

Vispek SDK IOS 端接入指引

通过集成 libWSBKSDK 移动端 SDK，使客户能够在 App 上快速具有连接蓝牙获取硬件设备信息的能力。

1 准备环境

请确保开发环境满足以下技术要求：

- iOS 10.0 或以上版本
- 支持真机, 暂无模拟器

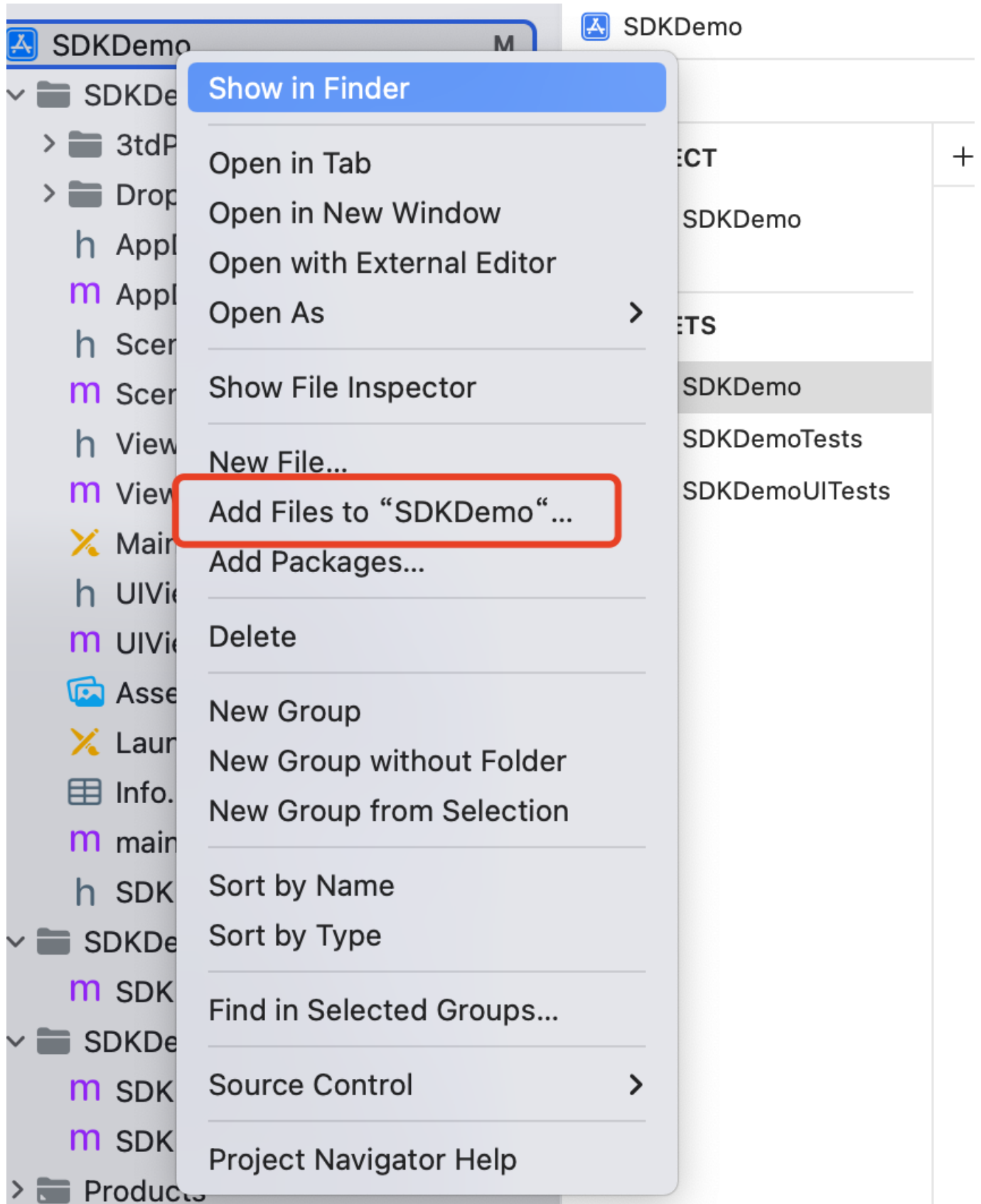
2 获取 SDK

请联系 Vispek 技术支持获取需要的 libWSBKSDK SDK。

3 集成 SDK

3.1 添加 SDK 动态库文件

1. 手动将 `libWSBKSDK.framework` 拷贝到项目目录下。
2. 打开 Xcode，使用 [Add Files to "xxx" (xxx 为用户的项目名)]，添加上述 SDK 动态库文件到项目。



