

iOS SDK 接入文档

一、简介

二、接入环境

1、集成sdk

2、初始化项目

三、批量操作

I、属性：

- 1、overtime
- 2、reconnectCount
- 3、timeout
- 4、sdkVersion

II、方法：批量操作

- 1、同步操作-设置分辨率和DPI指令
- 2、全局root控制指令
- 3、剪贴板指令
- 4、虚拟定位指令

III、方法：应用管理

- 1、启动指定的应用指令
- 2、停止指定的应用指令
- 3、卸载指定的应用指令
- 4、隐藏预置的应用指令
- 5、显示预置的应用指令
- 6、应用root权限控制指令
- 7、启用指定的应用指令
- 8、停用指定的应用指令
- 9、清理指定的应用指令
- 10、设备新机指令

IV、方法：预览图

- 1、开启预览图推流指令
- 2、停止预览图推流指令
- 3、停止所有预览图推流

四、音视频

I、属性：

- 1、isShowLogInfo
- 2、isCallBackLogInfo
- 3、shouldGetStats

- 4、url
- 5、ice
- 6、sn
- 7、fmt
- 8、isHWEncode
- 9、videoWidth
- 10、videoHeight
- 11、fps
- 12、bitrate
- 13、token
- 14、delegate
- 15、isHWEncodePre
- 16、fmtPre
- 17、fpsPre

II、方法：初始化

- 1、初始化
- 2、反初始化
- 3、创建RCPlayerView对象

III、方法：操作流程

- 1、切换主屏幕-重新连接 - 拉流
- 2、主动断开连接
- 3、暂停接收流
- 4、恢复接收流
- 5、音频开关
- 6、HD画质：自动码率和手动码率
- 7、设置渲染视频填充模式
- 8、设置是否回调流信息状态
- 9、设置输入键盘类型：

IV、方法：控制面板

- 1、设置视频类型 1是H264。2是vp8 5是H265 （需要重新拉流才生效）
- 2、拉流前调用设置软硬编码，YES硬编码，NO软编码，不调用默认硬编码
- 3、拉流过程中改变视频分辨率
- 4、拉流过程中改变HD画质（默认值：流畅-800 标清-1500 高清-2200 超清-2800）
- 5、设置帧率（需要重新拉流才生效）
- 6、调整音量 0-减 1-加
- 7、Home键

8、返回键

9、菜单键

V、方法：Websocket

1、开启websocket

2、关闭websocket

3、订阅主题与取消订阅主题

4、查询推流画质

5、查询分辨率

6、查询横竖屏状态 (ws指令)

7、查询隐藏应用列表

8、查询所有应用列表

9、显示应用

10、隐藏应用

11、打开应用

12、截图 (quality : 1~100)

13、吹一吹 取值范围 5-10秒

14、摇一摇 取值范围 3-10秒

15、设置输入键盘类型：

16、自定义内容透传给云机输入框

17、自定义内容透传给云机剪贴板

18、隐藏导航栏

19、显示导航栏

20、查询导航栏状态

21、查询屏幕状态 NO-竖屏 YES-横屏 (视频流横竖屏)

22、查询全局 root 状态

23、设置全局 root 状态

24、开启/关闭 传感器数据透传

VI、方法：其他

1、通过data channel发送字符串到云机

2、通过data channel发送BYTE到云机

3、通过data channel发送键盘事件到云机

4、自定义拉流 WS 发送指令

VII、代理RCPlayerViewDelegate：音视频状态回调

1、数据通道连接状态回调

2、音视频连接状态回调

3、分辨率和横竖屏状态回调

- 4、音视频鉴权状态回调
- 5、数据通道接收消息回调
- 6、音视频流数据 实时状态回调
- 7、音视频websocket 接收消息回调
- 8、音视频 参数信息回调
- 9、SDK 错误信息提示 Toast

五、屏幕旋转：画面横屏适配

- 1、引入分类文件
- 2、屏幕旋转配置
- 3、屏幕旋转代码

iOS SDK 接入文档

一、简介

SDK主要功能是为APP赋予云手机使用能力，可以通过SDK连接云手机，完成对云手机的一系列操作，包括RCOperation.framework和WebRTC.framework两个包、完整的Demo示例、接入文档。

二、接入环境

1、集成sdk

将RCOperation.framework和WebRTC.framework两个包拖入项目中，在targets->General->Framework中 将WebRTC.framework包设置为Enable&Sign。

2、初始化项目

在AppDelegate里，调用[RCPlayerView globalInitialization];方法初始化音视频，调用[RCPlayerView globalDeinitialization];方法，反初始化音视频。横竖屏相关代码如下示例。

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    // 初始化音视频
    [RCPlayerView globalInitialization];

    return YES;
}

- (void)applicationWillTerminate:(UIApplication *)application {
    // 反初始化音视频
    [RCPlayerView globalDeinitialization];
}
```

三、批量操作

RCOperation.framework中有个操作管理类RCOperateManager，所有批量操作相关API均可通过它调用，通过Block回调相关状态信息。

I、属性：

1、overtime

ws超时重连间隔时间，默认1秒。

2、reconnectCount

ws超时重连次数，默认5次。

3、timeout

ws超时时间，默认30秒。

4、sdkVersion

II、方法：批量操作

参数	类型	必选	说明
hosts	NSArray	是	ws 地址，来源:/api/v2/instance/list接口publicIp字段
ips	NSArray	是	实例ip 来源:/api/v2/instance/list接口ip字段
tokens	NSArray	是	控制 token 来源:/api/v2/instance/control/token/get接口token字段

1、同步操作-设置分辨率和DPI指令

```
- (void)resolutionWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:
(NSArray *)tokens width:(NSString *)width height:(NSString *)height
dpi:(NSString *)dpi progress:(RCProgress)progress complete:
(RCComplete)complete;
```

```
- (void)syscAlertSureBtnClick {
    if (self.syscAlert.widthTextField.text.length == 0) {
        [RCToast showToast:@"请输入宽度"];
        return;
    } else if (self.syscAlert.heightTextField.text.length == 0) {
        [RCToast showToast:@"请输入高度"];
        return;
    } else if (self.syscAlert.DPITextField.text.length == 0) {
        [RCToast showToast:@"请输入DPI"];
        return;
    }

    self.syscAlert.hidden = YES;
    [self.view endEditing:YES];
}
```

```
// 分辨率、DPI
NSString *width = self.syscAlert.widthTextField.text;
NSString *height = self.syscAlert.heightTextField.text;
NSString *dpi = self.syscAlert.DPITextField.text;

self.tokenModel.videoWidth = [width integerValue];
self.tokenModel.videoHeight = [height integerValue];
self.tokenModel.cardDensity = [dpi integerValue];

