YJOpenSDK-IOS 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
    - 3.1 Cocoapods自动集成（推荐）
    - 3.2 手动集成
4. 基础功能
    - 4.1 初始化
    - 4.2 账户及登录
        - 4.2.1 服务接口
        - 4.2.2 注册
        - 4.2.3 登录
        - 4.2.4 服务监听
5. API通道
    - 5.1 调用函数
        - 5.1.1 成功回调
        - 5.1.2 失败回调
    - 5.2 调用示例
6. 配网
    - 6.1 配网前检测：（所有类型配网前均可使用）
    - 6.2 不同方式配网绑定
        - 6.2.1 AP热点配网绑定
            - 6.2.1.1 AP热点绑定步骤
                - 6.2.1.1.1 AP热点配网配置
                - 6.2.1.1.2 发起AP热点配网绑定
                - 6.2.1.1.3 终止AP热点配网绑定
                - 6.2.1.1.4 其他接口补充
                    - 6.2.1.1.4.1 获取设备热点名称
                    -

6.2.1.1.4.2 清理缓存
- 6.2.1.2 代码示例
- 6.2.2 蓝牙配网绑定
  - 6.2.2.1 发起/停止蓝牙扫描
  - 6.2.2.2 发起蓝牙配网绑定
  - 6.2.2.3 终止蓝牙配网绑定
- 6.2.3 4G设备配网
  - 6.2.3.1 发起绑定
  - 6.2.3.2 终止配网
- 6.2.4 有线绑定
  - 6.2.4.1 发起绑定
  - 6.2.4.2 终止绑定

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. 流媒体RMPlaySDK

- 8.1 SDK集成
- 8.2 使用流程
  - 8.2.1 视频直播
    - 8.2.1.1 流程
    - 8.2.1.2 代码示例
      - 2.2 卡录像点播
  - 8.2.2 云录像点播
    - 8.2.2.1 流程
    - 8.2.2.2 代码示例
- 8.3 接口说明
  - 8.3.1 视频直播接口

- 8.3.2 卡录像点播接口
- 8.3.3 云录像点播接口
- 8.3.4 播放器代理接口

9. SDK发布说明

- 9.1 名词解释
- 9.2 功能介绍
- 9.3 隐私声明
  - 9.3.1 收集个人信息说明
  - 9.3.2 权限说明

10. SDK版本更新说明

 ↑ 请至钉钉文档上传文件 本文档用于说明神眸开放SDK iOS版本接口之间的关系以及接口调用顺序，对开放SDK iOS版本主要流程都有详细说明和代码示例。主要有功能介绍、安装说明、权限配置和主要流程介绍。

