帝视 iOS 端 SDK 编程指南

V1.0.0

声 明

非常感谢您购买我公司的产品，如果您有什么疑问或需 要请随时联系我们。

- 我们已尽量保证手册内容的完整性与准确性，但也不免出现 技术上不准确、与产品功能及操作不相符或印刷错误等情况， 如有任何疑问或争议，请以我司最终解释为准。
- 产品和手册将实时进行更新，恕不另行通知。
- 本手册中内容仅为用户提供参考指导作用，请以 SDK 实际内容为准。

SDK 简介

帝视是基于 IOT 领域的视频解决方案。iOS 端 SDK 可以将 IPC、NVR、猫眼、门铃等产品的音视频数据，通过网络拉

取数据到 iPhone 设备上进行展示播放。用户可以使用该 SDK 进行二次开发。

SDK 版本更新

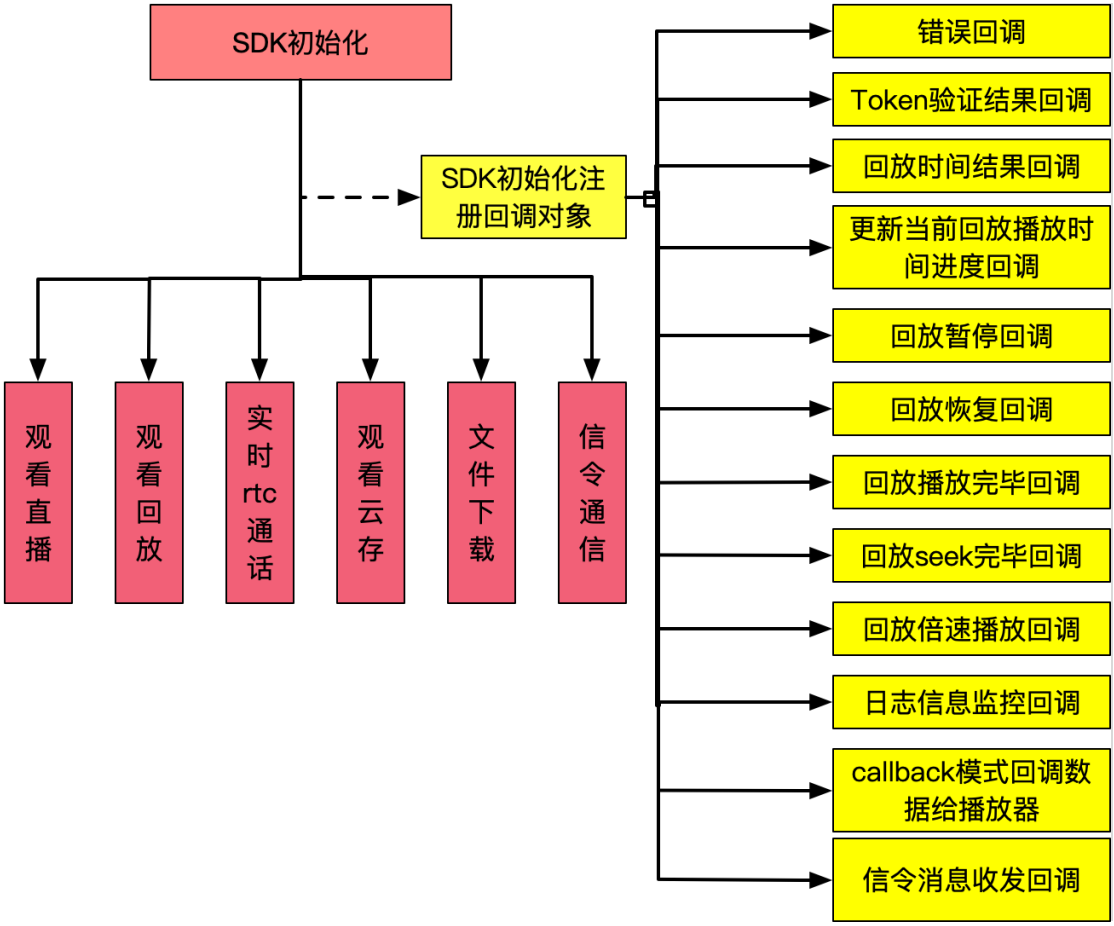
编程引导

- SDK 集成
  - SDK 下载

    iOS SDK 下载地址为:
  - demo 下载

    iOS demo 下载地址为：
  - Xcode 工程配置

- 函数调用流程
  - SDK 接口调用函数主要流程



图中黄色背景部分是 SDK 时注册实现的函数回调，包括错误回调、Token 验证结果回调、回放时间结果回调、更新当前回放播放时间进度回调、回放暂停回调、回放恢复回调、回放播放完毕回调、回放 seek 完毕回调、回放倍速播放回调、日志信息监控回调、callback 模式回调数据给播放器、信令消息收发回调。

功能模块简单解释如下：

- 初始化 SDK：使用帝视功能时，首先需要调用：

startLocalServer:cacheSize:options,SDK 会在本地启动一个代理服务，同时设置回调代理对象 godSeesDelegate.SDK 对信令消息提供两种通道：

a） QHVCNetLongConnectServiceSDK（SDK 自带长连，默认值）

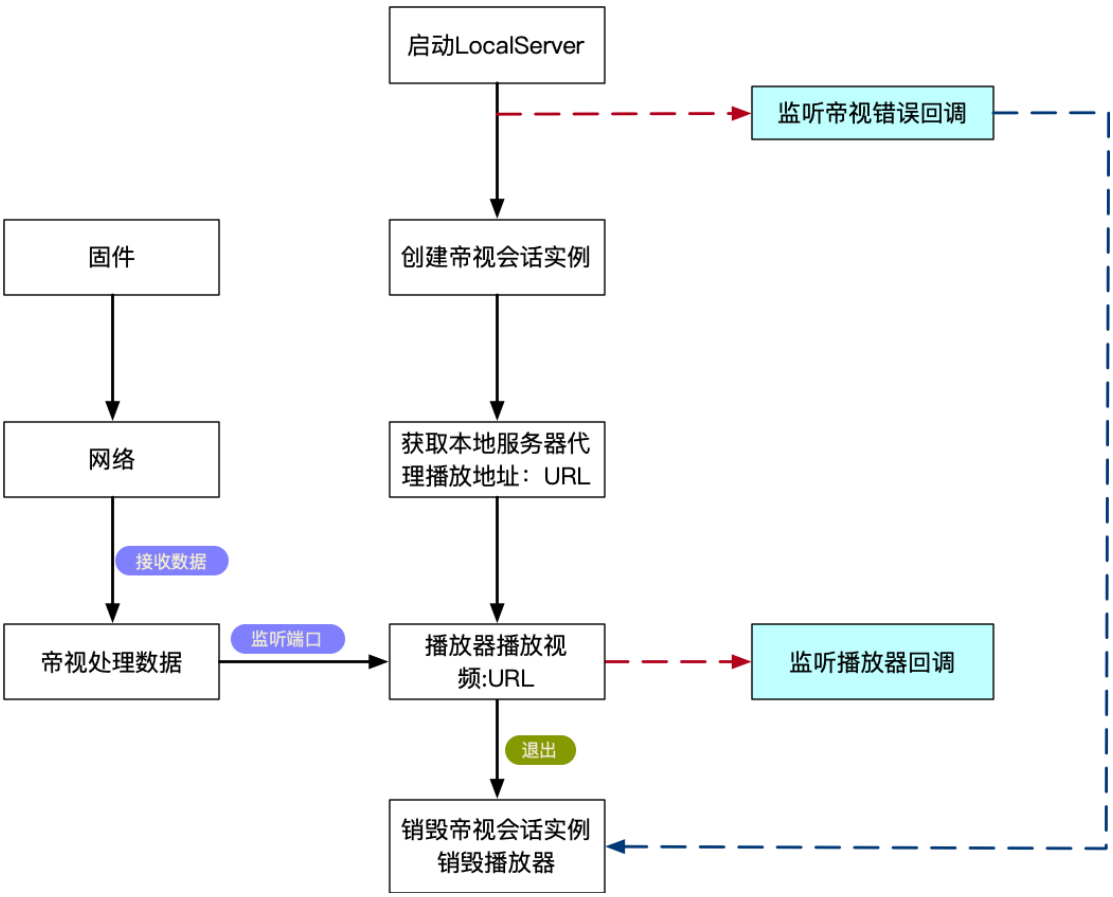
b） QHVCNetLongConnectServiceExternal（外部长连）

实际使用中业务可以根据实际情况设置不同的启动参数，具体的参数在 startLocalServer 的 options 传进来，具体使用方法见具体 API 函数注释。

- 观看直播：拉取固件端实时音视频数据通过网络传输在本地进行播放。
- 观看回放：从固件端拉取指定时间点开始的音视频数据通过网络传输到本地进行播放，支持 seek 操作、倍速播放等。
- 实时 RTC 通话：可与固件端进行实时通话，在同一通会话中，支持移动端之间相互音视频通话。
- 观看云存：用户如果选择把回放数据存放在云端，可以从云端拉取回放数据到本地播放。
- 文件下载：支持从指定设备上下载已经存在的图片、短视频资源。

- 信令通道：SDK 在工作的过程中，与固件端需要频繁的交互信息，SDK 提供两种信令通道的选择，即：外部业务长连或者 SDK 内部长连，默认值为 SDK 内部长连，如果需要使用业务自己的信令通道，需要在启动服务时显示设置。

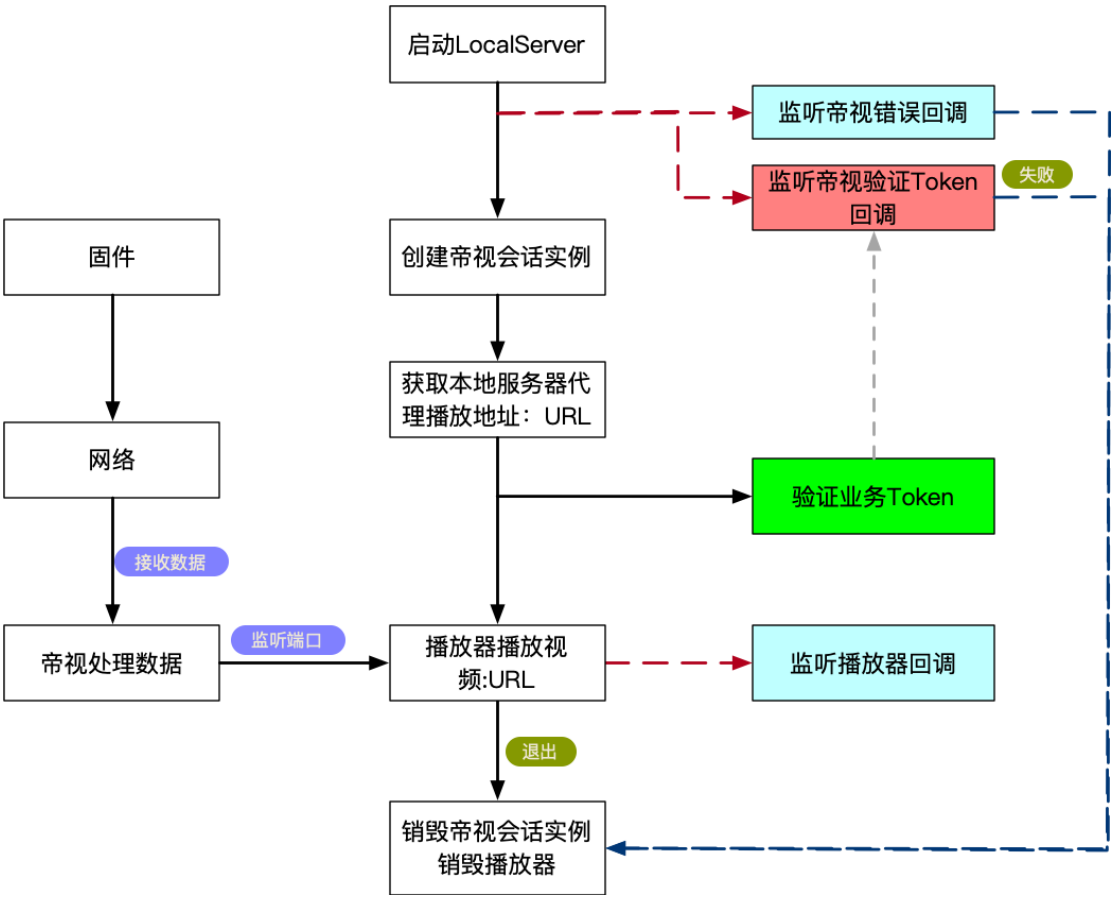
◆ 观看直播流程



帝视 SDK 运行依赖于一个本地服务器，启动后，创建帝视会

话实例，把设备号传给 SDK，SDK 给返回一个本地代理服务器地址给播放器并初始化播放器进行拉流播放，帝视 SDK 就会去指定的固件端拉取音视频数据给播放器播放。

