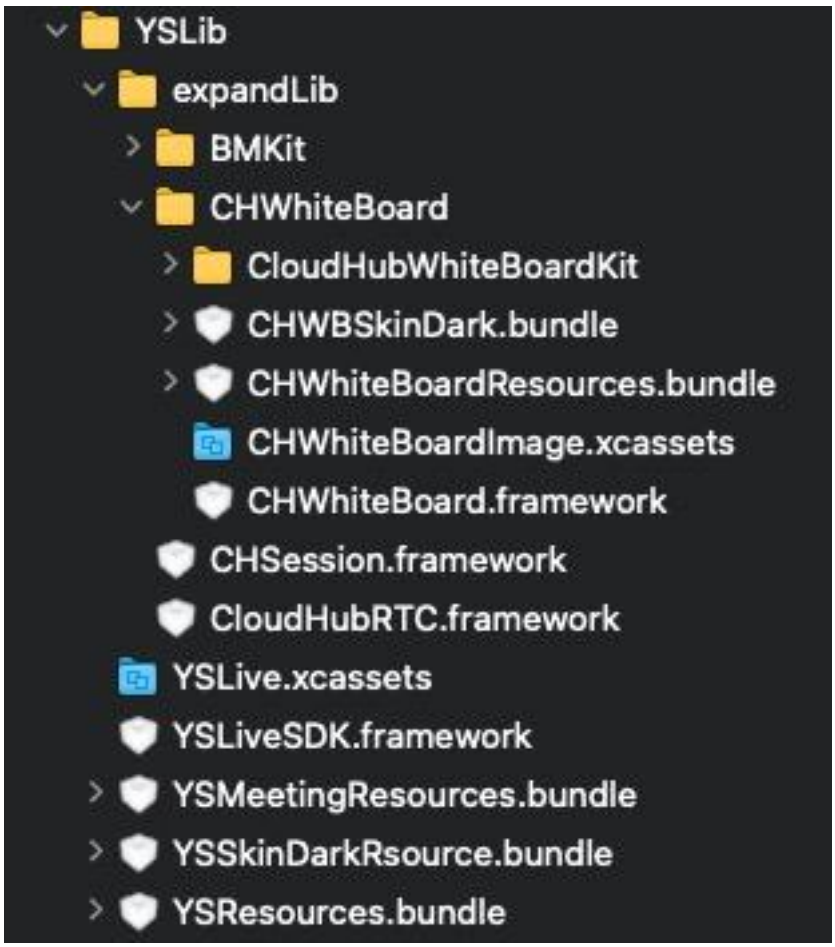
YSSDK For iOS 集成手册

V3.5.5.1

一. SDK工程结构



YSSDK.framwork: UI支持SDK

YSLive.xcassets: 图片资源

ExpendLib: 需要引用的库包括:

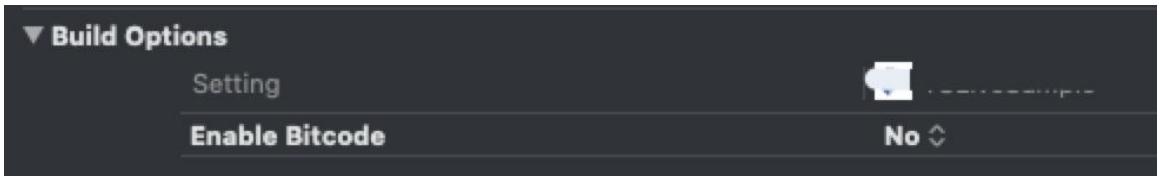
1. BMKit.framework: 基础库
2. CloudHubRTC.framework: 音视频SDK
3. CHSession.framework: 业务SDK
4. CHWhiteBoard.framework: 白板SDK
5. YSWhiteBoardResources.bundle: 白板资源文件
6. YSResources.bundle: 文本资源文件

