# 酷控红外解决方案

[酷控红外解决方案 1](#_Toc107391383)

[1：部分使用酷控红外码库方案的厂商 3](#_Toc107391384)

[2：红外覆盖范围 4](#_Toc107391385)

[机顶盒 4](#_Toc107391386)

[电视 4](#_Toc107391387)

[空调 4](#_Toc107391388)

[盒子 4](#_Toc107391389)

[DVD 5](#_Toc107391390)

[投影仪 5](#_Toc107391391)

[功放 5](#_Toc107391392)

[风扇 5](#_Toc107391393)

[单反相机 5](#_Toc107391394)

[灯泡开关 5](#_Toc107391395)

[空气净化器 5](#_Toc107391396)

[热水器 5](#_Toc107391397)

[电暖器 5](#_Toc107391398)

[扫地机器人 5](#_Toc107391399)

[电动窗帘 5](#_Toc107391400)

[设备类型代码定义 6](#_Toc107391401)

[3：酷控红外服务方案 6](#_Toc107391402)

[1:使用酷控SDK在线配对对码，客户离线保存使用。 7](#_Toc107391403)

[2:客户按年授权离线获取使用，可保存在客户的云端。 8](#_Toc107391404)

[4：酷控红外数据格式 10](#_Toc107391405)

[最长码 10](#_Toc107391406)

[翻转码 10](#_Toc107391407)

[连续重复码(需单独购买此项数据格式) 11](#_Toc107391408)

[空调的特殊性 11](#_Toc107391409)

[空调支持的功能 12](#_Toc107391410)

[非压缩格式 12](#_Toc107391411)

[压缩格式 13](#_Toc107391412)

[5：酷控红外码库提供方式红外数据格式 15](#_Toc107391413)

[在线红外码库接口 15](#_Toc107391414)

[离线原始红外码 15](#_Toc107391415)

[6：酷控红外数据结构 15](#_Toc107391416)

[设备类型(DeviceType): 15](#_Toc107391417)

[品牌(Brand): 15](#_Toc107391418)

[遥控器红外码(RemoteController): 15](#_Toc107391419)

[普通机顶盒红外关联: 16](#_Toc107391420)

[运营商(ServiceProvider): 16](#_Toc107391421)

[主面板设计: 16](#_Toc107391422)

[7：离线XML数据详细说明 17](#_Toc107391423)

[设备类型 17](#_Toc107391424)

[品牌 19](#_Toc107391425)

[品牌与遥控器红外码关系 20](#_Toc107391426)

[品牌与遥控器红外码关系（IPTV适用） 20](#_Toc107391427)

[地域 21](#_Toc107391428)

[运营商、地域与遥控器红外码关系 22](#_Toc107391429)

[运营商名称定义 22](#_Toc107391430)

[红外码数据文件 23](#_Toc107391431)

[空调手机端红外码库SDK 33](#_Toc107391432)

[酷控返回的空调红外数据的格式 33](#_Toc107391433)

# 1：部分使用酷控红外码库方案的厂商

小米手机

小米电视

小米音箱

米家

华为手机

荣耀手机

VIVO手机

百度音箱

海信

海尔

创维

美的

天猫精灵音箱

阿里云

乐视电视

欧瑞博

创米

绿米

涂鸦

英飞凌

海雀

猎豹

杜亚

荣事达

鸿雁

狄耐克

海康萤石

中移物联

金地智慧享联

深信服

立林

华脉

努比亚手机

金立手机

酷派手机

TCL手机

TCL电视

富可视手机

....................

