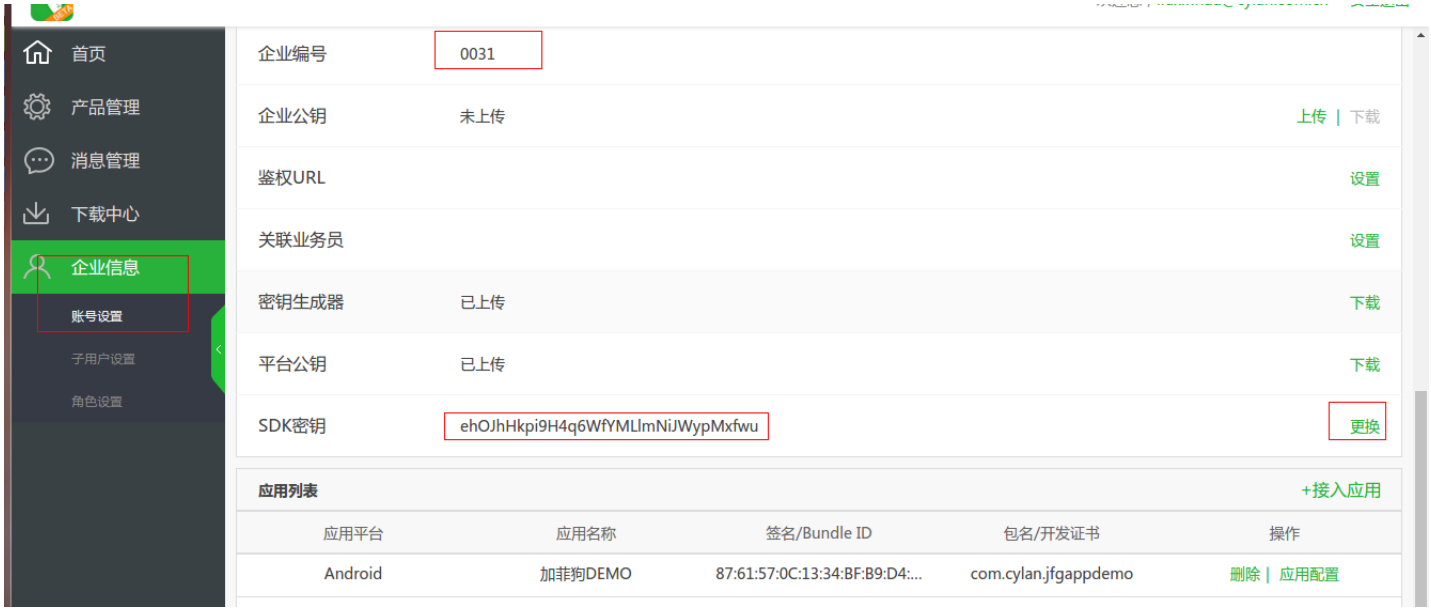
iOS APP开发文档

iOS APP开发-1准备工作

1.获取vkey，vid

第一步:首先需要在萝卜头运营平台注册一个开发者账号，老用户直接登录。

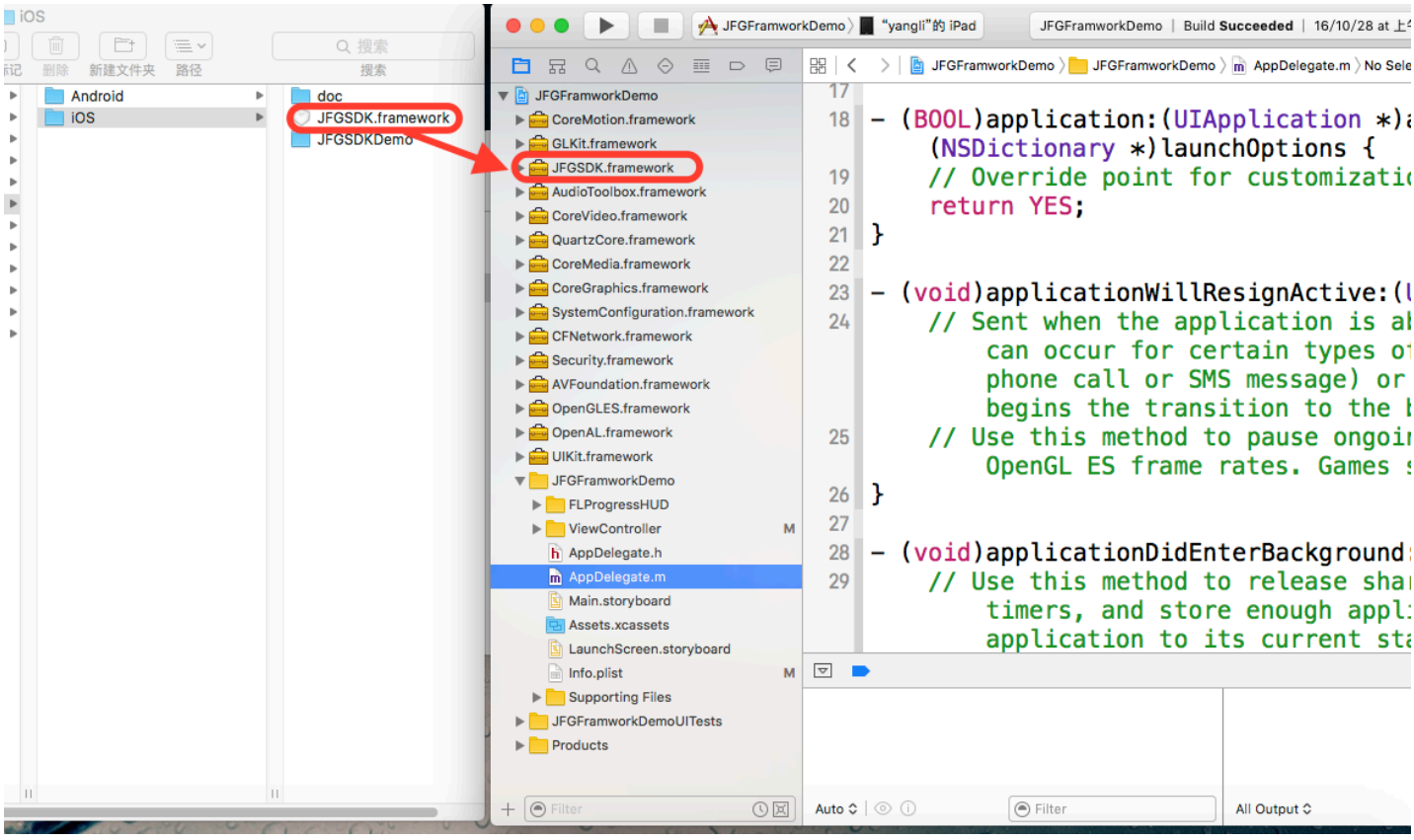
第二步:注册完成后，在“企业信息”-->“账号设置”中找到“企业编号”即是VID。“SDK密钥”即是VKEY。如果SDK密钥内容为空，则点击右侧的更换即可。