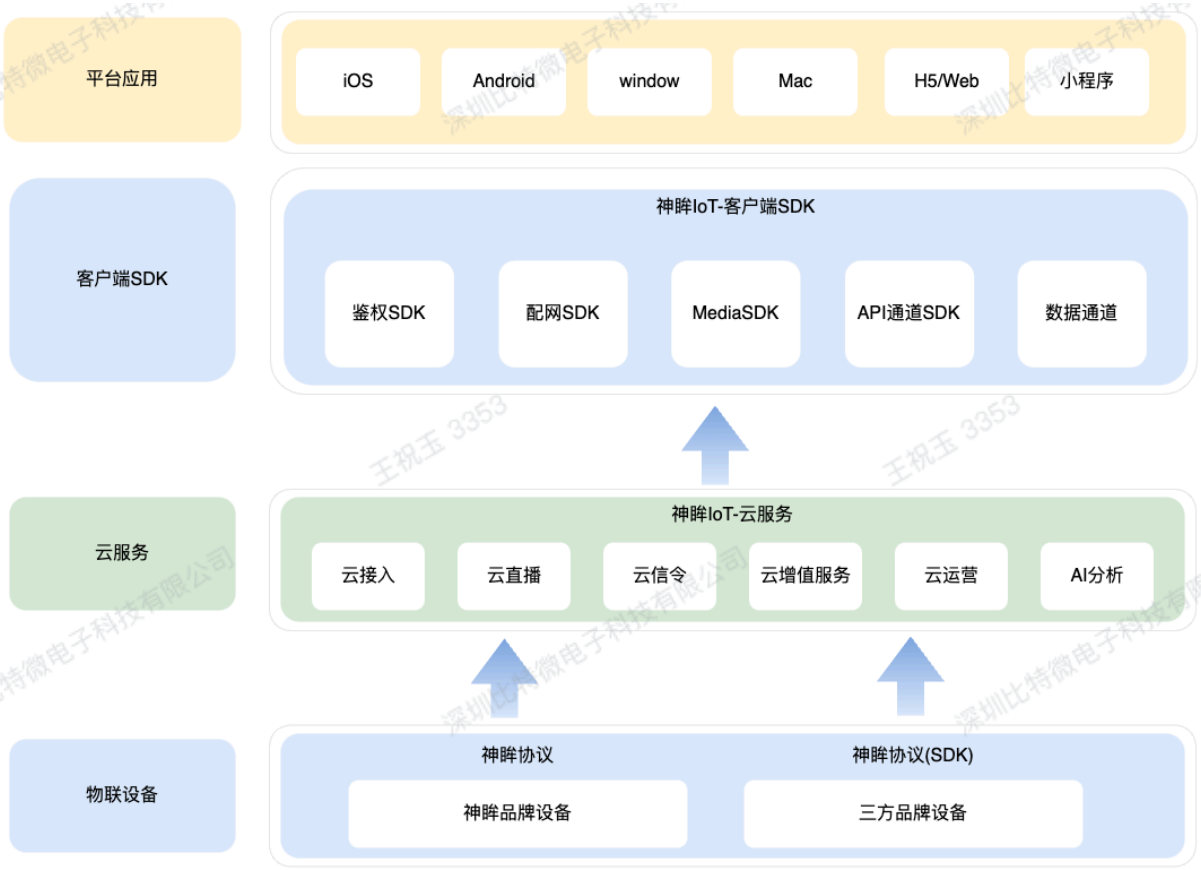
1. 获取App SDK

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。

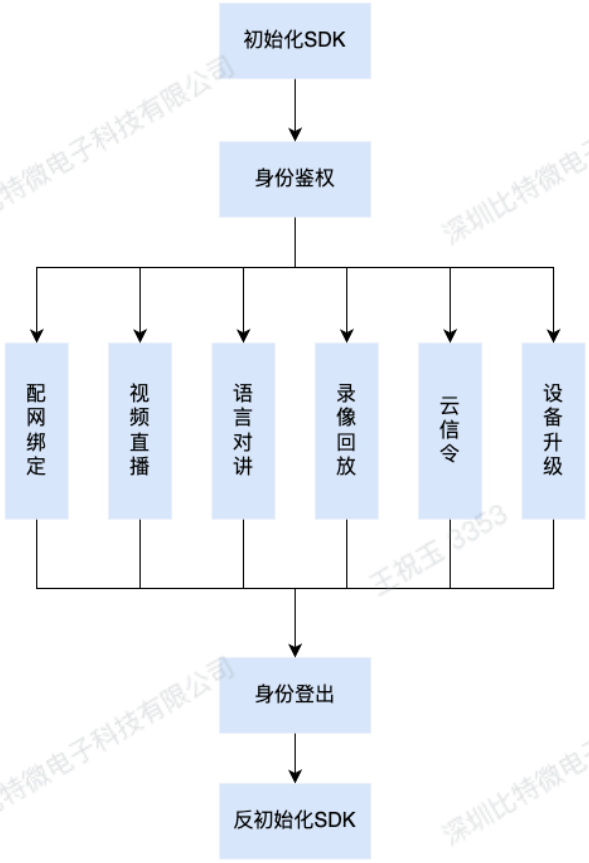
2. YJOpenSDK架构及调用流程

2.1 整体架构



> 该篇文档主要描述iOS端OpenSDK接入指南

2.2 基本调用流程



3. 集成流程

SDK为开发者提供了两种引入SDK方式：自动集成和手动集成。自动集成是指iOS的cocospod配置，手动集成是指在下载SDK后导入App工程。

iOS YJOpenSDK接入工程样例参见(接入Demo)[YJOpenSDKDemoAPP]。

3.1 Cocoapods自动集成（推荐）

1. 在Podfile中添加source，指定Master仓库和神眸云仓库。

```
source 'https://gitee.com/acmeapp/acme-spces.git'
source 'https://github.com/CocoaPods/Specs.git'
```

2. 在终端执行 `pod repo add` 命令，拉取神眸云Pod仓库到本地 。

```
pod repo add AcmeRepo https://gitee.com/acmeapp/acme-spces.git
```

或手动拉取Pod仓库工程到CocoaPods仓库（默认为 `~/.cocoapods/repos` ）。

```
git clone https://gitee.com/acmeapp/acme-spces.git ~/.cocoapods/repos/
```

>>> 注意：

- 可执行 `pod repo list` 查看本地 Pod 仓库信息。
- pod repo list或git clone失败一般为GitHub账号publicKey配置问题，请仔细核对错误信息，Pod命令使用参考 官方文档 。

3. 在您的工程中添加以下系统依赖库。

```
pod 'YJOpenKit'
```

4. 引入头文件

```
#import <YJOpenSDK/YJOpenSDK.h>
```

3.2 手动集成

1. 解压下载好的SDK包，在Xcode中，把SDK包目录中的framework拖入对应Target下即可，在弹出框勾选Copy items if needed。

```
YJOpenSDK.framework
YJOpenSDKUtils.framework
InnerHandyJson.framework
InnerAlamofire.framework
InnerMoya.framework
```

存在额外两个依赖库，同步拖入到对应Target下。

// 依赖的库

```
* SPPermissions/LocationWhenInUse ~> 7.1.5 用于权限申请
pod 'SPPermissions/LocationWhenInUse', '7.1.5'

*
```

4. 基础功能

4.1 初始化

通过调用如下接口完成初始化

```
/// YJVerifyInfo 结构体用于存储验证信息
public struct YJVerifyInfo {
    /// 应用的唯一键，开发者在神眸平台申请的appkey
    public let appKey: String
    /// 应用的秘密密钥，开发者在神眸平台申请的appSecret
    public let appSecret: String
}

/** 初始化YJOpenSDK
 @param verifyInfo 开发者传入的配置对象.
 */
public func setup(
```

```
    _ verifyInfo: YJVerifyInfo,
    completion: (Result<Void, YJError>) -> Void
```

示例代码

```
let acme_appKey = "您的应用AppKey"
let acme_appSecret = "您的应用AppSecret"
let config = YJVerifyInfo(appKey: acme_appKey, appSecret:
acme_appSecret)
YJOpenSDKManager.default.setup(config) { result in
    switch result {
        case .success: do{
            debugPrint("初始化成功")
        }
        case .failure(let error): do{
            debugPrint("初始化失败error: \(error)")
        }
    }
}
```

