XLINK iOS SDK集成文档

本文档是面向智能硬件的APP开发者，通过Xlink iOS SDK接入云智易物联平台。

一. 开发前的准备

1. 功能概述

云智易（XLINK）iOS SDK帮助开发者在iOS系统上开发用于和XLINK模块进行通讯的APP程序。

2. 集成准备

1. 注册云智易厂商账号：http://admin.xlink.cn/;
2. 新建产品，定义设备的数据端点。

> 数据端点：用于对设备运行状态的监控和调试，除有特殊需求，一般开发者不用处理。

3. 进入管理台系统设置\帐号信息，得到企业ID

> 企业ID为用户注册、认证须使用的必要字段。

3. 配置程序

- XLINKSDK采用原生ObjectC程序编写，提供的是静态链接库方式集成；
- XLINKSDK仅支持iOS系统，不支持OSX系统；
- XLINKSDK仅支持iOS7.0以上系统；
- XLINKSDK暂只对Xcode下的开发进行支持；

二. 配置SDK

1. 配置Xcode工程

- 配置头文件搜索路径:
    - 进入Xcode工程属性页面，
    - 进入BuildSettings标签，
    - 切换All选项，
    - 找到HeaderSearchPaths选项，
    - 将libxlink/include目录加入SearchPaths；
- 配置库引用:
    - 进入Xcode工程属性页面，
    - 进入BuildSettings标签，
    - 切换All选项，
    - 找到OtherLinkerFlags选项，
    - 加入-lxlinksdklib参数；
- 配置库文件搜索路径:
    - 进入Xcode工程属性页面，
    - 进入BuildSettings标签，
    - 切换All选项，
    - 找到LibrarySearchPaths选项，
    - 将libxlink/lib下针对模拟器和实体机加载的库的路径配置到SearhPaths中。

2. 初始化SDK

- 挂接delegate： 需要一个全局的对象作为XLINKShareObject的delegate接收器，一般情况下，我们会把App的AppDelegate作为全局delegate，如图：

- 启动XLinkExportObject
- 调用函数[[XLinkExportObject shareObject] start];
- 编译程序
- 若Xcode编译时没有报错误，则表示SDK初始化成功，后续便可以进行实际的开发了。

3. 注意事项

以上配置选项可以参考搭配SDK体统的Demo工程；

libxlinksdklib. a库文件，是区分模拟器和实机的，并且debug和release也是有区分，在配置工程时一定要注意。

三. 登录/注册厂商自己的用户体系

1.概述

通过厂商自己的用户体系uid+password换取用户在云智易平台唯一的AppID和AppAuthKey。获得AppID后，才能调用XLinkExportObject的loginWithAppID登录云智易平台，使用云端通讯功能。

2.使用

- 在企业管理平台/开发者服务/Accesskey下新建Accesskey，得到AccessKeyID和AccessKeySecret。
- 查看云智易RESTfulService-用户身份集成接口文档，并参考Demo程序中的

HttpRequest类注册登录流程（替换HttpRequest.h中的CorpId宏定义为自己的企业ID）。

3.注意事项

如果使用邮箱或者手机号作为uid，其中的邮箱验证，短信验证由厂商自行实现（先判断验证，验证成功后，再调用云智易的接口进行注册）。

我们提供重置密码接口，其中的找回密码之前的验证用户也是由厂商自行实现。
该注册之后的uid、name、password不在云智易SDK内通用，云智易唯一可识别的是AppID,AppAuthKey，这里的概念要区分开。

四. SDK通用调用流程

1.初始化SDK

- 使用全局变量实例话XLinkExportObjectDelegate，将该delegate赋值给XLinkExportObject；
- 调用XLinkExportObject的start函数；

2.注册XLINK账号

- 通过XLINK提供的身份接口，注册和获取AppID以及AppAuthKey，缓存到APP中。

3.登录到云端

- 通过上面获取的AppID和AppAuthKey，调用XLinkExportObject的loginWithUID函数，登录到XLINK云端平台。
- 注意：若厂商没有云端的需求，则上述的两个注册XLINK账号和登录到云端的动作可以不去实现。

4.扫描/添加设备

- 调用XLinkExportObject的scanByDeviceProductID函数扫描局域网内的指定产品ID（ProductID）下的所有设备。
- 其扫描结果将通过delegate的onGotDeviceByScan回调返回。