二. 将以上文件添加到工程中

三. 配置工程文件

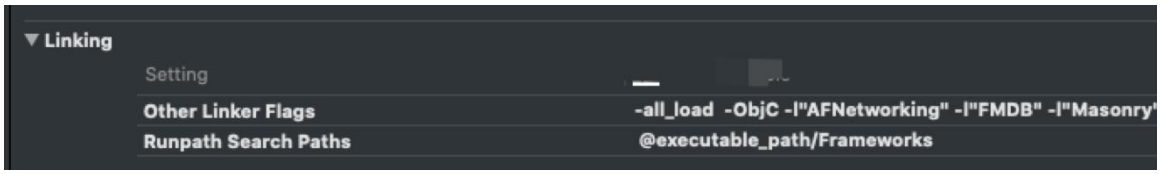
1. 工程配置修改(Build Settings)

1.1.Build Optins->Enable Bitcode->修改为NO

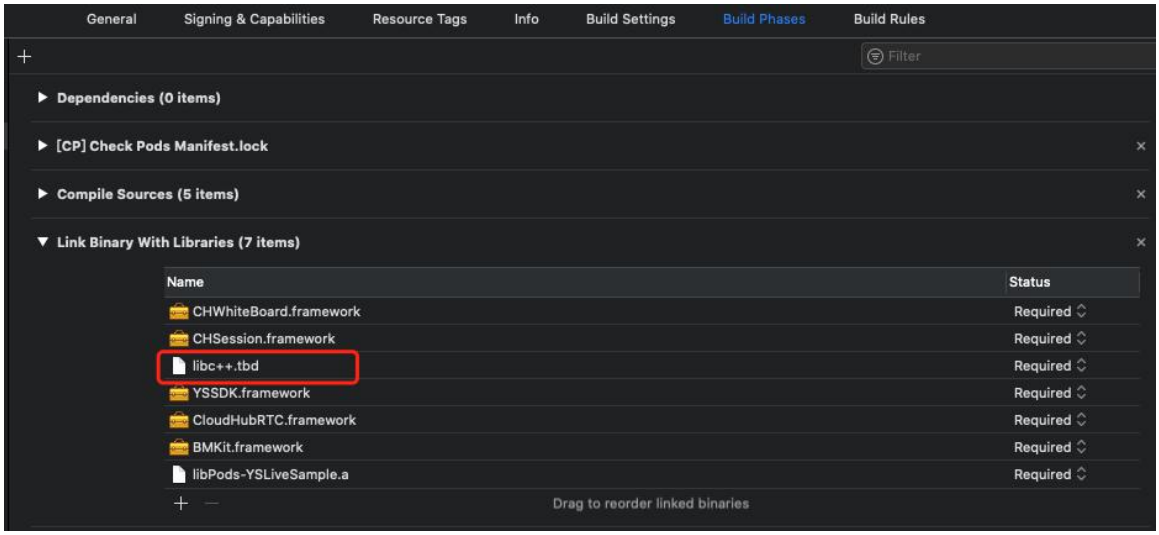


1.2.Linkink->Other Linker Flage添加

-all_load 和 -ObjC



2. Build Phases–>Link Binary With Libaries 添加必要依赖库: libc++.tbd

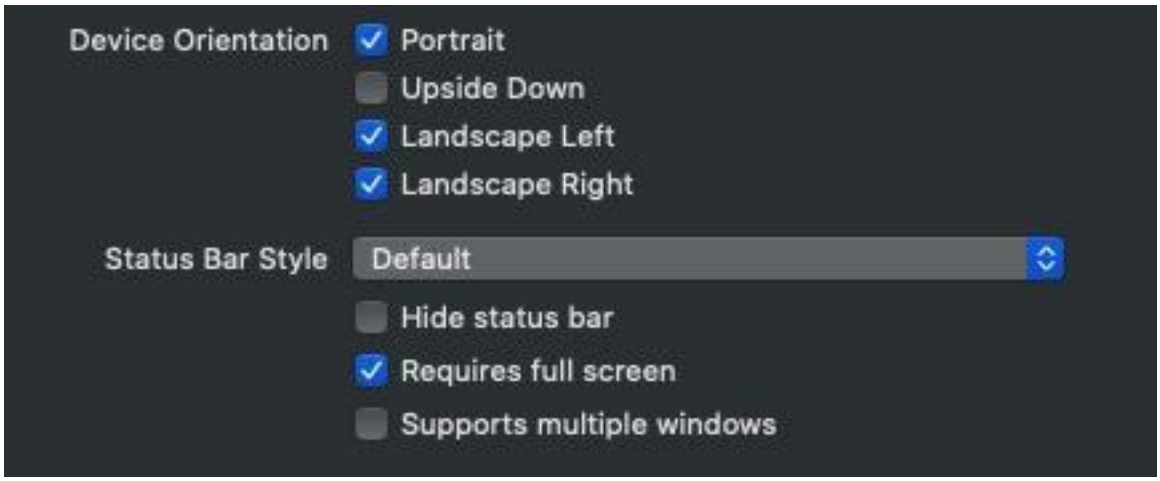


3. 需要添加第三方库 SSZipArchive，可通过CocoaPods安装也可手工添加，具体添加方法参照说明：

https://github.com/ZipArchive/ZipArchive

4. 工程支持旋转方向设置，请尽量参照sample来设置旋转

方法一：General–>Device Orientation设置如图