3.2 导入 SDK

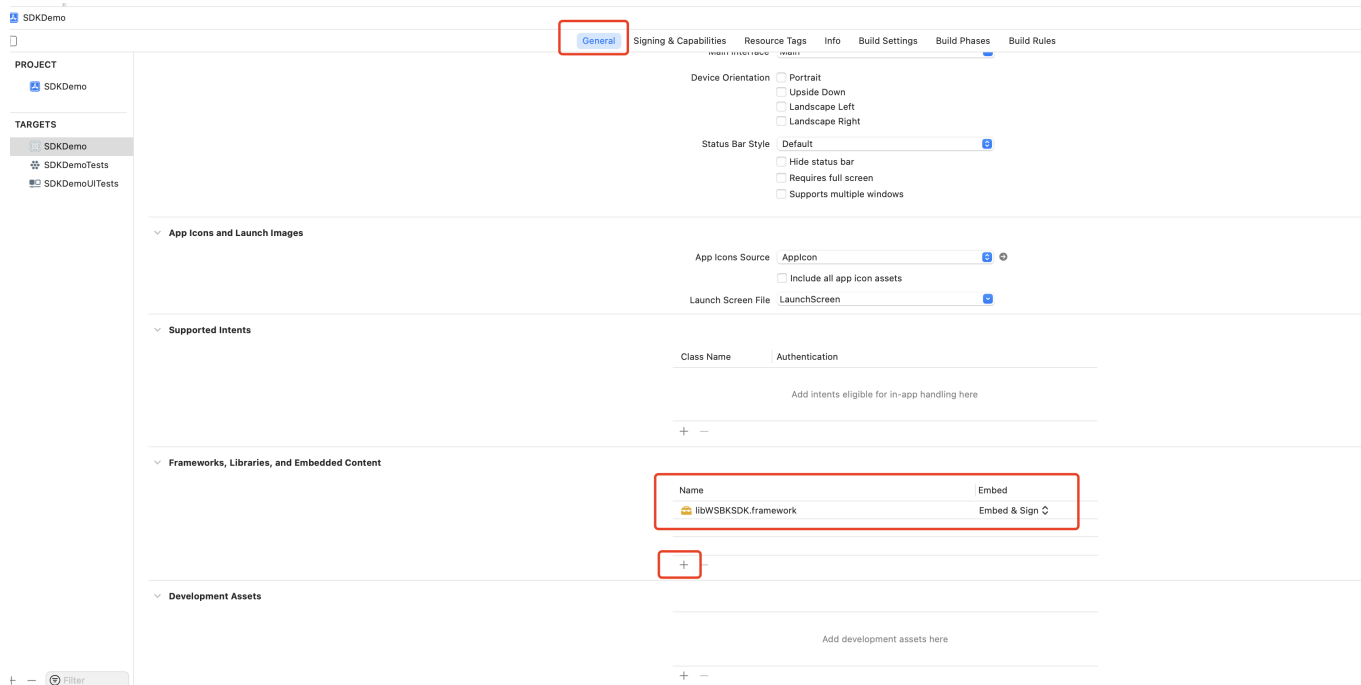
注意，SDK 库文件中有两个文件夹：iphoneos 和 iphoneos_simulator，区别如下：

- iphoneos 仅用于真机调试。用户在最终发布时，需要使用此文件下的 framework 文件，否则可能被苹果打回。

- iphonos_simulator 包含了真机和模拟器调试的库。如果用户开发过程中使用模拟器调试，需要导入此文件夹下的 framework 文件。但是最终发布时，要切换回 iphoneos 文件下的 framework 文件。

