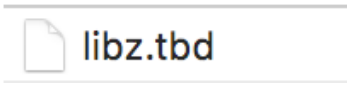
# GDBandSDK ios 文档

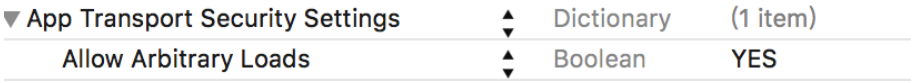
| 修改时间 | 修改人 | 修改内容 |
| --- | --- | --- |
| 2015-10-26 | Darren | 创建文档 |
| 2015-10-28 | Darren | 增加运动数据相关接口 |
| 2015-12-04 | Darren | 增加ANCS相关接口 |
| 2015-12-08 | Darren | 增加连接外部搜索到设备的接口 |
| 2016-02-02 | Darren | 增加获取实时步数、卡路里的接口 |
| 2016-03-22 | Darren | 修改初始化接口 |
| 2016-05-25 | Darren | 增加异步发送APDU指令的接口，增加APDU回调代理 |
| 2016-07-01 | Darren | 增加获取SDK版本号接口 |
| 2016-07-08 | Darren | 增加固件升级接口，修改初始化接口 |
| 2016-11-01 | Darren | 增加微信运动相关接口 |

# 一、接入说明

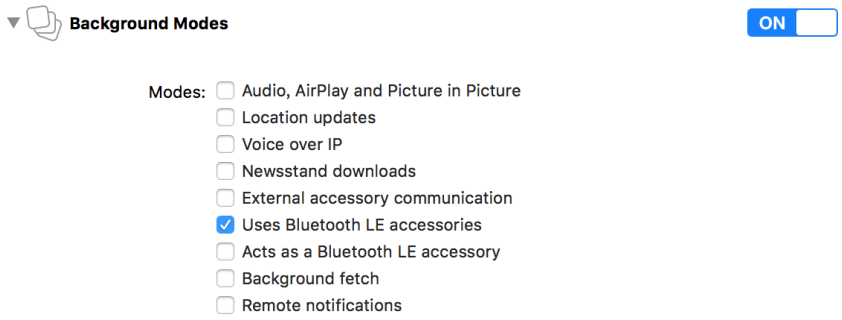
1、需要导入的依赖库：




2、ATS配置




3、固件升级过程中，App进入后台会导致升级中断，解决办法如下：




# 二、接口说明

```
NSString * const GDResponseResultName;                  // 操作结果
NSString * const GDResponseMajorVersionName;            // 当前固件主版本号
NSString * const GDResponseMinorVersionName;            // 当前固件副版本号
NSString * const GDResponseLatestMajorVersionName;      // 最新固件主版本号
NSString * const GDResponseLatestMinorVersionName;      // 最新固件副版本号
NSString * const GDResponseNeedUpdateName;              // 是否需要升级
NSString * const GDResponseLatestVersionProperty;       // 最新固件特性
NSString * const GDDFUInfoPercentageName;               // 升级进度

typedef NS_ENUM(NSInteger, GDErrorCode) {
    GDSucceed                       = 0,                // 成功
    GDNetworkError                  = -2101,            // 网络异常
    GDBLEError                      = -3002,            // 蓝牙异常
    GDConnectionError               = -3003,            // 连接异常
    GDTimeoutError                  = -3004,            // 异常超时
```

```
    GDNotSupportError               = -3101,                // 设备不支持
    GDInternalError                 = -3102,                // 内部异常
    GDAuthorizationError            = -4001,                // 授权验证失败，请检查
AppID、AAppKey
};

typedef NS_ENUM(NSInteger, GDFirmwareProperty) {
    GDFirmwarePropertyOptional      = 0,                    // 可选升级
    GDFirmwarePropertyRequired      = 1                     // 必须升级
};

@interface GDUserInfo : NSObject <NSCoding>
@property (nonatomic, assign) int height;                   // 身高 单位：厘米
@property (nonatomic, assign) int weight;                   // 体重 单位：千克
@property (nonatomic, assign) int age;                      // 年龄
@property (nonatomic, assign) int sex;                      // 性别 1:男 2:女
@property (nonatomic, assign) int walkingLength;            // 走路步长 单位：厘米
@property (nonatomic, assign) int runningLength;            // 跑步步长 单位：厘米
@end


typedef NS_OPTIONS(NSUInteger, GDDay) {
    GDDayMon                    = 1 <<  0,