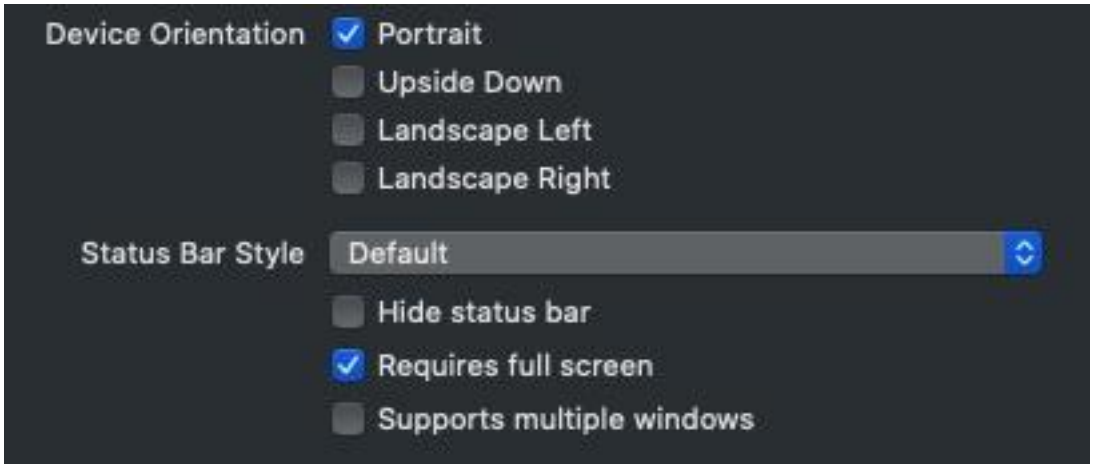
注意：如果支持pad，请检查并手工修改info.plist文件添加完整

对应需要设置sdk属性 useAppDelegateAllowRotation = NO

方法二：使用AppDelegate强制控制转屏方向，不建议使用此方法

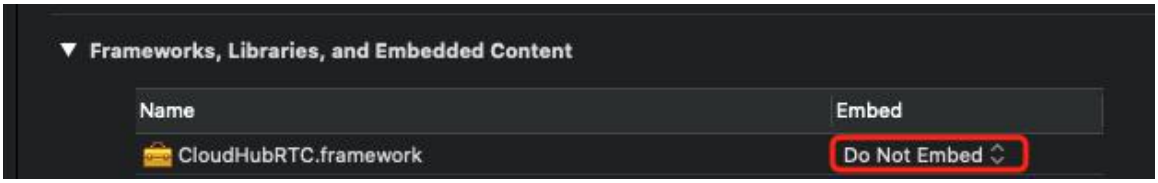
```
- (UIInterfaceOrientationMask)application:(UIApplication *)application
supportedInterfaceOrientationsForWindow:(UIWindow *)window
{
    if (self.allowRotation)
    {
        return UIInterfaceOrientationMaskLandscapeRight;
    }
    else
    {
        return UIInterfaceOrientationMaskPortrait;
    }}
```