5.连接设备

- 调用XLinkExportObject的connectDevice函数连接设备，连接结果将通过delegate的onConnectDevice回调返回。

6.控制设备

- 设备连接成功后，会获得设备的DeviceEntity实例，这个时候就可以通过XLinkExportObject的sendPipeData通过云端发送数据给设备，或通过sendLocalPipeData通过局域网发送数据给设备。
- 如何决定发送数据到云端还是本地，可以通过设备实体的isCloud数据决定，具体请参阅后面的说明。

7.接收设备数据

- 设备连接成功后，当设备发送数据给APP，APP可以通过delegate的onRecvLocalPipeData、onRecvPipeData和onRecvPipeSyncData三个回调，接收device发送过来的数据。
- onRecvLocalPipeData、onRecvPipeData和onRecvPipeSyncData三个函数的区别请参阅后面的说明。

五. XLinkExportObject类方法说明

1. 启动SDK

```
-(int)start
```

说明：

开始初始化操作监听的app本地UDP端口用于SDK监听WiFi设备数据回包，从休眠恢复之后，需要再次调用stop和start

参数：

- 无

返回值：

值	说明
0	成功

其他	失败

2. 初始化设备

```
-(int)initDevice:(DeviceEntity*)device;
```

说明：

- 初始化（更新）某个设备的基本信息，用在从APP缓存设置到SDK中时使用。

参数：

参数	说明
device	设备实体

返回值：

值	说明
0	成功
其它	失败

备注：

3. 连接到云端

函数：

```
-(int)loginWithAppID:(int)appId andAuthStr:(NSString*)authStr;
```

说明：

- APP登录到云端，登录到云端以后，才可以使用云端的功能。

参数：

参数	说明
appId	云端分配的唯一APPID，通过HTTP接口注册获取到。

authStr	云端分配的唯一APPID对应的AuthKey，通过HTTP接口注册获取到。

返回值：

值	说明
0	成功
其他	失败

4. 通过产品ID扫描设备

```
-(int)scanByDeviceProductID:(NSString*)productID;
```

说明：

- 通过产品ID扫描本地内网设备

参数：

参数	说明
productID	产品ID

返回值：

值	说明
0	成功
其他	失败

> 扫描结果通过onGotDeviceByScan异步返回。

5. 通过本地通讯设置设备授权码(v2版本使用)

函数：

```
-(int)setAccessKey:(NSNumber *)accessKey withDevice:(DeviceEntity
*)device;
```

说明：

- 设置内网中设备的授权码，仅能设置初始化状态（没有被设置过授权码）下的设备

参数：

参数	说明
accessKey	授权码(9位纯数字)
device	设备实体

返回值：

值	说明
0	成功
其它	失败

> 设置结果通过onSetDeviceAccessKey返回。

5. 通过本地通讯设置设备授权码(v2版本已淘汰)

函数：

```
-(int)setLocalDeviceAuthorizeCode:(DeviceEntity*)device andOldAut
hCode:(NSString*)oldAuth andNewAuthCode:(NSString*)newAuth;
```

说明：

- 设置内网中设备的授权码

参数：

参数	说明
device	设备实体

oldAuth	旧密码，如果设备本身并没有设置授权码的话，该参数置为@”“
newAuth	新密码

返回值：

值	说明
0	成功
其它	失败

> 设置结果通过onSetLocalDeviceAuthorizeCode返回。

6. 通过云端设置设备授权码(v2版本已淘汰)

函数：

```
-(int)setDeviceAuthorizeCode:(DeviceEntity*)device andOldAuthKey:
(NSString*)oldAuth andNewAuthKey:(NSString*)newAuth;
```

说明：

- 通过云端设置设备的授权码

参数：

参数	说明
device	设备实体;
oldAuth	旧密码，如果设备本身并没有设置授权码的话，该参数置为@”“;
newAuth	新密码;

返回值：

值	说明
0	成功;
其它	失败;

> 设置结果通过onSetDeviceAuthorizeCode返回

7. 连接一个设备

函数：

```
 -(int)connectDevice:(DeviceEntity *)device andAuthKey:(NSNumber
*)authKey;
```