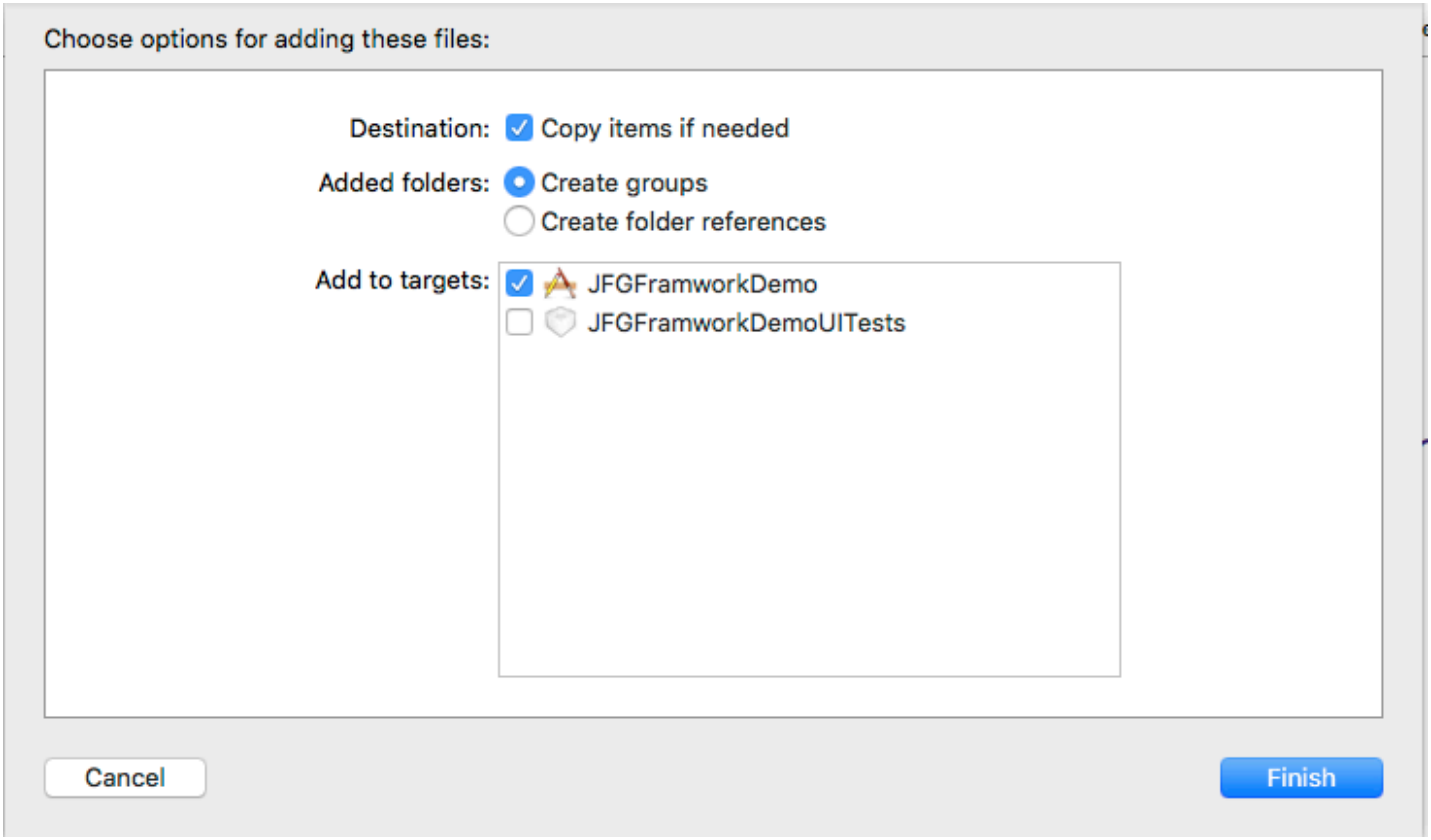
第三步：选择右下方的“+接入应用”，填上自己的应用平台和应用信息。此时，接入成功后，就可以在SDK登录，注册时使用第二步中的VKEY，VID。