以上排名不分先后，仅列出部分客户

# 2：红外覆盖范围

## 机顶盒

国内覆盖500个城市及地区，1470个运营商包括IPTV卫星电视机顶盒；

全球1733个品牌及运营商。

## 电视

国内391个品牌。

全球1900个品牌。

## 空调

国内281个品牌。

全球631个品牌 。

## 盒子

国内156个品牌。

全球242个品牌。

## DVD

国内201个品牌。

全球889个品牌。

## 投影仪

116个品牌

## 功放

国内180个品牌。

全球203个品牌。

## 风扇

170个品牌

## 单反相机

11个品牌

## 灯泡开关

36个品牌

## 空气净化器

29个品牌

## 热水器

21个品牌

## 电暖器

4个品牌

## 扫地机器人

5个品牌

## 电动窗帘

3个品牌

设备类型，品牌，红外码持续维护扩充中，每种设备保证覆盖市场设备98%以上。

酷控的红外码不在数量多， 在于高度专业的整理去除冗余红外码以及持续对码库的更新。

## 设备类型代码定义

\* public int STB = 1; //机顶盒

\* public int TV = 2; //电视

\* public int BOX = 3; //网络盒子

\* public int DVD = 4; //DVD

\* public int AC = 5; //空调

\* public int PRO = 6; //投影仪

\* public int PA = 7; //功放

\* public int FAN = 8; //风扇

\* public int SLR = 9; //单反相机

\* public int Light = 10; //开关灯泡

\* public int AirCleaner = 11;// 空气净化器

\* public int WaterHeater = 12;// 热水器

\* public int ElectricHeater = 13;// 电暖器

\* public int RobotVacuum = 14;// 扫地机器人

\* public int Blind = 15;// 电动窗帘

# 3：酷控红外服务方案

酷控红外服务支持多种红外硬件连接方案， 酷控的红外码库是纯软件码库，红外码都是时间码，单位us/微秒， 已经剥离了各种协议，可以任意传输，发送，和硬件没有关联关系，也就是说任何硬件都可以用，包括各种各种红外专用IC（集成电路，是指将多个电子器件，（如晶体管、二极管、电容等）集成在一起，形成一个可以完成特定功能的电路系统）和mcu（微控制单元，是一种集成了中央处理器、存储器、输入\输出接口和计时器等功能于一体的单片微型计算机）的IO口（输入输出口的简称。是MCU或其他数字电路经常使用的接口，用于实现数字信号的输入输出）模拟发射。以下各方案描述的是在各种场景下如何使用酷控的红外服务，实际硬件的产品方案会远超过下边各方案的种，请参考自己产品红外码库购买对接方式主要分为2种：

## 1:使用酷控SDK在线配对对码，客户离线保存使用。

压缩具体方案对接方式：一：配对，二：使用。

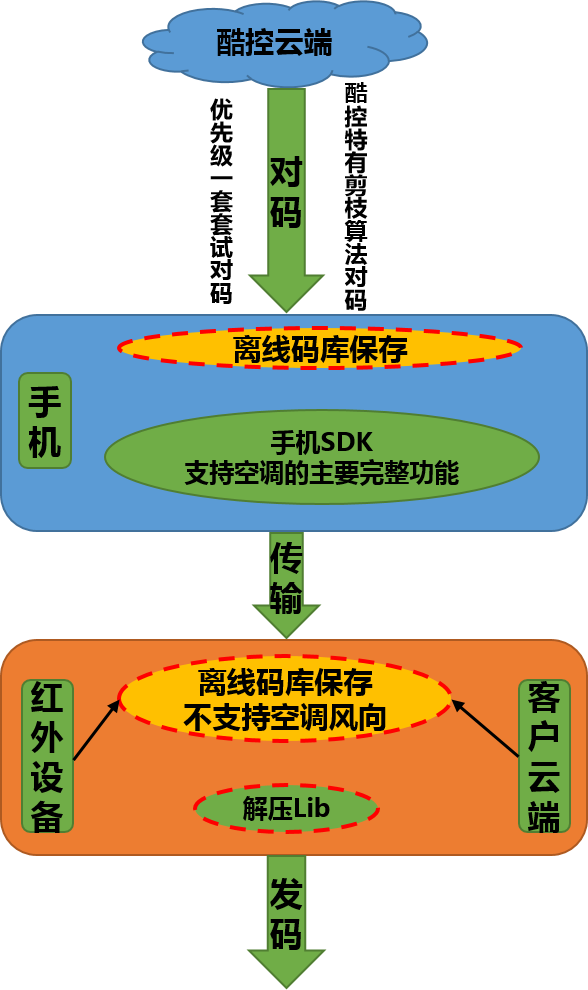
一：酷控手机SDK云端在线获取对码。

使用酷控手机SDK云端获取对码方式有也分为2种：

1是按照rank排序优先级One By One 一个接一个尝试对码，让用户主导选择确定使用哪套。

2是酷控特有的剪枝排除算法，比如创维电视有30套码，测试第一个按键X，其中有5套码的按键X的码值相同，发送按键X后如果好用那说明正确的码就在这5套之一，如果不好用那说明这5套确定是不好用只剩下25套， 这样再剪枝几次就可以快速得到正确的那套红外码了。

二：使用，通过第1步对码之后，已经确定使用哪套码，用户可以把这套码存在任意地方，比如手机端，红外转发设备端，客户自己的云端。这其中对于空调码库，如果存在手机端可以使用酷控的空调手机SDK获取空调的电源，模式，温度，风量，风向这几种完整功能。而如果存在红外设备或云端，则就没有空调SDK解码，只能是按照模式温度风量这三种状态组合解开，风向的状态则锁定为固定风向。这样一组状态对应一个按键红外时间码。

