常见问题

1 设置用户唯一标识的意义

用户唯一标识，多场景项目中称为 viewerId，有时也称为 userId（与保利威后台开发设置中的 userId 不同，后台开发设置中的 userId 指的是开发者账户ID，注意区分）。

用户昵称，多场景项目中称为 viewerName，有时也称为 nickName。

数据统计可以带上用户唯一标识和用户昵称，也能用来作聊天室的登录信息，建议客户设置。

设置用户唯一标识，有以下这些优点：

- 有效定位用户反馈遇到的问题。

- 大大减少与用户的沟通成本。
- 有效提高 SDK 这边单方面排查问题的效率。
- 可以归总某个用户唯一标识的错误日志。

2 在 iOS 9 及以上版本中运行企业版应用

在 iOS 9 中及以后的版本中，苹果对企业签名的应用运行时，进行了更严格的限制。因此，在 iOS 9 中，企业签名后的应用安装好之后，是无法直接启动的。默认情况下，在 iOS 9 中运行一款企业签名的应用时，会弹出这样的提示：

可以看到，应用不再是像之前的版本那样直接启动，而是弹出了一个安全提示。此时，如果我们确认要运行的应用是安全的，可以按照以下步骤来设置：

在系统中打开 `设置` - `通用` - `描述文件与设备管理` （在iOS 9.2以后叫： `设备管理` ），此时，可以看到有一个和刚刚弹出的提示中文字类似的描述文件。然后，点击对应描述文件进入后，再点击按钮 `信任` ，来让系统允许拥有这个证书的应用运行。

之后，我们就可以回到桌面，重新运行刚才的应用，就会发现应用可以正常打开了。

完整的步骤，如下图所示：

3 CocoaPods使用指南

CocoaPods 是 Objective-C 的一个依赖管理器，它可以自动化并简化在项目中使用第三方库的过程。

3.1 安装 CocoaPods

你可以用 RubyGems 工具包安装 CocoaPods，安装命令如下：

```
$ sudo gem install cocoapods
```

但有时运行该命令会报错：

```
$ ERROR:  While executing gem ... (Gem::FilePermissionError) $ You don't have write permissions for the /usr/bin directory.
```

原因是 “/usr/bin is protected by system integrity protection and is not writeable by anybody even root. ” 所以要把 CocoaPods 安装在 /usr/local/bin 路径下，使用如下命令：

```
$ sudo gem install -n /usr/local/bin cocoapods
```

install 加上参数 -n 表示 “Directory where executables are located”，参数含义可通过命令 "`gem help install"` 查询。

3.2 更新索引

安装完 CocoaPods 执行 `pod setup`，将所有的项目的 Podspec 文件更新到本地的 ~/.cocoapods/ 目录下。

```
$ pod setup
```

所有的项目的 Podspec 文件都托管在 https://github.com/CocoaPods/Specs.git 上，CocoaPods 在执行 `pod install` 和 `pod update` 时，会默认先更新一次 Podspec 索引。如果更新太慢，可修改 repo 地址为国内的镜像，如 gitcafe 的镜像操作命令如下：

```
$ pod repo remove master $ pod repo add master https://gitcafe.com/akuandev/Specs.git $ pod repo update
```

也可修改为 oschina 的镜像 http://git.oschina.net/akuandev/Specs.git。

如果用户 pod 操作时没找到我们最近的 SDK 版本，截图如下：

这种情况就是索引太旧需要更新了，执行命令 `pod repo update` 即可，或者在 `pod install` 的时候带上参数 `--repo-update`。

3.3 `pod install` 和 `pod update` 的区别

执行 `pod install` 之后，CocoaPods 会生成一个名为 Podfile.lock 的文件。Podfile.lock 会锁定当前各依赖库的版本，之后如果多次执行 `pod install` 不会更改版本，要 `pod update` 才会改 Podfile.lock 了。这样多人协作的时候，可以防止第三方库升级时造成大家各自的第三方库版本不一致。