需要修改General->Device Orientation设置如图



对应需要设置sdk属性 useAppDelegateAllowRotation = YES

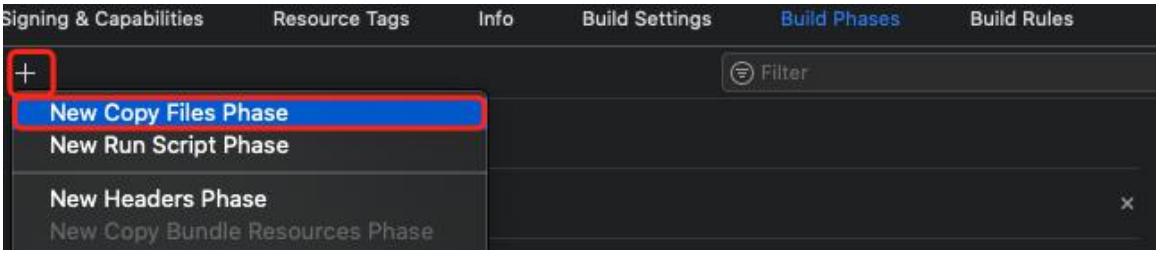
5. 将CloudHubRTC.framework 拷贝到工程中



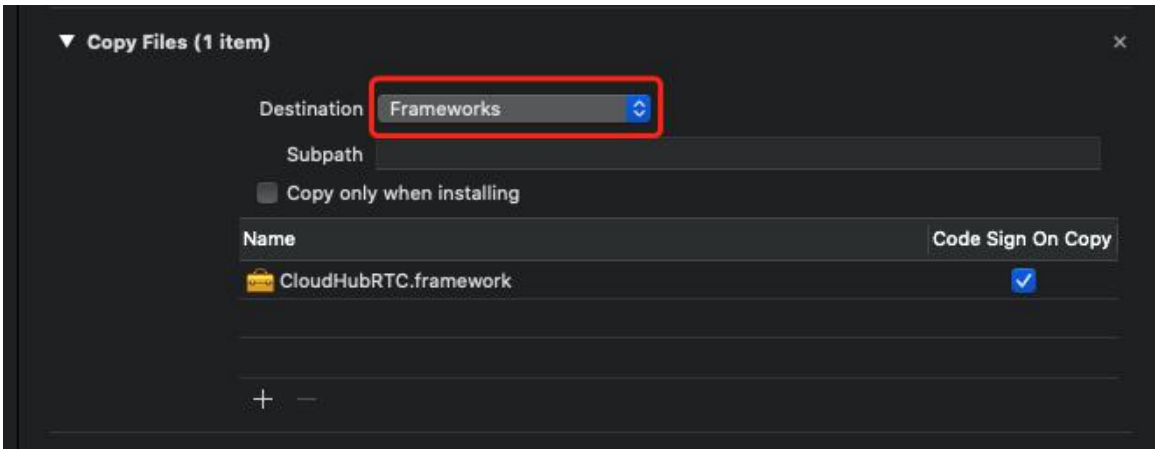
方法1：直接将CloudHubRTC.framework的 Embed选项设置为Embed & Sign

方法2：保持CloudHubRTC.framework 的Embed选项为Do Not Embed默认值

然后在Build Phases添加New Copy Files Phase项



将CloudHubRTC.framework加入



6. 语言设置 info.plist -> Localization native development region->China

   请根据自己需求设置语言，目前SDK支持中文简体，中文繁体和英文

7. 设置权限

   info.plist -> Privacy - Camera Usage Description -> 房间中需要进行视

频通话以及拍摄您是否允许打开相机

info.plist –> Privacy – Microphone Usage Description –> 房间中需要发送语音消息及发言您是否允许打开麦克风

info.plist –> Privacy – Photo Library Usage Description –> 房间中需要选择本地图片您是否允许访问相册

info.plist –> Privacy – Photo Library Additions Usage Description –> 房间中需要上传图片您是否允许添加图片

info.plist –>App Transport Security Settings–>Allow Arbitrary Loads–>YES

或将以下代码加入info.plist

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
<key>NSCameraUsageDescription</key>
<string>房间中需要进行视频通话以及拍摄您是否允许打开相机</string>
<key>NSLocationWhenInUseUsageDescription</key>
<string>房间中需要通过您的地理位置信息获取您周边的位置相关数据您是否允许开启位置</string>
<key>NSMicrophoneUsageDescription</key>
<string>房间中需要发送语音消息及发言您是否允许打开麦克风</string>
<key>NSPhotoLibraryAddUsageDescription</key>
<string>房间中需要上传图片您是否允许添加图片</string>
<key>NSPhotoLibraryUsageDescription</key>
<string>房间中需要选择本地图片您是否允许访问相册</string>
<key>UIBackgroundModes</key>
<array>
    <string>audio</string>
</array>
```

四. 代码调用

1. 导入头文件并初始化