NSLog(@"分辨率、DPI");
[self getControlTokenType:0 success:^(id _Nonnull responseObject)
{
    NSArray *tokenModelArray = responseObject;
    NSMutableArray *hosts = [NSMutableArray array];
    NSMutableArray *ips = [NSMutableArray array];
    NSMutableArray *tokens = [NSMutableArray array];
    for (RCDeviceModel *model in self.selectArray) {
        NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
        [hosts addObject:forward_server_addr];
        [ips addObject:model.ip];

        NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
        RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
        [tokens addObject:targetModel.token];
    }

    self.syscLoadingAlert.hidden = NO;
    [self.syscLoadingAlert refreshCount:hosts.count success:0
fail:0];
}
```

```

        [[RCOperateManager sharedInstance] resolutionWSHosts:hosts
ips:ips tokens:tokens width:width height:height dpi:dpi progress:^(int
success, int fail) {
    NSLog(@"指令发送成功: %d 失败: %d", success, fail);
    if (self.syscLoadingAlert.hidden) { return; }
    [self.syscLoadingAlert refreshCount:hosts.count
success:success fail:fail];
    } complete:^(BOOL isComplete) {
        NSLog(@"指令发送完成");
//        self.syscLoadingAlert.hidden = YES;
    }];
}];

}

```

2、全局root控制指令

```

- (void)rootSystemWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:
(NSArray *)tokens open:(**BOOL**)isOpen progress:(RCProgress)progress
complete:(RCComplete)complete;

```

```

- (void)rootSystemAlertSureBtnClick {
    // 开关
    BOOL isOpen = self.rootSystemAlert.openBtn.isSelected;
    self.rootSystemAlert.hidden = YES;

    // 全局ROOT
    NSLog(@"全局ROOT");
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
    {
        NSArray *tokenModelArray = responseObject;
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
    }

```



```

NSMutableArray *tokens = [NSMutableArray array];
for (RCDeviceModel *model in self.selectArray) {
    NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
    [hosts addObject:forward_server_addr];
    [ips addObject:model.ip];

    NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
    RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
    [tokens addObject:targetModel.token];
}

self.loadingAlert.hidden = NO;
[self.loadingAlert refreshCount:hosts.count success:0 fail:0];

[[RCOperateManager sharedInstance] rootSystemWSHosts:hosts
ips:ips tokens:tokens open:isOpen progress:^(int success, int fail) {
    NSLog(@"指令发送成功: %d 失败: %d", success, fail);
    if (self.loadingAlert.hidden) { return; }
    [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
    } complete:^(BOOL isComplete) {
        NSLog(@"指令发送完成");
//        self.loadingAlert.hidden = YES;
    }];
}];
}

```

3、剪贴板指令

```
- (void)clipboardWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:
(NSArray *)tokens text:(NSString *)text progress:(RCProgress)progress
complete:(RCComplete)complete;
```

```
- (void)clipboardAlertSureBtnClick {
    self.clipboardAlert.hidden = YES;

    // 内容
    UIPasteboard *pasteboard = [UIPasteboard generalPasteboard];
    NSString *clipboardContent = pasteboard.string;
    if (clipboardContent.length == 0) {
        [RCToast showToast:@"剪贴板为空"];
        return;
    }

    // 剪贴板
    NSLog(@"剪贴板 %@", clipboardContent);
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
    {
        NSArray *tokenModelArray = responseObject;
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
        NSMutableArray *tokens = [NSMutableArray array];
        for (RCDeviceModel *model in self.selectArray) {
            NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
            [hosts addObject:forward_server_addr];
            [ips addObject:model.ip];

            NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
            RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
            [tokens addObject:targetModel.token];
        }
    }];
}
```

```

        self.loadingAlert.hidden = NO;
        [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

        [[RCOperateManager sharedInstance] clipboardWSHosts:hosts
ips:ips tokens:tokens text:clipboardContent progress:^(int success,
int fail) {
            NSLog(@"指令发送成功: %d 失败: %d", success, fail);
            if (self.loadingAlert.hidden) { return; }
            [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
            } complete:^(BOOL isComplete) {
                NSLog(@"指令发送完成");
//                self.loadingAlert.hidden = YES;
            }]];
    }
}

```

4、虚拟定位指令

```

- (void)virtualLocationWSHosts:(NSArray *)hosts ips:(NSArray *)ips
tokens:(NSArray *)tokens longitude:(NSString *)longitude latitude:
(NSString *)latitude progress:(RCProgress)progress complete:
(RCComplete)complete;

```

```

- (void)locationAlertSureBtnClick {
    // 经纬度
    NSString *longitude = self.locationAlert.longitudeTextField.text;
    NSString *latitude = self.locationAlert.latitudeTextField.text;
    if (longitude.length == 0) {
        [RCToast showToast:@"请输入经度"];
        return;
    } else if (latitude.length == 0) {

```

```

        [RCToast showToast:@"请输入纬度"];;
        return;
    }

    self.locationAlert.hidden = YES;
    [self.view endEditing:YES];

    // 虚拟定位
    NSLog(@"虚拟定位");
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
    {
        NSArray *tokenModelArray = responseObject;
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
        NSMutableArray *tokens = [NSMutableArray array];
        for (RCDeviceModel *model in self.selectArray) {
            NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
            [hosts addObject:forward_server_addr];
            [ips addObject:model.ip];

            NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
            RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
            [tokens addObject:targetModel.token];
        }

        self.loadingAlert.hidden = NO;
        [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

        [[RCOperateManager sharedInstance] virtualLocationWSHosts:hosts
ips:ips tokens:tokens longitude:longitude latitude:latitude
progress:^(int success, int fail) {
            NSLog(@"指令发送成功: %d 失败: %d", success, fail);
            if (self.loadingAlert.hidden) { return; }
        }
    }];
    }

```

```

        [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
    } complete:^(BOOL isComplete) {
        NSLog(@"指令发送完成");
//        self.loadingAlert.hidden = YES;
    }];
}];
}

```

III、方法：应用管理

1、启动指定的应用指令