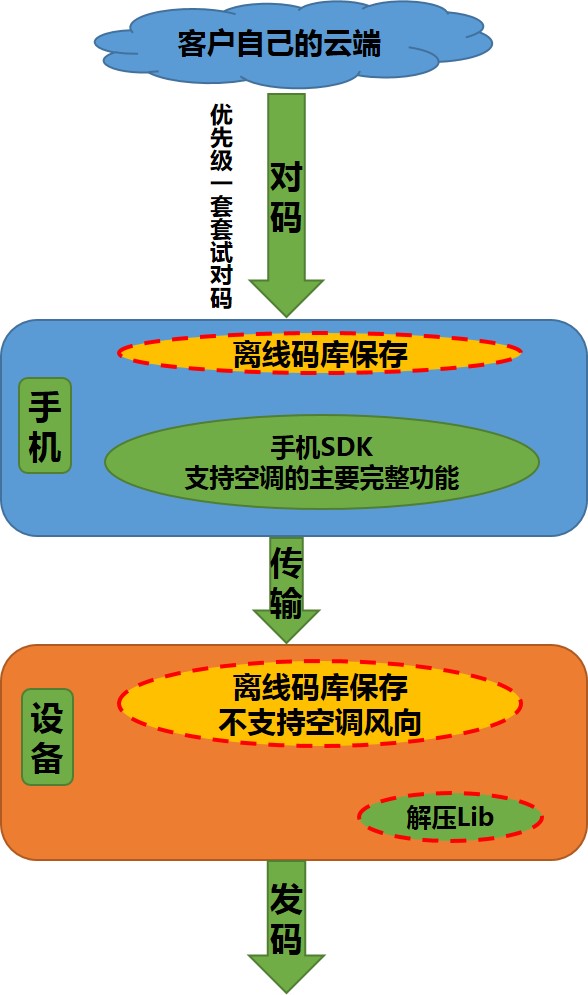


## 2:客户按年授权离线获取使用，可保存在客户的云端。

具体方案对接方式：一：配对，二：使用。

一：码库保存在客户的云端或者设备端，方式只能是按照rank排序优先级One By One一个接一个尝试对码，让用户主导选择确定使用哪套码。这种方案手机端的sdk只有空调解码的功能。

二：通过第1步对码之后，已经确定使用哪套码，用户可以把这套码存在任意地方，比如手机端，红外转发设备端，客户自己的云端。这其中对于空调码库，如果存在手机端可以使用酷控的空调手机SDK获取空调的电源，模式，温度，风量，风向这几种完整功能。而如果存在红外设备或云端，则就没有空调SDK解码，只能是按照模式温度风量这三种状态组合解开，风向的状态则锁定为固定风向。这样一组状态对应一个按键红外时间码。



注：对于不同的传输协议，又区分为压缩格式和非压缩格式，像wifi这种传输比较快的协议可以采用非压缩格式来实现，像zigbee这种传输比较慢的协议可以用压缩格式来实现。即虚线框“解压Lib”。

# 4：酷控红外数据格式

酷控采集的红外数据是一组连续的高低电平时间9000,4500,560,560,560,1680,560.....，属于时间波形码，单位us。已经剥离了各种协议可以直接使用。第1,3,5,7…奇数位个是高电平，第2,4,6,8…偶数位个是低电平。高电平的部分都是有载波调制的，低电平无载波调制。每套码都有自己的频率。范围是20K-60K左右。

C:\Users\Administrator\Desktop\QQ截图20171030144959.jpg

## 最长码



## 翻转码

有些遥控器同一按键连续按两次，第一次和第二次发送的码不相同，第三次和第一次的相同，也就是码中某些段根据按键的奇偶数来回切换，这种码定义为翻转码，酷控储存的波形码中使用&符号分隔开奇偶数按键对应的红外码。是指同一套红外码的任意一个键，比如按 "数字1" 键发1键&前面的，然后按 "数字2" 键是发2键&后面的，就是按同一套码来算，奇数次按任意键发&前面的，偶数次按任意键发&后面的。举个例子

数字 1：a&b

数字 2：c&d

发送数字11 就是发码a b

发送数字121 就是发码a d a

发送数字12 就是发码a d

发送数字212 就是发码c b c

发送数字111 就是发码a b a

注意1：是按照这一套码的按键奇偶次数来算的，&前后翻转。

注意2：由于我们添加了这种码不知道之前遥控器到底是发的&前，还是&后。所以一开始添加这套码就规定从&前开始翻转。

## 连续重复码(需单独购买此项数据格式)

如果红外发码硬件支持连续重复发码（即类似遥控器按下发码直到抬起按键停止发码），则可以使用此种格式的数据，主要有三种格式：

1：**完整重复**：9000,4500,560,560**|**

此种格式只有1个“**|**”符号，重复即是“**|**”符号之前的数据，拼接的连续重复数据：

9000,4500,560,560,9000,4500,560,560,9000,4500,560,560,9000,4500,560,560,……

2：**部分重复**：9000,4500,560,560**|**2500,560**|**

此种格式有2个“**|**”符号，重复即是2个“**|**”符号之间的数据，拼接的连续重复数据：

9000,4500,560,560,2500,560,2500,560,2500,560,2500,560,2500,560,2500,560……

3：**部分颠倒翻转重复**：9000,4500,560,560**|**2500,560**|**1500,560**|**

此种格式有3个“**|**”符号，重复即是后3个“**|**”符号之间的数据，但是颠倒翻转重复，拼接的连续重复数据：

9000,4500,560,560,2500,560,1500,560,2500,560,1500,560,2500,560,1500,560……

注意：如果此套码是翻转码格式，则首先是要取翻转的码，然后再看连续重复码：

这样三种格式的数据如下：

1：9000,4500,560,560**|&**9000,4500,1680,1680**|**

2：9000,4500,560,560**|**2500,560**|&**9000,4500,1680,1680**|**2500,1680**|**

3：9000,4500,560,560**|**2500,560**|**1500,560**|&**9000,4500,1680,1680**|**2500,1680**|**1500,1680**|**