```
#import <YSLiveSDK/YSSDKManager.h>

@interface YSLoginVC ()
<
    YSSDKDelegate
>
@property (nonatomic, weak) YSSDKManager *ysSDKManager;
@end
- (void)viewDidLoad
{
    [super viewDidLoad];

    self.ysSDKManager = [YSSDKManager sharedInstance];
    [self.ysSDKManager registerManagerDelegate:self];
}
```

2. 检查房间类型

```
__weak __typeof(self) weakSelf = self;
    [self.ysSDKManager checkRoomTypeBeforeJoinRoomWithRoomId:roomId success:^(YSSDKUseTheType
roomType, BOOL needpassword) {
        // roomType：房间类型 3：小班课  4：直播   6：会议
        // needpassword：参会人员(学生)是否需要密码
        if (self->userRole == YSSDKSUserType_Student)
        {
            // 学生登入
            // 注意：  直播只支持学生身份登入房间
            [weakSelf.ysSDKManager joinRoomWithRoomId:roomId nickName:nickName roomPassword:nil
userId:nil userParams:nil];
        }
        else
        {
            // 老师(会议主持)登入
            // 注意：  小班课和会议支持老师和学生身份登入房间
            [weakSelf.ysSDKManager joinRoomWithRoomId:roomId nickName:nickName roomPassword:nil
userRole:self->userRole userId:nil userParams:nil];
        }

    } failure:^(NSInteger code, NSString * _Nonnull errorStr) {
        NSLog(@"code:%@, message：%@", @(code), errorStr);
        [self.progressHUD hideAnimated:YES];
    }];
```

3. 进入房间

```
    // 学生登入
    // 注意：  直播只支持学生身份登入房间
    [weakSelf.ysSDKManager joinRoomWithRoomId:roomId nickName:nickName roomPassword:nil userId:nil
userParams:nil];
```

```
    // 老师(会议主持)登入
    // 注意： 小班课和会议支持老师和学生身份登入房间
    [weakSelf.ysSDKManager joinRoomWithRoomId:roomId nickName:nickName roomPassword:nil
userRole:self->userRole userId:nil userParams:nil];
```

4. 状态回调

```
#pragma mark -
#pragma mark YSSDKDelegate

/**
   成功进入房间
   @param ts 服务器当前时间戳，以秒为单位，如 1572001230
   @param roomType 房间类型
   @param userType 登入用户身份
*/
-  (void)onRoomJoined:(NSTimeInterval)ts  roomType:(YSSDKUseTheType)roomType
userType:(YSSDKUserRoleType)userType;

/**
    失去连接
 */
- (void)onRoomConnectionLost
{    NSLog(@"onRoomConnectionLost");
}

/**
    已经离开房间
 */
- (void)onRoomLeft
{    NSLog(@"onRoomLeft");
}

/**
    自己被踢出房间
    @param reason 被踢原因
 */
- (void)onRoomKickedOut:(NSDictionary *)reason
{    NSLog(@"onRoomKickedOut");
}

/**
    发生密码错误 回调
    需要重新输入密码

    @param errorCode errorCode
 */
- (void)onRoomNeedEnterPassWord:(YSSDKErrorCode)errorCode
{    NSLog(@"onRoomNeedEnterPassWord");
}
```

```
/**
    发生其他错误 回调
    需要重新登陆

    @param errorCode errorCode
*/
- (void)onRoomReportFail:(YSSDKErrorCode)errorCode descript:(NSString *)descript
{    NSLog(@"onRoomReportFail");
}

/**
   已经进入直播房间
*/
- (void)onEnterLiveRoom;