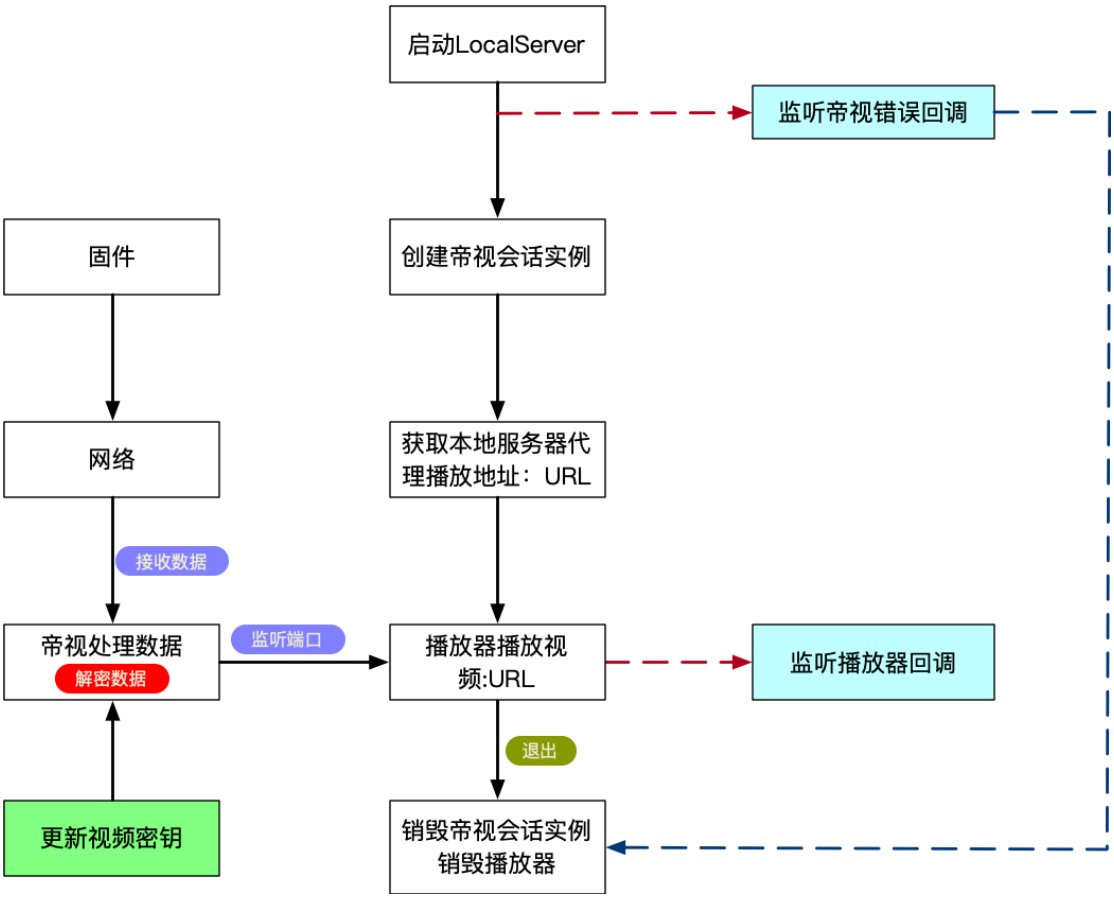
◆ 观看带 token 校验直播流程



固件端的流，需要业务用户身份合法才能观看，因此 SDK 设计了业务 Token 验证流程，其验证逻辑是在播放器初始化的同时，添加了异步 Token 验证流程，Token 由业务提供，帝视 SDK 会将 Token 传输到固件端，固件端验证成功后把结

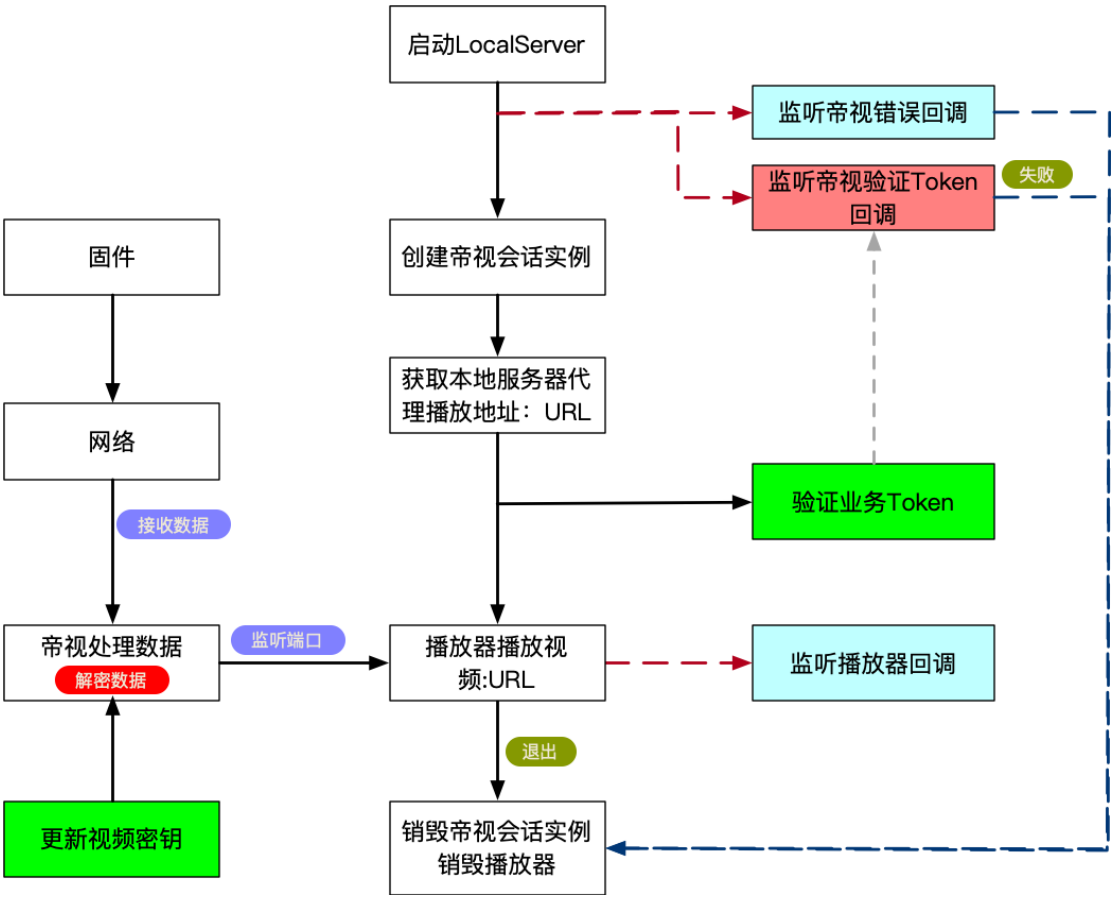
果返回并通过帝视验证 Token 回调函数通知结果，如果成功就开始推流，失败就拒绝推流。

◆ 观看加密直播流程



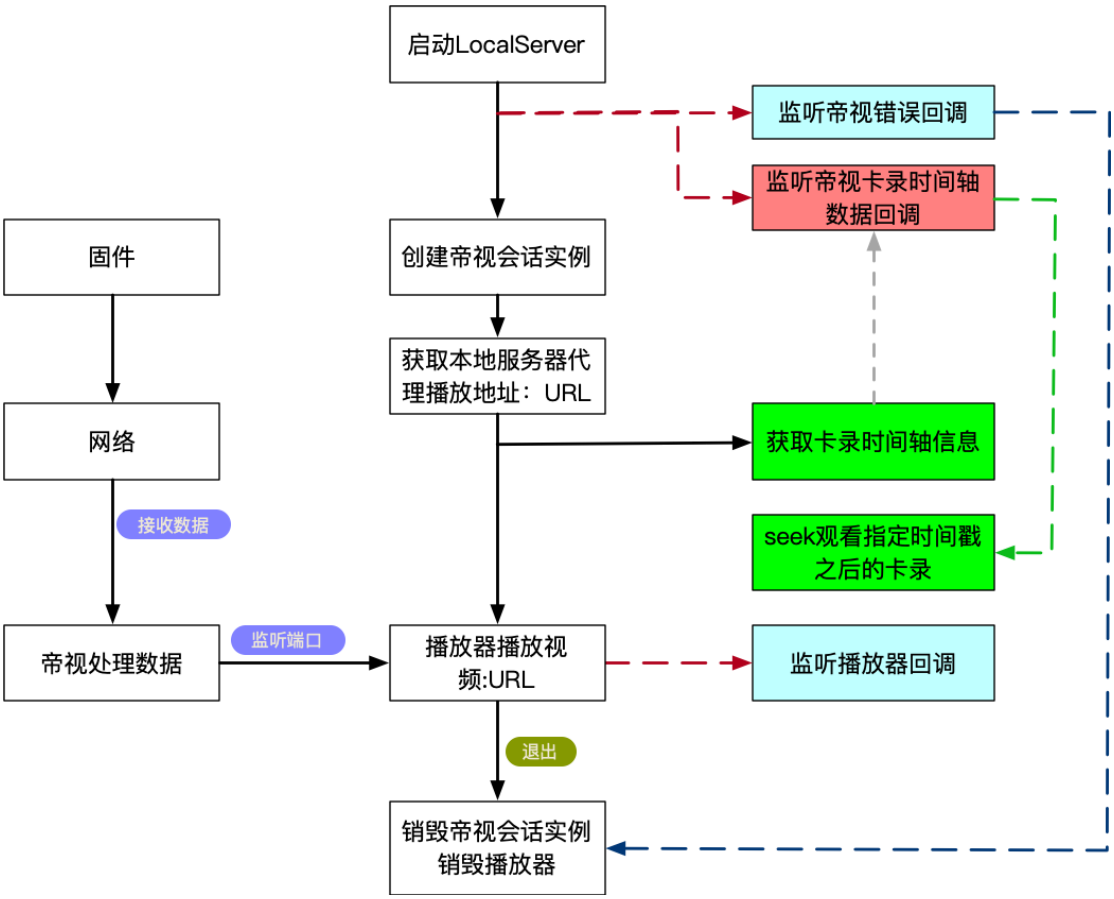
视频在网络传输的过程中需要保证视频内容的绝对安全，因此固件端会对每帧视频数据进行加密传输，客户端接收数据后对应的进行解密，解密后再把数据传给播放器播放。视频密钥由业务提供，业务自己保证密钥的安全性。

◆ 观看带 token 校验加密直播流程



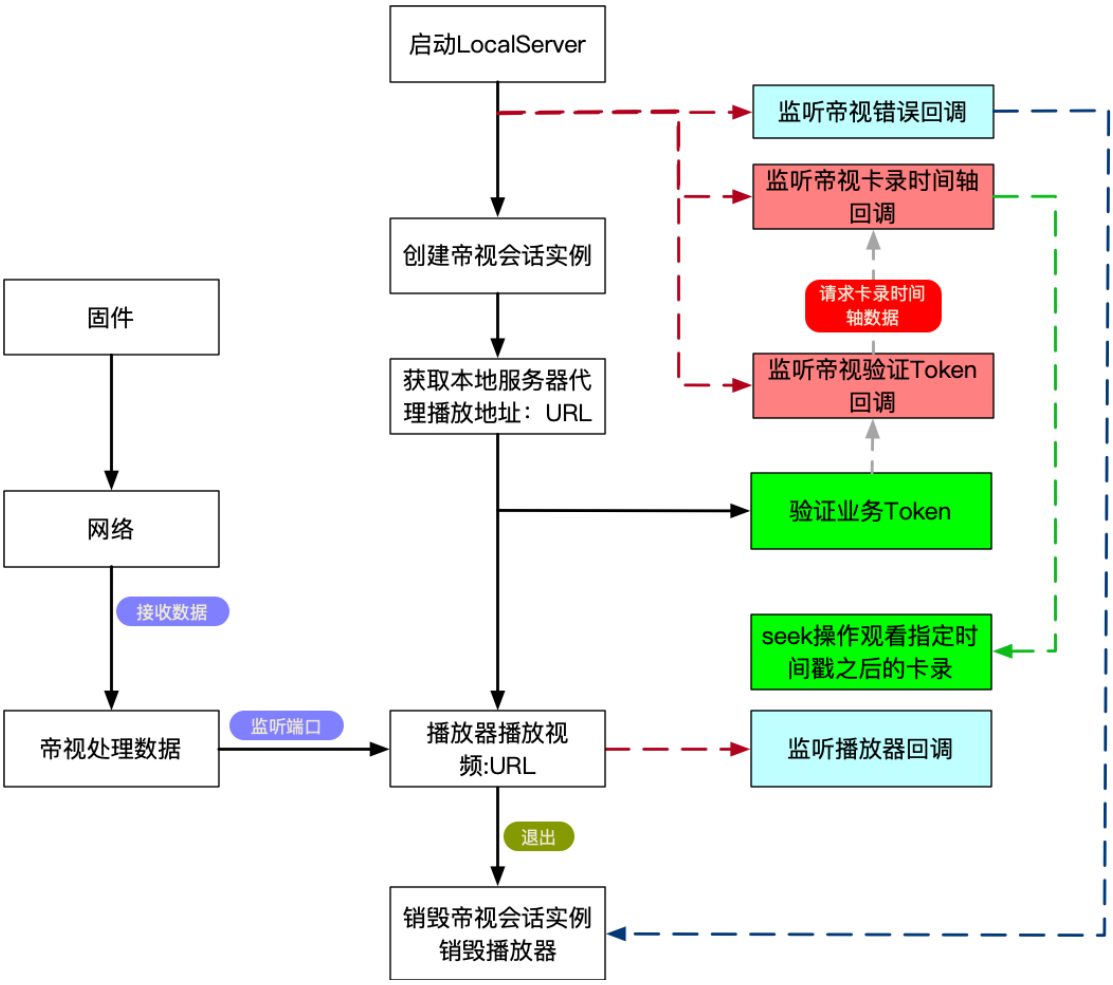
首先验证业务 Token，Token 验证成功后开始推流，固件端对视频数据加密传输，客户端接收数据后，业务把密钥传递给 SDK，SDK 解密后给播放器进行播放。