4.2 账户及登录

YJOpenSDK提供一整套鉴权及账号服务，主要内置账号方式集成；

包含用户注册、登录、登出、获取账号、会话管理、人机校验等功能；

4.2.1 服务接口

```
// 定义一个公开协议，用于管理租户账号的开放服务
public protocol YJOpenAccountService {

    // 当前登录的用户信息
    var currentUser: YJOpenAccountUser? { get }

    // 初始化设置方法，用于配置服务
    func setup()

    // 注销方法，用于退出当前用户登录状态
    func logout()

    // 注册任务方法，根据用户名创建注册任务
    // - Parameter userName: 用户名
```

```
    // - Returns: 返回注册任务的结果，包含任务对象或错误信息
    func registerTask(with userName: String) ->
Result<YJOpenAccountRegisterTask, YJOpenAccountError>

    // 密码登录任务方法，根据用户名创建密码登录任务
    // - Parameter userName: 用户名
    // - Returns: 返回密码登录任务的结果，包含任务对象或错误信息
    func passwordLoginTask(with userName: String) ->
Result<YJOpenAccountPasswordLoginTask, YJOpenAccountError>

    // 验证码登录任务方法，根据用户名创建验证码登录任务
    // - Parameter userName: 用户名
    // - Returns: 返回验证码登录任务的结果，包含任务对象或错误信息
    func authcodeLoginTask(with userName: String) ->
Result<YJOpenAccountAuthcodeLoginTask, YJOpenAccountError>

    /// 使用三方验证码登录。
    /// - Returns: 验证码的结果，包含成功或错误信息。
    func oauthLogin(authCode: String, completion: @escaping
(YJOpenAccountError?) -> Void)

    /// 使用三方平台信息进行登录
    func thirdPartyLogin(
        type: YJOpenAccountThirdPartyLoginType,
        completion: @escaping (YJOpenAccountError?) -> Void
    )

    // 忘记密码任务方法，根据用户名创建忘记密码任务
    // - Parameter userName: 用户名
    // - Returns: 返回忘记密码任务的结果，包含任务对象或错误信息
    func forgetPasswordTask(with userName: String) ->
Result<YJOpenAccountForgetPasswordTask, YJOpenAccountError>

    // 重置密码，需要登录态
    // params:
    //   oldPassword 旧密码
    //   newPassword 新密码
    func resetPassword(
        oldPassword: String, newPassword: String,
        completion: @escaping (YJOpenAccountError?) -> Void
    )

    // 重置密码任务方法，根据用户名创建验证码登录任务
    // - Returns: 返回重置密码任务的结果，包含任务对象或错误信息
```

```
    func forgotPasswordTask() ->
Result<YJOpenAccountForgotPasswordTask, YJOpenAccountError>
```

4.2.2 注册

App用户注册通过以下接口

复制代码

```
/// YJOpenAccountRegisterTask协议定义了开放账号注册任务的标准接口。
/// 它继承自YJOpenAccountTask协议，用于处理用户注册过程中的一系列操作。
public protocol YJOpenAccountRegisterTask: YJOpenAccountTask {

    /// 发送验证码。
    /// - Returns: 发送验证码的结果，包含成功或错误信息。
    func sendAuthcode(userName: String, completion: @escaping
(YJOpenAccountError?) -> Void)

    /// 重发验证码
    func resendAuthcode(completion: @escaping (YJOpenAccountError?) ->
Void)

    /// 验证验证码。
    /// - Parameter authcode: 验证码。
    /// - Returns: 验证结果，包含成功或错误信息。
    func verifyAuthcode(authcode: String, completion: @escaping
(YJOpenAccountError?) -> Void)

    /// 设置用户密码。
    /// - Parameter password: 密码。
    /// - Returns: 设置密码的结果，包含成功或错误信息。
    func signupWithPassword(_ password: String, completion: @escaping
(YJOpenAccountError?) -> Void)
}
```

4.2.3 登录

复制代码

```
/// YJOpenAccountPasswordLoginTask协议定义了使用密码登录开放账户所需的任务结
构
/// 它继承自YJOpenAccountTask，确保了基本的任务操作得以实现
public protocol YJOpenAccountPasswordLoginTask: YJOpenAccountTask {

    /// 启动登录流程
    ///
```

```
    /// - Parameter userName: 用户名
    /// - Returns: 返回登录任务的结果，包含成功时的任务对象或失败时的错误信息
    static func startLogin(with userName: String) ->
Result<YJOpenAccountPasswordLoginTask, YJOpenAccountError>

    /// 设置用户密码
    ///
    /// - Parameter password: 用户密码
    /// - Returns: 返回设置密码的结果，成功时不返回任何内容，失败时返回错误信
息
    func setPassword(_ password: String) async -> Result<Void,
YJOpenAccountError>
}

/// YJOpenAccountAuthcodeLoginTask协议定义了使用验证码进行登录的任务所需的方
法和属性
public protocol YJOpenAccountAuthcodeLoginTask: YJOpenAccountTask {

    /// 启动登录流程
    ///
    /// - Parameter userName: 用户名
    /// - Returns: 登录任务的结果，包含任务对象或错误信息
    static func startLogin(with userName: String) ->
Result<YJOpenAccountAuthcodeLoginTask, YJOpenAccountError>

    /// 发送验证码
    ///
    /// - Returns: 发送结果，包含成功或错误信息
    func sendAuthcode() async -> Result<Void, YJOpenAccountError>

    /// 验证验证码
    ///
    /// - Parameter authcode: 验证码
    /// - Returns: 验证结果，包含成功或错误信息
    func verifyAuthcode(authcode: String) async -> Result<Void,
YJOpenAccountError>
}

/// YJOpenAccountForgotPasswordTask协议定义了使用验证码进行密码重置任务所需
的方法和属性
public protocol YJOpenAccountForgotPasswordTask: YJOpenAccountTask {

    /// 当前登录任务的状态
    var state: YJOpenAccountForgotPasswordState { get }
```

```
    var userName: String? { get }
    var authCode: String? { get }
    var password: String? { get }

    /// 启动流程
    ///
    /// - Parameter userName: 用户名
    /// - Returns: 登录任务的结果，包含任务对象或错误信息
    static func startResetPassword() ->
Result<YJOpenAccountForgotPasswordTask, YJOpenAccountError>

    /// 发送验证码
    ///
    /// - Returns: 发送结果，包含成功或错误信息
    func sendAuthcode(userName: String, completion: @escaping
(YJOpenAccountError?) -> Void)

    /// 验证验证码
    ///
    /// - Parameter authcode: 验证码
    /// - Returns: 验证结果，包含成功或错误信息
    func verifiedAuthcode(authcode: String, completion: @escaping
(YJOpenAccountError?) -> Void)

    /// 设置新密码
    ///
    /// - Parameter password: 新密码
    /// - Returns: 设置结果，包含成功或错误信息
    func resetPassword(password: String, completion: @escaping
(YJOpenAccountError?) -> Void)
}
```

4.2.4 服务监听

账户登录登出情况通过监听广播

```
// 扩展Notification.Name，添加与账户登录状态相关的通知名称
extension Notification.Name {
    // 登录通知名称
    static let YJOpenAccountDidLogin =
Notification.Name("YJOpenAccountDidLogin")
    // 登出通知名称
    static let YJOpenAccountDidLogout =
Notification.Name("YJOpenAccountDidLogout")
```

```
    // 登录授权过期
    static let YJOpenAccountAuthorizationExpired =
Notification.Name("YJOpenAccountAuthorizationExpired")
```