```

- (void)startAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:
(NSArray *)tokens packageName:(NSString *)packageName progress:
(RCProgress)progress complete:(RCComplete)complete;

```

```

- (void)startAppAlertSureBtnClick {
    // 包名
    NSString *packageName =
self.startAppAlert.packageNameTextField.text;
    if (packageName.length == 0) {
        [RCToast showToast:@"请输入需启动的应用包名"];
        return;
    }

    self.startAppAlert.hidden = YES;
    [self.view endEditing:YES];

    // 启动应用
    NSLog(@"启动应用");
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
{

```

```

        NSArray *tokenModelArray = responseObject;
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
        NSMutableArray *tokens = [NSMutableArray array];
        for (RCDeviceModel *model in self.selectArray) {
            NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
            [hosts addObject:forward_server_addr];
            [ips addObject:model.ip];

            NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
            RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
            [tokens addObject:targetModel.token];
        }

        self.loadingAlert.hidden = NO;
        [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

        [[RCOperateManager sharedInstance] startAppWSHosts:hosts
ips:ips tokens:tokens packageName:packageName progress:^(int success,
int fail) {
            if (self.loadingAlert.hidden) { return; }
            [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
        } complete:^(BOOL isComplete) {
            NSLog(@"指令发送完成");
            //            self.loadingAlert.hidden = YES;
        }];
    }];
}

```

2、停止指定的应用指令

```
- (void)stopAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:
(NSArray *)tokens packageName:(NSString *)packageName progress:
(RCProgress)progress complete:(RCComplete)complete;
```

```
- (void)stopAppAlertSureBtnClick {
    NSString *packageName =
self.stopAppAlert.packageNameTextField.text;
    if (packageName.length == 0) {
        [RCToast showToast:@"请输入需停止的应用包名"];
        return;
    }

    self.stopAppAlert.hidden = YES;
    [self.view endEditing:YES];

    // 停止应用
    NSLog(@"停止应用");
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
    {
        NSArray *tokenModelArray = responseObject;
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
        NSMutableArray *tokens = [NSMutableArray array];
        for (RCDeviceModel *model in self.selectArray) {
            NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
            [hosts addObject:forward_server_addr];
            [ips addObject:model.ip];

            NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
            RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
            [tokens addObject:targetModel.token];
        }
    }];
}
```

```

        self.loadingAlert.hidden = NO;
        [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

        [[RCOperateManager sharedInstance] stopAppWSHosts:hosts ips:ips
tokens:tokens packageName:packageName progress:^(int success, int
fail) {
            if (self.loadingAlert.hidden) { return; }
            [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
            } complete:^(BOOL isComplete) {
                NSLog(@"指令发送完成");
//                self.loadingAlert.hidden = YES;
            }];
        }];
    }
}

```

3、卸载指定的应用指令

```

-(void)uninstallAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:
(NSArray *)tokens packageName:(NSString *)packageName progress:
(RCProgress)progress complete:(RCComplete)complete;

```

```

- (void)uninstallAlertSureBtnClick {
    // 包名
    NSString *packageName =
self.uninstallAlert.packageNameTextField.text;
    if (packageName.length == 0) {
        [RCToast showToast:@"请输入需卸载的应用包名"];
        return;
    }

    self.uninstallAlert.hidden = YES;
}

```



```

[self.view endEditing:YES];

// 卸载应用
NSLog(@"卸载应用");
[self getControlTokenType:0 success:^(id _Nonnull responseObject)
{
    NSArray *tokenModelArray = responseObject;
    NSMutableArray *hosts = [NSMutableArray array];
    NSMutableArray *ips = [NSMutableArray array];
    NSMutableArray *tokens = [NSMutableArray array];
    for (RCDeviceModel *model in self.selectArray) {
        NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
        [hosts addObject:forward_server_addr];
        [ips addObject:model.ip];

        NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
        RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
        [tokens addObject:targetModel.token];
    }

    self.loadingAlert.hidden = NO;
    [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

    [[RCOperateManager sharedInstance] uninstallAppWSHosts:hosts
ips:ips tokens:tokens packageName:packageName progress:^(int success,
int fail) {
        if (self.loadingAlert.hidden) { return; }
        [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
    } complete:^(BOOL isComplete) {
        NSLog(@"指令发送完成");
        // self.loadingAlert.hidden = YES;
    }]];
}];

```

```
    }];  
}
```

4、隐藏预置的应用指令

```
- (void)hideAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:  
(NSArray *)tokens packageNames:(NSArray *)packageNames progress:  
(RCProgress)progress complete:(RCComplete)complete;
```

```
- (void)hideAppAlertSureBtnClick {  
    NSString *packageName =  
self.hideAppAlert.packageNameTextField.text;  
    if (packageName.length == 0) {  
        [RCToast showToast:@"请输入需隐藏的应用包名"];  
        return;  
    }  
  
    self.hideAppAlert.hidden = YES;  
    [self.view endEditing:YES];  
  
    // 隐藏应用  
    NSLog(@"隐藏应用");  
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)  
    {  
        NSArray *tokenModelArray = responseObject;  
        NSMutableArray *hosts = [NSMutableArray array];  
        NSMutableArray *ips = [NSMutableArray array];  
        NSMutableArray *tokens = [NSMutableArray array];  
        for (RCDeviceModel *model in self.selectArray) {  
            NSString *forward_server_addr =  
model.forwardServer.firstObject.publicIp;  
            [hosts addObject:forward_server_addr];  
            [ips addObject:model.ip];  
        }  
    }];  
}
```

```

        NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
        RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
        [tokens addObject:targetModel.token];
    }

    self.loadingAlert.hidden = NO;
    [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

    [[RCOperateManager sharedInstance] hideAppWSHosts:hosts ips:ips
tokens:tokens packageNames:@[packageName] progress:^(int success, int
fail) {
        if (self.loadingAlert.hidden) { return; }
        [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
    } complete:^(BOOL isComplete) {
        NSLog(@"指令发送完成");
//        self.loadingAlert.hidden = YES;
    }];
}];
}

```

5、显示预置的应用指令

```

- (void)showAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:
(NSArray *)tokens packageNames:(NSArray *)packageNames progress:
(RCProgress)progress complete:(RCComplete)complete;

```

```

- (void)showAppAlertSureBtnClick {
    NSString *packageName =
self.showAppAlert.packageNameTextField.text;

```