- 观看回放流程



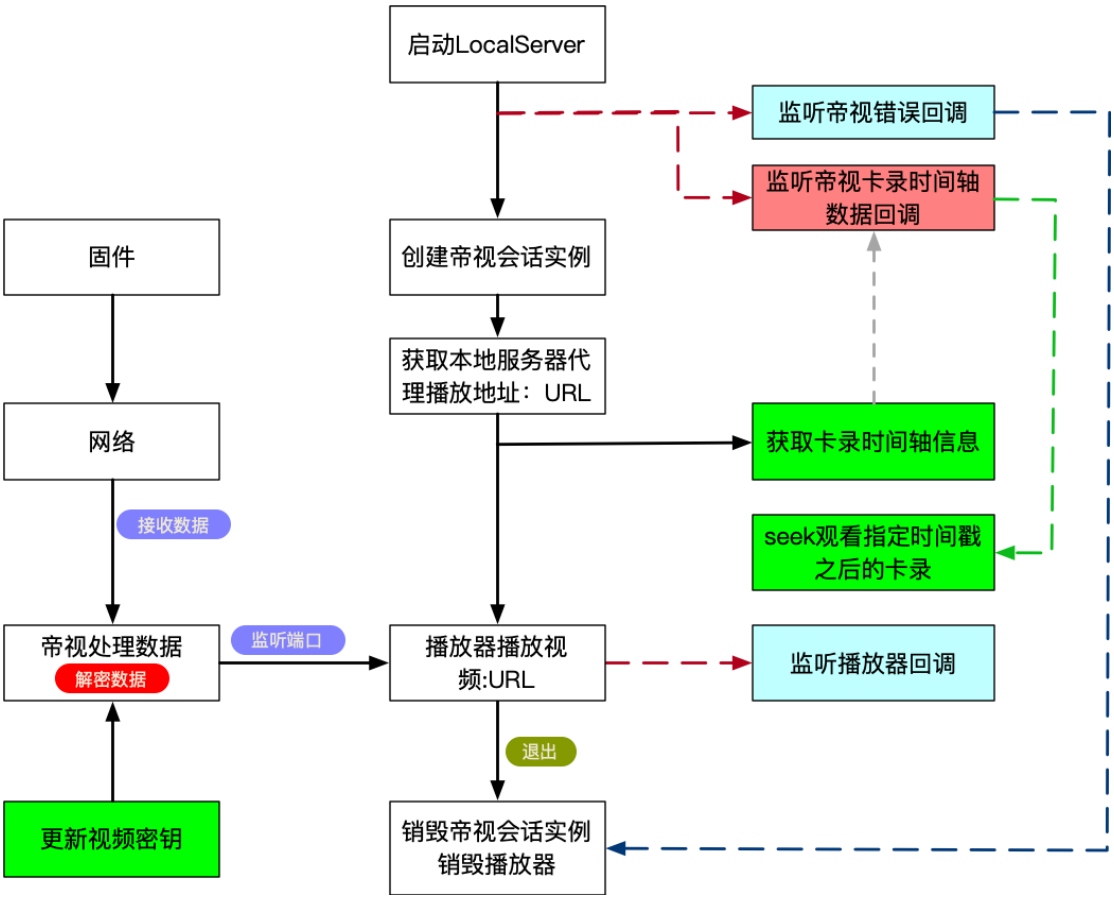
初始化播放器的同时，调用帝视获取卡录时间轴，待时间轴数据返回后，选择一个时间点执行 seek 操作，固件将会从指定时间点之后进行推送音视频数据。