如果数据里没有**|**符则说明不支持连续重复发码。

## 空调的特殊性

由于现代空调的遥控器需要显示空调的当前状态并与之保持同步，因此空调的遥控器红外的工作原理跟其它遥控器的工作原理不一致，其它遥控器的每个按键发送的红外码是固定的，但现代空调按键发送的红外码不是固定的，而是根据当前的状态进行组合发送一个完整的红外码（因此空调的红外一般来说比其它遥控器的红外码要长），对于空调的红外数据酷控采集的是一些空调红外的特征值，在发送的时候将空调的当前状态以及采集的特征值作为输入提交给酷控的手机SDK，SDK返回高低电平的波形码然后进行发送。

注：如设备端采用离线码库方式存储那只能是用固定的状态组合生成红外码，而非由手机sdk计算。此种格式属于全解空调红外码库，只能是简单的采用模式，温度，风量3个组合状态遍历，加上开关机的红外码，所以会有小概率的情况下控制失败，由于开关机码只有一组（开机码里包含的模式温度风量状态不确定），所以开机码发了之后可能不是上次关机时候的状态，所以开机之后可以发一个上次关机时候的状态同步一下。全解码的格式可参考xml离线说明里的全解格式。

注意：全解格式的码库只适用于非手机发码方案，以便脱离手机sdk使用。

## 空调支持的功能

空调支持全状态码既支持各种主要状态，切换随意状态后生成红外码。

（如果无手机空调SDK，离线数据目前不支持风向，风向默认固定风向）

目前空调支持的功能包括：

电源：开机，关机

模式：制冷，制热，自动，送风，除湿

温度：16-30℃

风量：自动，小，中，大

上下风向：扫风，风向（可能有多个风向状态）

注：如是采用非sdk，使用空调全解格式，则没有风向功能，只有开关机，模式，温度，风量。

## 非压缩格式

手机与红外转发中间通过wifi连接（传输快），每次均把红外时间码发给红外转发器（可发送时间序列或者脉冲个数）。此种传输的数据会很大目前有最大一套码是800多个高低电平的时间，按每个int的4个字节就是3kbytes。这种方案红外设备端没有解压lib处理数据，收到红外码直接发送。

非压缩的时间数组样例：4523,4483,576,554,576,1673,590,1677,590,1687,593,560,594,561,593,560,592,559,590,558,590,1677,590,1678,591,1680,590,557,591,558,590,557,590,557,590,557,590,557,590,1677,590,1677,590,559,591,557,590,557,590,557,590,1677,591,1680,591,559,591,557,591,1680,590,1677,590,1677,590,1677,591,45997,4525,4478,623,1673,571,95884,4525,4478,623,1673,571,95884,4525,4478,623,1673,571,95884

## 压缩格式

手机与红外转发中间通过通过zigbee传输（传输慢），受传输的速度限制，所以提供了压缩传输数据。手机端的SDK解出来的是压缩数据而不是高低电平时间数组。红外转发器上有一个小解压lib库(5-50Kbytes，根据编译软件mcu架构大小会不同)，用户在手机上按键后，（非空调设备的按键参数可以直接使用不需要解码）手机的空调SDK会解码出一个按键参数然后由网关转发给红外转发设备（如果是web端解码，那web会返回按键参数），红外转发设备里的小解压lib库会把这个按键参数并配合遥控器参数解码出高低电平的时间序列并发送。遥控器参数是每一套红外码确定唯一的，可以直接使用。按键参数则需要手机SDK解码得到，届时遥控器参数对于一套码只需传输一次，按键参数则每改变一次状态（按一次键）传输一次。

所以压缩格式在设备端可以有2种使用方式：

1）第一是确定使用哪几套红外码后，把remoteId和遥控器参数都先传给设备，设备上按照remoteId来对应存储，按键发码时候，手机只需传remoteId+按键参数就行。