```
if (packageName.length == 0) {
    [RCToast showToast:@"请输入需显示的应用包名"];
    return;
}

self.showAppAlert.hidden = YES;
[self.view endEditing:YES];

// 显示应用
NSLog(@"显示应用");
[self getControlTokenType:0 success:^(id _Nonnull responseObject)
{
    NSArray *tokenModelArray = responseObject;
    NSMutableArray *hosts = [NSMutableArray array];
    NSMutableArray *ips = [NSMutableArray array];
    NSMutableArray *tokens = [NSMutableArray array];
    for (RCDeviceModel *model in self.selectArray) {
        NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
        [hosts addObject:forward_server_addr];
        [ips addObject:model.ip];

        NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
        RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
        [tokens addObject:targetModel.token];
    }

    self.loadingAlert.hidden = NO;
    [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

    [[RCOperateManager sharedInstance] showAppWSHosts:hosts ips:ips
tokens:tokens packageNames:@[packageName] progress:^(int success, int
fail) {
        if (self.loadingAlert.hidden) { return; }
    }
}
```

```

        [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
    } complete:^(BOOL isComplete) {
        NSLog(@"指令发送完成");
//        self.loadingAlert.hidden = YES;
    }];
}];
}

```

6、应用root权限控制指令

```

- (void)rootAppsWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:
(NSArray *)tokens packageName:(NSString *)packageName open:
(**BOOL**)isOpen progress:(RCProgress)progress complete:
(RCComplete)complete;

```

```

- (void)rootAppAlertSureBtnClick {
    // 包名
    NSString *packageName =
self.rootAppAlert.packageNameTextField.text;
    if (packageName.length == 0) {
        [RCToast showToast:@"请输入ROOT应用包名"];
        return;
    }
    // 开关
    BOOL isOpen = self.rootAppAlert.openBtn.isSelected;
    self.rootAppAlert.hidden = YES;
    [self.view endEditing:YES];

    // ROOT应用
    NSLog(@"ROOT应用");
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
{

```

```

        NSArray *tokenModelArray = responseObject;
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
        NSMutableArray *tokens = [NSMutableArray array];
        for (RCDeviceModel *model in self.selectArray) {
            NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
            [hosts addObject:forward_server_addr];
            [ips addObject:model.ip];

            NSPredicate *predicate = [NSPredicate
predicateWithFormat:@"instanceId == %@", model.instanceId];
            RCControlTokenModel *targetModel = [[tokenModelArray
filteredArrayUsingPredicate:predicate] firstObject];
            [tokens addObject:targetModel.token];
        }

        self.loadingAlert.hidden = NO;
        [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

        [[RCOperateManager sharedInstance] rootAppsWSHosts:hosts
ips:ips tokens:tokens packageName:packageName open:isOpen
progress:^(int success, int fail) {
            NSLog(@"指令发送成功: %d 失败: %d", success, fail);
            if (self.loadingAlert.hidden) { return; }
            [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
        } complete:^(BOOL isComplete) {
            NSLog(@"指令发送完成");
            //            self.loadingAlert.hidden = YES;
        }];
    }];
}

```

7、启用指定的应用指令

```
- (void)enableAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:(NSArray *)tokens packageName:(NSString *)packageName progress:(RCProgress)progress complete:(RCComplete)complete;
```

```
- (void)enableAppAlertSureBtnClick {
    NSString *packageName =
self.enableAppAlert.packageNameTextField.text;
    if (packageName.length == 0) {
        [RCToast showToast:@"请输入需启用的应用包名"];
        return;
    }

    self.enableAppAlert.hidden = YES;
    [self.view endEditing:YES];

    // 启用应用
    NSLog(@"启用应用");
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
    {
        NSString *controlToken = [NSString stringWithFormat:@"%@",
responseObject];
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
        for (RCDeviceModel *model in self.selectArray) {
            NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
            [hosts addObject:forward_server_addr];
            [ips addObject:model.ip];
        }

        self.loadingAlert.hidden = NO;
        [self.loadingAlert refreshCount:hosts.count success:0 fail:0];
    }];
}
```

```

        [[RCOperateManager sharedInstance] enableAppWSHosts:hosts
ips:ips token:controlToken packageName:packageName progress:^(int
success, int fail) {
            if (self.loadingAlert.hidden) { return; }
            [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
        } complete:^(BOOL isComplete) {
            NSLog(@"指令发送完成");
//            self.loadingAlert.hidden = YES;
        }];
    }];
}

```

8、停用指定的应用指令

- (**void**)disableAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:(NSArray *)tokens packageName:(NSString *)packageName progress:(RCProgress)progress complete:(RCComplete)complete;

```

- (void)disableAppAlertSureBtnClick {
    NSString *packageName =
self.disableAppAlert.packageNameTextField.text;
    if (packageName.length == 0) {
        [RCToast showToast:@"请输入需停用的应用包名"];
        return;
    }

    self.disableAppAlert.hidden = YES;
    [self.view endEditing:YES];

    // 停用应用
    NSLog(@"停用应用");
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
{

```



```

        NSString *controlToken = [NSString stringWithFormat:@"%@",
responseObject];
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
        for (RCDeviceModel *model in self.selectArray) {
            NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
            [hosts addObject:forward_server_addr];
            [ips addObject:model.ip];
        }

        self.loadingAlert.hidden = NO;
        [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

        [[RCOperateManager sharedInstance] disableAppWSHosts:hosts
ips:ips token:controlToken packageName:packageName progress:^(int
success, int fail) {
            if (self.loadingAlert.hidden) { return; }
            [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
        } complete:^(BOOL isComplete) {
            NSLog(@"指令发送完成");
            // self.loadingAlert.hidden = YES;
        }];
    }];
}

```

9、清理指定的应用指令

- (**void**)clearAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:(NSArray *)tokens
packageName:(NSString *)packageName progress:(RCProgress)progress complete:
(RCComplete)complete;

```

- (void)clearAppAlertSureBtnClick {

```

```
NSString *packageName =
self.clearAppAlert.packageNameTextField.text;
if (packageName.length == 0) {
    [RCToast showToast:@"请输入需清理的应用包名"];
    return;
}

self.clearAppAlert.hidden = YES;
[self.view endEditing:YES];