- 观看带 token 校验回放流程



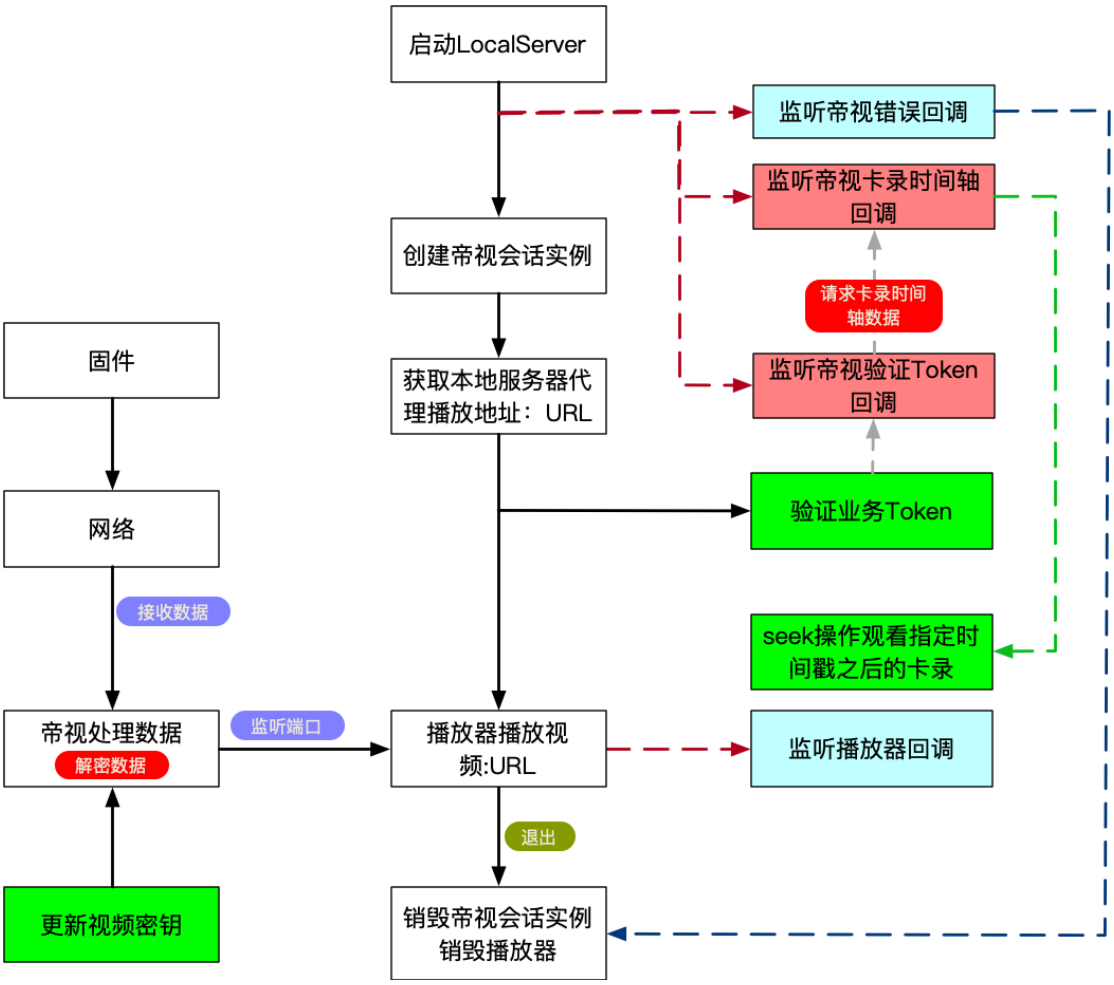
在观看回放的基础上，需要先验证用户身份，待身份验证成功后再发起观看回放流程。

- 观看加密回放流程

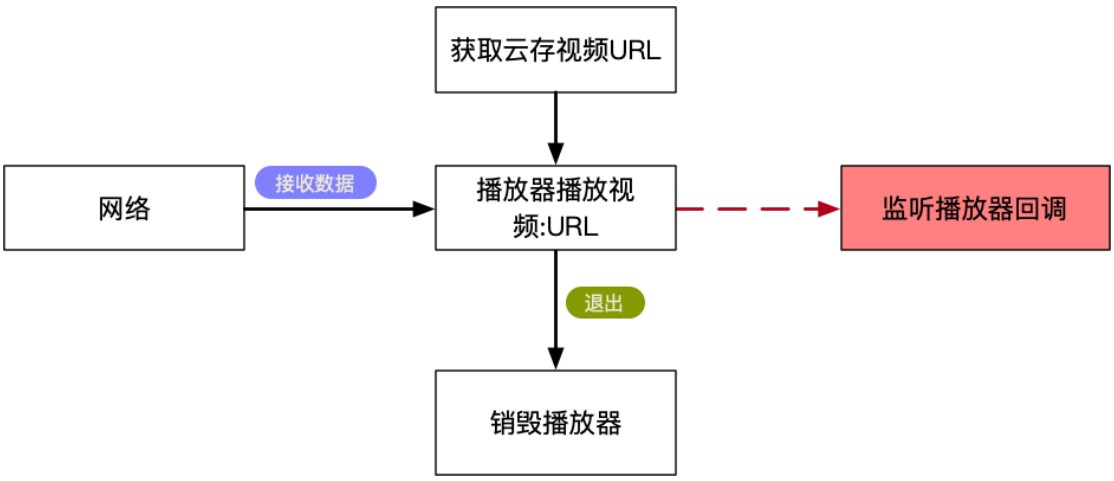


在观看回放的基础上，把当前视频的密钥给 SDK，SDK 解密后传递给播放器进行播放，业务需要保证密钥的正确性和安全性。

- 观看带 token 校验加密回放流程



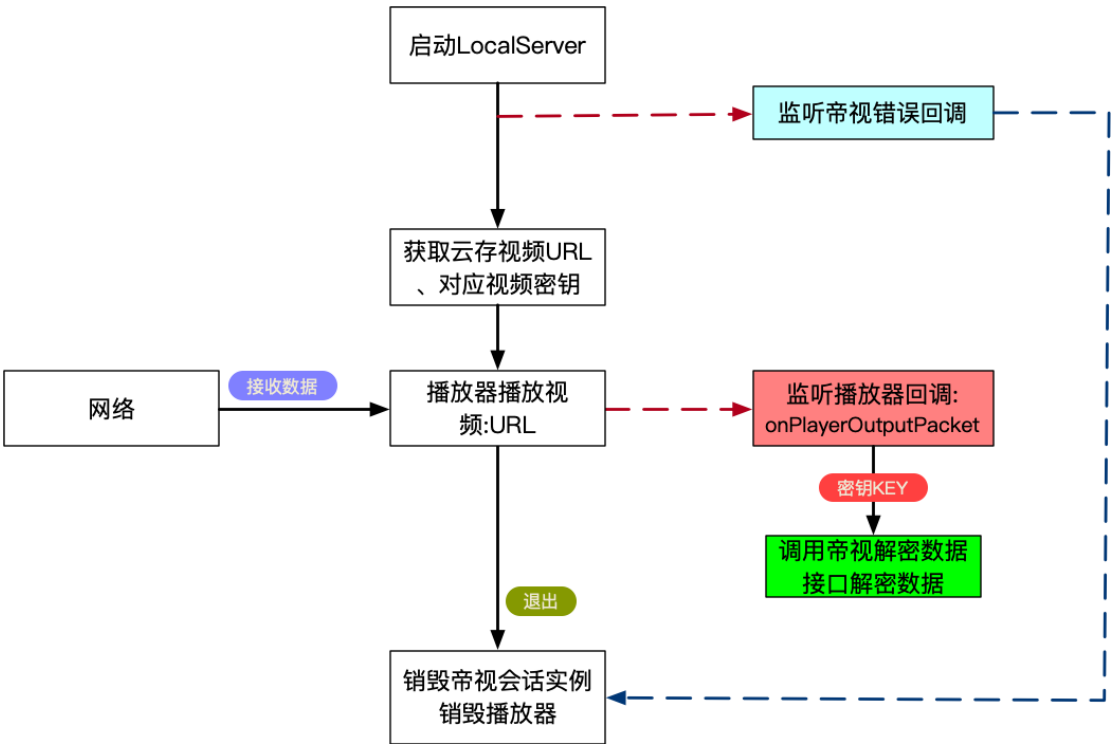
◆ 观看云存流程



一段回放视频数据被上传到云端，回放生成完毕后，帝视服务器会把云存地址通知给业务服务器，业务服务器把地址

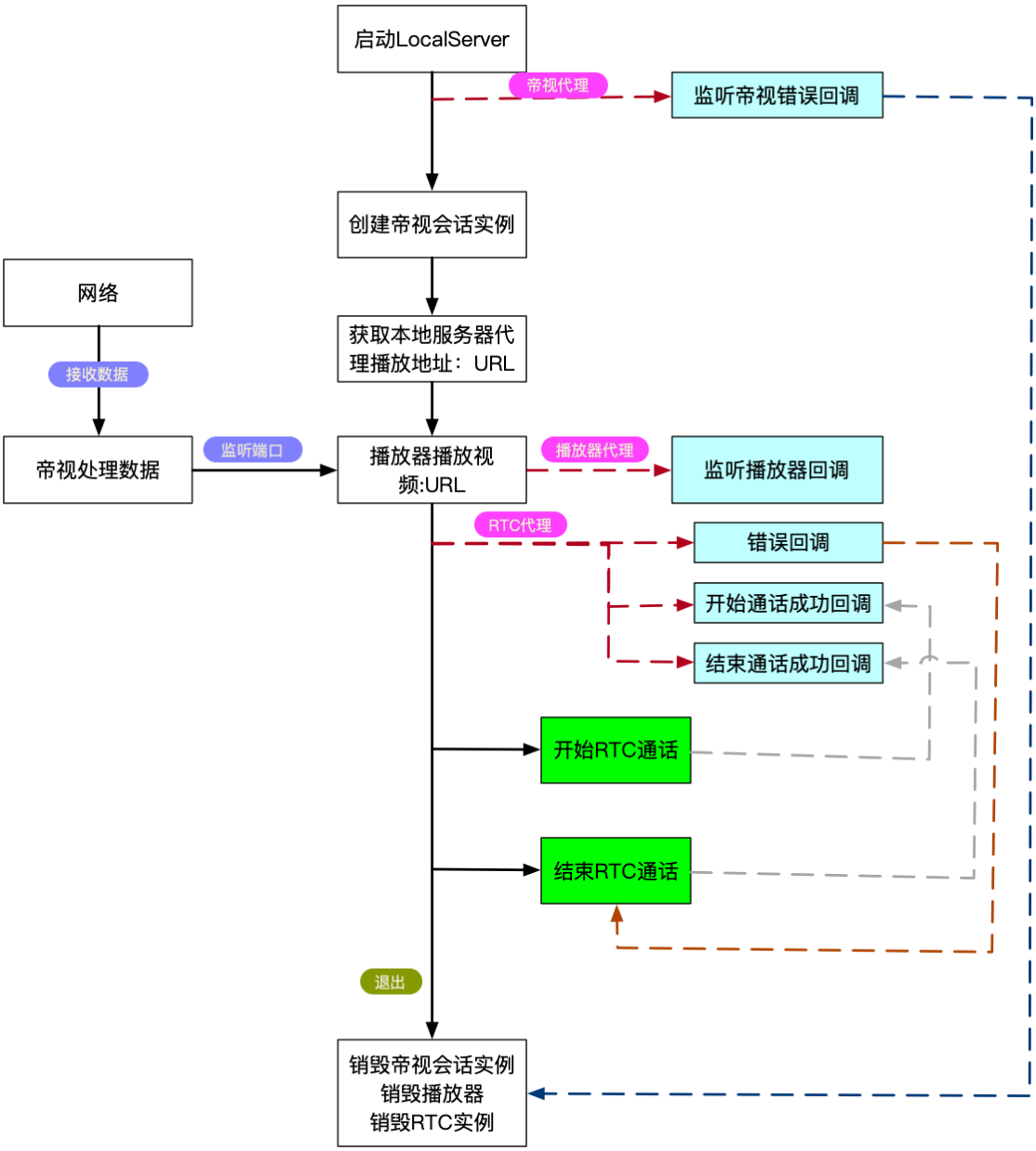
下发给观看端，观看端直接使用云存地址初始化播放器进行拉流观看。

◆ 观看加密云存流程



播放器在接收到原始视频数据包后，监听相关回调接口，在回调接口用事先拿到的密钥调用帝视提供的解密接口进行数据解密。

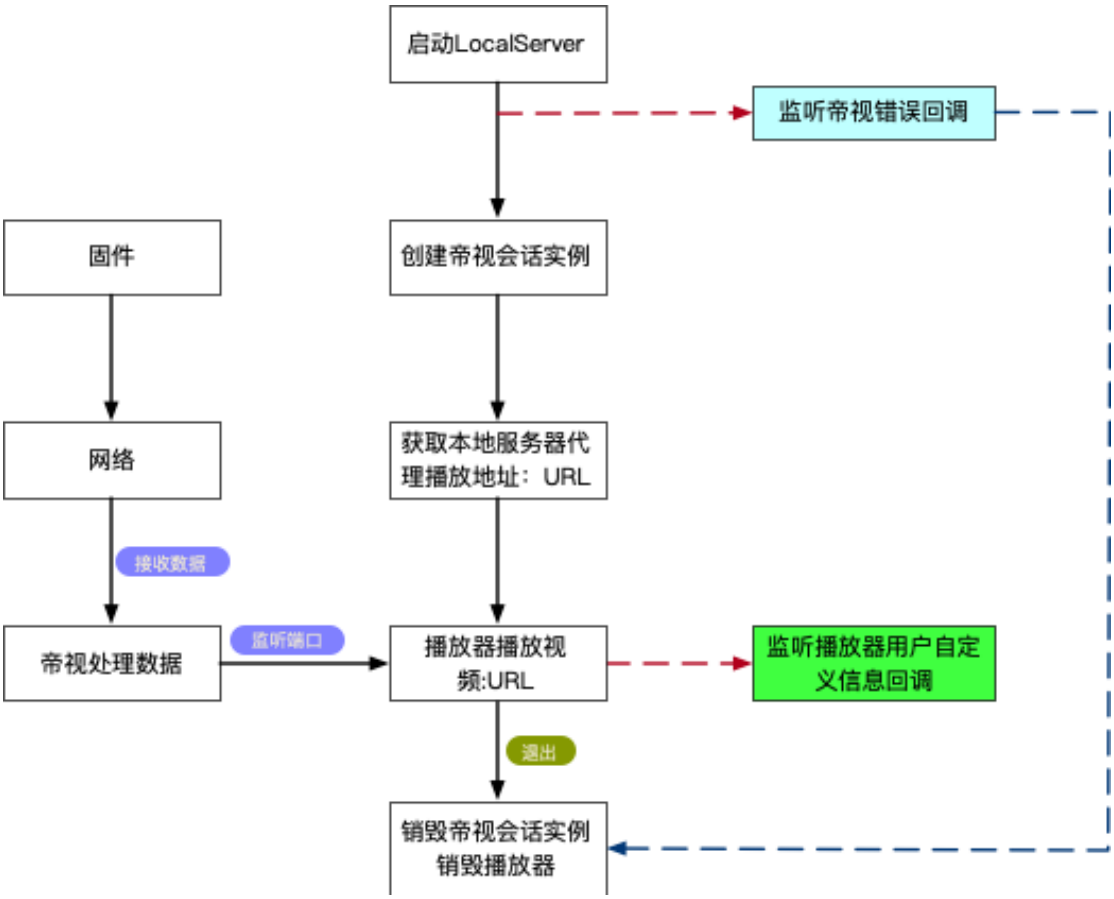
◆ 实时 RTC 通话流程



在观看直播的过程中，调用互动直播 SDK 中的高级接口：

`startAutomaticConversationWithDelegate` 即可进行 RTC 通话，通话成功后会回调开始通话成功回调，就可以和固件进行 RTC 通话，如果要结束通话，就调用 `stopAutomaticConversation` 接口进行结束通话，待结束通话成功回调成功后，就可以正常结束 RTC 通话。

- 获取视频流附加数据流程



视频数据在传输或者展示等各个环节有缓存，业务需要根据画面实际时间展示各种 AI 信息等，需要在每一帧上携带准确的信息，这个信息通过在 SEI 中携带，SDK 支持在视频数据帧中携带用户自定义的各种信息。播放器在接收到每帧数据中解析出来并通过用户自定义信息回调给业务进行处理。

- 文件下载流程

如果要观看固件上存储的小视频或者图片，可以调用帝视 `getDeviceFileDownloadUrlWithFileKey` 接口获取文件的完整下载地址，业务拿到这个下载地址即可进行自行下载。

■ SDK 使用实例

这些功能中，都需要事先启动 LocalServer，启动 LocalServer 代码如下：

```
QHVCNet* netkit = [QHVCNet sharedInstance];
[netkit setLogLevel:(QHVCNetLogLevel)[QHVCLogger
getLoggerLevel] detailInfo:0 callback:^(const char *buf,
size_t buf_size) {
      NSLog(@"NetSDK netLog:%@", [NSString
stringWithUTF8String:buf]);
   }];
NSString *path =
[NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES) firstObject];
path = [path
stringByAppendingPathComponent:@"videoCache"];
```

```
    NSFileManager *fileManager = [NSFileManager
defaultManager];
    if (![fileManager fileExistsAtPath:path]) {
        [fileManager createDirectoryAtPath:path
withIntermediateDirectories:YES attributes:nil
error:nil];
    }
    [netkit startLocalServer:path
                 cacheSize:500

options:@{kQHVCNetOptionsServicesKey:@(QHVCNetServiceGodS
ees | QHVCNetServicePrecache)}];
```

停止 LocalServer 代码如下：

```
[[QHVCNet sharedInstance] stopLocalServer];
```

- 观看直播实例代码

创建帝视会话实例：

```
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    [[QHVCNet sharedInstance] setGodSeesDelegate:self];
    _sessionId = [NSString stringWithFormat:@"%@_%@_%lld",
globalConfig.appId, config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
    [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
    [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
```

```
   NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN

deviceId:globalConfig.deviceId

appKey:globalConfig.appSecret];
   [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId

clientId:globalConfig.deviceId

appId:globalConfig.appId

userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
                         deviceChannelNumber:1
                               businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_LIVE

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongCon
nectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
_playerUrl = [[QHVCNet sharedInstance]
getGodSeesPlayUrl:_sessionId];
```

播放器播放视频：

```
    [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig* config = [QHVCGVConfig sharedInstance];
    int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                  userId:config.userName
                                playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency),@"in
putStream":@(inputStreamValue),@"streamType":@(config.pla
yerStreamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
```

```
}
```

◆ 观看带 token 校验直播实例代码

创建帝视会话实例：

```
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    [[QHVCNet sharedInstance] setGodSeesDelegate:self];
    _sessionId = [NSString stringWithFormat:@"%@_%@_%lld",
globalConfig.appId, config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
    [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
    [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
    NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN

deviceId:globalConfig.deviceId

appKey:globalConfig.appSecret];
    [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId

clientId:globalConfig.deviceId

appId:globalConfig.appId
```

```
userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
                          deviceChannelNumber:1
                                businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_LIVE

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongCon
nectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
_playerUrl       =       [[QHVCNet       sharedInstance]
getGodSeesPlayUrl:_sessionId];
```

验证业务 token:

```
[[QHVCNet sharedInstance]
verifyGodSeesBusinessTokenToDevice:_sessionId

token:[QHVCGVUserSystem sharedInstance].userInfo.token];
```

播放器播放视频：

```
   [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
   QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
   QHVCGVConfig* config = [QHVCGVConfig sharedInstance];
   int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
```

```
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                    userId:config.userName
                                  playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency),@"in
putStream":@(inputStreamValue),@"streamType":@(config.pla
yerStreamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

监听业务 token 验证回调方法：

```
- (void) onGodSees:(NSString *)sessionId
didVerifyToken:(NSInteger)status info:(NSString *)info {
    [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"onGodsees
didVerifyToken: sessionId = %@, status = %zd, info
= %@",sessionId,status,info]];
    if ([_sessionId isEqual:sessionId]) {
```

```
        if (status ==
QHVC_NET_GODSEES_TOKEN_ERROR_NO_ERROR) {
            return;
        }
        runOnMainQueueWithoutDeadlocking(^{
            [self stopPlayWhenException];//结束播放
        });
    }
}
```

◆ 观看加密直播实例代码

创建帝视会话实例：

```
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    [[QHVCNet sharedInstance] setGodSeesDelegate:self];
    _sessionId = [NSString stringWithFormat:@"%@_%@_%lld",
globalConfig.appId, config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
    [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
    [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
    NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN

deviceId:globalConfig.deviceId
```

```
appKey:globalConfig.appSecret];
    [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId

clientId:globalConfig.deviceId

appId:globalConfig.appId

userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
                          deviceChannelNumber:1
                                businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_LIVE

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongCon
nectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
_playerUrl = [[QHVCNet sharedInstance]
getGodSeesPlayUrl:_sessionId];
```

播放器播放视频：

```
    [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig* config = [QHVCGVConfig sharedInstance];
```

```
    int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                    userId:config.userName
                                  playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency),@"in
putStream":@(inputStreamValue),@"streamType":@(config.pla
yerStreamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

更新当前设备密钥：

```
- (void)setCurrentWatchingSN:(NSString
*)currentWatchingSN {
    _currentWatchingSN = currentWatchingSN;
    if (currentWatchingSN != nil) {
```

```
        NSArray *tmpArray = [_streamPwds copy];
        [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"秘钥 - 更新
serialNumer:%@ 的秘钥",currentWatchingSN]];
        for (QHVCGVStreamPasswordModel *pwdModel in
tmpArray) {
            if ([pwdModel.sn
isEqualToString:currentWatchingSN]) {
                [[QHVCNet sharedInstance]
updateGodSeesVideoStreamSecurityKeys:pwdModel.pwds
serialNumber:pwdModel.sn];
                [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"秘钥 - 找到秘钥，已更
新:%@",pwdModel.pwds]];
            }
        }
    }
}
```

◆ 观看带 token 校验加密直播实例代码

创建帝视会话实例：

```
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    [[QHVCNet sharedInstance] setGodSeesDelegate:self];
    _sessionId = [NSString stringWithFormat:@"%@_%@_%lld",
globalConfig.appId, config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
    [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
```

```
    [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
    NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN

deviceId:globalConfig.deviceId

appKey:globalConfig.appSecret];
    [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId

clientId:globalConfig.deviceId

appId:globalConfig.appId

userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
                          deviceChannelNumber:1
                                businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_LIVE

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongCon
nectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
_playerUrl = [[QHVCNet sharedInstance] getGodSeesPlayUrl:_sessionId];
```

验证业务 token:

```
[[QHVCNet sharedInstance] verifyGodSeesBusinessTokenToDevice:_sessionId

token:[QHVCGVUserSystem sharedInstance].userInfo.token];
```

播放器播放视频：

```
    [QHVCPlayer setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger getLoggerLevel]];
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig sharedInstance];
    QHVCGVConfig* config = [QHVCGVConfig sharedInstance];
    int inputStreamValue = [config playerReceiveDataModel] == QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                  userId:config.userName
                                playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency),@"inputStream":@(inputStreamValue),@"streamType":@(config.playerStreamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
```

```
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

监听业务 token 验证回调方法：

```
- (void) onGodSees:(NSString *)sessionId
didVerifyToken:(NSInteger)status info:(NSString *)info {
    [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"onGodsees
didVerifyToken: sessionId = %@, status = %zd, info
= %@",sessionId,status,info]];
    if ([_sessionId isEqual:sessionId]) {
        if (status ==
QHVC_NET_GODSEES_TOKEN_ERROR_NO_ERROR) {
            return;
        }
        runOnMainQueueWithoutDeadlocking(^{
            [self stopPlayWhenException];//结束播放
        });
    }
}
```

更新当前设备密钥：

```
- (void)setCurrentWatchingSN:(NSString
*)currentWatchingSN {
    _currentWatchingSN = currentWatchingSN;
    if (currentWatchingSN != nil) {
        NSArray *tmpArray = [_streamPwds copy];
```