2.下载SDK及Demo。

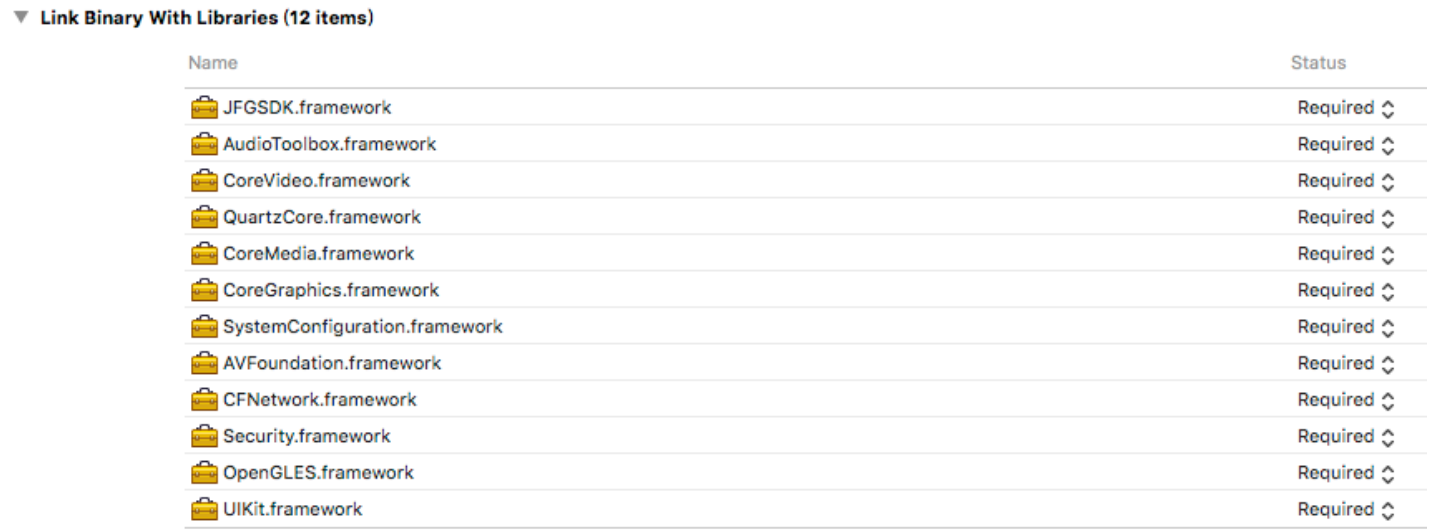
3.将下载好的framework拖入您的工程中



拖到工程中后弹出以下对话框，勾选"Copy items into destination group's folder(if needed)"，并点击“Finish“按钮, 如图