5. API通道

API通道SDK，提供IoT业务协议封装的HTTPS请求能力，并通过整合安全组件来提升通道的安全性。

> 通过API通道可以访问神眸云服务接口

5.1 调用函数

YJApiClient 复制代码

```
/// apiPath 接口路径
/// param  参数键值对
/// timeOut 接口超时时间(可选) 默认 15 秒超时
/// success 成功数据回调
/// fail 失败数据回调
public static func request(apiPath:String,param:
[String:Any],timeOut:Int = 15,success:@escaping
YJApiClientSuccess,fail:@escaping YJApiClientFailure)
```

5.1.1 成功回调

这里成功指的链路成功，并非逻辑上成功

各回调数据如下

- code 状态码 0 表示成功
- msg 接口影响回执信息，当code不为0时,可以根据 msg 来判断原因
- data 具体响应的业务数据，具体格式参照平台 api 接口

复制代码

```
public typealias YJApiClientSuccess = (_ code:Int, _ msg:String ,_
data: Any?) -> Void
```

5.1.2 失败回调

接口调用失败回调 error，一般网络网关层失败会进入回调

```
public typealias YJApiClientFailure = (_ error: NSError) -> Void
```

5.2 调用示例

以 /api/v1/event/query 获取设备告警列表为例

```
let apiPaht = "/api/v1/event/query"
var param:[String:Any] = [:]
let date = Date()
param["beginTime"] = date.timeIntervalSince1970
param["endTime"] = date.timeIntervalSince1970 + 86399
param["alarmType"] = 0
param["deviceIds"] = ["579831504939733990"] //设备Id数组
param["pageSize"] = 20                      //每页请求的数据

YJApiClient.request(apiPath: apiPaht, param: param) { code, msg, data in
    if let dataDic = data as? [String:Any]  //文档中data是Object类型 所以解析成字典
    {
        //TODO  解析数据
    }
    else
    {
        print("接口请求异常")
        print("code:\(code)")
        print("msg:\(msg)")
    }

} fail: { error in
    print("请求失败\(error)")
}
```

6. 配网

6.1 配网前检测：（所有类型配网前均可使用）

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；

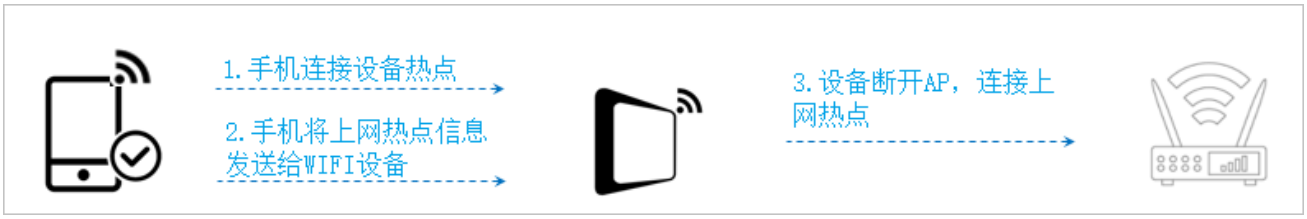
接口名：

```
/// YJBindDevice
open func getPrepareDeviceInfo(deviceName: String, comp: @escaping ( (_ info: YJPrepareDeviceInfo?, _ error: NSError?) -> Void ))
```

注意：如果设备已被其他人绑定，则不能再被重复绑定，可先通过该接口检查。

6.2 不同方式配网绑定

6.2.1 AP热点配网绑定



6.2.1.1 AP热点绑定步骤

6.2.1.1.1 AP热点配网配置

开始AP热点配网前，进行设备信息配置；

接口：

```
///YJBindDeviceBySoftAP
public static func configBindDevice(productKey: String, deviceName: String)
```

6.2.1.1.2 发起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. 最终陪我结果回调：设备添加绑定最终结果回调；

```
/// YJBindDeviceBySoftAP
    /// 开始配网绑定
    /// - Parameters:
    ///   - wifiSSID: 配网WiFi名称
    ///   - wifiPW: 配网WiFi密码
    ///   - bindStep: 配网过程已完成阶段
    ///   - result: 配网结果
    ///         - failStep: 配网失败阶段
    ///         - result: 配网成功返回设备数据，{"deviceId": deviceId}
    public static func bindDevice(wifiSSID: String,
                                  wifiPW: String,
                                  bindStep: ( (_ step: YJBindDeviceStep) ->
Void )?,
                                  result: ( (_ failStep: YJBindDeviceStep, _
result: [String : String]?, _ error: NSError?) -> Void )? )
```

绑定各节点：

复制代码

```
/// 绑定步骤
public enum YJBindDeviceStep: Int {
    case none = 0
    case readyBind = 1
    case connectedDeviceHot = 2
    case sendWifiInfo
    case deviceOnline
    case deviceBind
}
```

6.2.1.1.3 终止AP热点配网绑定

在发起配网（步骤3发起AP热点绑定）后，可以进行终止配网。

复制代码

```
/// YJBindDeviceBySoftAP
public static func stopBindDevice()
```

注意：如果设备已经在绑定中了，是无法真正终止配网，设备仍能完成绑定流转；

6.2.1.1.4 其他接口补充

6.2.1.1.4.1 获取设备热点名称

在完成步骤2（AP热点配网配置）后，可获取设备热点名称；如果步骤2没完成，会返回nil

```
/// YJBindDeviceBySoftAP
public static func deviceHotspotName() -> String?
```

6.2.1.1.4.2 清理缓存

配网结束后清理下缓存，快速释放

```
/// YJBindDeviceBySoftAP
public static func clearup()
```

6.2.1.2 代码示例

发起绑定：

```
YJBindDeviceBySoftAP.configBindDevice(productKey: productKey,
deviceName: deviceName)
YJBindDeviceBySoftAP.bindDevice(wifiSSID: wifi, wifiPW: pwd) { step in
    print("绑定过程回调：\(step)")
} result: { [weak self] failStep, result, error in
    self?.loading.send(false)
    if let error {
        print("绑定失败：\(error)")
        return
    }
    guard let result, let did = result["deviceId"] else {
        print("绑定失败：获取deviceId 为空")
        return
    }
    print("绑定成功")
}
```

