip解压，得到一个json，转对象内容如下：

eventId	string	非必须		事件id	
time	integer	非必须		发生时间，unix时间戳,服务器时区,东八区	
eventTime	string	非必须		事件发生时间，格式为yyyy-MM-ssHH:mm:ss。	
payload	string	非必须		设备上报事件的具体内容	
picUrl	string	非必须		图片url,服务已经转换，可以直接显示	

videoUrl	string	非必须		视频url，返回的是设备上传的索引文件，需要端上自己去转	
alarmType	integer	非必须		告警类型。1（表示移动侦测）；2（表示声音侦测）；3（表示人形侦测）.....	
alarmName	string	非必须		报警名称	
productKey	string	非必须		产品编码key	
deviceName	string	非必须		产品编码名，与产品编码key配套，设备唯一	
deviceId	integer	非必须		在平台的设备唯一标识符	

7.2.5 查询要展示的告警日期

接口路径：/message/api/v1/event/alarm/date

接口入参：

Headers：

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是	application/json	
timeZone		否		正整数，中国8

Body:

名称	类型	是否必须	默认值	备注	其他信息
startDate	string	必须		yyyy-mm-DD形式 startDate:如, 2022-10-01	
endDate	string	必须		yyyy-mm-DD形式 endDate: 如, 2022-10-03	
deviceIds	integer []	必须		设备ID集合	item 类型: integer
blnContainVideo	boolean	非必须		是否包含video, 如果查询结果一定要有 videoUrl,此值 true.默认false	

返回数据

名称	类型	是否必须	默认值	备注	其他信息
code	integer	非必须			
message	string	非必须			
data	object	非必须			
dateList	string []	非必须		存在事件发生的日期	

7.2.6 查询事件列表

接口路径: /message/api/v1/event/query

接口入参:

Headers:

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是	application/json	
Accept-Language		否		zh-CN
timeZone		否		正整数，中国8

Body:

名称	类型	是否必须	默认值	备注	其他信息
beginTime	integer	必须		查询开始时间。单位毫秒。	
endTime	integer	必须		查询结束时间。单位毫秒。	
alarmType	integer	非必须		告警类型。1（表示移动侦测）；2（表示声音侦测）；3（表示人形侦测）.....	
pageSize	integer	非必须		分页大小，不传默认值是20，最大100。	
deviceIds	integer []	必须		设备ID集合	item 类型: integer
nextToken	string	非必须		当符合查询条件的数据未读取完时，服务端会返回	

				nextToken，此时可以使用nextToken继续读取后面的数据。(nextToken在数据库技术上无法去除最后一次多查一次的问题。)使用token进行翻页时默认只能向后翻页。由于在一次查询的翻页过程中token长期有效，您可以通过缓存并使用之前的token实现向前翻页。	
--	--	--	--	--	--

返回数据：

名称	类型	是否必须	默认值	备注	其他信息
code	integer	非必须			
message	string	非必须			
data	object	非必须			
eventQueryRespDTOs	object []	非必须		查询数据集	item 类型: object
eventId	string	非必须		事件id	
productKey	string	非必须		产品编码key	
deviceName	string	非必须		产品编码名，与产品编码key	

				配套，设备唯一	
deviceId	integer	非必须		在平台的设备唯一标识符	mock: @datetime("")
alarmName	string	非必须		告警名称，如人形侦测，移动侦测等	
eventType	integer	非必须		事件类型。1 (alert) ; 2 (info) ; 3 (error)	
alarmType	integer	非必须		告警类型。1 (表示移动侦测) ; 2 (表示声音侦测) ; 3 (表示人形侦测)	
deviceNickname	string	非必须		设备别名,可设置的名称	
eventName	string	非必须		事件名称	
identifier	string	非必须		事件标识	
payload	string	非必须		设备上报事件的具体内容	
picUrl	string	非必须		图片url,服务已经转换，可以直接显示	
videoUrl	string	非必须		视频url，返回的是设备上传的索引文件，需要端上自己去转	

time	integer	非必须		发生时间, unix 时间戳,服务器 时区,东八区	
eventTime	string	非必须		事件发生时间, 格式为 yyyy-MM-ss HH:mm:ss。	
eventTimeUTC	string	非必须		事件发生UTC 时间, 格式为 yyyy-MM- ssTHH:mm:ssZ 。	
createTime	integer	非必须		入库时间	
deviceIcon	string	非必须		设备图标url	
iconUrl	string	非必须		报警图标url	
summary	string	非必须		消息摘要	
nextToken	string	非必须		当符合查询条件的数据未读取完时, 服务端会返回 nextToken, 此时可以使用 nextToken继续 读取后面的数据。(nextToken 在数据库技术上无法去除最后一次多查一次的问题。)	
totalCount	integer	非必须		每次查询, 能 匹配到的结果 总数	

8. SDK发布说明

8.1 名词解释

名词	注解
appKey	开发者在神眸平台申请的appKey
appSecret	开发者在神眸平台申请的appSecret
ProductKey	产品序列号，产品唯一标志
DeviceName	设备序列号，设备唯一标志
ChannelId	设备通道号
OSD	视频播放当前时间
PTZ	云台控制，可以通过终端控制操作设备

8.2 功能介绍

功能	说明
账号功能	神眸账号体系功能
摄像头列表	得到对应账号下设备
直播预览	直播预览，可设置直播分辨率
查看回放（SD卡、硬盘录像机、云存储）	回放
设备对讲	对讲（全双工对讲）
设备的设置功能	设备设置接口api
设备控制接口（云台、镜头画面）	云台控制
WiFi配置	设备wifi配置

直播、回放边播边录	播放过程中录像
直播、回放边播边截屏	播放过程中截屏
告警消息	告警消息获取

9. 隐私声明

9.1 收集个人信息说明

功能模块	收集个人信息类型	使用目的
设备配网	物联网硬件设备信息：设备序列号、设备验证码	为最终用户提供物联网硬件设备的配网功能
	客户端终端设备信息：客户端类型、客户端版本号、设备型号、设备硬件特征码、操作系统版本号	
	网络信息：WiFi账号、WiFi密码	
设备对讲	物联网硬件设备信息：设备序列号、设备验证码	为最终用户提供物联网硬件设备的语音对讲功能
	麦克风采集声音	
设备预览、回放	物联网硬件设备信息：设备序列号、设备验证码	为最终用户提供物联网硬件设备的视频预览、回放功能
	客户端终端设备信息：客户端类型、客户端版本号、设备型号、设备硬件特征码、操作系统版本号	
	网络信息：当前网络状态、网络连接方式	

请注意：基于不同的设备、系统及系统版本，以及开发者在集成、使用我们产品与/或服务时所决定的权限，我们实际接收到的信息可能会有所不同

9.2 权限说明

功能模块	权限名称	使用目的
设备配网	Camera 相机	用于扫描设备二维码以添加设备
	定位权限	用于搜索附近的WiFi信息以完成设备配网
设备对讲	麦克风录制	用于设备语音对讲功能，采集音频

SDK版本更新说明

SDK版本号	更新时间	更新说明
1.0.0	2024-12-12	新增开发YJOpenSDK，及对应Demo