`pod install` 只会按照 Podfile 的要求来请求类库，如果类库版本号有变化，那么将获取失败。但是 `pod update` 会更新所有的类库，获取最新版本的类库。每次更改了 Podfile 文件，你都需要重新执行一次 `pod update` 命令。

3.4 查询 CocoaPods 版本

```
$ pod --version
```

3.5 Homebrew & RubyGems

Homebrew 是 Mac 平台的一个包管理工具，提供了许多Mac下没有的Linux工具等。官网：https://brew.sh，官网上有安装命令：

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

安装完 Homebrew，就可以用 Homebrew 的命令 brew 来安装 RubyGems，命令安装如下：

```
$ brew install ruby
```

RubyGems 官网为 https://rubygems.org。

4 多场景SDK与点播SDK同时集成步骤说明

4.1 同时集成多场景SDK1.6.0版本与点播SDK2.16.3版本

使用如下语句集成时：

```
pod 'PLVLiveScenesSDK', '1.6.0'
pod 'PolyvVodSDK', '2.16.3'
```

会出现如下报错：

```
The 'XXXXX' target has frameworks with conflicting names: alicloudhttpdns.framework, alicloudutils.framework,
and utdid.framework.
```

解决方案如下：

```
# pod 'PLVLiveScenesSDK', '1.6.0'
# pod 'PolyvVodSDK', '2.16.3'

#注释上述代码，替换为下列代码

pod 'PLVLiveScenesSDK', '1.6.0', :subspecs => ['BaseSDK','Core', 'Player']
pod 'AgoraRtcEngine_iOS', '2.9.0.107'
pod 'PLVSocketIOClientSwift', '~> 0.1.0'
# 不使用use_Farmeworks!时使用pod "PLVSocket", "~> 0.2.0"  替换pod "PLVSocketIOClientSwift"
pod "UCloudRtcSdk", "1.9.1"
pod "TXLiteAVSDK_Professional", "8.3.9884"
pod "AliyunOSSiOS", "~> 2.10.8"
pod "PLVModel", "~> 1.0.4"

pod 'PolyvVodSDK', '~> 2.16.3', :subspecs => ['Core', 'Player']
pod 'SSZipArchive', '~> 2.1.5'
pod 'WCDB', '~> 1.0.6'
pod 'PLVAliHttpDNS', '~> 1.10.0'
```

出现报错的原因为多场景SDK与点播SDK同时依赖使用alicloudhttpdns.framework等framework。保利威多场景SDK以及点播SDK支持subspecs的pod集成方式，可灵活集成多场景SDK和点播SDK，只引入一次依赖库。

若出现旧版本点播SDK与多场景SDK1.6.0版本出现`PLVIJKPlayer`的冲突，推荐升级旧版本点播SDK至2.16.3版本。若仍无法解决，可联系保利威技术支持工作人员寻求协助。

5 聊天室模块升级到1.7.0版本变动说明

5.1 SDK

注意：若升级 SDK 后，直接升级我们 Common 层的代码，则无需关注 SDK 层的改动；如果只升级 SDK 而不升级 Common 层的源码，或者没有使用 Common 层的源码，则需要针对以下改动做相应的修改。

5.1.1 PLVChatroomManager

1. 发送文本消息、引用消息的 API 修改：

```
// 该方法被移除
- (BOOL)sendSpeakMessage:(PLVSpeakMessage *)message needIdCallback:(BOOL)needIdCallback;
// 替换为以下方法
- (BOOL)sendSpeakMessage:(PLVSpeakMessage *)message callback:(void (^ _Nullable)(NSString *msgId))callback;

// 以下两个方法被移除
- (BOOL)sendSpeakMessage:(PLVSpeakMessage *)message replyMsgId:(NSString * _Nullable)replyMsgId
needIdCallback:(BOOL)needIdCallback;
- (BOOL)sendSpeakMessage:(PLVSpeakMessage *)message replyMsgId:(NSString * _Nullable)replyMsgId
needIdCallback:(BOOL)needIdCallback callback:(void (^ _Nullable)(NSString *msgId))callback;
// 替换为以下两个方法
- (BOOL)sendQuoteMessage:(PLVQuoteMessage *)message;
- (BOOL)sendQuoteMessage:(PLVQuoteMessage *)message callback:(void (^ _Nullable)(NSString *msgId))callback;

// 该方法被移除
- (BOOL)sendImageEmotionMessage:(PLVImageEmotionMessage *)message callback:(void (^ _Nullable)
(PLVImageEmotionMessage *message))callback;
// 替换为以下方法
- (BOOL)sendImageEmotionMessage:(PLVImageEmotionMessage *)message;
```