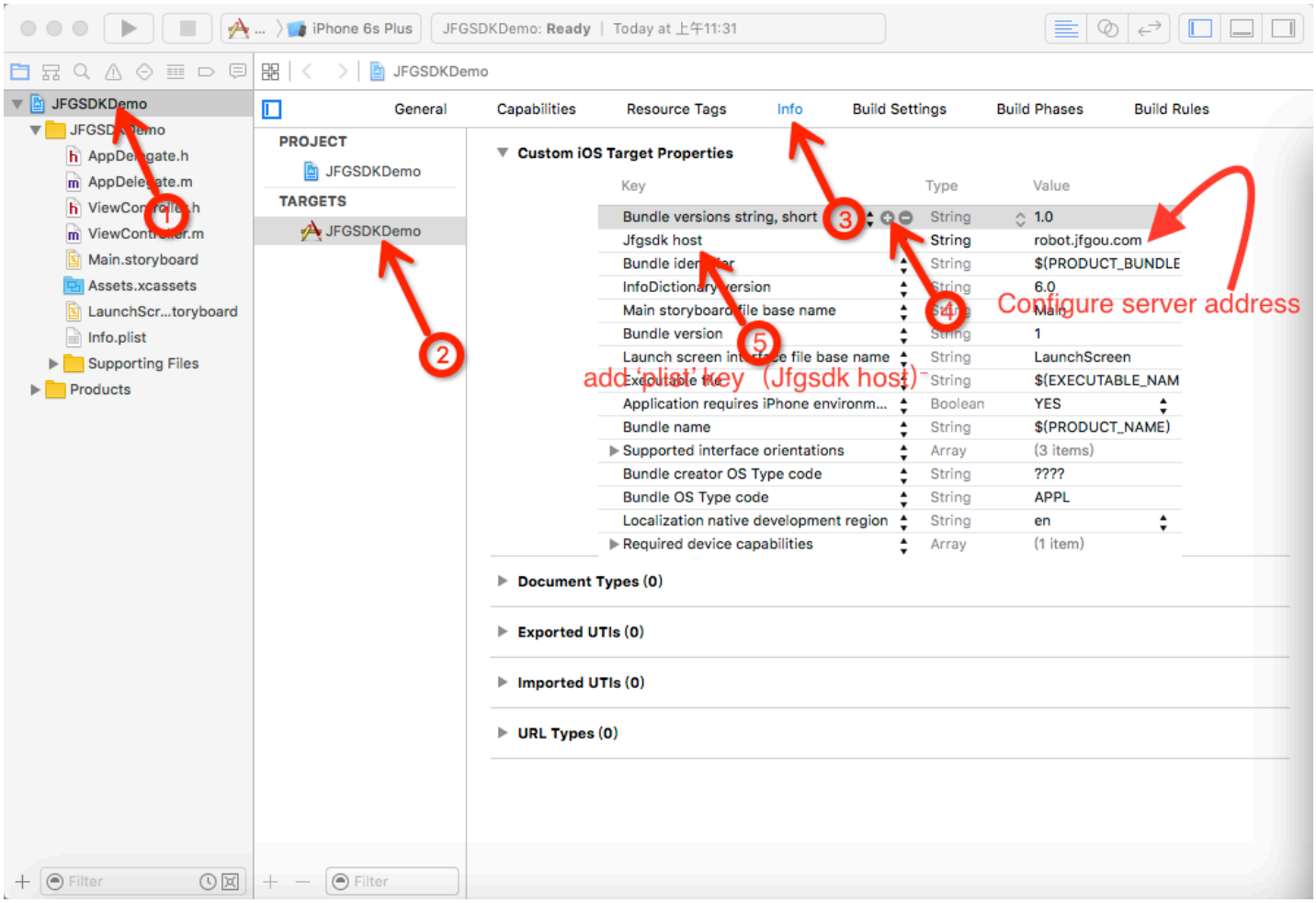


4.添加如下依赖库，如图



5.找到Build Settings，把Enable Bitcode设置为NO

6.设置SDK服务器地址（默认robot.jfgou.com）



iOS APP开发-2初始化

1.初始化SDK

打开 *AppDelegate.m (*代表你的工程名字)导入文件头JFGSDK.h

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

SDK初始化

```
//初始化SDK，设置日志保存路径
[JFGSDK connectForWorkDir:path];

//添加SDK回调代理
[JFGSDK addDelegate:self];

//开启SDK操作日志记录功能
[JFGSDK logEnable:YES];
```

iOS APP开发-3用户管理

1.用户注册（JFGSDK.h）

第一步:发送注册手机号验证码（邮箱注册可以直接跳到第五步）

> [JFGSDK sendSMSWithPhoneNumber:phoneNumber type:0];

note:具体参数定义可以参考头文件（JFGSDK.framework/Headers/JFGSDK.h）中的方法。

```
110
111 /**
112  *  send SMS code
113  *
114  *  @param phone     phone
115  *  @param type      0:register/binding phone   1:forget password   2:Modify
         password
116  */
117 +(void)sendSMSWithPhoneNumber:(NSString *)phone type:(int)type;
118
```

第二步:获取验证码发送结果（JFGSDK回调）

```
-(void)jfgSendSMSResult:(JFGErrorType)errorType token:(NSString *)token
{
   if (errorType == 0) {
        //验证码发送至手机成功
   }else{
        //验证码发送至手机失败
   }
   //记录这个token，注册第五步将会用到
   smsToken = token;
}
```

第三步：验证验证码是否过期

```
[JFGSDK verifySMSWithAccount:注册手机号
                        code:手机收到的验证码