```
        [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"秘钥 - 更新
serialNumer:%@ 的秘钥",currentWatchingSN]];
        for (QHVCGVStreamPasswordModel *pwdModel in
tmpArray) {
            if ([pwdModel.sn
isEqualToString:currentWatchingSN]) {
                [[QHVCNet sharedInstance]
updateGodSeesVideoStreamSecurityKeys:pwdModel.pwds
serialNumber:pwdModel.sn];
                [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"秘钥 - 找到秘钥，已更
新:%@",pwdModel.pwds]];
            }
        }
    }
}
```

- 观看回放实例代码

创建帝视会话实例：

```
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    [[QHVCNet sharedInstance] setGodSeesDelegate:self];
    _sessionId = [NSString
stringWithFormat:@"%@_%@_%lld",globalConfig.appId,
config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
    [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
```

```
    [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
    NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN

deviceId:globalConfig.deviceId

appKey:globalConfig.appSecret];
    [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId

clientId:globalConfig.deviceId

appId:globalConfig.appId

userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
                          deviceChannelNumber:1
                                businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_RECORD

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongCon
nectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
    _playerUrl      =      [[QHVCNet      sharedInstance]
getGodSeesPlayUrl:_sessionId];
```

播放器播放视频：

```
    [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    QHVCGlobalConfig * globalConfig = [QHVCGlobalConfig
sharedInstance];
    int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                  userId:config.userName
                                playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency|QHVCP
layModeLive_IOT),@"max_buffering_delay":@(600),@"inputStr
eam":@(inputStreamValue),@"streamType":@(config.playerStr
eamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
```

```
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

获取卡录时间轴信息：

```
[[QHVCNet sharedInstance]
getGodSeesRecordTimeline:_sessionId startTime:0 endTime:-
1];
```

监听帝视卡录时间轴信息回调：

```
- (void) onGodSees:(NSString *)sessionId
didRecordTimeline:(NSArray<QHVCNetGodSeesRecordTimeline
*> *)data {
    if ([_sessionId isEqual:sessionId]) {
        [self analysisCardRecordTimeline:data];
    }
}
```

seek 观看指定时间戳之后的卡录：

```
[[QHVCNet sharedInstance] setGodSeesRecordSeek:_sessionId
timeStamp:self.currentSeekTime];
```

监听更新当前播放回放时间进度值：

```
-       (void)       onGodSees:(NSString       *)sessionId
didRecordUpdateCurrentTimeStamp:(NSUInteger)timeStampByMS
{
 //更新 UI
}
```

- 观看带 token 校验回放实例代码

创建帝视会话实例：

```
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    [[QHVCNet sharedInstance] setGodSeesDelegate:self];
    _sessionId = [NSString
stringWithFormat:@"%@_%@_%lld",globalConfig.appId,
config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
    [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
    [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
    NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN

deviceId:globalConfig.deviceId

appKey:globalConfig.appSecret];
    [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId

clientId:globalConfig.deviceId

appId:globalConfig.appId

userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
```

```
                            deviceChannelNumber:1
                                 businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_RECORD

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongCon
nectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
    _playerUrl      =      [[QHVCNet      sharedInstance]
getGodSeesPlayUrl:_sessionId];
```

验证业务 token:

```
[[QHVCNet                                  sharedInstance]
verifyGodSeesBusinessTokenToDevice:_sessionId
token:[QHVCGVUserSystem sharedInstance].userInfo.token];
```

播放器播放视频：

```
    [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    QHVCGlobalConfig * globalConfig = [QHVCGlobalConfig
sharedInstance];
    int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                    userId:config.userName
                                 playType:QHVCPlayTypeLive
```

```
options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency|QHVCP
layModeLive_IOT),@"max_buffering_delay":@(600),@"inputStr
eam":@(inputStreamValue),@"streamType":@(config.playerStr
eamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

监听帝视验证 token 回调：

```
- (void) onGodSees:(NSString *)sessionId
didVerifyToken:(NSInteger)status info:(NSString *)info
{
    [QHVCLogger printLogger:QHVC_LOG_LEVEL_TRACE
content:[NSString stringWithFormat:@"didVerifyToken:
sessionId = %@, status = %zd, info
= %@",sessionId,status,info]];
    if ([_sessionId isEqual:sessionId])
    {
        if (status == QHVCNetErrorCodeNoError)
        {
```

```
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
            [[QHVCNet sharedInstance] getGodSeesRecordTimeline:_sessionId startTime:0 endTime:-1];
        });
        return;
    }
  }
}
```

在验证 token 成功后获取卡录时间轴信息：

```
[[QHVCNet sharedInstance] getGodSeesRecordTimeline:_sessionId startTime:0 endTime:-1];
```

监听帝视卡录时间轴信息回调：

```
- (void) onGodSees:(NSString *)sessionId didRecordTimeline:(NSArray<QHVCNetGodSeesRecordTimeline *> *)data {
  if ([_sessionId isEqual:sessionId]) {
      [self analysisCardRecordTimeline:data];
  }
}
```

seek 观看指定时间戳之后的卡录：

```
[[QHVCNet sharedInstance] setGodSeesRecordSeek:_sessionId timeStamp:self.currentSeekTime];
```

监听更新当前播放回放时间进度值：

```
-      (void)      onGodSees:(NSString      *)sessionId didRecordUpdateCurrentTimeStamp:(NSUInteger)timeStampByMS
{
```

```
 //更新 UI
}
```

◆ 观看加密回放实例代码

的

创建帝视会话实例：

```
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    [[QHVCNet sharedInstance] setGodSeesDelegate:self];
    _sessionId = [NSString
stringWithFormat:@"%@_%@_%lld",globalConfig.appId,
config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
    [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
    [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
    NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN

deviceId:globalConfig.deviceId

appKey:globalConfig.appSecret];
    [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId
```

```
clientId:globalConfig.deviceId

appId:globalConfig.appId

userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
                          deviceChannelNumber:1
                               businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_RECORD

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongCon
nectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
    _playerUrl      =      [[QHVCNet      sharedInstance]
getGodSeesPlayUrl:_sessionId];
```

播放器播放视频：

```
    [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    QHVCGlobalConfig * globalConfig = [QHVCGlobalConfig
sharedInstance];
    int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl
```

```
channelId:globalConfig.appId
                                    userId:config.userName
                                  playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency|QHVCP
layModeLive_IOT),@"max_buffering_delay":@(600),@"inputStr
eam":@(inputStreamValue),@"streamType":@(config.playerStr
eamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

获取卡录时间轴信息：

```
[[QHVCNet sharedInstance]
getGodSeesRecordTimeline:_sessionId startTime:0 endTime:-
1];
```

监听帝视卡录时间轴信息回调：

```
- (void) onGodSees:(NSString *)sessionId
didRecordTimeline:(NSArray<QHVCNetGodSeesRecordTimeline
*> *)data {
    if ([_sessionId isEqual:sessionId]) {
        [self analysisCardRecordTimeline:data];
    }
}
```

seek 观看指定时间戳之后的卡录：

```
[[QHVCNet sharedInstance] setGodSeesRecordSeek:_sessionId
timeStamp:self.currentSeekTime];
```

监听更新当前播放回放时间进度值：

```
-       (void)       onGodSees:(NSString       *)sessionId
didRecordUpdateCurrentTimeStamp:(NSUInteger)timeStampByMS
{
 //更新 UI
}
```

更新当前播放视频的密钥：

```
- (void)setCurrentWatchingSN:(NSString
*)currentWatchingSN {
    _currentWatchingSN = currentWatchingSN;
    if (currentWatchingSN != nil) {
        NSArray *tmpArray = [_streamPwds copy];
        [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"秘钥 - 更新
serialNumer:%@ 的秘钥",currentWatchingSN]];
        for (QHVCGVStreamPasswordModel *pwdModel in
tmpArray) {
            if ([pwdModel.sn
isEqualToString:currentWatchingSN]) {
```

```
            [[QHVCNet sharedInstance]
updateGodSeesVideoStreamSecurityKeys:pwdModel.pwds
serialNumber:pwdModel.sn];
            [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"秘钥 - 找到秘钥，已更
新:%@",pwdModel.pwds]];
        }
    }
  }
}
```

◆ 观看带 token 校验加密回放实例代码

创建帝视会话实例：

```
  QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
  QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
  [[QHVCNet sharedInstance] setGodSeesDelegate:self];
  _sessionId = [NSString
stringWithFormat:@"%@_%@_%lld",globalConfig.appId,
config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
  [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
  [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
  NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN
```

```
deviceId:globalConfig.deviceId

appKey:globalConfig.appSecret];
    [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId

clientId:globalConfig.deviceId

appId:globalConfig.appId

userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
                          deviceChannelNumber:1
                                businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_RECORD

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongCon
nectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
    _playerUrl      =      [[QHVCNet      sharedInstance]
getGodSeesPlayUrl:_sessionId];
```

验证业务 token:

```
[[QHVCNet                                  sharedInstance]
verifyGodSeesBusinessTokenToDevice:_sessionId
token:[QHVCGVUserSystem sharedInstance].userInfo.token];
```

播放器播放视频：

```
    [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    QHVCGlobalConfig * globalConfig = [QHVCGlobalConfig
sharedInstance];
    int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                  userId:config.userName
                                playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency|QHVCP
layModeLive_IOT),@"max_buffering_delay":@(600),@"inputStr
eam":@(inputStreamValue),@"streamType":@(config.playerStr
eamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
```

```
}
```

监听帝视验证 token 回调：

```
- (void) onGodSees:(NSString *)sessionId
didVerifyToken:(NSInteger)status info:(NSString *)info
{
    [QHVCLogger printLogger:QHVC_LOG_LEVEL_TRACE
content:[NSString stringWithFormat:@"didVerifyToken:
sessionId = %@, status = %zd, info
= %@",sessionId,status,info]];
    if ([_sessionId isEqual:sessionId])
    {
        if (status == QHVCNetErrorCodeNoError)
        {

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_P
RIORITY_DEFAULT, 0), ^{
                [[QHVCNet sharedInstance]
getGodSeesRecordTimeline:_sessionId startTime:0 endTime:-
1];
            });
            return;
        }
    }
}
```

在验证 token 成功后获取卡录时间轴信息：

```
[[QHVCNet sharedInstance]
getGodSeesRecordTimeline:_sessionId startTime:0 endTime:-
1];
```

监听帝视卡录时间轴信息回调：

```
- (void) onGodSees:(NSString *)sessionId
didRecordTimeline:(NSArray<QHVCNetGodSeesRecordTimeline
*> *)data {
    if ([_sessionId isEqual:sessionId]) {
        [self analysisCardRecordTimeline:data];
    }
}
```

seek 观看指定时间戳之后的卡录：

```
[[QHVCNet sharedInstance] setGodSeesRecordSeek:_sessionId
timeStamp:self.currentSeekTime];
```

监听更新当前播放回放时间进度值：

```
-       (void)       onGodSees:(NSString       *)sessionId
didRecordUpdateCurrentTimeStamp:(NSUInteger)timeStampByMS
{
 //更新 UI
}
```

更新当前播放视频的密钥：

```
- (void)setCurrentWatchingSN:(NSString
*)currentWatchingSN {
    _currentWatchingSN = currentWatchingSN;
    if (currentWatchingSN != nil) {
        NSArray *tmpArray = [_streamPwds copy];
        [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"秘钥 - 更新
serialNumer:%@ 的秘钥",currentWatchingSN]];
        for (QHVCGVStreamPasswordModel *pwdModel in
tmpArray) {
            if ([pwdModel.sn
isEqualToString:currentWatchingSN]) {
```

```
            [[QHVCNet sharedInstance]
updateGodSeesVideoStreamSecurityKeys:pwdModel.pwds
serialNumber:pwdModel.sn];
            [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString stringWithFormat:@"秘钥 - 找到秘钥，已更
新:%@",pwdModel.pwds]];
        }
    }
  }
}
```

◆ 观看云存实例代码

把云存地址给播放器播放并实例化播放器：

```
  [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
  QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
  QHVCGVConfig* config = [QHVCGVConfig sharedInstance];
  int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
  _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                  userId:config.userName
                                playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency),@"in
putStream":@(inputStreamValue),@"streamType":@(config.pla
yerStreamType)}];
  _player.playerDelegate = self;
```

```
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

- 观看加密云存实例代码

的

把云存地址给播放器播放并实例化播放器：

```
    [QHVCPlayer
setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger
getLoggerLevel]];
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig* config = [QHVCGVConfig sharedInstance];
    int inputStreamValue = [config playerReceiveDataModel]
==
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                    userId:config.userName
                                  playType:QHVCPlayTypeLive
```

```
options:@{@"hardDecode":@([config
isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency),@"in
putStream":@(inputStreamValue),@"streamType":@(config.pla
yerStreamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

监听播放器代理回调 onPlayerOutputPacket，在里面调用帝视解密函数，密钥由业务服务器给出。

```
- (void)onPlayerOutputPacket:(QHVCPacket)packet
{
    QHVCNetGodSeesFrameDataType frameDataType;
    if (packet.flags == QHVC_PACKET_TYPE_AAC ||
packet.flags == QHVC_PACKET_TYPE_OPUS)
    {
        frameDataType =
QHVC_NET_GODSEES_FRAME_DATA_TYPE_AAC;
    } else if (packet.flags == QHVC_PACKET_TYPE_H264)
    {
```

```
        frameDataType =
QHVC_NET_GODSEES_FRAME_DATA_TYPE_H264;
    } else if (packet.flags ==  QHVC_PACKET_TYPE_H265)
    {
        frameDataType =
QHVC_NET_GODSEES_FRAME_DATA_TYPE_H265;
    }
    [[QHVCNet sharedInstance]
decryptGodSeesMediaDataWithSecretKey:encryptedKey

decryptionRules:QHVC_NET_GODSEES_DECRYPTION_RULES_XOR

frameDataType:frameDataType

originalMediaData:packet.pData

originalDataLength:packet.size

decryptedDataLength:&packet.size];
}
```