    GDDayTues                   = 1 <<  1,
    GDDayWed                    = 1 <<  2,
    GDDayThur                   = 1 <<  3,
    GDDayFri                    = 1 <<  4,
    GDDaySat                    = 1 <<  5,
    GDDaySun                    = 1 <<  6
};

@interface GDBandInfo : NSObject <NSCoding>
@property (nonatomic, assign) int alertBeginHour;           // 活动提醒开始时间 范
围：0~23
@property (nonatomic, assign) int alertEndHour;             // 活动提醒结束时间 范
围：0~23
@property (nonatomic, assign) int alertInterval;            // 活动提醒间隔 可取值
为：10,20,30,40,50,60,70,80,90,100,110,120
// 例：bandInfo.alertDayFlag = GDDayMon | GDDayTues | GDDayWed;表示星期一、二、三开
启
@property (nonatomic, assign) int alertDayFlag;             // 活动提醒各天开关
@property (nonatomic, assign) BOOL alertEnable;             // 活动提醒总开关
@property (nonatomic, assign) int alarmHour;                // 智能闹钟时
@property (nonatomic, assign) int alarmMin;                 // 智能闹钟分
@property (nonatomic, assign) int alarmDayFlag;             // 智能闹钟各天开关
@property (nonatomic, assign) BOOL alarmEnable;             // 智能闹钟总开关
@property (nonatomic, assign) int battery;                  // 电池电量 剩余百分比
@end

@interface GDSportData : NSObject
@property (nonatomic, assign) NSTimeInterval time;          // 运动周期起始时间
@property (nonatomic, assign) int step;                     // 该运动周期产生的步数
@property (nonatomic, assign) float calorie;                // 该运动周期产生的卡路
里 单位：大卡
@property (nonatomic, assign) int distance;                 // 该运动周期产生的米数
单位：米
@end
```

```objc
@interface GDSleepData : NSObject
@property (nonatomic, assign) NSTimeInterval time;          // 睡眠时间段起始时间
@property (nonatomic, assign) int step;                     // 该睡眠周期内产生的运动量
@end

typedef NS_ENUM(NSInteger, GDSleepState) {
    GDSleepStateGotoSleep = 1,              // 入睡
    GDSleepStateLightSleep,                 // 浅度睡眠
    GDSleepStateDeepSleep,                  // 深度睡眠
    GDSleepStateWakeup,                     // 起夜
    GDSleepStateGetUp                       // 起床
};

@interface GDSleepDetail : NSObject
@property (nonatomic, assign) NSTimeInterval time;
@property (nonatomic, assign) GDSleepState state;
@end

@interface GDSleepDay : NSObject
@property (nonatomic, readonly) NSTimeInterval gotoSleepPoint;
    // 入睡时间 时间戳
@property (nonatomic, readonly) NSTimeInterval getUpPoint;
    // 起床时间 时间戳
@property (nonatomic, readonly) NSTimeInterval lightSleepTime;
    // 浅度睡眠时间 单位：秒
@property (nonatomic, readonly) NSTimeInterval deepSleepTime;
    // 深度睡眠时间 单位：秒