                       token:第二步中记录的token];
```

第四步：获取检验验证码（第三步）结果（JFGSDK回调）

```
-(void)jfgVerifySMSResult:(JFGErrorType)errorType
{
    if (errorType == 0) {
        //验证码有效，进行第五步
    }else{
        //验证码无效，从第一步重新开始
    }
}
```

第五步：注册

```
[JFGSDK userRegister:注册账号（手机号/邮箱）
             keyword:密码
        registerType:注册类型（0：手机  1：邮箱）
               token:第二步中记录的token（邮箱注册填充@""）
                 vid:萝卜头平台获取的VID];
```

note:请确保VID，项目BundleID与萝卜头平台注册时的一致。否则会报160错误。

第六步：注册结果

```
-(void)jfgRegisterResult:(JFGErrorType)errorType
{
    if (errorType == 0) {
       //注册成功
    }else{
       //注册失败
    }
}
```

2.用户登录(JFGSDK.h)

第一步：登录操作

```
[JFGSDK userLogin:账号
          keyword:密码
              vid:萝卜头平台获取的VID
             vkey:萝卜头平台获取的VKEY];
```

note:请确保VID，项目BundleID与萝卜头平台注册时的一致。否则会报160错误。

第二步：获取登录结果（JFGSDK回调）

```
-(void)jfgLoginResult:(JFGErrorType)errorType
{
   if (errorType == 0) {
      //登录成功
   }else{
      //登录失败
   }
}
```

iOS APP开发-4设备绑定

第一步：设备起AP，iphone手机连接设备wifi。

第二步：初始化

导入头文件JFGSDKBindingDevice.h，遵从代理协议

> #import <JFGSDK/JFGSDKBindingDevice.h>

创建全局JFGSDKBindingDevice对象

```
bindDevice = [[JFGSDKBindingDevice alloc]init];
bindDevice.delegate = self;
```