说明：

- 控制设备之前，先要去连接设备；
- 该函数会自动识别设备是本地可用还是云端可用；

参数：

```
device设备实体;
authKey设备授权码;
```

返回值：

```
0成功;
其他失败;
```

备注：

```
连接结果通过onConnectDevice回调
```

8.发送本地透传数据

函数：

```
 -(int)sendLocalPipeData:(DeviceEntity*)device andPayload:(NSDat
a*)payload;
```

说明：

- 向内网中的设备发送透传数据

参数：

参数	说明
device | 设备实体;
payload | 数据值，二进制的。

返回值：

值	说明
0 | 成功
其它 | 失败

> 其发送结果通过onSendLocalPipeData回调返回。

9. 通过云端发送透传数据

函数：

```
-(int)sendPipeData:(DeviceEntity*)device andPayload:(NSData*)payl
oad;
```

说明：

- 通过云向设备发送透传数据

参数：

参数	说明
device | 设备实体
payload | 数据值，二进制的

返回值：

值	说明
0 | 成功
其它 | 失败

> 其发送结果通过onSendLocalPipeData回调返回。

10. 探测云端设备状态

函数：

```
-(int)probeDevice:(DeviceEntity*)device;
```

说明：

- 探测设备状态

参数：

参数	说明
device ｜ 设备实体

返回值：

值	说明
0 | 成功
其他 | 失败

> 探测结果异步通过onDeviceProbe回调

11. 获取SDK中所有设备列表

函数：

```
-(NSArray*)getAllDevice;
```

说明：

- 得到所有缓存的设备列表，返回的NSArray中包含的是DeviceEntity对象

参数：

- 无

返回值：

值	说明
NSArray | DeviceEntity * 实体的队列

12. 释放SDK

函数：

```
-(void)stop;
```

说明：

- 释放SDK，清理本地资源。在退出程序，或者从休眠恢复之后，都需要再次调用stop和start

参数：

- 无

返回值：

- 无

六. XLinkExportObjectDelegate代理回调说明

1. onStart

定义：

```
-(void)onStart;
```

说明：

- SDK的start接口结果回调

参数：

- 无

2. onLogin

函数：

```
-(void)onLogin:(int)result;
```

说明：

- 登录状态回调

参数：

- 无

result结果：

定义	值	说明
`CODE_SUCCEED`	0	登录成功;
`CODE_SERVER_KICK_DISCONNECT`	13	被踢下线;
`CODE_STATE_OFFLINE`	-101	与服务器socket连接断开;

3. onGotDeviceByScan

函数：

```
-(void)onGotDeviceByScan:(DeviceEntity*)device;
```

说明：

- SDK扫描到的设备结果回调;

参数：

参数	说明
device | 设备实体

> 如果扫描到了多个设备，该回调会多次调用;

4. onSetDeviceAccessKey(v2版本使用)

函数：

```
-(void)onSetDeviceAccessKey:(DeviceEntity *)device withResult:(unsigned char)result withMessageID:(unsigned short)messageID;
```

说明：

- 内网中设置用户访问授权码的结果回调。

参数：

参数	说明
device	设备实体;
result	设置结果;
messageID	消息ID，用于定位消息。APP可以忽略;

4. onSetLocalDeviceAuthorizeCode(v2版本已淘汰)

函数：

```
-(void)onSetLocalDeviceAuthorizeCode:(DeviceEntity*)device withResult:(int)result withMessageID:(int)messageID;
```

说明：

- 内网中设置用户访问授权码的结果回调。

参数：

参数	说明
device	设备实体;
result	设置结果;
messageID	消息ID，用于定位消息。APP可以忽略;

5. onSetDeviceAuthorizeCode(v2版本已淘汰)

函数：

```
-(void)onSetDeviceAuthorizeCode:(DeviceEntity*)device withResult:
(int)result withMessageID:(int)messageID;
```

说明：

- 云端设置授权结果回调

参数：

参数	说明
device	设备实体
result	设置结果
messageID	消息ID，用于定位消息。APP可以忽略

6. onSendLocalPipeData

函数：

```
-(void)onSendLocalPipeData:(DeviceEntity*)device withResult:(in
t)result withMessageID:(int)messageID;
```

说明：

- 发送本地透传消息结果回调

参数：

参数	说明
device	设备实体;
result	设置结果;
messageID	消息ID，用于定位消息。APP可以忽略;

7. onSendPipeData

函数：

```
-(void)onSendPipeData:(DeviceEntity*)device withResult:(int)resul
t withMessageID:(int)messageID;
```