@property (nonatomic, readonly) NSTimeInterval wakeupTime;
    // 起夜时间 单位：秒
@property (nonatomic, readonly) NSArray<GDSleepDetail *> *detailArray;
    // 详细睡眠数据
@end

typedef NS_ENUM(NSInteger, GDBandManagerState) {
    GDBandManagerStateDisconnected  = 0,
    GDBandManagerStateScaning,
    GDBandManagerStateConnecting,
    GDBandManagerStateConnected
};

typedef NS_ENUM(NSInteger, GDBandANCSStatus) {
    GDBandANCSStatusOFF             = 0,        // ANCS功能关闭
    GDBandANCSStatusAllOn           = 1,        // 来电和消息提醒都开启
    GDBandANCSStatusCallON          = 2,        // 只开启来电提醒
    GDBandANCSStatusMessageON       = 3         // 只开启消息提醒
};

typedef NS_ENUM(NSInteger, GDBandDFUState) {
    GDBandDFUStateIdle              = 0,        // 空闲状态
    GDBandDFUStatePreparing         = 1,        // 准备中
    GDBandDFUStateWaiting           = 2,        // 等待中
    GDBandDFUStateTransfering       = 3,        // 升级中
};

@protocol GDBandManagerDelegate <NSObject>
@optional
// 搜索到手环回调
- (void)didDiscoverDevice:(CBPeripheral *)peripheral RSSI:(NSNumber *)RSSI;
```

```
// 连上手环回调
- (void)didConnectDevice;
// 发送六位随机码回调，需返回六位随机码字符串
- (NSString *)didSendValidateCode;
// 连接失败回调
- (void)didFailToConnectDevice;
// 手环断开连接
- (void)didDisconnectDevice;
// 手机蓝牙状态回调
- (void)centralManagerDidUpdateState:(CBCentralManagerState)state;
@end

@protocol GDBandManagerAPDUDelegate <NSObject>
// 异步发送APDU指令的回调    result:发送是否成功   responseData:返回数据
- (void)didSendApduResult:(BOOL)result responseData:(NSData *)responseData;
@end

@protocol GDBandDFUDelegate <NSObject>
// 固件升级开始
- (void)bandDFUDidStart;
// 回调升级状态
- (void)bandDFUDidUpdateState:(GDBandDFUState)state info:(NSDictionary *)info;
// 固件升级成功
- (void)bandDFUDidComplete;
// 固件升级中断
- (void)bandDFUDidCancelWithError:(NSInteger)errorCode;
@end

@interface GDBandManager : NSObject
@property (nonatomic, readonly) GDBandManagerState state;
@property (nonatomic, readonly) GDBandDFUState dfuState;
@property (nonatomic, readonly) CBCentralManagerState centralManagerState;
@property (nonatomic, readonly) CBPeripheral *currentDevice;
@property (nonatomic, weak) id<GDBandManagerDelegate> delegate;
@property (nonatomic, weak) id<GDBandManagerAPDUDelegate> apduDelegate;
// 初始化方法，请务必填写AppID、AppKey。（showAlert:蓝牙未打开时是否弹出系统提示框）
+ (void)initializeWithAppID:(NSString *)appid appKey:(NSString *)appKey
showSystemAlert:(BOOL)showAlert;
+ (GDBandManager *)sharedManager;
// 获取SDK当前版本号
+ (NSString *)sdkVersion;
// 开始搜索手环
- (BOOL)startDiscoverDevice;
// 停止搜索
- (void)stopDiscoverDevice;
// 连接手环。isfirst：是否新绑定手环