第三步：开始绑定（JFGSDKBindingDevice.h）

```
[bindDevice bindDevWithSn:nil
                     ssid:周边可用wifi名
                      key:周边可用wifi密码];
```

note:具体参数定义可以参考头文件（JFGSDK.framework/Headers/JFGSDKBindingDevice.h）中的方法

```
/*!
 *  绑定设备（绑定成功后会触发JFGSDKCallbackDelegate #jfgDeviceList:回调）
 *  需在连接设备wifi后执行此操作
 *  @param sn  新设备填写设备sn，旧设备无需填写（cid）
 *  @param ssid  wifi ssid
 *  @param key   wifi 密码
 */
-(void)bindDevWithSn:(NSString *)sn ssid:(NSString *)ssid key:(NSString *)key;
```

第四步：获取绑定结果

绑定过程回调

```
-(void)jfgBindDeviceProgressStatus:(JFGSDKBindindProgressStatus)status
```

绑定成功回调

```
-(void)fjgBindDeviceSuccessForPeer:(NSString *)peer
```

绑定失败回调

```
-(void)jfgBindDeviceFailed:(JFGSDKBindindProgressStatus)errorType
```

note：具体参数定义可以参考头文件（JFGSDK.framework/Headers/JFGSDKBindingDevice.h)

```objc
typedef NS_ENUM (NSInteger,JFGSDKBindindProgressStatus){

    JFGSDKBindindProgressStatusPing,//ping操作

    JFGSDKBindindProgressStatusConfigureStart,//开始配置摄像头参数
    JFGSDKBindindProgressStatusConfigureSuccess,//配置摄像头参数成功
    JFGSDKBindindProgressStatusStartBinding,//开始绑定

    JFGSDKBindindProgressStatusSuccess,//绑定成功

    JFGSDKBindindProgressStatusSetWifiFailed,//设置摄像头wifi失败
    JFGSDKBindindProgressStatusBindTimeout,//绑定超时

    JFGSDKBindindProgressStatusCIDNotExist = 200,//CID不存在。关联消息：客户端绑定
    JFGSDKBindindProgressStatusCIDBinding,// 绑定中，正在等待摄像头上传随机数与CID关
        联关系，随后推送绑定通知

};
```

iOS APP开发-5设备音视频

1.直播

第一步：初始化

导入头文件JFGSDKVideoView.h，遵从代理协议

> #import <JFGSDK/JFGSDKVideoView.h>

创建全局JFGSDKVideoView对象

```objc
playView = [[JFGSDKVideoView alloc]initWithFrame:CGRectMake(0, 64, self.view.bound
s.size.width, 255)];
playView.center = CGPointMake(self.view.bounds.size.width*0.5, 64+300);
playView.delegate = self;
[self.view addSubview:playView];
```

第二步：开始视频直播

```objc
[playView startLiveRemoteVideo:设备标示];
```

第三步：获取直播代理回调

1）成功连接直播设备会首先回调以下函数，返回绘制直播画面的尺寸

```
-(void)jfgResolutionNotifyWidth:(int)width
                         height:(int)height
                           peer:(NSString *)peer
```

note：直播画面以初始化时给定的宽为基准，经过等比拉伸。如果此方案不能满足您的需求，请联系我司，提供另外一套完全自定义API。

2）成功连接后，此方法会每秒回调一次

```
-(void)jfgRTCPNotifyBitRate:(int)bitRate
                 videoRecved:(int)videoRecved
                   frameRate:(int)frameRate
                   timesTamp:(int)timesTamp
```

note:具体参数含义请参考头文件（JFGSDK.framework/Headers/JFGSDKPlayVideoDelegate.h）

```
/*!
 *  视频RTCP通知(每秒一次回调)
 *
 *  @param bitRate      码率,单位为 KB/s
 *  @param videoRecved  本次连接以来总共收到多少数据,单位为KB
 *  @param frameRate    帧率
 *  @param timesTamp    设备发送的时间戳,用于在查看历史录像时同步本地进度,看实时画面时此参数为0
 */
-(void)jfgRTCPNotifyBitRate:(int)bitRate
                 videoRecved:(int)videoRecved
                   frameRate:(int)frameRate
                   timesTamp:(int)timesTamp;

```

3)视频连接发生错误回调

```
-(void)jfgTransportFail:(NSString *)remoteID
                  error:(JFGErrorType)errorType
```

2.播放历史记录

第一步：初始化

同直播第一步

第二步：获取历史记录

```
[playView getHistoryVideoList:设备标示];
```

第三步：获取历史记录回调

```
-(void)jfgHistoryVideoList:(NSArray <JFGSDKHistoryVideoInfo *>*)list
{
    //播放历史记录需要用到JFGSDKHistoryVideoInfo对象中的beginTime属性
    dataArray = [[NSMutableArray alloc]initWithArray:list];
}
```