◆ 实时 RTC 通话实例代码

创建帝视会话实例：

```
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig
sharedInstance];
    QHVCGVConfig * config = [QHVCGVConfig sharedInstance];
    [[QHVCNet sharedInstance] setGodSeesDelegate:self];
    _sessionId = [NSString stringWithFormat:@"%@_%@_%lld",
globalConfig.appId, config.userName, [QHVCToolUtils
getCurrentDateByMilliscond]];
```

```
    [[QHVCNet sharedInstance]
enableGodSeesMonitorVideoState:config.shouldShowPerforman
ceInfo];
    [[QHVCNet sharedInstance]
setGodSeesNetworkConnectType:config.networkConnectType];
    NSString* businessSign = [QHVCGlobalConfig
generateBusinessSubscriptionSignWithAppId:globalConfig.ap
pId

serialNumber:_deviceModel.bindedSN

deviceId:globalConfig.deviceId

appKey:globalConfig.appSecret];
    [[QHVCNet sharedInstance]
createGodSeesSession:_sessionId

clientId:globalConfig.deviceId

appId:globalConfig.appId

userId:[[QHVCGVUserSystem sharedInstance]
userInfo].userId

serialNumber:_deviceModel.bindedSN
                          deviceChannelNumber:1
                                businessSign:businessSign

sessionType:QHVC_NET_GODSEES_SESSION_TYPE_LIVE

options:@{kQHVCNetGodSeesOptionsStreamTypeKey:@([config
streamType]),kQHVCNetGodSeesOptionsPlayerReceiveDataModel
Key:@([config
```

```
playerReceiveDataModel]),kQHVCNetGodSeesOptionsUseLongConnectKey:@([config longConnectService])}];
```

获取本地服务器代理播放地址：

```
_playerUrl = [[QHVCNet sharedInstance] getGodSeesPlayUrl:_sessionId];
```

播放器播放视频：

```
    [QHVCPlayer setLogLevel:(QHVCPlayerLogLevel)[QHVCLogger getLoggerLevel]];
    QHVCGlobalConfig* globalConfig = [QHVCGlobalConfig sharedInstance];
    QHVCGVConfig* config = [QHVCGVConfig sharedInstance];
    int inputStreamValue = [config playerReceiveDataModel] == QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK?1:0;
    _player = [[QHVCPlayer alloc] initWithURL:_playerUrl

channelId:globalConfig.appId
                                  userId:config.userName
                                playType:QHVCPlayTypeLive

options:@{@"hardDecode":@([config isHardDecode]),@"playMode":@(QHVCPlayModeLowLatency),@"inputStream":@(inputStreamValue),@"streamType":@(config.playerStreamType)}];
    _player.playerDelegate = self;
    _player.playerAdvanceDelegate = self;

    [_player createPlayerView:_playerView];
    [_player setSystemVolumeCallback:YES];
    [_player setSystemVolumeViewHidden:NO];
    [_player prepare];
```

播放器真实开始播放

```
/**
 播放器首次加载缓冲准备完毕，在此回调中调用play开始播放
 */
- (void)onPlayerPrepared:(QHVCPlayer *_Nonnull)player {
    [_player play];
}
```

开始 RTC 通话：

```
    [self mutePlayer:YES];
    QHVCGVUserSystem* userSystem = [QHVCGVUserSystem
sharedInstance];
    [QHVCInteractiveKit
openLogWithLevel:(QHVCITLLogLevel)[QHVCLogger
getLoggerLevel]];//设置日志级别
    [[QHVCInteractiveKit sharedInstance]
setPublicServiceInfo:[QHVCGlobalConfig
sharedInstance].appId

appKey:[QHVCGlobalConfig sharedInstance].appKey

userSign:[[QHVCGVUserSystem sharedInstance]
getUserSign]];
    NSMutableDictionary* optionInfoDict =
[NSMutableDictionary dictionary];
    [QHVCToolUtils setBooleanToDictionary:optionInfoDict
key:@"openVideo" value:[QHVCGVConfig
sharedInstance].isRTCOpenVideo];
    [QHVCToolUtils setIntToDictionary:optionInfoDict
key:@"dataCollectModel" value:QHVCITLDataCollectModeSDK];
    [[QHVCInteractiveKit sharedInstance]
startAutomaticConversationWithDelegate:self

roomId:userSystem.roomInfo.roomId
```

```
channelNo:1

userId:userSystem.userInfo.talkId

beInvitedUserId:userSystem.roomInfo.deviceTalkId

optionInfoDict:optionInfoDict];
```

监听开始自动通话成功回调：

```
- (void)interactiveAutomationEngine:(QHVCInteractiveKit
*)engine didStartAutomaticConversation:(NSString
*)channel withUid:(NSString *)uid
{
    [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString
stringWithFormat:@"interactiveAutomationEngine
didStartAutomaticConversation channel:%@, uid
= %@",channel,uid]];
    [_speakBoardView didStartRTC];
}
```

结束 RTC 通话：

```
[[QHVCInteractiveKit                          sharedInstance]
stopAutomaticConversation];
```

监听结束自动通话成功回调：

```
- (void)interactiveAutomationEngine:(QHVCInteractiveKit
*)engine didStopAutomaticConversation:(NSString *)channel
withUid:(NSString *)uid
{
    [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString
stringWithFormat:@"interactiveAutomationEngine
didStopAutomaticConversation channel:%@, uid
= %@",channel,uid]];
```

```
    [QHVCInteractiveKit destory];
    [self mutePlayer:NO];
    [_speakBoardView didStopRTC];
}
```

监听 RTC 通话信令信息发送回调以及帝视发送信息消息：

```
- (void)interactiveAutomationEngine:(QHVCInteractiveKit
*)engine didSendingSignalingMessage:(NSString *)message
toDestId:(NSString *)destId
{
    [QHVCLogger printLogger:QHVC_LOG_LEVEL_DEBUG
content:[NSString
stringWithFormat:@"interactiveAutomationEngine
didSendingSignalingMessage destId:%@, toDestId
= %@",message,destId]];
    //判断信令使用方式
    QHVCNetLongConnectService longConnectService =
[QHVCGVConfig sharedInstance].longConnectService;
    if (longConnectService ==
QHVCNetLongConnectServiceSDK) {
        [[QHVCNet sharedInstance]
sendSignallingData:message toDestId:destId];
    }
}
```

监听帝视信令发送回调以及 RTC 通话接收信令消息：

```
- (void) onGodSees:(NSString *)destId
didReceiveSignallingData:(NSString *)data {
    if ([[QHVCGVConfig sharedInstance] longConnectService]
== QHVCNetLongConnectServiceSDK) {
        if ([QHVCToolUtils isNullString:data]) {
            return;
        }
        NSData* nData = [data
dataUsingEncoding:kCFStringEncodingUTF8];
```

```
        NSDictionary* dataDict = [QHVCToolUtils
resolveJsonDataToDictionary:nData];
        NSString *modelName = [QHVCToolUtils
getStringFromDictionary:dataDict
key:kQHVCNetGodSeesOptionsModelKey defaultValue:nil];
        if ([modelName
isEqualToString:kQHVCNetGodSeesOptionsModelRtcSDKKey]) {
            [[QHVCInteractiveKit sharedInstance]
receiveSignalingMessages:data];
        }
    }
}
```

◆ 获取视频流附加数据实例代码

实例化播放器监听播放器回调：

```
/**
 seiMeta信息
 @param type 类型
 @param data 数据
 */
- (void)seiMeta:(long)type data:(NSString *_Nullable)data
player:(QHVCPlayer *_Nonnull)player;
```

◆ 文件下载实例代码

调用帝视获取设备文件下载链接:

```
        [[QHVCNet sharedInstance]
getDeviceFileDownloadUrlWithFileKey:model.imageName

clientId:globalConfig.deviceId

appId:globalConfig.appId
```

```
userId:userInfo.userId

serialNumber:serialNumber

businessSign:businessSign

token:@"业务验证码"

rangeStart:0

rangeEnd:0

resultUrlString:&resultString];
```

拿到 resultString 后用任何下载工具都可自行下载。

- SDK 函数说明

QHVCNet.h

```
//
//  QHVCNet.h
//  QHVCNet
//
//  Created by yangkui on 2017/8/29.
//  Copyright © 2017年 yangkui. All rights reserved.
//

#import <Foundation/Foundation.h>
#import "QHVCNetEnumerates.h"

NS_ASSUME_NONNULL_BEGIN

/**
```

```
 * 需要开启服务的KEY，值见：QHVCNetServiceType。默认值：
QHVCNetServicePrecache
 * 如果需要同时开启多个服务，多个服务的值相加即可，例如：
QHVCNetServicePrecache|QHVCNetServiceGodSees
 */
extern NSString* const kQHVCNetOptionsServicesKey;//开启服
务类型

#pragma mark - SDK具体实现 -

@interface QHVCNet: NSObject

#pragma mark - 基础功能 -

/**
 该方法生成一个单例QHVCNet

 @return QHVCNet对象
 */
+ (instancetype)sharedInstance;

/**
 获取QHVCNetKit的版本信息

 @return 字符串版本号
 */
- (NSString *) getVersion;

/**
 设置日志开关

 @param logLevel 日志等级
 @param detailInfo 日志是否包含详细信息（时间、进程号、线程号）
```

```
 NSLog和Android的Log函数可以自带这些信息时，可以设置成0
 存入文件或其他不带详细信息的日日志打印函数的场合建议把detailInfo设置成1
 */
- (void)setLogLevel:(QHVCNetLogLevel)logLevel
         detailInfo:(int)detailInfo
           callback:(void(^)(const char* buf, size_t buf_size))callback;

/**
 开启LocalServer，全局只用调用一次，需要和stopLocalServer配对使用

 @param cacheDir 缓存目录，要保证是一个当前存在且有读写权限的文件夹
 @param cacheSizeInMB 初始缓存大小 单位：MB
 @param options @{kQHVCNetOptionsServicesKey:@(QHVCNetServicePrecache | QHVCNetServiceGodsees),

kQHVCNetGodSeesOptionsUseLongConnectKey:@(QHVCNetLongConnectService)
                }
 */
- (void)startLocalServer:(NSString *)cacheDir
              cacheSize:(int)cacheSizeInMB
                options:(NSDictionary *)options;

/**
 停止本地服务，需要和startLocalServer配对使用
 */
- (void)stopLocalServer;
```

```
/**
 获取localserver播放的链接

 @param rid 资源的唯一标识，由于url在加防盗链时是可变的，所以需要一个唯一标识来匹配缓存文件
 @param url 资源的链接
 @return 走预缓存播放的链接，不成功将会返回空
 */
- (NSString *)getLocalServerPlayUrl:(NSString *)rid url:(NSString *)url;

/**
 获取localserver播放的链接

 @param rid 资源的唯一标识，由于url在加防盗链时是可变的，所以需要一个唯一标识来匹配缓存文件
 @param url 资源的链接
 @param options @{@"open_p2p":@"boolValue",@"open_localserver":@"boolValue"}
 @return 走预缓存播放的链接，不成功将会返回空
 */
- (NSString *)getLocalServerPlayUrl:(NSString *)rid url:(NSString *)url options:(NSDictionary * _Nullable)options;

/**
 是否开启LocalServer

 @return YES 开启 NO 未开启
 */
- (BOOL)isStartLocalServer;
```

```
/**
 设置和调整缓存占用空间大小。 这个接口可以中途调用，可以调用多次

 @param cacheSizeInMB 缓存空间的大小， 单位:MB
 @return yes:成功 no:失败
 */
- (BOOL)setCacheSize:(int)cacheSizeInMB;

/**
 * 获取网络状态
 */
- (QHVCNetNetworkStatus)networkStatus;

/**
 恢复/禁止QHVCNetKit的所有访问网络主动拉取数据的行为

 @param enable YES:允许，NO：禁止，默认：允许
 */
- (void)enableNetwork:(BOOL)enable;

/**
 是否允许QHVCNetKit所有访问网络主动拉取数据的行为

 @return YES：允许，NO：禁止
 */
- (BOOL)isEnableNetwork;

/**
 获取p2p信息

 @return 详情
 */
- (nullable NSDictionary *)getP2pInfo;
```

```
@end

NS_ASSUME_NONNULL_END
```