说明：

- 发送云端透传数据结果

参数：

参数	说明
device	设备实体;
result	设置结果;
messageID	消息ID，用于定位消息。APP可以忽略;

8. onRecvLocalPipeData

函数：

```
-(void)onRecvLocalPipeData:(DeviceEntity*)device withPayload:(NSD
ata*)data;
```

说明：

- 接收到设备发送过来的透穿消息

参数：

参数	说明
device	设备实体;
data	消息内容;

9. onRecvPipeData

函数：

```
-(void)onRecvPipeData:(DeviceEntity*)device withPayload:(NSDat
a*)payload;
```

说明：

- 接收到云端设备发送回来的透传数据

参数：

参数	说明
device	设备实体;
data	消息内容;

10. onRecvPipeSyncData

函数：

```
-(void)onRecvPipeSyncData:(DeviceEntity*)device withPayload:(NSDa
ta*)payload;
```

说明：

- 接收到云端设备发送的广播透传数据

参数：

参数	说明
device	设备实体;

data	消息内容;

11. onDeviceProbe

函数：

```
-(void)onDeviceProbe:(DeviceEntity*)device withResult:(int)result
withMessageID:(int)messageID;
```

说明：

- 云端探测返回回调

参数：

参数	说明
device	设备实体;
result	设置结果;
messageID	消息ID，用于定位消息。APP可以忽略;

12. onConnectDevice

函数：

```
-(void)onConnectDevice:(DeviceEntity*)device andResult:(int)resul
t andTaskID:(int)taskID;
```

说明：

- 连接设备结果回调

参数：

参数	说明
device	设备实体;
result	设置结果;

messageID	消息ID，用于定位消息。APP可以忽略;

13. onDeviceStateChanged

函数：

```
-(void)onDeviceStateChanged:(DeviceEntity*)device andState:(int)s
tate;
```

说明：

* 设备上下线状态回调

参数：

参数	说明
device	设备实体
state	状态

14. onDataPointUpdata

函数：

-(void)onDataPointUpdata:(DeviceEntity*)device withIndex:(int)index withDataBuff:(NSData*)dataBuff
```

### 说明：

- 数据端点数据回调;

### 参数：

| 参数 | 说明 |
| --- | --- |
| device | 设备实体 |
| index | 端点索引 |
| dataBuff | 索引值 |
| channel | 通道：云端还是本地 |

> 纯透传APP，该功能用不到；

# 七. DeviceEntity属性说明代理回调说明

APP开发者只用关心几个属性即可；

## -(BOOL)getInitStatus;

> 设备是否初始化过，初始化的概念就是设备有没有被设置过授权码，如果没有就需要先设置授权码才可以使用。

## -(BOOL)isCloud

> 设备是云端设备还是本地设备，这个值的判定，由connectDevice时确定，如果是云端设备，相应的发送数据和控制指令的函数，是需要调用云端对应的函数，否则的话就需要调用对应的本地函数。
```

-(NSDictionary*)getDictionaryFormatWithProtocol:(int)protocol;

```
将设备序列化成Dictionary，并且可以用JSON表示出来，JSON格式如下：
{
    "protocol": 1,
    "device": {
        "version": 1,
        "deviceID": 1234,
        "deviceName": "设备名称",
        "deviceIP": "192.168.1.127",
        "devicePort": 5987,
        "macAddress": "8C8C8C8C8C8C",
        "init": true,
        "mcuHardVersion": 1,
        "mucSoftVersion": 1,
        "productID": "faf9a0964c3c450a9c2a6dbbe0028391"
    }
}
```

参数：

```
protocol：现在只支持1
```

-(NSString*)getLocalAddress;

```
获取设备内网的通讯地址，如果设备是公网设备，将返回空；
```

八. XLINK设备的操作

设备和用户的关系

内网模式:

- 内网通讯不经过云智易服务器，在无互联网环境下也能使用设备;
- 这种模式下，用户和设备唯一凭证就是设备密码。

公网模式:

- 设备激活后，设备有属性deviceID，该属性是每个设备在云智易后台唯一不重复的标识;
- 如果APP有公网环境，调用connectDevice函数后，SDK会先在内网尝试连接设备，如果内网的环境下，无法和设备进行通讯;就会尝试在云端通道下和设备进行通讯。云端通讯的前提，需要在云端和设备产生一个订阅关系。订阅动作是需要进行设备授权认证的，只有设备授权成功以后，才可以订阅完成。所以，订阅过程是需要保证设备在线的。
- 订阅动作是由函数内部实现，开发者不用手动进行订阅动作。
- 订阅成功以后，就可以通过云端和设备进行通讯。
- 当设备被重置后，设备会清除云端中所有的订阅关系;

connectDevice函数：

- SDK为了简化开发复杂度，APP开发者只需要统一的调用connectDevice便可以完成云端或者是内网的识别;
- 设备便会有对应的属性产生（isCloud），然后APP根据这个属性，便可做后续的处理。

设备的获取

- 当用户第一次使用SDK时，获取设备只能通过SDK的scanDeviceByProductID函数获取设备实例。同时APP可以将该设备实体通过DeviceEntity的getDictionaryFormat函数将实体序列化成Dict后存储在本地。
- 再次使用时可通过DeviceEntity的initWithDictionary函数，还有SDK的initDevice函数把本地纪录的DeviceDict反序列化成设备实例。

设备的存储:

- SDK不提供存储设备,SDK有提供了设备序列化接口:DeviceEntity的getDictionaryFormat函数，APP可将获得的Dict存储在本地，Demo程序中有对应的示例代码。
- 如果需要云端同步设备，可使用云智易数据服务集成接口进行云端同步设备；(详情请查阅文档云智易RESTfulService(数据服务集成接口))

设备的操作：

- 扫描到设备后可通过isInit()属性判断设备是否初始化过，如果未初始化设备需要设置初始授权码才能使用设备。流程如下：
    1. 扫描设备

2. 判断getInitStatus属性:
    1. 如果为true：连接设备需要提示，让用户输入之前设置过的设备密码->连接设备成功->控制设备。
    2. 如果为false：提示用户设置设备授权码 - >连接设备成功->控制设备。

3. 序列化接口不提供存储设备密码，设备密码存储由第三方自行决定;可参考Demo；如果设备安全等级不高，上述步骤可有APP自动实现（Demo云智易开发版就是这种方式内置了设备密码8888）。

数据透传：

1. 通过各种PIPE接口发送数据给设备；

九. SDK返回的Code说明

字段	值	说明
CODE_SUCCEED	0	成功
CODE_INVALID_PARAM	1	参数不正确
CODE_INVALID_KEY	2	KEY不正确
CODE_UNAVAILABLE_ID	3	ID没有找到
CODE_SERVER_ERROR	4	服务器内部错误
CODE_UNAUTHORIZED	5	消息没授权，由于订阅关系不正确引起
CODE_ILLEGAL_SENDER	6	消息发送者角色不对，比如Device到Device发送set，probe等消息
CODE_DEVICE_CLOUD_OFFLINE	10	设备在云端离线
CODE_SERVER_KICK_DISCONNECT	13	云端有相同账号登录，本地登录账号被踢下线
CODE_DEVICE_OFFLINE	110	设备不在线
CODE_DEVICE_RECONNECT	111	开始重新连接设备
CODE_DEVICE_CONNECTED	112	连接设备成功
CODE_DEVICE_DISCONNECTED	113	断开连接
CODE_DEVICE_CONNECTING	114	正在连接设备

CODE_TIMEOUT	200	CODE_TIMEOUT
CODE_DEVICE_UNINIT	201	设备还未初始化
CODE_FUNC_PARAM_ERROR	-8	函数调用参数错误
CODE_FUNC_DEVICE_ERROR	-9	Device属性错误
CODE_FUNC_DEVICE_NOT_ACTIVATION	-10	Device还未激活
CODE_FUNC_DEVICE_MAC_ERROR	-11	Device mac地址错误;
CODE_FUNC_DEVICE_IP_ERROR	-12	Device IP地址错误;
CODE_FUNC_NETWOR_ERROR	-13	网络错误，不能连接设备;
CODE_FUNC_DEVICE_SESSION_ID_ERROR	-14	Device SessionId错误;
CODE_FUNC_DEVICE_CONNECTING	-15	Device已在连接状态;
CODE_STATE_OFFLINE	-101	APP下线;
CODE_STATE_NO_WIFI	-102	没有WIFI环境;
CODE_STATE_KICK_OFFLINE	-103	APP被踢下线;