停止绑定

```
/// YJBindDeviceBySoftAP
YJBindDeviceBySoftAP.stopBindDevice()
```

退出绑定流程

```
/// YJBindDeviceBySoftAP
YJBindDeviceBySoftAP.clearup()
```

6.2.2 蓝牙配网绑定

蓝牙绑定会先发起蓝牙扫描，扫描到设备后发起绑定

6.2.2.1 发起/停止蓝牙扫描

会通过蓝牙持续扫描Camer设备(需已重置成功)，申请蓝牙权限，并持续返回扫描到的蓝牙设备；

发起蓝牙扫描：

```
/// YJBindDeviceByBLE
/// 发起蓝牙扫描
  /// - Parameter result:会多次回调扫描到的设备
  public static func scanDevice(result: ( (_ scanDeviceInfo:
YJBLEScanDeviceInfo?, _ error: NSError?) -> Void )? ) {
      YJBindDeviceByBLE.shared.bindDevVModel?.scanDevice(result: result)
  }
```

停止蓝牙扫描

```
/// YJBindDeviceByBLE
/// 停止蓝牙扫描
public static func stopScan() {
        YJBindDeviceByBLE.shared.bindDevVModel?.stopSearch()
}
```

6.2.2.2 发起蓝牙配网绑定

复制代码

```
/// 发起蓝牙配网绑定
/// - Parameters:
///   - scanDeviceInfo: 扫码到的设备信息
///   - bindStep: 配网过程已完成阶段
///   - result:
public static func bindDevice(scanDeviceInfo: YJBLEScanDeviceInfo,
                              wifiSSID: String,
                              wifiPW: String,
                              bindStep: ( (_ step:
YJBindDeviceStepByBLE) -> Void )?,
                              result: ( (_ failStep:
YJBindDeviceStepByBLE, _ result: [String : String]?, _ error: NSError?)
-> Void )? )
```

发起配网，需传入配网的WiFi相关信息，正式发起配网流程；

该流程会有两个回调：

1. 配网过程回调：在配网流程中将配网进度节点回调；
2. 最终陪我结果回调：设备添加绑定最终结果回调；

> 如果已被别人绑定，

6.2.2.3 终止蓝牙配网绑定

复制代码

```
/// YJBindDeviceByBLE
/// 终止配网
public static func stopBind()
```

6.2.3 4G设备配网

针对支持4G的Camer设备可以通过4G蜂窝网络绑定，完成设备绑定

6.2.3.1 发起绑定

复制代码

```
// YJBindDeviceByCellular
    /// 4G蜂窝网络绑定
   /// - Parameters:
   ///   - productKey: productKey
```

```
    ///    - deviceName: deviceName
    ///    - result: {"deviceStatus":"1",
"bindStatus":"2","deviceId":"xxxxx","bindUser":"xxxxx"}
    ///       - deviceStatus: 设备状态。-1 (查询异常)  0（表示未激活）；1（表示在线）；3（表示离线）；8（表示禁用）
    ///       - bindStatus:  绑定结果:0是初始化状态;1是在线状态;2是已绑定状态;3是网关设备与子设备建立连接失败;4是子设备认证失败;5是子设备认证成功;6是被别人绑定;7绑定通知失败;8绑定关系建立失败
    ///       - deviceId:   bindStatus==2时已绑定状态，存在设备id，否则为空
    ///       - bindUser:   bindStatus==6 时，返回绑定用户的用户名（脱敏后），否则为空
    ///     error:  异常报错
    public static func bindDevice(productKey: String, deviceName:
String,
                                  result: ( (_ result: [String : String]?, _
error: NSError?) -> Void )? )
```

6.2.3.2 终止配网

复制代码

```
/// YJBindDeviceByCellular
public static func stopBindDevice()
```

6.2.4 有线绑定

针对支持插有线的Camer设备可以通过4G蜂窝网络绑定，完成设备绑定

6.2.4.1 发起绑定

需传入设备deviceName、productKey、等配网初始化参数；

复制代码

```
/// YJBindDeviceByWired
    /// 有线网络绑定
    /// - Parameters:
    ///    - productKey: productKey
    ///    - deviceName: deviceName
    ///    - result: {"deviceStatus":"1",
"bindStatus":"2","deviceId":"xxxxx","bindUser":"xxxxx"}
    ///       - deviceStatus: 设备状态。-1 (查询异常)  0（表示未激活）；1（表示在线）；3（表示离线）；8（表示禁用）
    ///       - bindStatus:  绑定结果:0是初始化状态;1是在线状态;2是已绑定状态;3是网关设备与子设备建立连接失败;4是子设备认证失败;5是子设备认证成功;6是被别人绑定;7绑定通知失败;8绑定关系建立失败
```

```
    ///      - deviceId:   bindStatus==2时已绑定状态，存在设备id，否则为空
    ///      - bindUser:   bindStatus==6 时，返回绑定用户的用户名（脱敏后），否则为空
    ///      error:  异常报错
    public static func bindDevice(productKey: String, deviceName: String,
                                  result: ( (_ result: [String : String]?, _ error: NSError?) -> Void )? )
```

6.2.4.2 终止绑定

复制代码

```
/// YJBindDeviceByCellular
public static func stopBindDevice()
```

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：属性操作用thing.service.property.set，服务使用thing.service.${tsl.service.identifier}	
params	object	必须		jsonSchema支持的数据类型	备注: jsonSchema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Capabilitys	固定值Capabilitys

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-ss HH: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-ssHH: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. 流媒体RMPlaySDK

8.1 SDK集成

RMPlayer 随 OpenSDK一起发布，集成见OpenSDK集成流程

8.2 使用流程

播放器创建前，需要调用RMPEngine的接口初始化播放器引擎，该初始化接口调用时机是在设置环境参数后 播放器创建前调用一次即可

复制代码

```
var isInited:Bool = YJRMPEngine.getDefault().isInited()
if !isInited
  YJRMPEngine.getDefault().initialized()
```