QHVCNet+GodSees.h

```
//
//  QHVCNet+GodSees.h
//  QHVCNetKit
//
//  Created by yangkui on 2019/4/25.
//  Copyright © 2019 yangkui. All rights reserved.
//

#import "QHVCNet.h"

NS_ASSUME_NONNULL_BEGIN

#pragma mark - 外部属性定义 -

//用于设置码流，类型见QHVCNetGodSeesStreamType
extern NSString* const
kQHVCNetGodSeesOptionsStreamTypeKey;

//用于设置播放模式，类型见
QHVCNetGodSeesPlayerReceiveDataModel
extern NSString* const
kQHVCNetGodSeesOptionsPlayerReceiveDataModelKey;

//最大重连次数，默认3次
extern NSString* const
kQHVCNetGodSeesOptionsMaxReconnectCountKey;
```

```
//业务决定是否使用外部长连，值见：QHVCNetLongConnectService。默认值：QHVCNetLongConnectServiceSDK
extern NSString* const kQHVCNetGodSeesOptionsUseLongConnectKey;

//如果业务使用SDK长连，相关字典返回数据的<K,V>,例如{kQHVCNetGodSeesOptionsModelKey:kQHVCNetGodSeesOptionsRtcModelKey}
extern NSString* const kQHVCNetGodSeesOptionsModelKey;
extern NSString* const kQHVCNetGodSeesOptionsModelRtcSDKKey;
extern NSString* const kQHVCNetGodSeesOptionsModelNetSDKKey;

#pragma mark - 数据结构定义 -

/**
 卡录时间轴数据结构
 */
@interface QHVCNetGodSeesRecordTimeline : NSObject

@property(nonatomic, assign) NSUInteger startMS;//开始时间。单位：毫秒
@property(nonatomic, assign) NSUInteger durationMS;//持续时间。单位：毫秒
@property(nonatomic, assign) QHVCNetGodSeesRecordDataType recordDataType;//数据录制类型

@end

/**
 音频流格式
 */
```

```objc
@interface QHVCNetGodSeesAudioStreamFormat : NSObject

@property (nonatomic, assign) int sampleRate;//采样率
@property (nonatomic, assign) int sampleBits;//采样深度
@property (nonatomic, assign) int channelNumbers;//声道数

@end

#pragma mark - 代理回调 -

@protocol QHVCNetGodSeesDelegate <NSObject>

@optional

/**
 错误监听回调

 @param sessionId 会话ID
 @param errorCode 错误码
 */
- (void) onGodSees:(NSString *)sessionId
didError:(QHVCNetErrorCode)errorCode;

/**
 业务token验证结果回调

 @param sessionId 会话ID
 @param status 验证结果，由业务定义
 @param info 附带信息
 */
- (void) onGodSees:(NSString *)sessionId
didVerifyToken:(NSInteger)status info:(NSString *)info;
```

```
/**
 请求卡录时间轴结果

 @param sessionId 会话ID
 @param data 卡录时间数据
 */
- (void) onGodSees:(NSString *)sessionId
didRecordTimeline:(NSArray<QHVCNetGodSeesRecordTimeline
*> *)data;

/**
 更新当前播放的时间点

 @param sessionId 会话ID
 @param timeStampByMS 当前时间戳
 */
- (void) onGodSees:(NSString *)sessionId
didRecordUpdateCurrentTimeStamp:(NSUInteger)timeStampByMS
;

/**
 卡录已经暂停

 @param sessionId 会话ID
 */
- (void) onGodSeesRecordPause:(NSString *)sessionId;

/**
 卡录已经恢复

 @param sessionId 会话ID
 */
- (void) onGodSeesRecordResume:(NSString *)sessionId;
```

```
/**
 卡录播放完毕

 @param sessionId 会话ID
 */
- (void) onGodSeesRecordPlayComplete:(NSString
*)sessionId;

/**
 卡录seek完成

 @param sessionId 会话ID
 */
- (void) onGodSeesRecordSeekComplete:(NSString
*)sessionId;

/**
 倍速播放回调

 @param sessionId 会话ID
 @param rate 卡录播放速度值
 */
- (void) onGodSees:(NSString *)sessionId
didRecordPlaybackRate:(double)rate;

/**
 监控网络日志信息

 @param sessionId 会话ID
 @param info 监控信息
 */
```

```
- (void) onGodSees:(NSString *)sessionId
didMonitorNetworkInfo:(NSString *)info;

/**
 NetSDK将帧数据抛出，业务端可以将相关数据直接吐给播放器播放。
 注意：此代理只会在开启
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK时才会
触发该回调，此时需要业务把该回调数据直接传递给播放器使用。(开启方
法：在createGodSeesSession方法的参数中，设置播放器接收数据模式
QHVCNetGodSeesPlayerReceiveDataModel为
QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK)

 @param sessionId 会话ID
 @param frameDataType 数据帧类型
 @param frameData 帧数据
 @param length 帧数据长度
 @param pts 显示时间戳
 @param dts 解码时间戳
 @param isKeyFrame 是否是关键帧
 @param audioStreamFormat 音频格式
 */
- (void) onGodSees:(NSString *)sessionId
didReceiveFrameDataType:(QHVCNetGodSeesFrameDataType)fram
eDataType
        frameData:(const uint8_t *)frameData
     frameDataLen:(int)length
              pts:(uint64_t)pts
              dts:(uint64_t)dts
       isKeyFrame:(BOOL)isKeyFrame
 audioStreamFormat:(QHVCNetGodSeesAudioStreamFormat
*)audioStreamFormat;

/**
```

```
 借用业务信令通道发送数据回调方法
 注意：只有在设置 QHVCNetLongConnectService =
QHVCNetLongConnectServiceExternal 模式下，该方法调用才有效

 @param data 发送的数据体
 @param destId 设备ID
 */
- (void) onGodSees:(NSString *)destId
didSendSignallingData:(NSString *)data;



/**
 使用SDK信令通道发送消息回调方法
 注意：只有在设置 QHVCNetLongConnectService =
QHVCNetLongConnectServiceSDK 模式下，该方法调用才有效

 @param data 发送的数据体
 @param destId 设备ID
 */
- (void) onGodSees:(NSString *)destId
didReceiveSignallingData:(NSString *)data;

@end

#pragma mark - SDK实现 -
@interface QHVCNet (GodSees)

@property (nonatomic, weak) id<QHVCNetGodSeesDelegate>
godSeesDelegate;//代理回调

#pragma mark - 信令相关 -

/**
```

```
 帝视接收外部信令通道数据
 注意：只有在设置 QHVCNetLongConnectService =
QHVCNetLongConnectServiceExternal 模式下，该方法调用才有效
 相关回调：onGodSees:didSendSignallingData:

 @param data 信令数据
 @return 0成功，非0表示失败
 */
- (int) receiveGodSeesSignallingData:(NSString *)data;

/**
 利用SDK内部通道发送信令消息
 注意：只有在设置 QHVCNetLongConnectService =
QHVCNetLongConnectServiceSDK 模式下，该方法调用才有效
 相关回调：onGodSees:didReceiveSignallingData:

 @param data 信令数据
 @param destId 设备ID
 @return 0成功，非0表示失败
 */
- (int) sendSignallingData:(NSString *)data
toDestId:(NSString *)destId;

#pragma mark - 安全相关 -

/**
 更新帝视视频流解密秘钥

 @param keys 密钥表，形如
[{"27":"sfee23"},{"28":"fsjei"},@{"29":"ejis"}]
 @param serialNumber 设备唯一标识
 @return 0成功，非0表示失败
 */
```

```
- (int)
updateGodSeesVideoStreamSecurityKeys:(NSArray<NSDictionar
y<NSString *, NSString *> *> *)keys
                             serialNumber:(NSString
*)serialNumber;

/**
 解密帝视媒体数据。
 原地解密数据。

 @param secretKey 解密秘钥
 @param decryptionRules 解密规则
 @param frameDataType 待解密的数据类型
 @param originalMediaData 待解密的原始媒体数据
 @param originalDataLength 待解密的原始媒体数据长度
 @param decryptedDataLength 解密后的媒体数据大小
 @return 0成功，非0表示失败
 */
- (int) decryptGodSeesMediaDataWithSecretKey:(NSString
*)secretKey

decryptionRules:(QHVCNetGodSeesDecryptionRules)decryption
Rules

frameDataType:(QHVCNetGodSeesFrameDataType)frameDataType
                        originalMediaData:(uint8_t
*)originalMediaData

originalDataLength:(int)originalDataLength
                      decryptedDataLength:(int
*)decryptedDataLength;
```

```
#pragma mark - 文件下载 -

/**
 获取设备文件下载的链接

 @param fileKey 指定设备上的文件的标识符
 @param clientId 客户端ID,每个设备需要唯一，用户多端登录时用来区分链接
 @param appId 应用appId，在帝视官网后台申请
 @param userId 用户ID
 @param serialNumber 设备唯一标识
 @param businessSign 业务签名,帝视鉴定SDK业务的合法性，签名需要由业务服务器下发，密钥在帝视官网获取
 @param token 待设备验证的标识，该标识由业务服务器验证合法性，字符串长度不能超过63个字符
 @param rangeStart 请求文件内容的起始位置，与http的range语义相同
 @param rangeEnd 请求文件内容的结束位置，与http的range语义相同
 @param resultUrlString 获取的下载链接，如果返回成功的，不为nil，否则为nil
 @return 0成功，非0表示失败
 */
- (int) getDeviceFileDownloadUrlWithFileKey:(NSString *)fileKey
                                   clientId:(NSString *)clientId
                                      appId:(NSString *)appId
                                     userId:(NSString *)userId
                               serialNumber:(NSString *)serialNumber
                               businessSign:(NSString *)businessSign
                                      token:(NSString *)token
```

```
rangeStart:(NSUInteger)rangeStart

rangeEnd:(NSUInteger)rangeEnd
                        resultUrlString:(NSString
*_Nonnull *_Nullable)resultUrlString;

#pragma mark - 全局公共方法 -

/**
 启用测试环境

 @param testEnv YES：启用；NO：关闭
 @return 0：成功，非0:失败
 */
- (int) enableTestEnvironment:(BOOL)testEnv;



/**
 设置帝视网络连接方式
 可以是多种方式的组合，默认值: netConnectType =
QHVC_NET_GODSEES_NETWORK_CONNECT_TYPE_P2P |
QHVC_NET_GODSEES_NETWORK_CONNECT_TYPE_RELAY |
QHVC_NET_GODSEES_NETWORK_CONNECT_TYPE_DIRECT_IP

 @param networkConnectType 网络连接类型
 */
- (void)
setGodSeesNetworkConnectType:(QHVCNetGodSeesNetworkConnec
tType)networkConnectType;



/**
```

```
 设置帝视设备IP直连网络地址

 @param serialNumber 设备唯一标识
 @param ip ip地址。值为NULL时，删除原来的ip地址
 @param port 设备侦听端口
 @return 0成功，非0表示失败
 */
- (int) setGodSeesDeviceNetworkAddress:(NSString
*)serialNumber
                                    ip:(NSString *)ip
                                  port:(NSInteger)port;


/**
 设置p2p连接成功最大等待时间
 调用时机：createGodSeesSession之前调用

 @param timeMS 等待时间，单位：毫秒，默认值：1000毫秒
 */
- (void)
setGodSeesP2PConnectionSucceedMaxWaitTime:(NSInteger)time
MS;


/**
 开启/关闭视频状态监控
 开启监控，SDK将会触发didMonitorNetworkInfo回调

 @param enable YES 监控,NO 不监控，默认值:NO
 */
- (void) enableGodSeesMonitorVideoState:(BOOL)enable;
```

```
/**
 预连接设备，为每个通道提前建立连接
 用于提前建立网络通道，节省用户拉取视频等待时间，提高用户体验

 @param serialNumber 需要预连的设备Id
 @param deviceChannelNumber 设备通道数量，如果一个NVR设备有64个通道，实际同时观看16个通道，就传16
 @return 0成功，非0表示失败
 */
- (int) preConnectGodSeesDevice:(NSString *)serialNumber

deviceChannelNumber:(NSInteger)deviceChannelNumber;



/**
 帝视创建网络会话实例

 @param sessionId 实例会话ID
 @param clientId 客户端ID,每个设备需要唯一，用户多端登录时用来区分链接
 @param appId 应用appId，在帝视官网后台申请
 @param userId 用户ID
 @param serialNumber 设备唯一标识
 @param deviceChannelNumber 设备管道号，从索引1开始[1, ...]
 @param businessSign 业务签名,帝视鉴定SDK业务的合法性，签名需要由业务服务器下发，密钥在帝视官网获取
 @param sessionType 会话类型:直播/卡录/文件下载等
 @param options 可为nil
@{@"kQHVCNetGodSeesOptionsStreamTypeKey":@(QHVCNetGodSeesStreamType),

@"kQHVCNetGodSeesOptionsPlayerReceiveDataModelKey":@(QHVCNetGodSeesPlayerReceiveDataModel)
```

```
                    }
 @return 0成功，非0表示失败
 */
- (int) createGodSeesSession:(NSString *)sessionId
                    clientId:(NSString *)clientId
                       appId:(NSString *)appId
                      userId:(NSString *)userId
                serialNumber:(NSString *)serialNumber

deviceChannelNumber:(NSInteger)deviceChannelNumber
                businessSign:(NSString *)businessSign

sessionType:(QHVCNetGodSeesSessionType)sessionType
                     options:(NSDictionary *)options;



/**
 销毁帝视网络会话实例

 @param sessionId 实例会话ID
 @param serialNumber 设备唯一标识
 */
- (int) destroyGodSeesSession:(NSString *)sessionId
                 serialNumber:(NSString *)serialNumber;



/**
 获取帝视指定会话的播放链接

 @param sessionId 实例会话ID
 @return 成功:对应的播放链接；不成功:nil
 */
- (NSString *) getGodSeesPlayUrl:(NSString *)sessionId;
```

```
/**
 设备验证业务token
 该方法是异步操作，SDK将会触发didVerifyToken回调，然后，业务才可以进行一系列操作设置。若验证失败，SDK将会触发didError回调，请根据相关错误码进行逻辑处理。

 @param sessionId 实例会话ID
 @param token 验证信息，字符串长度不能超过500个字符
 @return 0成功，非0表示失败
 */
- (int) verifyGodSeesBusinessTokenToDevice:(NSString *)sessionId
                                     token:(NSString *)token;

/*

**********************************************************************
**********************************************************
 * 特别注意：下面的方法一定要等设备授权通过后才能调用。
 * 即调用verifyGodSeesBusinessTokenToDevice函数通知设备验证token成功，待回调didVerifyToken执行后，才能调用下面的接口

**********************************************************************
**********************************************************
 */

#pragma mark - 直播相关 -

#pragma mark - 卡录相关 -

/**
```

```
 获取卡录时间轴信息
 该方法是异步操作，SDK将会触发didRecordTimeline回调，然后，业务才可以进行一系列操作设置。若加载失败，SDK将会触发didError回调，请根据相关错误码进行逻辑处理。
 如果要获取全部卡录信息，srartTimeMS:0,  endTimeMS:-1

 @param sessionId 实例会话ID
 @param srartTimeMS 卡录开始时间
 @param endTimeMS 卡录结束时间
 @return 0成功，非0表示失败
 */
- (int) getGodSeesRecordTimeline:(NSString *)sessionId
                       startTime:(NSUInteger)srartTimeMS
                         endTime:(NSUInteger)endTimeMS;



/**
 观看指定时间戳之后的卡录
 该方法是异步操作，SDK将会触发onGodSeesRecordSeekComplete回调，然后，业务才可以进行一系列操作设置。若加载失败，SDK将会触发didError回调，请根据相关错误码进行逻辑处理。

 @param sessionId 实例会话ID
 @param timeStampByMS 指定时间点（单位：毫秒）
 @return 0成功，非0表示失败
 */
- (int) setGodSeesRecordSeek:(NSString *)sessionId
                   timeStamp:(NSUInteger)timeStampByMS;



/**
 暂停观看卡录
```

```
 该方法是异步操作，SDK将会触发onGodSeesRecordPause回调，然后，业务才可以进行一系列操作设置。若加载失败，SDK将会触发didError回调，请根据相关错误码进行逻辑处理。

 @param sessionId 实例会话ID
 @return 0成功，非0表示失败
 */
- (int) setGodSeesRecordPause:(NSString *)sessionId;



/**
 恢复观看卡录
 该方法是异步操作，SDK将会触发onGodSeesRecordResume回调，然后，业务才可以进行一系列操作设置。若加载失败，SDK将会触发didError回调，请根据相关错误码进行逻辑处理。

 @param sessionId 实例会话ID
 @return 0成功，非0表示失败
 */
- (int) setGodSeesRecordResume:(NSString *)sessionId;



/**
 设置观看卡录速度
 该方法是异步操作，SDK将会触发didRecordPlaybackRate回调，然后，业务才可以进行一系列操作设置。若加载失败，SDK将会触发didError回调，请根据相关错误码进行逻辑处理。

 @param sessionId 实例会话ID
 @param rate 速度倍数，默认:1倍速
 @return 0成功，非0表示失败
 */
- (int) setGodSeesRecordPlayRate:(NSString *)sessionId
```

```
                          rate:(double)rate;


@end

NS_ASSUME_NONNULL_END
```