- (BOOL)connectDevice:(CBPeripheral *)device isFirst:(BOOL)isfirst;
// 连接外部搜索到的设备（非GDBandManager搜索到的设备）。
- (BOOL)connectOuterDevice:(CBPeripheral *)device isFirst:(BOOL)isfirst;
// 断开连接
- (BOOL)disconnectDevice;
// SE芯片上下电，并返回ATR。enable为YES时上电，NO为下电。
- (BOOL)SEChipOffOn:(BOOL)enable ATR:(uint8_t *)ATR length:(uint8_t *)length
timeout:(uint32_t) TimeOut;
// 发送APUD指令并接收返回信息
- (BOOL)APDUSendReceive:(SC_ADPU_Commands *)SC_ADPU response:(SC_ADPU_Response
*)SC_Response timeout:(uint32_t) TimeOut;
//  透传APDU指令并接收返回信息
- (NSData *)sendAPDUData:(NSData *)data timeout:(uint32_t) TimeOut;
```

```
// 异步发送APDU指令，通过apduDelegate回调发送结果（返回值为YES时表示调用成功且必有回调，请等待回调后再进行下一次调用）。
- (BOOL)asyncSendAPDUData:(NSData *)data;

/*
 运动数据相关接口
 */

// 获取设备类型以及固件版本号
- (BOOL)getDeviceTypeAndVersion:(void(^)(BOOL success, int type, int versionMajor, int versionMinor))completion;
// 获取设备内存储的用户信息
- (BOOL)getUserInfo:(void(^)(BOOL success, GDUserInfo *userInfo))completion;
// 获取设备内存储的手环信息
- (BOOL)getBandInfo:(void(^)(BOOL success, GDBandInfo *bandInfo))completion;
// 更新用户信息
- (BOOL)updateUserInfo:(GDUserInfo *)userInfo completion:(void(^)(BOOL success))completion;
// 更新手环信息
- (BOOL)updateBandInfo:(GDBandInfo *)bandInfo completion:(void(^)(BOOL success))completion;
// 更新设备时间
- (BOOL)updateDeviceTime:(void(^)(BOOL success))completion;
// 获取设备内存储的运动数据和睡眠数据
- (BOOL)getSportAndSleepData:(void(^)(BOOL success, NSArray *sportDataAry, NSArray *sleepDataAry))completion;
// 获取设备内存储的运动数据和睡眠数据(睡眠算法结果)
- (BOOL)getSportAndSleepResult:(void(^)(BOOL success, NSArray *sportDataAry, GDSleepDay *sleepDay))completion;
// 清除设备内存储的运动数据和睡眠数据
- (BOOL)clearSportAndSleepData:(void(^)(BOOL success))completion;
// 获取手环当前的ANCS状态
- (BOOL)getBandANCSStatus:(void(^)(BOOL success, GDBandANCSStatus status))completion;
// 设置手环当前的ANCS状态
/*
    注意：如果手环连接手机的时候状态为GDBandANCSStatusOFF，
    此时设置为其他开启状态后，都需要手环和手机重新连接才能正常收到ANCS消息。
 */
- (BOOL)updateBandANCSStatus:(GDBandANCSStatus)status completion:(void(^)(BOOL success))completion;

// 获取手环当前的实时步数、卡路里和距离
- (BOOL)getRealTimeStepCalorieDistance:(void(^)(BOOL success, int step, float calorie, float distance))completion;
// 清除手环显示的步数卡路里
- (BOOL)clearRealTimeDataWithCompletion:(void(^)(BOOL success))completion;

// 检查固件是否有更新
- (BOOL)checkBandFirmwareVersion:(void(^)(NSDictionary *response))completion;
// 升级最新的固件（必须先调用检查接口）
-(BOOL)startDFUWithDelegate:(id<GDBandDFUDelegate>)delegate;

// 注册微信运动
- (BOOL)registerWechatSport:(void(^)(BOOL success))completion;
// 查询微信运动注册状态
- (BOOL)queryWechatSportStatus:(void(^)(BOOL success, BOOL isOpen))completion;

@end
```