8.2.1 视频直播

8.2.1.1 流程

8.2.1.2 代码示例

复制代码

```
//定义直播播放器对象
private var player: YJRMPNetLivePlayer?
//父窗口
private var videoBaseView:      UIView? = nil
//视频显示窗口
private var playerVideoView:    YJRMPVideoView? = nil
//设备名称
private var deviceName: String = ""
//产品秘钥
private var productKey: String = ""

//创建播放器配置对象并以设备名称和产品秘钥初始化
let config = YJRMPNetPlayerConfig()
config.deviceName = self.deviceName
config.productKey = self.productKey
config.engine = YJRMPEngine.getDefault()

//根据配置创建直播播放器对象
```

```
self.player = YJRMPNetLivePlayer.create(with: config)
self.player?.delegate = self//设置代理对象 用于获取流相关状态和错误码
self.playerVideoView = YJRMPVideoView(frame: CGRectZero)
self.player?.setVideoView(self.playerVideoView)
//添加视频显示窗口
self.videoBaseView?.addSubview(self.playerVideoView!)
//开始播放
self.player?.start()
```

8.2.2 卡录像点播

8.2.2.1 流程

8.2.2.2 代码示例

复制代码

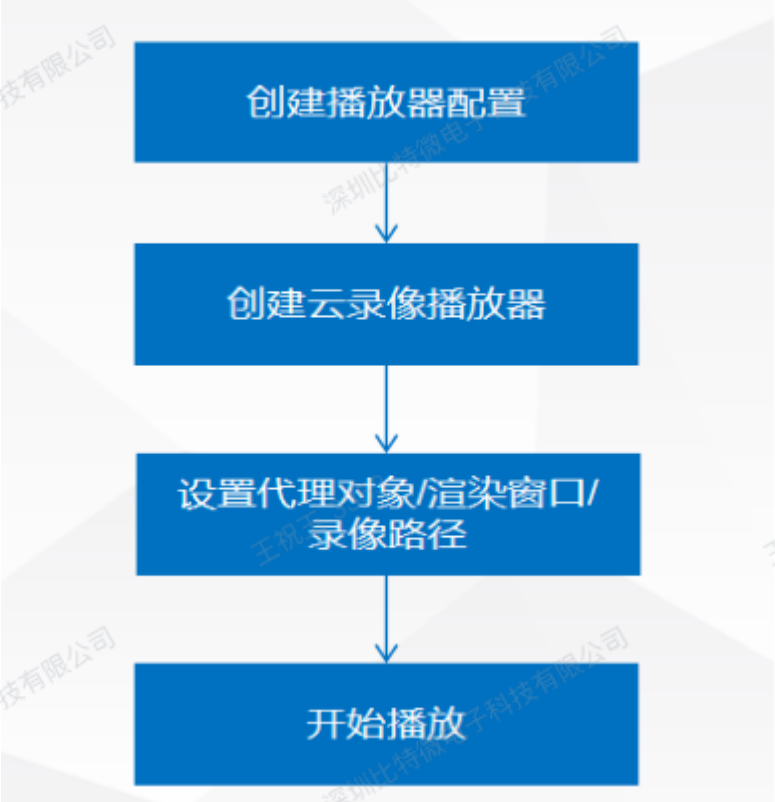
```
//定义卡录像点播播放器对象
private var player: YJRMPNetVodPlayer?
//父窗口
private var videoBaseView:      UIView = UIView()
//视频显示窗口
private var playerVideoView:    YJRMPVideoView? = nil
//设备名称
private var deviceName: String = ""
//产品秘钥
private var productKey: String = ""
private var startTime:  Int    = 0 //开始时间 单位秒
private var seekTime:   Int    = 0 //定位时间 单位秒
private var endTime:    Int    = 0 //结束时间 单位秒
```

```
//创建播放器配置对象并以设备名称和产品秘钥初始化
let config = YJRMPNetPlayerConfig()
config.deviceName = self.deviceName
config.productKey = self.productKey
config.engine = YJRMPEngine.getDefault()

//根据配置创建卡录像点播播放器对象
self.player = YJRMPNetVodPlayer.create(with: config)
self.player?.delegate = self //设置代理对象 用于获取流相关状态和错误码
self.playerVideoView = YJRMPVideoView(frame: CGRectZero)
self.player?.setVideoView(self.playerVideoView)
//添加视频显示窗口
self.videoBaseView.addSubview(self.playerVideoView!)
//设置卡录像开始时间、定位时间、结束时间
self.player?.setDeviceSource(startSec: self.startTime, endSec:
self.endTime, seekSec: self.seekTime)
//开始播放
self.player?.start()
```

8.2.3 云录像点播

8.2.3.1 流程



8.2.3.2 代码示例

```
//定义云录像点播播放器对象
private var player: YJRMPNetCloudVodPlayer?
//父窗口
private var videoBaseView:      UIView = UIView()
//视频显示窗口
private var playerVideoView:    YJRMPVideoView? = nil
//设备名称
private var deviceName: String = ""
//产品秘钥
private var productKey: String = ""
private var url:        String = "" //播放链接
private var meta:       String = "" //json元数据

//创建播放器配置对象并以设备名称和产品秘钥初始化
let config = YJRMPNetPlayerConfig()
config.deviceName = self.deviceName
config.productKey = self.productKey
config.engine = YJRMPEngine.getDefault()

//根据配置创建卡录像点播播放器对象
self.player = YJRMPNetCloudVodPlayer.create(with: config)
self.player?.delegate = self //设置代理对象 用于获取流相关状态和错误码
self.playerVideoView = YJRMPVideoView(frame: CGRectZero)
self.player?.setVideoView(self.playerVideoView as? YJRMPVideoView)

//添加视频显示窗口
[self.videoBaseView addSubview:self.playerVideoView];

//设置云录像播放URL及元数据(由其他接口获取URL及元数据)
[self.player setCloudSource:self.url meta:self.meta
mode:RMPNetCloudVodPlayMode_All];
//开始播放
[self.player start];
```