// 清理应用
NSLog(@"清理应用");
[self getControlTokenType:0 success:^(id _Nonnull responseObject)
{
    NSString *controlToken = [NSString stringWithFormat:@"%@",
responseObject];
    NSMutableArray *hosts = [NSMutableArray array];
    NSMutableArray *ips = [NSMutableArray array];
    for (RCDeviceModel *model in self.selectArray) {
        NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
        [hosts addObject:forward_server_addr];
        [ips addObject:model.ip];
    }

    self.loadingAlert.hidden = NO;
    [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

    [[RCOperateManager sharedInstance] clearAppWSHosts:hosts
ips:ips token:controlToken packageName:packageName progress:^(int
success, int fail) {
        if (self.loadingAlert.hidden) { return; }
        [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
    } complete:^(BOOL isComplete) {
        NSLog(@"指令发送完成");
    }];
}
```

```

//          self.loadingAlert.hidden = YES;
        }];
    }];
}

```

10、设备新机指令

- (void)renewAppWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:(NSArray *)tokens brand:(NSString *)brand type:(NSString *)type progress:(RCProgress)progress complete:(RCComplete)complete;

```

- (void)renewAlertClickSureBtn {
    NSString *brand = self.renewAlert.longitudeTextField.text;
    NSString *type = self.renewAlert.latitudeTextField.text;
    if (brand.length == 0) {
        [RCToast showToast:@"请输入品牌"];
        return;
    } else if (type.length == 0) {
        [RCToast showToast:@"请输入型号"];
        return;
    }
    self.renewAlert.hidden = YES;
    [self.view endEditing:YES];

    // 设备新机
    NSLog(@"ROOT应用");
    [self getControlTokenType:0 success:^(id _Nonnull responseObject)
    {
        NSString *controlToken = [NSString stringWithFormat:@"%@",
        responseObject];
        NSMutableArray *hosts = [NSMutableArray array];
        NSMutableArray *ips = [NSMutableArray array];
        for (RCDeviceModel *model in self.selectArray) {

```

```

        NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
        [hosts addObject:forward_server_addr];
        [ips addObject:model.ip];
    }

    self.loadingAlert.hidden = NO;
    [self.loadingAlert refreshCount:hosts.count success:0 fail:0];

    [[RCOperateManager sharedInstance] renewAppWSHosts:hosts
ips:ips token:controlToken brand:brand type:type progress:^(int
success, int fail) {
        NSLog(@"指令发送成功: %d 失败: %d", success, fail);
        if (self.loadingAlert.hidden) { return; }
        [self.loadingAlert refreshCount:hosts.count
success:success fail:fail];
    } complete:^(BOOL isComplete) {
        NSLog(@"指令发送完成");
//        self.loadingAlert.hidden = YES;
    }];
}];
}

```

IV、方法：预览图

1、开启预览图推流指令

- (**void**)startPreviewWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:(NSArray *)tokens width:(NSString *)width height:(NSString *)height interval:(NSString *)interval quality:(NSString *)quality preview:(RCPreview)preview;

```

/// 发送预览图指令
- (void)sendPreviewOrder {

```

```

[self getControlTokenType:1 success:^(id _Nonnull responseObject)
{
    NSString *controlToken = [NSString stringWithFormat:@"%@",
responseObject];
    if ([controlToken isEqualToString:@"-1"]) { return; }

    NSMutableArray *hosts = [NSMutableArray array];
    NSMutableArray *ips = [NSMutableArray array];
    for (RCDeviceModel *model in self.previewArray) {
        NSString *forward_server_addr =
model.forwardServer.firstObject.publicIp;
        if (forward_server_addr.length == 0) {
            [RCToast showToast:@"数据异常 请检查实例WS连接地址是否为
空"];

            return;
        }
        [hosts addObject:forward_server_addr];
        [ips addObject:model.ip];
    }

    [[RCOperateManager sharedInstance] startPreviewWSHosts:hosts
ips:ips token:controlToken width:@"640" height:@"480" interval:@"1"
quality:@"90" preview:^(NSString *key, UIImage * _Nullable image) {

        int index = (int)[self.urlDataSource indexOfObject:key];
        if (index >= 0) {
            //          NSLog(@"预览图 index: %d key:%@", index, key);
            RCDeviceListCell *cell = [self.collectionView
cellForItemAtIndexPath:[NSIndexPath indexPathForItem:index
inSection:0]];
            cell.imageView.image = image;

            RCDeviceModel *model = [self.dataSource
objectAtIndex:index];
            model.image = image;
        }
    }
}

```

```
        }];  
    }];  
}
```

2、停止预览图推流指令

```
- (void)stopPreviewWSHosts:(NSArray *)hosts ips:(NSArray *)ips tokens:(NSArray *)tokens preview:(RCPreview)preview;
```

3、停止所有预览图推流

```
-(void)stopAllConnect;
```

四、音视频

RCOperation.framework中有个音视频管理类RCPlayerView，所有音视频相关API均可通过它调用，通过RCPlayerViewDelegate回调相关状态信息。

I、属性：

1、isShowLogInfo

是否显示调试信息。

2、isCallBackLogInfo

是否回调调试信息。

3、shouldGetStats

标识状态回调状态。

4、url

信令服务器地址。

5、ice

中转服务器地址

6、sn

卡序列号

7、fmt

视频格式：1是264、2是vp8、5是265

8、isHWEncode

编码方式：yes是硬编码。no是软编码

9、videoWidth

视频分辨率 宽

10、videoHeight

视频分辨率 高

11、fps

帧率

12、bitrate

码率

13、token

推流Token

14、delegate

状态回调代理

15、isHWEncodePre

预设-编码方式： yes是硬编码。 no是软编码

16、fmtPre

预设-视频格式： 1是264、 2是vp8、 5是265

17、fpsPre

预设-帧率

II、方法：初始化

1、初始化

