

---

# JoyLink

## 模块端 SDK 开发文档 V1.3.2

京东智能协议组

京东智能Joylink协议

本文档可能包含公司技术机密以及其他需要保密的信息，本文档所包含的所有信息均为北京京东智能集团公司版权所有。未经本公司书面许可，不得向授权许可方以外的任何第三方泄露本文档内容，不得以任何形式擅自复制或传播本文档。若使用者违反本版权保护的约定，本公司有权追究使用者由此产生的法律责任。

---

## 目 录

目 录 .....	2
1 简介 .....	3
2 主要文件说明 .....	3
2 编译 .....	4
3 设备要实现的接口 .....	5
4 调试 .....	7
5 一键配置相关接口 .....	7

京东智能Joylin协议

# 1 简介

模块 SDK 主要用来方便设备端厂家开发自己的应用程序，快速接入微联。模块端的 SDK 是根据 Joylink 协议实现设备与微联 APP 通过局域网、远程云的交互。包括设备发现、feedid、accesskey、localkey 写入、获取设备快照，控制设备等功能。

文档中以一个已经接入微联的路由器为案例，展示 Joylink 协议在设备端的数据交互过程。

在开发过程中建议，首先在 PC 端开发调试，（如在 ubuntu 下），将云端，局域网内的协议调试通过。其次将调试通过的代码移植到设备中去。此 SDK 是在 ubuntu-14-10 的环境下调试通过的。

joylink\_dev\_sdk 在 ubuntu 运行时资源占用情况：

包括子设备部分：

ram: 4-6K

rom: 184K

其中的代码可以根据设备特性裁剪，如没有子设备，则将 xxx\_sub\_dev.c 踢出编译，减小代码空间。

## 2 主要文件说明

SDK 主要包括：

名称: joylink\_dev\_sdk\_v1.0

目录结构：

./joylink\_dev\_sdk\_v1.0

├── auth	加解密头文件，实现以库的形式提供。
├── extern	设备相关的逻辑处理
├── example	设备相关的逻辑处理参考案例
├── joylink	协议相关的逻辑处理
├── json	cJSON 相关的解析打包
├── lua	lua 脚本的参考源码
└── target	编译后生成文件
├── bin	编译后生成的可执行文件
└── lib	编译后生成的静态、动态库

设备厂商和模块只关注 extern 下的文件，其余的都是协议相关的流程。

extern 路径下实现的是设备相关的逻辑处理，包括 joylinkExtern.c joylinkExtern\_sub\_dev.c joylinkExtern.h 文件。

设备端必须要提供如下信息的存储和获取的接口：

feedid -- 动态设备识别码

accesskey -- 设备远程通讯加密 key

localkey -- 局域网通讯 key

uuid -- 静态设备的唯一设备码

version -- 设备的固件版本

serverinfo -- 智能云的域名和端口如 live.smart.jd.com:2002

注意：在开发者中心我们建议采用支持 lua 的方式控制设备，sdk 中默认的 joylink\_dev\_sdk.c 代码头部 .jlp.trantype = 1 默认支持 lua，所以参找 sdk/lua/ 下的脚本编辑设备自己的脚本，并上传到开发者中心，不然会出现云端链接不了的情况。如果设备自身也支持 json 格式的控制，建议上传 sdk/lua/only\_trans.lua, 这个 lua 是个空实现，仅为匹配流程，便于以后扩展。

开发者中心配置如图：

选择支持 lua 脚本



## 2 编译

SDK 的编译采用 make，管理编译文档，为了不和厂家的自己的 Makefile 冲突，所以将 SDK 的 Makefile 指定为 Make\_jl, 每个路径下都有 Make\_jl, 并且可以在各个路径下独立编译测试，详细请参考 Make\_jl。Makefile.rule 是配置基本的编译规则。

编译步骤：

1 修改配置文件 Makefile.rule

```
PROJECT_ROOT_PATH:=/home/steven/joylink_dev_sdk_v1.0
```

指定 SDK 在 PC 端的路径。

2 编译

```
make -f Make_jl
```

3 运行

生成的可执行文件

target/bin/jt

### 3 设备要实现的接口

设备只关注 extern 下的 joylink\_extern.c joylink\_extern\_sub\_dev.c joylink\_extern.h 文件。实现标注“todo”的空接口。以下文档主要说明接口作用。