2）第二是遥控器参数和按键参数都是按键发码的时候才传，这样设备端上就不需要保存任何数据，只需要收到2个参数直接解码发送。

以下是2个参数的码库数据统计：

**注意：可能会有0.1%的红外码还没做压缩处理或无法压缩处理，这些码的按键参数可能会大于100个字节。**

1：空调（有状态的组合码）：

1）遥控器参数 数据长度（一套遥控码只需要传输一次存储在转发设备上即可）：

长度（字节） 所占百分比

0-30 69.4%

30-40 26.2%

40-50 0.9%

50-60 3.2%

70-80 0.3%

2）按键参数 数据长度（每次按键都需要传输一次）：

长度（字节） 所占百分比

0-10 48.8%

10-20 42.9%

20-30 2.1%

30-40 5.0%

40-50 0.3%

50-60 0.3%

60-80 0.6%

2：固定按键遥控码（非空调设备和小部分空调）：

1）遥控器参数 数据长度（一套遥控码只需要传输一次存储在转发设备上即可）：

长度（字节） 所占百分比

0-30 88.8%

30-40 11.1%

40-50 0.1%

2）按键参数 数据长度（每次按键都需要传输一次）：

长度（字节） 所占百分比

0-5 83.1%

5-10 9.4%

10-20 1.3%

20-30 1.0%

30-40 4.6%

40-50 0.3%

50-60 0.2%

60-80 0.1%

压缩数据样例：

遥控器参数：00040EDB0100104D0015001500150040010004015A00AD000000

按键参数：000E0E0CF3

这两个参数经过设备端的解压lib库解压后得到上面非压缩格式的样例数据。

**压缩方案是为了缩短传输时间，例如zigbee这种传输慢的协议，特别适合对数据量的大小有限制传输方式。**

# 5：酷控红外码库提供方式红外数据格式

### 在线红外码库接口

酷控提供多个SDK接口，用于在线访问酷控的品牌列表， 单个遥控器的红外码。

在线模式: 不限制设备id数量，按年付费授权使用。

### 离线原始红外码

酷控通过XML提供原始的码库，厂商自己进行存贮或者解析到自己的云端。

### 

# 6：酷控红外数据结构

## 设备类型(DeviceType):

酷控目前支持设备类型有机顶盒，电视，空调，盒子，DVD，投影仪，功放，单反，灯泡，

空气净化器，热水器。

## 品牌(Brand):

一个设备类型有多个品牌，比如电视有创维，海信，三星，长虹等品牌，一个品牌也可能有多种设备类型，比如海信同时有电视，空调等设备类型。

## 遥控器红外码(RemoteController):

一套遥控器红外码与一个物理遥控器相对应，一套遥控器红外码包含有多个按键(RemoteControllerKey)，每个按键对应一段固定的红外码（空调是个例外，绝大多数空调一个按键不是对应一段规定的红外码而是根据状态组合得到的），大部分情况是：一套遥控器红外码有相同的红外格式，相同的客户码，但有些遥控器不同的键客户码不相同，设置格式都不相同，因此酷控为了通用性对于遥控器的每个按键都保存了红外格式，客户码，键码信息。

## 普通机顶盒红外关联:

由于绝大多数运营商对机顶盒的红外有要求，即使不同品牌的机顶盒设备使用的红外也必须相同，因此对于机顶盒来说，遥控器红外码不能跟非机顶盒的设备一样关联到设备的品牌上，而必须关联到运营商上面。

## 运营商(ServiceProvider):

一个运营商在一个地区可能使用多套红外码，酷控的方案是维护运营商+地区的多套红外码的优先级，在确定了运营商和地区之后根据优先级让用户测试选择红外。

**注意：IPTV运营商的红外跟非机顶盒的红外模型是一致的，即跟机顶盒的品牌有关联关系。在线方案里sdk每2个月之内只能选择7个城市，这是出于数据安全保护考虑，对普通用户不会造成影响。**

## 主面板设计:

主要按键如下：



所有按键如下 中英文：



也可以参考酷控app

酷控APP可以扫码下载，目前只支持安卓手机，可以用带有红外功能的安卓手机体验发码。



# 7：离线XML数据详细说明

整个流程就是：设备类型(非机顶盒)🡪品牌列表🡪红外码列表🡪对码🡪确定用哪套码.