```
+(void)globalInitialization;
```

2、反初始化

```
+(void)globalDeinitialization;
```

3、创建RCPlayerView对象

参数	类型	必选	说明
url	NSString	是	信令服务器地址 格式: test-pass-szsy.phone.androidsccloud.com:4432(仅供参考,以接口返回为准) 来源: /api/v2/instance/stream/token/get 接口 signalTcp字段
ice	NSString	是	中转服务器地址 来源: /api/v2/net/forward 接口 forwardAdds字段
sn	NSString	是	实例id 来源: /api/v2/instance/list接口instanceId字段
direct	NSInteger	是	是否传输媒体流 0正常拉流, 1不拉流只用于数据传输 默认传 0
fmt	NSInteger	是	fmt 视频类型: 1是H264、5是H265、2是vp8 默认传 1
videoWidth	NSInteger	是	视频流宽 例如: 720, 1080 根据需要设置 默认传 1080
videoHeight	NSInteger	是	视频流高 例如: 1280, 1920 根据需要设置 默认传 1920
fps	NSInteger	是	拉流帧率 值选择: 15、20、30、35、40、45、50、55、60 默认传 30
bitrate	NSInteger	是	拉流码率 默认传 3200

cardWidth	NSInteger	是	卡分辨率 宽 默认传 0
cardHeight	NSInteger	是	卡分辨率 高 默认传 0
cardDensity	NSInteger	是	DPI 默认传 0
token	NSString	是	拉流鉴权token 来源: /api/v2/instance/stream/token/get接口token字段
delegate	id	是	音视频回调代理, 代理协议 RCPlayerViewDelegate

- (instancetype)initWithUrl:(NSURL *)url ice:(NSString *)ice sn:(NSString *)sn direct:(NSInteger)direct fmt:(NSInteger)fmt videoWidth:(NSInteger)videoWidth videoHeight:(NSInteger)videoHeight fps:(NSInteger)fps bitrate:(NSInteger)bitrate cardWidth:(NSInteger)cardWidth cardHeight:(NSInteger)cardHeight cardDensity:(NSInteger)cardDensity token:(NSString)token delegate:(id)delegate;

[illegible]

```

direct:direct
fmt:fmt

videoWidth:videoWidth

videoHeight:videoHeight

fps:fps

bitrate:bitrate

cardWidth:0
cardHeight:0
cardDensity:0
token:token
delegate:self];

// websocket
[self startWebsocket];
self.playerView = playerView;
[self.view addSubview:playerView atIndex:0];
CGFloat height = SCREEN_W_RC*videoHeight/videoWidth;
[playerView mas_makeConstraints:^(MASConstraintMaker *make) {
    make.centerY.mas_equalTo(self.view.mas_centerY);
    make.left.right.mas_equalTo(0);
    make.height.mas_equalTo(height);
}];

// 系统工具条
RCSysmBar *systemBar = [RCSysmBar share];
self.systemBar = systemBar;
[self.view addSubview:systemBar aboveSubview:self.playerView];
[systemBar mas_makeConstraints:^(MASConstraintMaker *make) {
    make.top.mas_equalTo(self.playerView.mas_bottom);
    make.left.mas_equalTo(0);
    make.right.mas_equalTo(0);
    make.height.mas_equalTo(40);
}];

```

```
[self.systemBar.menuBtn addTarget:self
action:@selector(systemMenuBtnClick)
forControlEvents:UIControlEventTouchUpInside];

[self.systemBar.homeBtn addTarget:self
action:@selector(systemHomeBtnClick)
forControlEvents:UIControlEventTouchUpInside];

[self.systemBar.backBtn addTarget:self
action:@selector(systemBackBtnClick)
forControlEvents:UIControlEventTouchUpInside];

// 侧边栏
RCSyscView *syscView = [RCSyscView share];
self.syscView = syscView;
syscView.hidden = YES;
[self.view addSubview:syscView belowSubview:self.queryView];
[syscView mas_makeConstraints:^(MASConstraintMaker *make) {

make.top.mas_equalTo(self.playerView.mas_top).mas_offset(130+20);

make.left.mas_equalTo(self.playerView.mas_left).mas_offset(-10);
    make.width.mas_equalTo(100);
    make.height.mas_equalTo(60);
}];

[self.syscView.queryView addGestureRecognizer:
[[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(queryViewClick)]];

[self.syscView.endView addGestureRecognizer:
[[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(endViewClick)]];

[self.syscView addGestureRecognizer:[[UITapGestureRecognizer
alloc] initWithTarget:self action:@selector(expandViewClick)]];

// 侧边栏-遮挡
UIView *maskView = [[UIView alloc] init];
maskView.backgroundColor = [UIColor blackColor];
[self.view addSubview:maskView belowSubview:self.queryView];
```

```

[maskView mas_makeConstraints:^(MASConstraintMaker *make) {
    make.top.mas_equalTo(self.playerView.mas_top);
    make.bottom.mas_equalTo(self.playerView.mas_bottom);
    make.right.mas_equalTo(self.playerView.mas_left);
    make.width.mas_equalTo(300);
}];
}

```

III、方法：操作流程

1、切换主屏幕-重新连接 – 拉流

- (void)reConnectForUrl:(NSURL *)url ice:(NSString *)ice sn:(NSString *)sn token:(NSString *)token;

```

// 获取推流token
- (void)getPushToken:(NSString *)instanceID {

    NSString *deviceId = [[[UIDevice currentDevice]
    identifierForVendor] UUIDString];
    NSDictionary *params = @{@"instanceId": instanceID,
                              @"deviceId": deviceId};

    [[RCNetwork sharedInstance] postWithUrl:getPushTokenAPI
    Parameters:params success:^(id _Nonnull responseObject) {

        NSDictionary *responseDict = (NSDictionary *)responseObject;
        RCModel *model = [RCModel
        mj_objectWithKeyValues:responseDict];
        if (model.status == 0) {
            self.tokenModel.sn = instanceID;

            RCPushTokenModel *pushModel = [RCPushTokenModel
            mj_objectWithKeyValues:responseDict[@"data"]];

```

```

        self.tokenModel.pushModel = pushModel;
        NSLog(@"pushToken: %@", self.tokenModel.pushModel.token);

        [self getForwardAdds];

    } else {
        [RCToast hideLoading];
        [RCToast showToast:@"切换主屏失败"];
    }

} failure:^(NSError * _Nonnull error) {
    [RCToast hideLoading];
    [RCToast showToast:@"切换主屏失败"];
}];
}

- (void)getForwardAdds {

    NSString *url = self.tokenModel.pushModel.coturnInfo.coturnHttp;
    if ([url hasPrefix:@"http"]) {
        url = [NSString stringWithFormat:@"%@@%", url, getForwardAPI];
    } else {
        url = [NSString stringWithFormat:@"http://%@@%", url,
getForwardAPI];
    }

    [[RCNetwork sharedInstance] getWithUrl:url Parameters:nil
success:^(id _Nonnull responseObject) {

        NSDictionary *responseDict = (NSDictionary *)responseObject;
        NSString *IP = [NSString stringWithFormat:@"%@",
responseDict[@"ip"]];
        NSString *port = [NSString stringWithFormat:@"%@",
responseDict[@"port"]];
        self.tokenModel.forwardAdds = [NSString
stringWithFormat:@"%@:%@", IP, port];
    }];
}

```