5.1.2 PLVSocketManager

1. 接收图片表情消息（emotion）消息的回调发生如下变动：

```
- (void)socketMananger_didReceiveMessage:(NSString *)subEvent
                                    json:(NSString *)jsonString
                              jsonObject:(id)object {
    NSDictionary *jsonDict = (NSDictionary *)object;
    if (![jsonDict isKindOfClass:[NSDictionary class]]) {
        return;
    }
    if ([subEvent isEqualToString:@"EMOTION"]) {   // someone logged in chatroom
        //1.7.0以前的版本接收到图片表情消息的位置
    }
}

- (void)socketMananger_didReceiveEvent:(NSString *)event subEvent:(NSString *)subEvent json:(NSString
*)jsonString jsonObject:(id)object {
    NSDictionary *jsonDict = PLV_SafeDictionaryForValue(object);
    if ([event isEqualToString:@"emotion"]) {// someone send a image emotion
       //1.7.0以后（含1.7.0版本）接收到图片表情消息的位置
   }
}
```

5.2 Common

若升级完 SDK 和 Common 层的源码，同步也升级了 Scene 层的源码，则无需关注以下改动；若只升级 SDK 和 Common 层的源码，不升级 Scene 层的源码，或没有使用 Scene 层的源码的客户，则需要针对以下改动做相应的修改。

5.2.1 PLVChatroomPresenter

1. `PLVChatroomPresenterProtocol` 有以下两个 delegate 方法发生变动

```
/// 以下回调被删除
- (void)chatroomPresenter_sendImageEmotionMessageStatus:(PLVImageEmotionMessage *)message;

/// 以下回调有更改
- (void)chatroomPresenter_loadImageEmotionsSuccess:(NSArray <NSDictionary *> *)dictArray;
/// 改为
- (void)chatroomPresenter_loadImageEmotionsSuccess;
```

2. API 的改动如下：

```
// 该方法被移除
- (PLVChatModel * _Nullable)chatModelWithMsgStateForSendSpeakMessage:(NSString *)content replyChatModel:
(PLVChatModel * _Nullable)replyChatModel;
// 可用以下方法代替
- (PLVChatModel * _Nullable)sendSpeakMessage:(NSString *)content replyChatModel:(PLVChatModel *
_Nullable)replyChatModel;

// 该方法被删除
- (PLVChatModel * _Nullable)sendImageEmotionId:(NSString *)imageId;
// 改为以下方法
- (PLVChatModel * _Nullable)sendImageEmotionId:(NSString *)imageId imageUrl:(NSString *)imageUrl;

// 该方法被删除，改为直接调用 [[PLVChatroomManager sharedManager] sendCloseRoom:close];
- (BOOL)sendCloseRoom:(BOOL)closeRoom;
```

3. 以下属性改动：

```
@property (nonatomic, assign) BOOL specialRole;
// 布尔属性 specialRole 改为对外只读，无需外部配置
@property (nonatomic, assign, readonly) BOOL specialRole;

// 以下属性删除，改为直接通过 [PLVChatroomManager sharedManager].closeRoom 读取
@property (nonatomic, assign, readonly) BOOL closeRoom;

// 新增以下属性，可用于获取图片表情资源数组
@property (nonatomic, strong, readonly) NSArray *imageEmotionArray;
```