其中机顶盒与其他设备类型不同，不是按照品牌列表来区分。

机顶盒也分为广电运营商和IPTV两种

1：广电机顶盒🡪地区🡪运营商列表🡪红外码列表🡪对码🡪确定用哪套码

2：IPTV🡪所有的IPTV品牌列表 🡪红外码列表🡪对码🡪确定用哪套码

所有地区的移动/电信/联通IPTV都是用IrIPTV.xml里的品牌和红外码列表对应关系。

## 设备类型

IrDeviceType.xml

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<device\_type>

<id><![CDATA[1]]></id>

<!-- 机顶盒 -->

<name><![CDATA[STB]]></name>

</device\_type>

<device\_type>

<id><![CDATA[2]]></id>

<!-- 电视 -->

<name><![CDATA[TV]]></name>

</device\_type>

<device\_type>

<id><![CDATA[3]]></id>

<!-- 盒子 -->

<name><![CDATA[BOX]]></name>

</device\_type>

<device\_type>

<id><![CDATA[4]]></id>

<!-- DVD -->

<name><![CDATA[DVD]]></name>

</device\_type>

<device\_type>

<id><![CDATA[5]]></id>

<!-- 空调 -->

<name><![CDATA[AC]]></name>

</device\_type>

<device\_type>

<id><![CDATA[6]]></id>

<!-- 投影仪-->

<name><![CDATA[Pro]]></name>

</device\_type>

<device\_type>

<id><![CDATA[7]]></id>

<!-- 功放 -->

<name><![CDATA[PA]]></name>

</device\_type>

<device\_type>

<id><![CDATA[8]]></id>

<!-- 电风扇 -->

<name><![CDATA[FAN]]></name>

</device\_type>

<device\_type>

<id><![CDATA[9]]></id>

<!-- 单反 -->

<name><![CDATA[SLR]]></name>

</device\_type>

<device\_type>

<id><![CDATA[10]]></id>

<!-- 灯泡 -->

<name><![CDATA[Light]]></name>

</device\_type>

<device\_type>

<id><![CDATA[11]]></id>

<!— 空气净化器 -->

<name><![CDATA[AirCleaner]]></name>

</device\_type>

<device\_type>

<id><![CDATA[12]]></id>

<!— 加湿器 -->

<name><![CDATA[WaterHeater]]></name>

</device\_type>

</DOCUMENT>

## 品牌

IrBrand.xml

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<brand>

<id><![CDATA[2]]></id>//品牌id

<name><![CDATA[创维]]></name>//品牌中文名称

<en\_name><![CDATA[Skyworth]]></en\_name>//品牌英文名称

</brand>

<brand>

<id><![CDATA[7]]></id>

<name><![CDATA[海信]]></name>

<en\_name><![CDATA[Hisense]]></en\_name>

</brand>

</DOCUMENT>

## 品牌与遥控器红外码关系

IrBrandRemoteRel.xml

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<item>

<device\_type\_id><![CDATA[2]]></device\_type\_id>//设备类型id

<brand\_id><![CDATA[2]]></brand\_id> //品牌id某设备下有哪些品牌也是从此xml得到

<remote\_id><![CDATA[147]]></remote\_id>//遥控器remoteId

<rank><![CDATA[1]]></rank>//排序同一设备类型和品牌下多个红外码的优先级，值越小的越优先使用

</item>

<item>

<device\_type\_id><![CDATA[2]]></device\_type\_id>

<brand\_id><![CDATA[2]]></brand\_id>

<remote\_id><![CDATA[282]]></remote\_id>

<rank><![CDATA[25]]></rank>

</item>

</DOCUMENT>

## 品牌与遥控器红外码关系（IPTV适用）

//IPTV不需要按照地区获取remoteId，所有地区的移动，联通，电信的IPTV都是从

//此xml文件里获取品牌和remoteId。

IrIPTV.xml

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<item>

<brand\_id><![CDATA[157]]></brand\_id>//品牌id

<remote\_ids><![CDATA[6547,6492,6497,6507,6352,6657,6627,6632]]></remote\_ids>

//remoteId,多个用逗号分隔

</item>

<item>

<brand\_id><![CDATA[1017]]></brand\_id>

<remote\_ids><![CDATA[5262,6027,5129,5850,6492]]></remote\_ids>

</item>

</DOCUMENT>

## 地域

IrCityArea.xml  
<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<city>

<id><![CDATA[370200]]></id>//地级市id

<name><![CDATA[青岛市]]></name>//城市名称

<province><![CDATA[山东省]]></province>//省名称

<province\_id><![CDATA[370000]]></province\_id>//省id

</city>

<city>

<id><![CDATA[370100]]></id>

<name><![CDATA[济南市]]></name>

<province><![CDATA[山东省]]></province>

<province\_id><![CDATA[370000]]></province\_id>

</city>

</DOCUMENT>

## 运营商、地域与遥控器红外码关系

IrSpRemoteRel.xml

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<item>

<sp\_id><![CDATA[17]]></sp\_id> //运营商id -->

<city\_id><![CDATA[370200]]></city\_id> //地级市id

<remote\_id><![CDATA[457]]></remote\_id> //remoteid

<rank><![CDATA[1]]></rank>//排序：同一运营商和地区下多个红外码的优先级，值越小的越优先使用

</item>

<item>

<sp\_id><![CDATA[17]]></sp\_id>

<city\_id><![CDATA[370200]]></city\_id>

<remote\_id><![CDATA[872]]></remote\_id>

<rank><![CDATA[5]]></rank>

</item></DOCUMENT>

## 运营商名称定义

IrServiceProvider.xml

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<service\_provider id="17">//运营商sp\_id

<provider\_name><![CDATA[山东广电-青岛]]></provider\_name>//运营商名称

<type>0</type>

</service\_provider>

<service\_provider id="12">

<provider\_name><![CDATA[山东广电-济南]]></provider\_name>

<type>0</type>

</service\_provider>

</DOCUMENT>

**遥控器按键定义**

IrKey.xml

<DOCUMENT>

<key>

<id><![CDATA[1]]></id>//按键id

<name><![CDATA[POWER]]></name>//按键英文名

<display\_name><![CDATA[电源]]></display\_name>//按键中文名

</key>

<key>

<id><![CDATA[3]]></id>

<name><![CDATA[TEMP+]]></name>

<display\_name><![CDATA[温度+]]></display\_name>

</key>

</DOCUMENT>

## 红外码数据文件

IrRemoteController.xml

1：无状态遥控器红外码 （非空调设备和小部分空调）

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<remote\_controller>

<id><![CDATA[347]]></id>/遥控器id

<frequency><![CDATA[38000]]></frequency>//频率，每套码都有自己的频率

<type><![CDATA[1]]></type> //1代表无状态红外码可直接发送，2代表有状态要SDK解码

<exts>//扩展信息，特征值等。

<ext>

<tag><![CDATA[99999]]></tag>//遥控器参数，压缩方案会有此项

<value><![CDATA[00040ECF010010390015001500150040010004015A00AD000000]]></value>

</ext>

</exts>

<keys>

<key>

<id><![CDATA[1]]></id>按键id，根据此id在按键定义中查询名称

//红外波形/按键参数

<pulse><![CDATA[9002,4496,576,554,576,555,576,549,576,555,576,555,549,578,576,555,576,564,575,1680,576,1681,576,1679,576,1679,576,1679,576,1680,576,1679,576,1679,576,554,576,1679,576,554,576,1679,575,555,575,555,575,555,575,555,575,1679,576,554,576,1676,576,547,576,1680,576,1679,575,1678,576,1679,576,40164,9003,2253,576,96773]]></pulse>