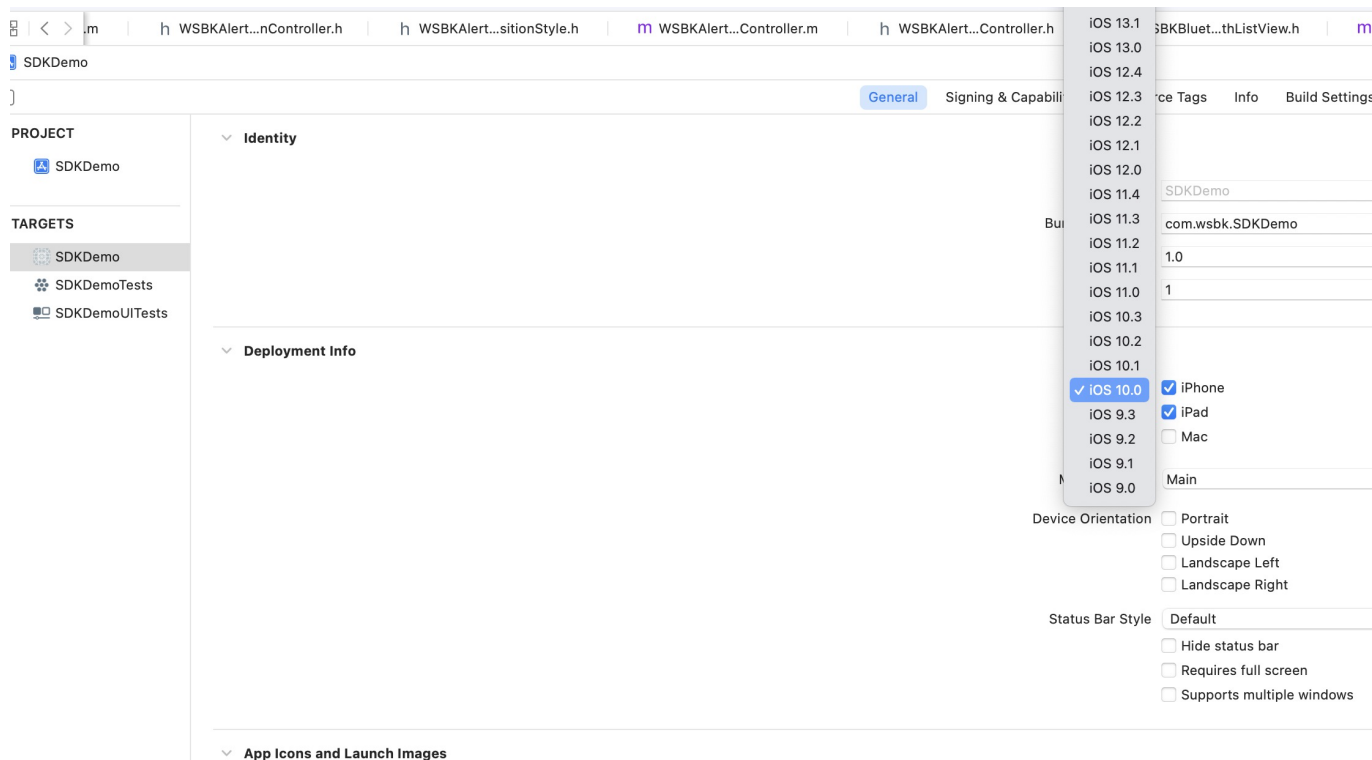
注意，在下面的设置步骤中，请选择符合开发要求的 framework 文件。

1. 打开 Xcode，选择：项目 TARGET -> General -> Frameworks, Libraries, and Embedded Content，添加 libWSBKSDK.framework，并设置 Embed 为 Embed & Sign。



3.3 项目设置

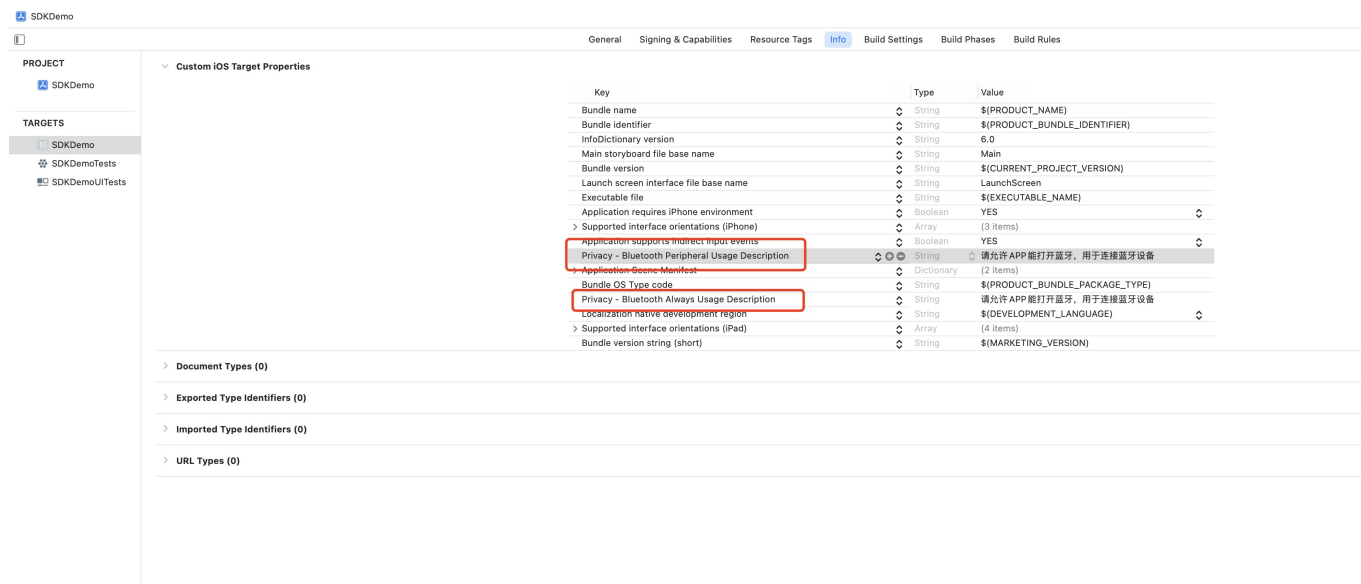
1. 打开 Xcode，选择：项目 TARGET -> General -> Deployment Target，设置 10.0 或以上版本。