/**
   已经进入小班课(会议)房间
*/
- (void)onEnterClassRoom;
```

回调- (void)onRoomReportFail:descript:中的

错误编码YSSDKErrorCode参照 YSSDKDefine.h 中的定义

5. 房间需要重新输入密码时的操作

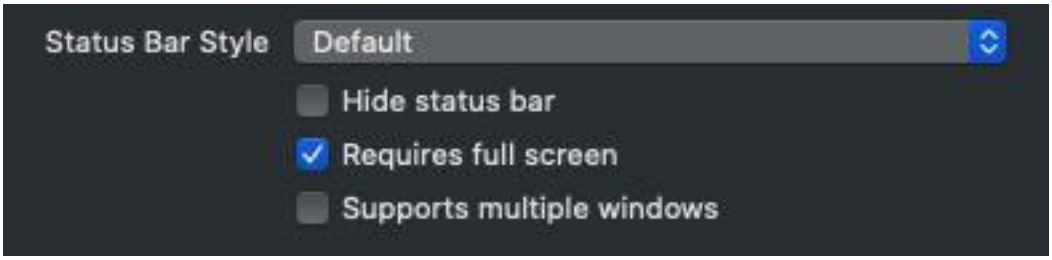
```
- (void)roomManagerNeedEnterPassWord:(YSSDKErrorCode)errorCode
{
    NSLog(@"roomManagerNeedEnterPassWord");

    // 重新设置密码 password 后调用
    [self.ysSDKManager joinRoomWithRoomId:roomId nickName:nickName roomPassword:password
userId:nil userParams:nil];
}
```

五． 补充一些屏幕旋转的说明

由于本SDK在运行时会将屏幕旋转，在退出房间后返还前置页面窗口时需要还原屏幕方向，需要注意以下几点：

1. 设置fullScreen开关



2. 根据自己的UI架构选择适配方法

```
// 根控制器是导航控制器 UINavigation，那么在这个导航控制器中实现下面三个方法
-(BOOL)shouldAutorotate {
    return [[self.viewControllers lastObject] shouldAutorotate];
}

-(NSUInteger)supportedInterfaceOrientations {
    return [[self.viewControllers lastObject] supportedInterfaceOrientations];
}

- (UIInterfaceOrientation)preferredInterfaceOrientationForPresentation {
    return [[self.viewControllers lastObject] preferredInterfaceOrientationForPresentation];
}
```

```
// 根控制器是 tabBar 控制器 UITabController，那么在这个 tabBar 控制器中实现下面三个方法
-(BOOL)shouldAutorotate {
    return [self.selectedViewController shouldAutorotate];
}

-(NSUInteger)supportedInterfaceOrientations {
    return [self.selectedViewController supportedInterfaceOrientations];
}

- (UIInterfaceOrientation)preferredInterfaceOrientationForPresentation {
    return [self.selectedViewController preferredInterfaceOrientationForPresentation];
}
```

以上都是为了将控制权转交给UIViewController，需要在你的ViewController实现屏幕方向设置

例如：竖屏

```
// 竖屏方向
- (BOOL)shouldAutorotate {
    return YES;
}

- (UIInterfaceOrientationMask)supportedInterfaceOrientations {
    return UIInterfaceOrientationMaskPortrait;
}