E\_JLRetCode\_t

joylink\_dev\_is\_net\_ok()

功能描述：检查设备是否能连接外网。

E\_JLRetCode\_t

joylink\_dev\_set\_connect\_st(int st);

功能描述：通知设备与云端连接的状态。

E\_JLRetCode\_t

joylink\_dev\_set\_attr\_jlp(JLPInfo\_t \*jlp);

功能描述：存储协议相关的 feedid, accesskey, localkey 等信息。

注意事项：存储的这些信息，重新上电后能够正确获得。

E\_JLRetCode\_t

joylink\_dev\_get\_jlp\_info(JLPInfo\_t \*jlp);

功能描述：获取协议相关的 feedid, accesskey, localkey 等信息。

E\_JLRetCode\_t

joylink\_dev\_set\_attr(XXX\_t \*wi);

功能描述：存储设备相关的属性等信息，例如：冰箱的温度，湿度等。

注意事项：与设备相关，数据结构需要依据设备而定。

int

joylink\_dev\_get\_snap\_shot(char \*out\_snap, int32\_t out\_max);

功能描述：获取设备快照。

注意事项：要注意判断 out\_max, 不能内存越界。

int

joylink\_dev\_get\_json\_snap\_shot(char \*out\_snap, int32\_t out\_max, int code, char \*feedid);

功能描述：获取 json 格式的设备快照

注意事项：json 格式要正确，请在 json.net 网站验证。

E\_JLRetCode\_t

joylink\_dev\_lan\_json\_ctrl(const char \*json\_cmd);

功能描述：局域网 json 格式的控制指令控制

E\_JLRetCode\_t

---

joylink\_dev\_script\_ctrl(const char \*cmd, JLContrl\_t \*ctr, int from\_server);

功能描述： 局域网脚本转换后的控制指令控制，cmd 是设备上传的 lua 转化后的二进制。

E\_JLRetCode\_t

joylink\_dev\_sub\_add(JLDevInfo\_t \*dev, int num);

功能描述： 子设备添加

注意事项： 子设备信息重新上电后能够获得

E\_JLRetCode\_t

joylink\_sub\_dev\_del(JLDevInfo\_t \*dev, int num);

功能描述： 子设备删除

注意事项：

E\_JLRetCode\_t

joylink\_dev\_sub\_get\_by\_feedid(char \*feedid, JLDevInfo\_t \*dev);

功能描述： 通过 feedid 获得子设备的信息。

E\_JLRetCode\_t

joylink\_sub\_dev\_get\_by\_uuid\_mac(char \*uuid, char \*mac, JLDevInfo\_t \*dev);

功能描述： 通过 uuid 和 mac 获得子设备信息。

注意事项：

E\_JLRetCode\_t

joylink\_dev\_sub\_update\_keys\_by\_uuid\_mac(char \*uuid, char \*mac, JLDevInfo\_t \*dev);

功能描述： 通过 uuid 和 mac 来更新 accesskey, localkey, feedid 等信息，设备绑定的时候用。

注意事项： 子设备信息重新上电后能够正确获得。

JLDevInfo\_t \*

joylink\_dev\_sub\_devs\_get(int \*count, int scan\_type);

功能描述： 通过 scan\_type 获得设备信息。

注意事项： 获取三类设备信息：所有设备，等待配置，已经配置。

E\_JLRetCode\_t

joylink\_dev\_sub\_ctrl(const char \*cmd, int cmd\_len, char\* feedid);

功能描述： 控制子设备

注意事项：

char \*

joylink\_dev\_sub\_get\_snap\_shot(char \*feedid, int \*out\_len);

功能描述： 获取子设备快照

注意事项： 返回的是 malloc 的 char\*， 要返回长度 \*out\_len = 长度。

E\_JLRetCode\_t

joylink\_dev\_sub\_unbind(const char \*feedid);

功能描述： 子设备解绑  
注意事项： 将子设备 feedid 清空

E\_JLRetCode\_t  
joylink\_dev\_ota(JLOtaOrder\_t \*otaOrder);