```

        // 重新拉流
        NSString *urlString = [NSString stringWithFormat:@"http://%@",
self.tokenModel.pushModel.signalInfo.signalTcp];
        NSURL *url = [NSURL URLWithString:urlString];
        NSString *ice = self.tokenModel.forwardAdd;

        // 切换主屏幕 重新拉流
        [self.playerView reconnectForUrl:url ice:ice
sn:self.tokenModel.sn token:self.tokenModel.pushModel.token];
        [self stopWebsocket]; // 关闭Websocket
        dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(1 *
NSEC_PER_SEC)), dispatch_get_main_queue(), ^{
            [self startWebsocket]; // 重连Websocket
        });

    } failure:^(NSError * _Nonnull error) {
        [RCTToast hideLoading];
        [RCTToast showToast:@"切换主屏失败"];
    }];
}

```

2、主动断开连接

- (**void**)disconnect;

3、暂停接收流

- (**BOOL**)pauseStream;

4、恢复接收流

- (**BOOL**)resumeStream;

5、音频开关

- **(BOOL)**setAudioEnable:**(BOOL)**enable;

6、HD画质：自动码率和手动码率

- **(BOOL)**setAutoBitrateEnabled:**(BOOL)**enable bitrate:(NSNumber*)bitrate;

自动：

提高流畅性,降低延迟,画质会跟随网络变化而变化,网络差时会降低画质保流畅度,网络恢复时画质提升。

手动：

保证推流过程中画质稳定性,可以有效控制推流流量,网络差时会出现卡顿,丢包,高延迟,声音断续等。

7、设置渲染视频填充模式

- **(void)**setVideoContentMode:(UIViewContentMode)mode;

8、设置是否回调流信息状态

- **(void)**setShouldGetStats:**(BOOL)**shouldGetStats;

9、设置输入键盘类型：

1、设置真机输入法 2、设置云手机输入法

- **(void)**setKeyboardMode:**(int)**mode;

IV、方法：控制面板

1、设置视频类型 1是H264。2是vp8 5是H265 （需要重新拉流才生效）

- **(void)**setVideoType:**(int)**fmt;

2、拉流前调用设置软硬编码，YES硬编码，NO软编码，不调用默认硬编码

-**(void)**setVideoCodecHwAcceleration:**(BOOL)**videoCodecHwAcceleration;

3、拉流过程中改变视频分辨率

- **(BOOL)**setVideoSize:(NSInteger)videoWidth videoHeight:(NSInteger)videoHeight;

4、拉流过程中改变HD画质（默认值：流畅-800 标清-1500 高清-2200 超清-2800）

- **(BOOL)**setHDVaule:**(int)**value;

5、设置帧率（需要重新拉流才生效）

- **(void)**setMaxFps:(NSInteger)fps;

6、调整音量 0-减 1-加

- **(BOOL)**setVoice:(NSInteger)type;

7、Home键

- **(BOOL)**systemHomeBtnClick;

8、返回键

- **(BOOL)**systemBackBtnClick;

9、菜单键

- **(BOOL)**systemMenuBtnClick;

V、方法：Websocket

1、开启websocket

参数	类型	必选	说明
host	NSString	是	控制转发服务器 来源:/api/v2/instance/list接口publicIp字段
ip	NSString	是	实例ip 来源:/api/v2/instance/list接口ip字段
token	NSString	是	控制 token 来源:/api/v2/instance/control/token/get接口token字段

- **(void)**startWebsocketHost:(NSString *)host ip:(NSString *)ip token:(NSString *)token complete:(**(void (^)(BOOL))**)complete;

2、关闭websocket

- **(void)**stopWebsocket;

3、订阅主题与取消订阅主题

```

/// 订阅主题：屏幕亮度
- (void)subscribeBrightness;
/// 取消订阅主题：屏幕亮度
- (void)unsubscribeBrightness;

/// 订阅主题：剪贴板透传
- (void)subscribeClipboard;
/// 取消订阅主题：剪贴板透传
- (void)unsubscribeClipboard;

/// 订阅主题：屏幕方向
- (void)subscribeScreenOrientation;
/// 取消订阅主题：屏幕方向
- (void)unsubscribeScreenOrientation;

/// 订阅主题：分辨率变化
- (void)subscribeResolution;
/// 取消订阅主题：分辨率变化
- (void)unsubscribeResolution;

```

```

/// 订阅主题：创建快捷方式
- (void)subscribeAppShortcut;
/// 取消订阅主题：创建快捷方式
- (void)unsubscribeAppShortcut;

/// 订阅主题：音量变化
- (void)subscribeVolume;
/// 取消订阅主题：音量变化
- (void)unsubscribeVolume;
/**
 * 音量设置 streamType:音量类型 (0-5) percentage-音量百分比 (0-100)
 * streamType: 0 通话音量
 * streamType: 2 铃声音量
 * streamType: 3 媒体音量
 * streamType: 4 闹钟音量
 * streamType: 5 通知音量
 */
- (void)setVolumeWithStreamType:(uint)streamType percentage:
(uint)percentage;

/// 订阅主题：消息透传
- (void)subscribeAppMessage;
/// 取消订阅主题：消息透传
- (void)unsubscribeAppMessage;
/// 消息透传 - 发送消息
- (void)appMsgTransmission:(NSString *)packageName message:(NSString
*)message;

```

4、查询推流画质