- (UIInterfaceOrientation)preferredInterfaceOrientationForPresentation {
    return UIInterfaceOrientationPortrait;
}
```

3. - (BOOL)shouldAutorotate的返回值区别

YES: 使用supportedInterfaceOrientations及preferredInterfaceOrientationForPresentation设置当前页面旋转，不能超出全局设置范围

NO: 根据系统全局设置，- (UIInterfaceOrientationMask)application:(UIApplication *)application supportedInterfaceOrientationsForWindow:(UIWindow *)window相关的设置进行页面旋转

具体操作：

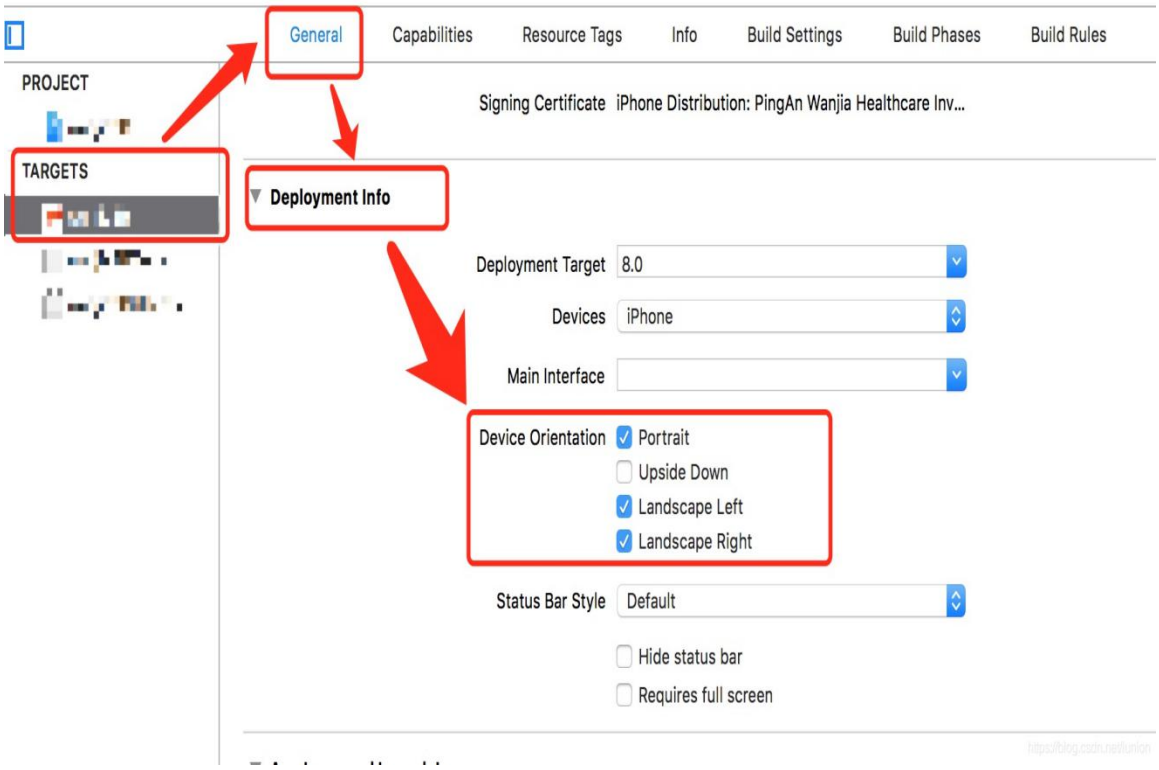
首先，全局控制

方式一 Info.plist 文件中，有一个 Supported Interface Orientations，可以配置整个应用的屏幕方向，此处为全局控制。

例：Info.plist 关于旋转的设置