- 错误码及枚举定义说明

```
//
//  QHVCNetEnumerates.h
//  QHVCNetKit
//
//  Created by yangkui on 2019/8/5.
//  Copyright © 2019 yangkui. All rights reserved.
//

#ifndef QHVCNetEnumerates_h
#define QHVCNetEnumerates_h

#import <Foundation/Foundation.h>

/**
 SDK错误码
 */
typedef NS_ENUM(NSInteger, QHVCNetErrorCode) {
    QHVCNetErrorCodeNoError = 0,//无错误
    QHVCNetErrorCodeFailed = 1,//一般错误，原因未分类
    QHVCNetErrorCodeInvalidArgument = 2,//无效的参数
    QHVCNetErrorCodeSignallingModelError = 3,//信令通道模式使用错误
```

```
    QHVCNetErrorCodeUninitOnAppStart = 100,
    QHVCNetErrorCodeRepeatInitAppStart,
    QHVCNetErrorCodeUninit,
    QHVCNetErrorCodeRepeatInit,
    QHVCNetErrorCodeEventCallbackIsEmpty,
    QHVCNetErrorCodeLongConnectCallbackIsEmpty,
    QHVCNetErrorCodeInputBufferSizeTooSmaller,
    QHVCNetErrorCodeLocalSercerListenAddressInvalid,
    QHVCNetErrorCodeSessionIsEmpty,
    QHVCNetErrorCodeSessionIsUsed,
    QHVCNetErrorCodeSessionInvalid,
    QHVCNetErrorCodeTokenTooLong,
    QHVCNetErrorCodeFileKeyIsEmpty,
    QHVCNetErrorCodeFileDownloadRangeNoLegal,
    QHVCNetErrorCodeDownloadFileOpenFileFailed,
    QHVCNetErrorCodeDownloadFileGetAbsFilePathFailed,
    QHVCNetErrorCodeDownloadFileRepeatDownloadSameFileKey,
    QHVCNetErrorCodeRecordRangeNoLegal,
    QHVCNetErrorCodeRecordPlaySpeedNoLegal,
    QHVCNetErrorCodeP2PPunchHoleFailed,
    QHVCNetErrorCodeDirectIPConnectFailed,
    QHVCNetErrorCodeDisableAllConnectType,
    QHVCNetErrorCodeNoFoundDecryptKey,
    QHVCNetErrorCodeNoSupportedDecryptAlgo,
    QHVCNetErrorCodeVerifyTokenFailed,
    QHVCNetErrorCodeWaitTokenTimeoutInP2P,
    QHVCNetErrorCodeWaitTimelineResTimeoutInP2P,
    QHVCNetErrorCodeWaitStreamConnectTimeoutInP2P,
    QHVCNetErrorCodeStreamConnectFailedInP2P,
    QHVCNetErrorCodeHeartBeatTimeoutInP2P,
    QHVCNetErrorCodeTooLongTimeNoReceivePeerDataInP2P,
    QHVCNetErrorCodeStreamDisconnectInP2P,
```

```
QHVCNetErrorCodeWaitLongConnnectConnectSucessTimeoutInRel
ay,
    QHVCNetErrorCodeWaitTokenTimeoutInRelay,
    QHVCNetErrorCodeWaitTimelineResTimeoutInRelay,
    QHVCNetErrorCodeHeartbeatTimeoutInRelay,
    QHVCNetErrorCodeWaitStreamConnectTimeoutInRelay,
    QHVCNetErrorCodeCallScheuleFailedInRelay,
    QHVCNetErrorCodeScheduleCallbackFailedInRelay,
    QHVCNetErrorCodeStreamConnectFailedInRelay,
    QHVCNetErrorCodeTooLongTimeNoReceivePeerDataInRelay,
    QHVCNetErrorCodeStreamDisConnectInRelay,
    QHVCNetErrorCodeWaitTokenTimeoutInDirectIP,
    QHVCNetErrorCodeWaitTimelineResTimeoutInDirectIP,
    QHVCNetErrorCodeWaitStreamConnenctTimeoutInDirectIP,
    QHVCNetErrorCodeStreamConnectFailedInDirectIP,
    QHVCNetErrorCodeHeartbeatTimeoutInDirectIP,

QHVCNetErrorCodeTooLongtimeNoReceivePeerDataInDirectIP,
    QHVCNetErrorCodeStreamDisconnectInDirectIP,
    QHVCNetErrorCodeRepeatCallGetRecordTimeline,

    //comm
    QHVCNetErrorCodeDeviceSerialNumberIsEmpty = 1000,
    QHVCNetErrorCodeClientIdIsEmpty,
    QHVCNetErrorCodeAppIdIsEmpty,
    QHVCNetErrorCodePlatformIdIsEmptyOnAppStart,
    QHVCNetErrorCodeVersionIsEmptyOnAppStart,
    QHVCNetErrorCodeOperatingSystemIsEmptyOnAppStart,
    QHVCNetErrorCodeMIDIsEmptyOnAppStart,
    QHVCNetErrorCodeModelIsEmptyOnAppStart,
    QHVCNetErrorCodeDeviceChannelNumNoLegal,
    QHVCNetErrorCodeNoSupportedBitrateType,
```

```
    QHVCNetErrorCodeNoSupportedPlayType,
    QHVCNetErrorCodeNoSupportedPlayMode,
    QHVCNetErrorCodeUserIdIsEmpty,
    QHVCNetErrorCodeChannelIdIsEmpty,
    QHVCNetErrorCodeUserSignIsEmpty,
    QHVCNetErrorCodeSecretKeyNumNoLegal,
    QHVCNetErrorCodeLongConnectTopicIsEmpty,
    QHVCNetErrorCodeLongConnectDataIsEmpty,
    QHVCNetErrorCodeLongConnectDataParseFailed,
    QHVCNetErrorCodeLongConnectDataUnknownModel,
    QHVCNetErrorCodeLongConnectDataNoFoundModel,
    QHVCNetErrorCodeLongConnectDataNoFoundData,
    QHVCNetErrorCodeNetSDKParseFailed,
    QHVCNetErrorCodeNetSDKUnknownType,
    QHVCNetErrorCodeNetSDKNoFoundContext,

QHVCNetErrorCodeLongConnectNetSDKNoFoundResValueOfREQRela
yRes,

QHVCNetErrorCodeLongConnectNetSDKNoFoundTokenResOfREQRela
yRes,
    QHVCNetErrorCodeLongConnectNetSDKUnknownFailed,
    QHVCNetErrorCodeNoSupportedLogLevel,
    QHVCNetErrorCodeLogPathIsEmpty,
    QHVCNetErrorCodeMediaDeviceHandleInvalid,
};

/**
 * 日志等级
 */
typedef NS_ENUM(NSInteger, QHVCNetLogLevel) {
    QHVCNetLogLevelTrace = 0,//trace
    QHVCNetLogLevelDebug = 1,//debug
```

```
    QHVCNetLogLevelInfo  = 2,//info
    QHVCNetLogLevelWarn  = 3,//warn
    QHVCNetLogLevelError = 4,//error
    QHVCNetLogLevelAlarm = 5,//alarm
    QHVCNetLogLevelFatal = 6,//fatal
    QHVCNetLogLevelNone  = 7,//none
};

/**
 *  网络状态
 */
typedef NS_ENUM(NSInteger, QHVCNetNetworkStatus) {
    QHVCNetNetworkNotReachable = 0,
    QHVCNetNetworkReachableViaWiFi,
    QHVCNetNetworkReachableViaWWAN
};

/**
 * QHVCNetKit提供的服务类型
 */
typedef NS_OPTIONS(NSInteger, QHVCNetServiceType) {
    QHVCNetServicePrecache        =   1   <<  0,     // 预缓
存服务
    QHVCNetServiceGodSees         =   1   <<  1      // 帝视
服务
};

/**
 长连服务
 */
typedef NS_ENUM(NSInteger, QHVCNetLongConnectService) {
    QHVCNetLongConnectServiceSDK = 0,//SDK自带长连
    QHVCNetLongConnectServiceExternal,//外部长连
```

```
};

/**
 网络连接类型
 */
typedef NS_ENUM(NSInteger,
QHVCNetGodSeesNetworkConnectType) {
    QHVC_NET_GODSEES_NETWORK_CONNECT_TYPE_P2P            =
1,//p2p
    QHVC_NET_GODSEES_NETWORK_CONNECT_TYPE_RELAY          =
(1<<1),//云端
    QHVC_NET_GODSEES_NETWORK_CONNECT_TYPE_DIRECT_IP
= (1<<2),//IP直连
};

/**
 会话类型
 */
typedef NS_ENUM(NSInteger, QHVCNetGodSeesSessionType) {
    QHVC_NET_GODSEES_SESSION_TYPE_LIVE = 1,  //直播
    QHVC_NET_GODSEES_SESSION_TYPE_RECORD,    //回放
    QHVC_NET_GODSEES_SESSION_TYPE_FILE_DOWNLOAD, //文件下载
};

/**
 码流类型
 码流大小根据硬件设备性能自定义
 */
typedef NS_ENUM(NSInteger, QHVCNetGodSeesStreamType) {
    QHVC_NET_GODSEES_STREAM_TYPE_MAIN = 1,  // 主码流
    QHVC_NET_GODSEES_STREAM_TYPE_SUB,       // 子码流
    QHVC_NET_GODSEES_STREAM_TYPE_THIRD,     // 第三码流
};
```

```
/**
 播放器接收数据的方式
 */
typedef
NS_ENUM(NSInteger,QHVCNetGodSeesPlayerReceiveDataModel) {
    QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_CALLBACK =
1,  //SDK将音视频帧数据通过回调抛出来，由业务塞给播放器播放。此模式
可以在码流里附加自定义数据，同时让业务有机会对音视频帧数据进行中间处
理
    QHVC_NET_GODSEES_PLAYER_RECEIVE_DATA_MODEL_HTTP_FLV,
//QHVCNetKit生成标准http-flv格式数据，业务使用本地代理产生的url
给播放器进行播放
};

/**
 数据帧类型
 */
typedef NS_ENUM(NSInteger, QHVCNetGodSeesFrameDataType) {
    QHVC_NET_GODSEES_FRAME_DATA_TYPE_NONE = 0,
    QHVC_NET_GODSEES_FRAME_DATA_TYPE_H264,
    QHVC_NET_GODSEES_FRAME_DATA_TYPE_H265,
    QHVC_NET_GODSEES_FRAME_DATA_TYPE_AAC,
    QHVC_NET_GODSEES_FRAME_DATA_TYPE_PCM_S16LE,
    QHVC_NET_GODSEES_FRAME_DATA_TYPE_PCM_MULAW,
    QHVC_NET_GODSEES_FRAME_DATA_TYPE_PCM_ALAW,
};

/**
 解密规则
 */
typedef NS_ENUM(NSInteger, QHVCNetGodSeesDecryptionRules)
{
```

```
    QHVC_NET_GODSEES_DECRYPTION_RULES_NONE = 0,   //无密
    QHVC_NET_GODSEES_DECRYPTION_RULES_XOR = 1,    //XOR方式
    QHVC_NET_GODSEES_DECRYPTION_RULES_RC4,        //RC4方式
};



/**
 卡录数据录制类型
 */
typedef NS_ENUM(NSInteger, QHVCNetGodSeesRecordDataType)
{
    QHVC_NET_GODSEES_RECORD_DATA_TYPE_NORMAL = 0,    //正常模式，全部录制
    QHVC_NET_GODSEES_RECORD_DATA_TYPE_PICTURE_CHANGE,//图像变化模式，简单判断图像内容变化时录制
    QHVC_NET_GODSEES_RECORD_DATA_TYPE_INTELLIGENT,   //智能模式，检测到特定物体出现时录制
};
#endif /* QHVCNetEnumerates_h */
```

- 附录