“如果是压缩方案传输，按键参数则是：<pulse><![CDATA[0040BF00FF]]></pulse>” 按键参数,注意有空格一定要去除。

有状态的红外码如空调的则是需要手机SDK解出这个参数来。

<exts>//扩展信息，特征值等。

</exts>

</key>

</keys>

</remote\_controller>

</DOCUMENT>

2：有状态遥控器红外码 用手机SDK解码（大部分空调）

<remote\_controller>

<id><![CDATA[10737]]></id>//遥控器id

<frequency><![CDATA[38880]]></frequency>//频率，每套码都有自己的频率

<type><![CDATA[2]]></type>//1代表无状态红外码可直接发送，2代表有状态要SDK解码

<!-- 遥控器扩展信息，特征值等 -->//以下exts各个值put手机空调SDK里配合按键状态即可解码

<exts>

<ext>

<tag><![CDATA[99999]]></tag>

<!-- 遥控器参数(压缩方案会有此项) --> //遥控器参数可以直接使用，按键参数则需要手机SDK配合状态解码

<value><![CDATA[00040ECF010010390015001500150040010004015A00AD000000]]></value>

</ext>

<ext>

<tag><![CDATA[300]]></tag>

<value><![CDATA[4342,4342]]></value>

</ext>

<ext>

<tag><![CDATA[301]]></tag>

<value><![CDATA[545,545]]></value>

</ext>

<ext>

<tag><![CDATA[302]]></tag>

<value><![CDATA[545,1610]]></value>

</ext>

<ext>

<tag><![CDATA[303]]></tag>

<value><![CDATA[-1&5165]]></value>

</ext>

<ext>

<tag><![CDATA[1508]]></tag>

<value><![CDATA[2]]></value>

</ext>

<ext>

<tag><![CDATA[305]]></tag>

<value><![CDATA[312]]></value>

</ext>

<ext>

<tag><![CDATA[306]]></tag>

<value><![CDATA[1]]></value>

</ext>

<ext>

<tag><![CDATA[1002]]></tag>

<value><![CDATA[064DB2FD0200FF]]></value>

</ext>

<ext>

<tag><![CDATA[1001]]></tag>

<value><![CDATA[000802DE0321040705F8]]></value>

</ext>

<ext>

<tag><![CDATA[1004]]></tag>

<value><![CDATA[04040205FD04043305CC0802F80307041105EE04042705D80802F80307042205DD]]></value>

</ext>

<ext>

<tag><![CDATA[1011]]></tag>

<value><![CDATA[00062428002C300F062428082C30070624280C2C3003062428042C300B062428062C30090624280E2C30010624280A2C3005062428022C300D062428032C300C0624280B2C3004062428092C3006062428012C300E062428052C300A0624280D2C3002]]></value>

</ext>

<ext>

<tag><![CDATA[1503]]></tag>

<value><![CDATA[T&16|S]]></value>

</ext>

<ext>

<tag><![CDATA[1505]]></tag>

<value><![CDATA[T&16|S]]></value>

</ext>

<ext>

<tag><![CDATA[1504]]></tag>

<value><![CDATA[T]]></value>

</ext>

<ext>

<tag><![CDATA[1010]]></tag>

<value><![CDATA[0D0700AD015202AF0350045505AA]]></value>

</ext>

<ext>

<tag><![CDATA[1005]]></tag>

<value><![CDATA[0402FD03020402F903060402FA03050402FC0303]]></value>

</ext>

<ext>

<tag><![CDATA[1501]]></tag>

<value><![CDATA[T&16]]></value>

</ext>

<ext>

<tag><![CDATA[1502]]></tag>

<value><![CDATA[T&16]]></value>

</ext>

<ext>

<tag><![CDATA[1514]]></tag>

<value><![CDATA[6&&1A00060F3001010000001700170017003F01000400B400B40000004DB2F00F07F8]]></value>

</ext>

</exts>

<keys>

</keys>

</remote\_controller>

//以下第三种格式是对应无需sdk里空调码库格式，

//开关机，模式，温度，风量，没有风向功能

3：手机SDK固定状态解开红外码 即：**全解空调红外码**

//注意由于开关机码只有一组（开机码里包含的模式温度风量状态不确定），所以开机码发了之//后可能不是上次关机时候的状态，所以开机之后可以发一个上次关机时候的状态同步一下。