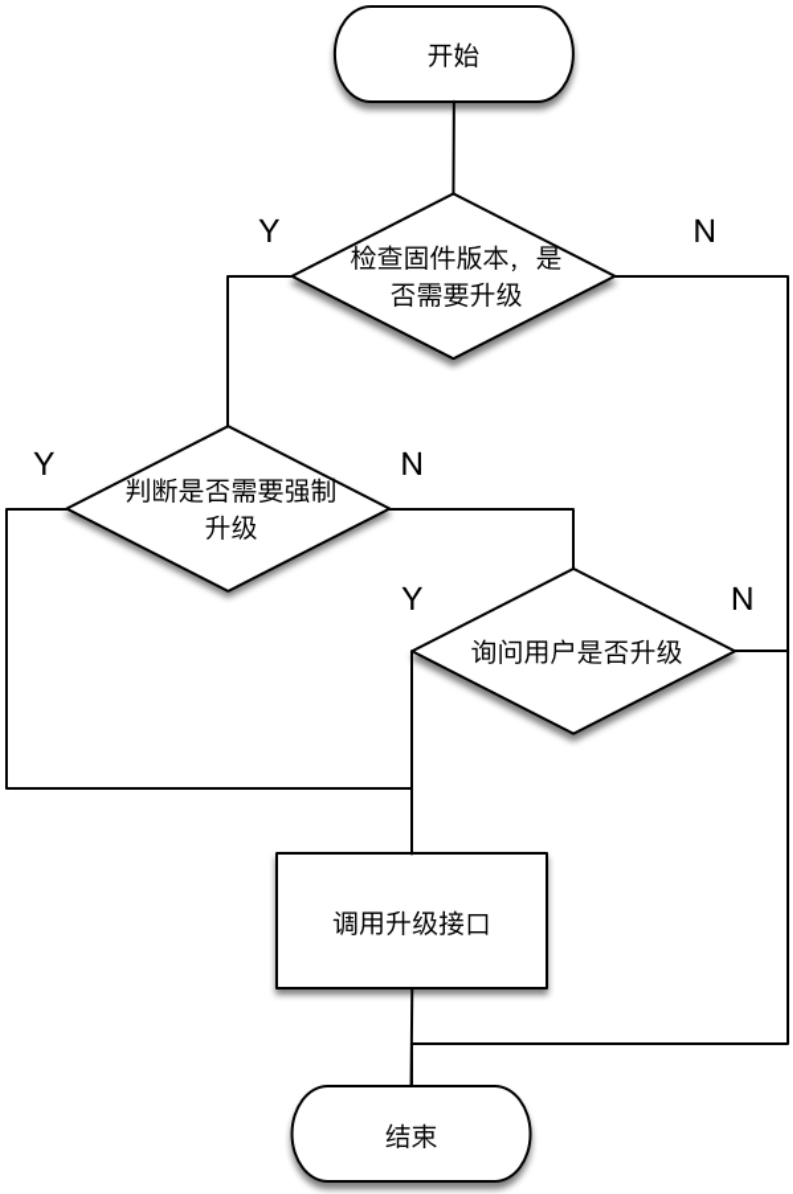
# 三、固件升级流程说明

## 功能描述

App调用SDK的固件升级功能来升级手环内的固件。

## 流程描述

App调用检查固件版本接口，SDK返回是否需要升级，以及最新版本的固件信息。如果需要升级，App调用升级固件接口完成升级。升级流程图如下：



### 注意事项

1、请务必在初始化时，填写正确的AppID、AppKey。

2、调用升级接口前，请务必调用检查接口。检查接口返回需要升级时，才能调用升级接口。

3、固件升级功能需要联网，由于苹果ATS限制，请在info.plist中配置，配置方法见文档开头的接入说明。

4、App进入后台时，升级过程会被中断，解决方法见文档开头的接入说明。

# 四、微信运动接入流程

### 功能描述

App调用微信运动相关接口，使手环能成功接入微信运动，同步运动数据。

### 流程描述

手环需要注册微信运动功能才能被微信运动识别并绑定。因此，手环在第一次启用微信运动前，需要调用SDK的注册微信运动接口，完成注册。一个手环只需要注册一次。

# 四、注意事项

1、连接手环接口中，isfirst为YES时，表示新绑定手环，绑定流程为：手环显示一个6位随机码并震动，敲击手环确认，则绑定成功，回调连接成功。六位随机码需要通过 didSendValidateCode返回给sdk，类型为字符串，例如“123456”。如果isfirst为NO，表示不走绑定流程直接连接，此时不会回调didSendValidateCode接口，直接回调连接成功。

2、所有APDU相关的发送数据指令的接口都是线程阻塞并等待超时的，请不要在主线程中调用。所有运动相关接口为即时返回，异步获取并通过block回调，可以在主线程直接调用，block会在主线程回调，所有传输数据的接口，只能同时调用一个，请等待返回后再调用下一个。

具体使用流程请参考Demo