```
- (void)queryStreamQuality;
```

5、查询分辨率

```
- (void)querySysResolution;
```

6、 查询横竖屏状态 (ws指令)

- (void)queryScreenOrientation;

7、 查询隐藏应用列表

- (void)queryAppHideList;

8、 查询所有应用列表

- (void)queryAppAllList;

9、 显示应用

- (void)showAppPackageNames:(NSArray *)packageNames;

10、 隐藏应用

- (void)hideAppPackageNames:(NSArray *)packageNames;

11、 打开应用

- (void)openAppPackageName:(NSString *)packageName;

12、 截图 (quality : 1~100)

- (void)screenshot:(int)quality;

13、 吹一吹 取值范围 5-10秒

- (void)sendBlowOrder:(int)time;

14、 摇一摇 取值范围 3-10秒

- (void)sendShakeOrder:(int)time;

15、 设置输入键盘类型：

1： 设置真机输入法 2： 设置云手机输入法

- **(void)**setInputMode:(**int**)inputMode;

16、 自定义内容透传给云机输入框

- **(void)**setKeyboardInfo:(NSString *)content;

17、 自定义内容透传给云机剪贴板

- **(void)**setClipboardInfo:(NSString *)content;

18、 隐藏导航栏

- **(void)**hideNavBar;

19、 显示导航栏

- **(void)**showNavBar;

20、 查询导航栏状态

- **(void)**queryNavBarStatus;

21、 查询屏幕状态 NO-竖屏 YES-横屏 (视频流横竖屏)

- **(BOOL)**queryScreenStatus;

22、 查询全局 root 状态

- **(void)**queryRootStatus;

23、 设置全局 root 状态

- **(void)**rootSystemStatus:(**BOOL**)isOpen;

24、 开启/关闭 传感器数据透传

- **(void)**openSensorData:(**BOOL**)isOpen;

VI、方法：其他

1、通过data channel发送字符串到云机

- (**BOOL**)sendData:(NSString*)strData;

2、通过data channel发送BYTE到云机

- (**BOOL**)sendBinary:(NSData*)data;

3、通过data channel发送键盘事件到云机

- (**BOOL**)sendKey:(**int**)value;

4、自定义拉流 WS 发送指令

- (**void**)sendWSOrder:(NSString *)order;

VII、代理RCPlayerViewDelegate：音视频状态回调

1、数据通道连接状态回调

- (**void**)dataChannelDidChangeFromPeerName:(NSString *)peerName State:
(RCDataChannelState)state;

2、音视频连接状态回调

- (**void**)onChangeConnectionStateFromPeerName:(NSString *)peerName
didChangeIceConnectionState:(RCIceConnectionState)state;

3、分辨率和横竖屏状态回调

- (**void**)onFrameResolutionChangedFromPeerName:(NSString *)peerName videoWidth:
(**int**)videoWidth videoHeight:(**int**)videoHeight rotation:(**int**)rotation;

4、音视频鉴权状态回调

- **(void)**onAuthResultFromPeerName:(NSString *)peerName code:(**int**)code descriptions:(NSString *)descriptions;

5、数据通道接收消息回调

- **(void)**onChannelDataFromPeerName:(NSString *)peerName buffer:(RCDataBuffer *)buffer;

6、音视频流数据 实时状态回调

- **(void)**didGetStats:(NSString *)peerName stats:(RCStatisticsReport *)stats;

7、音视频websocket 接收消息回调

- **(void)**webSocketCallBack:(RCWSType)type content:(NSDictionary *)content code:(**int**)code;

8、音视频 参数信息回调

- **(void)**videoParamsCallBack:(NSDictionary *)content;

9、SDK 错误信息提示 Toast

- **(void)**toastMsg:(NSString *)msg;

五、屏幕旋转：画面横屏适配

1、引入分类文件

UIInterface+HXRotation.h 和 UIInterface+HXRotation.m

2、屏幕旋转配置

AppDelegate.m 文件里

```
// 在这里写支持的旋转方向，为了防止横屏方向，应用启动时候界面变为横屏模式
- (UIInterfaceOrientationMask)application:(UIApplication *)application
supportedInterfaceOrientationsForWindow:(UIWindow *)window {
    return self.allowAutoRotate ? UIInterfaceOrientationMaskLandscape
: UIInterfaceOrientationMaskPortrait;
}
```

3、屏幕旋转代码

RCDevicePlayViewController.m 文件

```
#pragma mark - 屏幕旋转
- (void)setDeviceRotation:(BOOL)isRotation {
    NSLog(@"屏幕旋转: %d", isRotation);
    dispatch_async(dispatch_get_main_queue(), ^{
        AppDelegate *app = (AppDelegate *)[UIApplication
sharedApplication].delegate;
        app.allowAutoRotate = isRotation;

        UIInterfaceOrientation orientation = isRotation ?
UIInterfaceOrientationLandscapeRight :
UIInterfaceOrientationLandscapeRight;
        [self hx_rotateToInterfaceOrientation:orientation];

        CGFloat videoWidth = self.playerView.videoWidth;
        CGFloat videoHeight = self.playerView.videoHeight;
        if (videoWidth == 0 || videoHeight == 0) {
            NSString *selectResolution = self.tokenModel.videoWidth <
1080 ? @"720X1280": @"1080X1920";
            NSArray *array = [selectResolution.lowercaseString
componentsSeparatedByString:@"x"];
            videoWidth = [array.firstObject integerValue];
            videoHeight = [array.lastObject integerValue];
        }
    });
}
```



```

        CGFloat scale = MAX(videoWidth, videoHeight) / MIN(videoWidth,
videoHeight);

        NSLog(@"屏幕旋转 %f %f %f %f %f", SCREEN_W_RC, SCREEN_H_RC,
videoWidth, videoHeight, scale);

        if (isRotation) {
            CGFloat width = MIN(SCREEN_W_RC-40, SCREEN_H_RC-40)*scale;
            NSLog(@"屏幕旋转 width %f", width);
            [self.playerView
mas_remakeConstraints:^(MASConstraintMaker *make) {
                make.centerX.mas_equalTo(self.view.mas_centerX);
                make.top.mas_equalTo(0);
                make.bottom.mas_equalTo(-40);
                make.width.mas_equalTo(width);
            }];

            [self.systemBar mas_remakeConstraints:^(MASConstraintMaker
*make) {
                make.bottom.mas_equalTo(0);
                make.centerX.mas_equalTo(self.view.mas_centerX);
                make.width.mas_equalTo(width);
                make.height.mas_equalTo(40);
            }];

        } else {
            CGFloat height = MIN(SCREEN_W_RC, SCREEN_H_RC)*scale;
            NSLog(@"屏幕旋转 height %f", height);
            [self.playerView
mas_remakeConstraints:^(MASConstraintMaker *make) {
                make.centerY.mas_equalTo(self.view.mas_centerY);
                make.left.right.mas_equalTo(0);
                make.height.mas_equalTo(height);
            }];

            [self.systemBar mas_remakeConstraints:^(MASConstraintMaker
*make) {

```

```
        make.top.mas_equalTo(self.playerView.mas_bottom);  
        make.left.mas_equalTo(0);  
        make.right.mas_equalTo(0);  
        make.height.mas_equalTo(40);  
    }];  
}  
});  
}
```