```
<key>UISupportedInterfaceOrientations</key>
<array>
<string>UIInterfaceOrientationPortrait</string>
<string>UIInterfaceOrientationLandscapeLeft</string>
<string>UIInterfaceOrientationLandscapeRight</string>
</array>
<key>UISupportedInterfaceOrientations~ipad</key>
<array>
<string>UIInterfaceOrientationPortrait</string>
<string>UIInterfaceOrientationPortraitUpsideDown</string>
<string>UIInterfaceOrientationLandscapeLeft</string>
```

<string>UIInterfaceOrientationLandscapeRight</string>
</array>



方式二，UIWindow 的方向控制（iOS6 及以上版本才有效）

UIApplicationDelegate 提供了下述方法，能够指定 UIWindow 中的界面的屏幕方向：

- (UIInterfaceOrientationMask)application:(UIApplication *)application supportedInterfaceOrientationsForWindow:(nullable UIWindow *)window API_AVAILABLE(ios(6.0)) API_UNAVAILABLE(tvos);

此方法默认值为 Info.plist 中配置的 Supported Interface Orientations 项的值。iOS 中通常只有一个 window，所以此处的控制也可以视为全局控制。

所以，全局 Interface Orientation 的确定依据，方式二，无方式二时依据方式一

UIViewController 的方向控制

iOS6 以及以后版本中控制屏幕旋转相关方法：

限制

- 当前 controller 是 window 的 rootViewController
- 当前 controller 是 modal 模式的(present 出来的)

iOS10 以及之前版本变化

- iOS6~iOS9 中 modal 的 controller 需要 rootViewController 做特殊处理调用旋转
- iOS10+ model 的 controller 可以独立处理

只有以上两种情况时候，以下的 orientations 相关方法才会起作用(才会被调用)，当前 controller 及其所有的 childViewController 都在此作用范围内。

- #pragma mark 横竖屏
-
- /// 1.决定当前界面是否开启自动转屏，如果返回 NO，后面两个方法也不会被调用，只是会支持只会支持默认的 UIInterfaceOrientationMaskPortrait 方向
- //@property(nonatomic, readonly) BOOL shouldAutorotate API_AVAILABLE(ios(6.0)) API_UNAVAILABLE(tvos);
- - (BOOL)shouldAutorotate NS_AVAILABLE_IOS(6_0) __TVOS_PROHIBITED;
-
- /// 2.返回支持的旋转方向
- /// iPhone 设备上，默认返回值 UIInterfaceOrientationMaskAllButUpSideDwon
- /// iPad 设备上，默认返回值是 UIInterfaceOrientationMaskAll
- //@property(nonatomic, readonly) UIInterfaceOrientationMask supportedInterfaceOrientations API_AVAILABLE(ios(6.0)) API_UNAVAILABLE(tvos);
- (UIInterfaceOrientationMask)supportedInterfaceOrientations NS_AVAILABLE_IOS(6_0) __TVOS_PROHIBITED;
-
- /// 3.返回进入界面优先方向
- //@property(nonatomic, readonly) UIInterfaceOrientation preferredInterfaceOrientationForPresentation API_AVAILABLE(ios(6.0)) API_UNAVAILABLE(tvos);
- - (UIInterfaceOrientation)preferredInterfaceOrientationForPresentation NS_AVAILABLE_IOS(6_0) __TVOS_PROHIBITED;

- - shouldAutorotate

方法决定是否支持多方向旋转屏，如果返回 NO 则后面的两个方法都不会再被调用，而且只会支持默认的 UIInterfaceOrientationMaskPortrait 方向；

- - supportedInterfaceOrientations

  直接返回支持的旋转方向，该方法在 iPad 上的默认返回值是 UIInterfaceOrientationMaskAll，iPhone 上的默认返回值是 UIInterfaceOrientationMaskAllButUpsideDown，详情见官方 Q&A 文档；

- - preferredInterfaceOrientationForPresentation

  返回最优先显示的屏幕方向，比如同时支持 Portrait 和 Landscape 方向，但想优先显示 Landscape 方向，那软件启动的时候就会先显示 Landscape，在手机切换旋转方向的时候仍然可以在 Portrait 和 Landscape 之间切换；

最终支持的屏幕方向

- 前面所述的 3 种控制规则的交集就是一个 controller 的最终支持的方向，就是说：一个界面最后支持的屏幕方向，是取（全局控制 ∩ UIWindow 中的界面控制 ∩ 单个界面控制）的交集，如果全局控制支持所有屏幕方向，UIWindow 中的界面控制支持横屏，当个界面中只是支持横屏向右，那么最后界面只会以横屏向右显示，并且不支持旋转到其他的方向。
- 如果以上三种控制支持的屏幕方向最后的交集为空，在 iOS 6 以上会直接抛出 UIApplicationInvalidInterfaceOrientationException 的异常，然后直接崩溃，所以还是要保持这三个值的交集为非空。

关于：'UIApplicationInvalidInterfaceOrientation'，reason：'Supported orientations has no common orientation with the application，and [XXX shouldAutorotate] is returning YES'
terminating with uncaught exception of type NSException
原因是某个 VC 的 shouldAutorotate 方法返回 YES，而－(UIInterfaceOrientationMask)supportedInterfaceOrientations 返回的支持方向超出了全局的设置
如：全局设置 UIInterfaceOrientationPortrait 时，supportedInterfaceOrientations 返回了 UIInterfaceOrientationLandscapeRight 就会提示上述崩溃信息