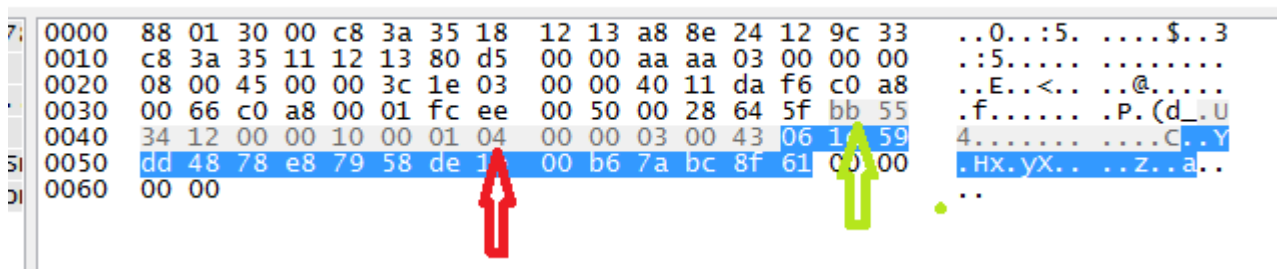
功能描述： 设备升级  
注意事项： 如果设备类型为网关，则要考虑是否本地是否由于相应固件。如果有，则不需要重新下载。产品（模块）端收到升级指令从固件资源端下载固件采用 **http** 协议。

void  
joylink\_dev\_ota\_status\_upload();  
功能描述： 设备升级状态上报  
注意事项： 设备升级状态有变化时进行升级状态主动上报。下载完固件后，须计算固件的 **crc32** 的值是否正确，如果正确说明固件下载成功，否则升级失败，可以在 **status\_desc** 字段中详细描述状态的相关描述信息。  
**厂商必须保证无论下载失败还是安装失败，都不能影响设备正常使用。**

int  
joylink\_dev\_register\_attr\_cb(  
    const char \*name,  
    E\_JL\_DEV\_ATTR\_TYPE type,  
    E\_JL\_DEV\_ATTR\_GET\_CB attr\_get\_cb,  
    E\_JL\_DEV\_ATTR\_SET\_CB attr\_set\_cb);  
功能描述： 通过注册回调的方式，管理设备属性，参考 **example** 中的案例。

## 4 调试

建议调试的时候现在 **pc** 端调试整个协议，可以造假数据，协议流程调试通过后再移植到设备。下图是调试时候抓包的图，可以分析其中 **type** 字段看到数据在手机，云端，设备端的交互。



71	0000	88 01 30 00 c8 3a 35 18 12 13 a8 8e 24 12 9c 33	..0...:5. ....\$.3
	0010	c8 3a 35 11 12 13 80 d5 00 00 aa aa 03 00 00 00	..:5.....
	0020	08 00 45 00 00 00 3c 1e 03 00 00 40 11 da f6 c0 a8	..E..<.. ..@.....
	0030	00 66 c0 a8 00 01 fc ee 00 50 00 28 64 5f bb 55	.f..... .P.(d..U
	0040	34 12 00 00 10 00 01 04 00 00 03 00 43 06 15 59	4..... ..C..Y
51	0050	dd 48 78 e8 79 58 de 1a 00 b6 7a bc 8f 61 c0 00	.Hx.yX.. ..z..a..
51	0060	00 00	..

Wireshark 抓包分析 **payload** 对应的从绿色 **bb** 开始就是 **JLPacket\_t** 对应的数据，第 10 个字节就是 **04** 对应的包体内容就是 **PT\_SCRIPTCONTROL**，说明此包是控制报文。在调试协议的时候用此方式可分析报文的交互。

## 5 一键配置相关接口

一键配置 **joylink** 这边只提供相应的库，具体使用的时候联系产品，厂商提供交叉编译环境，我们交叉编译

---

后提供库。

`int joylink_cfg_init();`

功能描述： 一键配置初始化

注意事项：

```
joylink_cfg_status_t joylink_cfg_rcv (  
    unsigned char *da,  
    unsigned char *sa,  
    int len,  
    void *user_data
```

```
);
```

功能描述： 处理空中抓取到的数据包

注意事项：

```
int joylink_cfg_result(  
    joylink_cfg_result_t *result  
);
```

功能描述： 获取一键配置结果

注意事项：

```
char* joylink_cfg_dinit (void);
```

功能描述： 获取当前库的版本信息

注意事项：