8.3 接口说明

8.3.1 视频直播接口

```
open class YJRMPNetLivePlayer {
```

```
    /// 播放器回调代理
    weak open var delegate: (any YJOpenSDK.YJRMPlayerDelegate)?

    /// 创建播放器实例，非单例
    open class func create(with config: YJOpenSDK.YJRMPNetPlayerConfig)
-> Self?

    /// 设置渲染远端摄像头画面的窗口
    open func setVideoView(_ view: YJOpenSDK.YJRMPVideoView?)

    /// 设置渲染本地摄像头画面的窗口
    open func setLocalVideoView(_ view: YJOpenSDK.YJRMPVideoView?)

    /// 通话模式需要配置，直播模式可省略
    open func configLivePlay(_ config: YJOpenSDK.YJLivePlayConfig) ->
Bool

    /// 开启本地视频的采集预览画面
    /// 使用该接口前需通过`configLivePlay` 设置 `config.audioSend=YES`,
`config.videoSend=YES`
    /// - Parameters:
    ///   - position: 摄像头方向类型，前置、后置。
    open func startLocalPreview(_ position:
YJOpenSDK.YJRMPCameraPosition) -> Bool

    /// 停止本地视频采集及预览
    open func stopLocalPreview()

    /// 切换摄像头
    open func switchCamera(_ position: YJOpenSDK.YJRMPCameraPosition) -
> Bool

    /// 音频流发送开关
    /// - Parameters:
    ///   - mute: true 暂停， false 恢复
    open func muteLocalAudio(_ mute: Bool) -> Bool

    /// 视频流发送开关
    /// - Parameters:
    ///   - mute: true 暂停， false 恢复
    open func muteLocalVideo(_ mute: Bool) -> Bool

    /// 设置接收视频帧YUV数据的接收器
    open func setVideoSink(_ sink: (any
YJOpenSDK.YJRMPVideoSinkDelegate)?)
```

```
    /// 远端音频播放的开关
    /// - Parameters:
    ///   - mute: true 静音开启，false 静音关闭
    open func muteRemoteAudio(_ mute: Bool) -> Bool

    /// 开始播放
    open func start() -> Bool

    /// 停止播放
    open func stop()

    /// 获取收发码率、帧率统计
    open func getStats() -> YJOpenSDK.YJRMPlayerStats?

    /// 截图
    /// - Parameters:
    ///   - path: 截图保存的文件，示例：/save/snapshot.jpg
    open func snapshot(_ path: String) -> Bool

    /// 开始录制本地视频流
    /// - Parameters:
    ///   - path: 录制保存的文件，示例：/save/record.mp4
    open func startFileRecording(_ path: String) -> Bool

    /// 结束录制本地视频流
    open func stopFileRecording() -> Bool

    /// 设置获取SEI数据的回调
    open func setSeiDataCallback(_ callback: (any
YJOpenSDK.YJRMPSeiDelegate)?)

    /// 获取录制时长，单位 ms，无录制操作返回-1
    open func getFileRecordingDuration() -> Int

    /// 开启对讲
    open func startTalk() -> Bool

    /// 结束对讲
    open func stopTalk()

    /// 对讲状态
    open func isTalking() -> Bool
```

8.3.2 卡录像点播接口

复制代码

```
open class YJRMPNetVodPlayer {

    /// 播放器回调代理
    weak open var delegate: (any YJOpenSDK.YJRMPlayerDelegate)?

    /// 创建播放器实例，非单例
    open class func create(with config: YJOpenSDK.YJRMPNetPlayerConfig)
-> Self?

    /// 设置渲染远端摄像头画面的窗口
    open func setVideoView(_ view: YJOpenSDK.YJRMPVideoView?)

    /// 设置播放起始位置、结束位置、偏移位置。
    /// 例如：setDeviceSource(1709026797, 1709053456, 60)，则播放的范围时
间是 [1709026797 + 60,  1709053456]
    /// - Parameters:
    ///   - startSec: 播放起始位置，Unix时间戳，单位秒
    ///   - endSec: 播放结束位置，Unix时间戳，单位秒
    ///   - seekSec: 相对开始时间的偏移量，相对时间，单位秒
    open func setDeviceSource(startSec: Int, endSec: Int, seekSec: Int)
-> Bool

    /// 远端音频播放的开关
    /// - Parameters:
    ///   - mute: true静音开启，false静音关闭
    open func muteRemoteAudio(_ mute: Bool) -> Bool

    /// 开始播放
    open func start() -> Bool

    /// 停止播放
    open func stop()

    /// 获取收发码率、帧率统计
    open func getStats() -> YJOpenSDK.YJRMPlayerStats?

    /// 截图
    /// - Parameters:
    ///   - path: 截图保存的文件，示例：/save/snapshot.jpg
    open func snapshot(_ path: String) -> Bool

    /// 开始录制本地视频流
    /// - Parameters:
    ///   - path: 录制保存的文件，示例：/save/record.mp4
```

```
    open func startFileRecording(_ path: String) -> Bool

    /// 结束录制本地视频流
    open func stopFileRecording() -> Bool

    /// 设置获取SEI数据的回调
    open func setSeiDataCallback(_ callback: (any
YJOpenSDK.YJRMPSeiDelegate)?)

    /// 获取录制时长，单位 ms，无录制操作返回-1
    open func getFileRecordingDuration() -> Int

    /// 暂停播放
    open func pause()

    /// 恢复播放
    open func resume()

    /// 跳到到某一位置后会自动播放
    /// - Parameters:
    ///   - offsetSec: 相对时间，相对 setDeviceSource 接口的 startSec 的时
间，单位秒。
    open func seek(_ offsetSec: Int)

    /// 倍速，支持 1x，4x，8x，16x
    open func setPlaybackSpeed(_ speed: Int32)
}
```