4. 内部逻辑改动：

```
// Scene 层无需再调用 ViewModel 的以下方法加载图片表情资源，Common 层的 PLVChatroomPresenter 初始化时会自动调用
- (void)loadImageEmotions;
```

5.3 Scene

若升级了 SDK、Common 层、Scene 层 ViewModel 的源码后，同步升级 UI 部分的源码，则无需关注以下改动；若 Scene 层未升级 UI 部分的源码，只升级了聊天室的 ViewModel，则需要针对以下改动做相应的修改。

5.3.1 PLVLCChatroomViewModel

1. 协议 `PLVLCChatroomViewModelProtocol` 的 delegate 方法有以下改动：

```
// 该方法被删除
- (void)chatroomViewModel_loadEmotionSuccess;
// 改为以下方法代替
- (void)chatroomViewModel_loadImageEmotionSuccess:(NSArray<NSDictionary *> *)dictArray;

// 新增以下方法，加载图片表情资源失败时触发
- (void)chatroomViewModel_loadImageEmotionFailure;
```

5.3.2 PLVECChatroomViewModel

1. 协议 `PLVECChatroomViewModelProtocol` 的 delegate 方法有以下改动：

```
// 删除以下图片表情资源加载相关的回调，因为直播带货场景没有图片表情消息功能
- (void)chatroomViewModel_loadEmotionSuccess;
```

5.3.3 PLVLSChatroomViewModel

1. 协议 `PLVLSChatroomViewModelProtocol` 的 delegate 方法有以下改动：

```
// 该方法被删除
- (void)chatroomViewModel_loadEmotionSuccess;
// 改为以下方法代替
- (void)chatroomViewModel_loadImageEmotionSuccess:(NSArray<NSDictionary *> *)dictArray;

// 新增以下方法，加载图片表情资源失败时触发
- (void)chatroomViewModel_loadImageEmotionFailure;

// 新增以下方法，当发送消息包含违禁词或图片违规时触发
- (void)chatroomViewModel_didSendProhibitMessgae;
```

5.3.4 PLVSAChatroomViewModel

1. 协议 `PLVSAChatroomViewModelProtocol` 的 delegate 方法有以下改动：

```
// 该方法被删除
- (void)chatroomViewModel_loadEmotionSuccess;
// 改为以下方法代替
- (void)chatroomViewModel_loadImageEmotionSuccess:(NSArray<NSDictionary *> *)dictArray;

// 新增以下方法，加载图片表情资源失败时触发
- (void)chatroomViewModel_loadImageEmotionFailure;

// 新增以下方法
// 重发公聊消息后触发
- (void)chatroomViewModelDidResendMessage:(PLVSAChatroomViewModel *)viewModel;
// 当发送消息包含违禁词或图片违规时触发
- (void)chatroomViewModelDidSendProhibitMessgae:(PLVSAChatroomViewModel *)viewModel;
```

2. 删除以下属性：

```
// 以下属性删除，改为直接通过 [PLVChatroomManager sharedManager].closeRoom 读取
@property (nonatomic, assign, readonly) BOOL closeRoom;
```

3. 以下 API 改动：

```
// 该方法被删除，改为直接调用 [[PLVChatroomManager sharedManager] sendCloseRoom:close];
- (BOOL)sendCloseRoom:(BOOL)closeRoom;

// 该方法被删除
- (BOOL)resendSpeakMessage:(NSString *)content replyChatModel:(PLVChatModel * _Nullable)replyChatModel;
// 改为以下方法
- (BOOL)resendSpeakMessage:(PLVChatModel *)model replyChatModel:(PLVChatModel * _Nullable)replyChatModel;

// 该方法被删除
- (BOOL)resendImageMessage:(UIImage *)image imageId:(NSString *)imageId;
// 改为以下方法
- (BOOL)resendImageMessage:(PLVChatModel *)model;

// 该方法被删除
- (BOOL)resendImageEmotionMessage:(NSString *)imageId imageUrl:(NSString *)imageUrl;
// 改为以下方法
- (BOOL)resendImageEmotionMessage:(PLVChatModel *)model;
```