2. 添加蓝牙相关权限。选择项目 TARGETS -> Info -> Custom iOS Target Properties, 点击 + 添加按钮, 添加蓝牙权限。添加的项有:

Privacy - Location Always and When In Use Usage Description
Privacy - Bluetooth Always Usage Description

添加完成后如图所示:



△注意: AppStore 上架审核时, 对于使用了蓝牙相关的框架, 需要自己录制app的所有操作生成视频链接, 放到苹果审核备注中

4 实现

本节介绍如何调用 libWSBKSDK.framework 实现业务:

4.1 调起 libWSBKSDK

根据场景需要, 在合适的入口调起 libWSBKSDK 流程, 同时在相关文件中调用 import **libWSBKSDK** 的头文件。

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

在文件声明中遵循代理

使用如下 API 初始化以及配置蓝牙相关设置代理:

```
[WSBKBlueToothSDK shared].delegate = self;
```

4.2 开始扫描蓝牙设备

使用如下 API 开始扫描蓝牙设备：

```
/**
 开始扫描周边蓝牙设备
 */
- (void)startScanPeripheral;
```

4.3 选择需要连接到蓝牙设备和停止开始扫描蓝牙设备

```
/**
 停止扫描
 */
- (void)stopScanPeripheral;

/**
 连接所选中的蓝牙外设
 @param peripheral 所选择蓝牙外设的perioheral
 */
- (void)connectPeripheral:(CBPeripheral *)peripheral;
```

4.4 根据需要在用到的地方实现代理

```
/**
 系统蓝牙被关闭、提示用户去开启蓝牙
 */
- (void)systemBluetoothClose;

/**
 系统蓝牙已开启、开始扫描周边的蓝牙设备
 */
- (void)sysytemBluetoothOpen;

/**
 扫描到的设备回调

 @param peripheralInfoArr 扫描到的蓝牙设备数组
 */
- (void)getScanResultPeripherals:(NSArray *)peripheralInfoArr;

/**
 连接成功
 */
- (void)connectSuccess;
```

```

/**
 连接失败
 */
- (void)connectFailed;

/**
 当前设备断开连接

@param peripheral 断开的peripheral信息
 */
- (void)disconnectPeripheral:(CBPeripheral *)peripheral;

```

4.5 发送蓝牙指令获取数据

使用如下 API 发送蓝牙指令获取数据：

```

/// 相对应的指令
typedef NS_ENUM(NSUInteger, WSBKBlueToothSDKCommandType) {
    /// 电量
    WSBKBlueToothSDKBatteryCommandType,
    /// 光谱
    WSBKBlueToothSDKSpectralCommandType,
    /// 温度
    WSBKBlueToothSDKTemperatureCommandType,
    /// 硬件版本
    WSBKBlueToothSDKDeviceVersionCommandType,
    /// 设备编码
    WSBKBlueToothSDKDeviceSNCommandType
};

/// 发送指令获取蓝牙设备返回数据
/// @param commandType 指令类型
/// @param successBlock 获取数据成功回调
/// @param failedBlock 获取数据失败回调
- (void)sendCommandType:(WSBKBlueToothSDKCommandType)commandType
  getDeviceData:(void (^)(NSString *response, NSString
*response2))successBlock failed:(void (^)(NSString *failed))failedBlock;

```

该方法返回成功数据注意点：发送WSBKBlueToothSDKBatteryCommandType 、WSBKBlueToothSDKTemperatureCommandType、WSBKBlueToothSDKDeviceVersionCommandType、WSBKBlueToothSDKDeviceSNCommandType等类型发送指令返回的数据在response参数，response2为nil，发送WSBKBlueToothSDKSpectralCommandType指令获取成功的返回数据response为原始光谱数据，response2为参比值，具体使用可参考提供的Demo

4.5 断开蓝牙连接

使用如下 API 断开蓝牙连接：

```
/**
 断开当前连接的所有蓝牙设备
 */
- (void)disconnectAllPeripherals;

/**
 断开所选择的蓝牙设备

@param peripheral 所选择蓝牙外设的perioheral
 */
- (void)disconnectLastPeripheral:(CBPeripheral *)peripheral;
```