8.3.3 云录像点播接口

复制代码

```
open class YJRMPNetCloudVodPlayer {

    /// 播放器回调代理
    weak open var delegate: (any YJOpenSDK.YJRMPlayerDelegate)?

    /// 创建播放器实例，非单例
    open class func create(with config: YJOpenSDK.YJRMPNetPlayerConfig)
-> Self?

    /// 设置渲染远端摄像头画面的窗口
    open func setVideoView(_ view: YJOpenSDK.YJRMPVideoView?)
```

```
    /// 云存播放设置参数
    /// - Parameters:
    ///   - url:  视频链接
    ///   - meta: 视频描述信息，json格式
    ///   - mode: 播放模式 `all`播放包含前卷、普通视频，
    ///                   `normal`只播放普通视频，
    ///                   `preroll`只播放前卷
    open func setCloudSource(url: String?, meta: String?, mode:
YJOpenSDK.YJRMPNetCloudVodPlayMode) -> Bool

    /// 一直播接口，把 {url_1, meta_1}， {url_2, meta_2} ..... {url_n,
meta_n} 设置进去，
    /// 播放器就会无缝播放设置进去的链接视频，直到url_n播放结束
    /// - Parameters:
    ///   - url:  视频链接
    ///   - meta: 视频描述信息，json格式
    open func appendCouldPlaylist(url: String?, meta: String?) -> Bool

    /// 播放的总时长，单位 ms
    open func getTotalDuration() -> Int

    /// 静音开关
    /// - Parameters:
    ///   - mute: true静音开启，false静音关闭
    open func muteRemoteAudio(_ mute: Bool) -> Bool

    /// 开始播放
    open func start() -> Bool

    /// 停止播放
    open func stop()

    /// 暂停播放
    open func pause()

    /// 恢复播放
    open func resume()

    /// 跳到到某一位置后会自动播放
    /// - Parameters:
    ///   - offsetSec: 相对时间，单位秒。
    open func seek(_ offsetSec: Int)
}
```

8.3.4 播放器代理接口

复制代码

```
@objc public protocol YJRMPlayerDelegate {

    /// 播放器错误回调
    /// - Parameters:
    ///   - type: 错误类型
    ///   - code: 错误码
    ///   - description: 错误描述
    @objc optional func onError(player: Any?, type:
YJOpenSDK.YJRMPlayerErrorType, code: YJOpenSDK.YJRMPlayerErrorCode,
description: String?)

    /// 播放器状态回调
    /// - Parameters:
    ///   - state: 播放器的状态
    @objc optional func onPlayerStateChange(player: Any?, state:
YJOpenSDK.YJRMPlayerState)

    /// 对讲状态回调
    /// - Parameters:
    ///   - state: 对讲状态
    @objc optional func onTalkStateChange(player: Any?, state:
YJOpenSDK.YJRMPlayerTalkState)

    /// 倍速状态回调
    /// - Parameters:
    ///   - speed: 倍速改变后的值
    @objc optional func onPlaybackSpeedUpdate(player: Any?, speed: Int)

    /// seek回调
    /// - Parameters:
    ///   - success: 是否seek成功
    @objc optional func onSeekComplete(player: Any?, success: Bool)

    /// 缓冲状态改变回调。超过 3s 无帧播放，回调 Loading 状态，恢复回调
Ready 状态
    /// - Parameters:
    ///   - state: 缓冲状态
    ///   - bufferDurationMillis: 状态改变的时间点的buffer长度，单位毫秒。
返回 -1 表示buffer长度未知，因为可能未拿到数据。
    @objc optional func onBufferStateUpdate(player: Any?, state:
YJOpenSDK.YJRMPlayerBufferState, bufferDurationMillis: Int)

    /// 首帧回调
    /// - Parameters:
```

```
    ///   - elapseMills: `start()`调用到出现首帧的耗时，单位毫秒
    @objc optional func onFirstFrameRendered(player: Any?, elapseMills:
Int)

    /// 分辨率变化回调
    /// - Parameters:
    ///   - size: 分辨率改变后的值
    @objc optional func onVideoSizeChanged(player: Any?, size: CGSize)

    /// 开始本地录制的成功回调
    /// - Parameters:
    ///   - file: 录制的文件路径
    @objc optional func onFileRecordingStart(player: Any?, file:
String?)

    /// 录制过程中本地录制的错误回调
    /// - Parameters:
    ///   - file: 录制的文件路径
    ///   - code: 错误码
    ///   - description: 错误描述
    @objc optional func onFileRecordingError(player: Any?, file:
String?, code: YJOpenSDK.YJRMPlayerRecordingError, description: String?)

    /// 停止本地录制的成功回调
    /// - Parameters:
    ///   - file: 录制的文件路径
    @objc optional func onFileRecordingFinish(player: Any?, file:
String?)

    /// 截图回调
    /// - Parameters:
    ///   - file: 截图的文件路径
    ///   - code: 状态码
    ///   - description: 错误描述
    @objc optional func onSnapshotResult(player: Any?, file: String?,
code: YJOpenSDK.YJRMPlayerSnapshotResult, description: String?)

    /// 云存播放、卡录像播放进度回调，单位毫秒
    /// - Parameters:
    ///   - millis: 播放进度时间戳
    @objc optional func onVodPlayProgress(player: Any?, millis: Int)

    /// 云存播放、卡录像播放结束回调
    @objc optional func onVodPlayComplete(player: Any?)
}
```

9. SDK发布说明

9.1 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
countryCode	国家码，取值参考： https://zh.wikipedia.org/wiki/ISO_3166-1 countryCode 二位代码

9.2 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

9.3 隐私声明

9.3.1 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

9.3.2 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
	LocationAlwaysAndWhenInUse 使用期间始终访问位置	用于搜索附近的WiFi信息以完成设备配网
	LocationAlways 始终访问位置	用于搜索附近的WiFi信息以完成设备配网
	LocationWhenInUse 使用期间访问位置	用于搜索附近的WiFi信息以完成设备配网
设备对讲	Microphone 麦克风	用于设备语音对讲功能，采集音频
设备预览、回放	PhotoLibrary 读取照片库	用于保存视频的录像和截图到手机相册
	PhotoLibraryAdd 写照片库	用于保存视频的录像和截图到手机相册
	LocalNetwork 局域网设备搜索、预览	用于局域网设备搜索、预览

10. SDK版本更新说明

SDK版本号	更新时间	更新说明
0.5.0	2024-12-12	新增YJOpenSDK，及对应Demo

RMPlayer_v1.3.3_release_7f5c0963_20241212.zip

8.3MB