第四步：播放历史记录

```
//获取历史记录对象
JFGSDKHistoryVideoInfo *info  = dataArray[indexPath.row];
//播放某个时间点开始的历史记录
[self startHistoryVideo:info.beginTime];
```

第五步：获取视频代理回调

1）成功连接直播设备会首先回调以下函数，返回绘制直播画面的尺寸

```
-(void)jfgResolutionNotifyWidth:(int)width
                         height:(int)height
                           peer:(NSString *)peer
```

note：1）直播过程中可以直接切换历史视频（第四步），但不会有此回调。 2）历史录像播放过程中若需要切换直播，需要先停止视频播放，在重新从直播第二步开始。

2）成功连接后，此方法会每秒回调一次

```
-(void)jfgRTCPNotifyBitRate:(int)bitRate
                 videoRecved:(int)videoRecved
                   frameRate:(int)frameRate
                   timesTamp:(int)timesTamp
```

3)视频连接发生错误回调

```
-(void)jfgTransportFail:(NSString *)remoteID
                  error:(JFGErrorType)errorType
```

3.停止播放

```
[playView stopVideo];
```

iOS APP开发-6设备设置

1.导入头文件

```
#import <JFGSDK/JFGSDKDataPoint.h>
```

2.查阅具体消息ID含义

请参考我司DP消息文档,或者贵司自己协商的文档，了解每个消息ID，具体含义及数据结构类型。

3.示例

1.请求数据

```
DataPointIDVerSeg *_seg = [[DataPointIDVerSeg alloc]init];
_seg.msgId = 505;
_seg.version = 0;//某个时间点的数据（0代表最近时间点）

[[JFGSDKDataPoint sharedClient] robotGetDataWithPeer:@"200000149340" msgIds:@[_seg
] asc:NO limit:100 success:^(NSString *identity, NSArray<NSArray<DataPointSeg *> *
> *idDataList) {

    for (NSArray *subArr in idDataList) {

        for (DataPointSeg *seg in subArr) {

            //数据传递经过了msgpack打包，所以这里需要解析出原来的数据
            id obj = [MPMessagePackReader readData:seg.value error:nil];
            //obj数据类型要根据当初的协定判断
            if ([obj isKindOfClass:[NSArray class]]) {
                NSArray *arr = obj;
                NSLog(@"%@",[arr description]);
            }
        }
    }
} failure:^(RobotDataRequestErrorType type) {



}];
```

2.设置数据

```
DataPointSeg *setSeg = [[DataPointSeg alloc]init];
setSeg.msgId = 501;
setSeg.version = 0;
//消息msgpack打包
NSData *packData =[MPMessagePackWriter writeObject:[NSNumber numberWithBool:YES] e
rror:nil];
setSeg.value = packData;

[[JFGSDKDataPoint sharedClient] robotSetDataWithPeer:@"200000149340" dps:@[setSeg]
 success:^(NSArray<DataPointIDVerRetSeg *> *dataList) {

    for (DataPointIDVerRetSeg *resultSeg in dataList) {
        if (resultSeg.ret == 0) {
            NSLog(@"success");
        }else{
            NSLog(@"failed");
        }
    }

} failure:^(RobotDataRequestErrorType type) {

}];
```

iOS APP开发-7错误码定义

见头文件JFGSDK.framework/Headers/JFGTypeDefine.h

```
/*!
 *  错误码
 */
typedef NS_ENUM (NSUInteger, JFGErrorType){
    // EOK 成功
    JFGErrorTypeNone = 0,

    // P2P 错误
    JFGErrorTypeP2PDns,
    JFGErrorTypeP2PSocket,
    JFGErrorTypeP2PCallerRelay,
    JFGErrorTypeP2PCallerStun,
    JFGErrorTypeP2PCalleeStun = 5,
    JFGErrorTypeP2PCalleeWaitCallerCheckNetTimeOut,
    JFGErrorTypeP2PPeerTimeOut,
    JFGErrorTypeP2PUserCancel,
    JFGErrorTypeP2PConnectionCheck,
    JFGErrorTypeP2PChannel = 10,
    JFGErrorTypeP2PDisconetByUser,

    // 直播类
    // 对端不在线
    JFGErrorTypeVideoPeerNotExist = 100,
```