<remote\_controller>

<id><![CDATA[2607]]></id>//遥控器id

<frequency><![CDATA[38000]]></frequency>//频率，每套码都有自己的频率

<type><![CDATA[2]]></type>//格式这里还是采用type=2表示是空调固定状态全解开的红外码

<exts>

<ext>

<tag><![CDATA[99999]]></tag>//如果是压缩方案则有此遥控器参数

<value><![CDATA[000E0ED801000000001500150015004000000800760076007600AE000000]]>

</value>

</ext>

</exts>

<functions>

<!—- 模式0:制冷,1:制热,2:自动,3:送风,4:除湿 -->

<modes>

<!—- 制冷 -->

<mode id="0">

<!—- 制冷模式支持的温度范围16-30度 -->

<temperature>16,17,18,19,20,21,22,23,24,25,26,27,28,29,30</temperature>

<!—- 制冷模式支持的风量范围 0:自动,1:小风量,2:中风量,3:大风量”-->

<speed>0,1,2,3</speed>

</mode>

<!—- 制热模式-->

<mode id="1">

<!—- 制热模式支持的温度范围 -->

<temperature>16,17,18,19,20,21,22,23,24,25,26,27,28,29,30</temperature>

<!—- 制热模式支持的风量范围 -->

<speed>0,1,2,3</speed>

</mode>

<!—- 自动模式-->

<mode id="2">

<!—- 自动模式不支持温度调节-->

<temperature></temperature>

<!—- 自动模式支持的风量范围-->

<speed>0,1,2,3</speed>

</mode>

<!—- 送风模式-->

<mode id="3">

<!—- 送风模式支持的温度范围-->

<temperature>16,17,18,19,20,21,22,23,24,25,26,27,28,29,30</temperature>

<!—- 送风模式支持的风量范围-->

<speed>0,1,2,3</speed>

</mode>

<!—- 除湿模式-->

<mode id="4">

<!—- 除湿模式支持的温度范围-->

<temperature>16,17,18,19,20,21,22,23,24,25,26,27,28,29,30</temperature>

<!—- 除湿模式支持的风量范围-->

<speed>0,1,2,3</speed>

</mode>

</modes>

</functions>

<keys>

<key>

<!—- 开机红外码 -->

<id><![CDATA[power\_on]]></id>

<pulse><![CDATA[9000,4500,645,1680,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,1680,645,541,645,541,645,1680,645,541,645,20700,645,541,645,1680,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1000]]></pulse>

</key>

//按键参数,如果是压缩方案则就是此种格式。注意有空格一定要去除。

<id><![CDATA[power\_on]]></id>

<pulse><![CDATA[00A6A2000040A0002000000000054D]]></pulse>

</key>

<key>

<!—- 制冷\_23度\_自动风量 -->

<id><![CDATA[M0\_T23\_S0]]></id>

<pulse><![CDATA[9000,4500,645,1680,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,1680,645,1680,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,1680,645,541,645,541,645,1680,645,541,645,20700,645,541,645,1680,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,1680,645,1680,645,1000]]></pulse>

</key>

<key>//如果是压缩方案则就是此种格式

<id><![CDATA[M0\_T23\_S0]]></id>

<pulse><![CDATA[00A672000040A00020000000000018]]></pulse>

</key>

<key>

<!—- 关机红外码 -->

<id><![CDATA[power\_off]]></id>

<pulse><![CDATA[9000,4500,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,1680,645,541,645,541,645,1680,645,541,645,20700,645,1680,645,541,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,541,645,1680,645,1680,645,1680,645,1000]]></pulse>

</key>

</keys>

<key>//如果是压缩方案则就是此种格式

<id><![CDATA[power\_off]]></id>

<pulse><![CDATA[00A50000000C000000B1]]></pulse>

</key>

</remote\_controller>

## 

//**以下2个文件是全球码库独有的**。

**国家定义（全次球码库）**

IrCountry.xml

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<item>

<lang\_code><![CDATA[ar]]></lang\_code>//语言代码，(格式: [ISO-639-1标准语言代码\_国家代码])

<country><![CDATA[AD]]></country>//国家代码(ISO-3166-3标准 2位字母代码

<name><![CDATA[أندورا]]></name>//国际名字（会用此国语言表示）

</item>

</DOCUMENT>

**设备类型下的品牌排序（全球码库）.**

**//此文件的主要作用就是每个国家都有自己的品牌排序**

IrDeviceTypeBrandRank.xml

<?xml version="1.0" encoding="utf-8"?>

<DOCUMENT>

<item>

<device\_type\_id><![CDATA[1]]></device\_type\_id>//设备类型id

<brand\_id><![CDATA[882]]></brand\_id>//品牌id

<country><![CDATA[AD]]></country>//国家代码(ISO-3166-3标准 2位字母代码

<rank><![CDATA[2]]></rank>//排序同一设备类型和品牌下多个红外码的优先级，值越小的越优先使用

</item>

</DOCUMENT>

### 空调手机端红外码库SDK

如前文所述, 由于空调的特殊性, 空调需要通过SDK来生成设备可以发送的红外波形码 为了方便第三方人员进行开发, 酷控对空调的解码进行了封装, 并提供了demo。

有Android和IOS两种手机SDK。

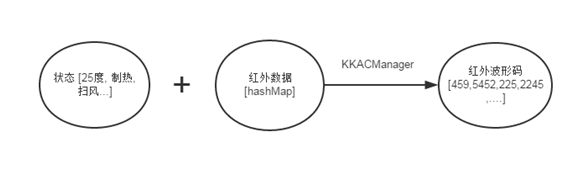
KKACManager的功能:

管理空调的状态.并提供了空调必备的切换状态的函数

根据空调的不同状态, 并生成红外波形码

返回当前空调状态的字符串信息便于App保存, 同时提供是restore前面保存的空调状态的接口

主体功能:



## 酷控返回的空调红外数据的格式

酷控返回的空调的数据格式如下,

[1, "", {//1表示返回正确数据，0表示返回错误，具体code错误如下

/\*

\* 服务器返回错误代码含义

\* 代码 意义

\* 0,"code 1" secret 错误

\* 0,"code 2" secret 不存在

\* 0,"code 3" 客户已被禁用

\* 0,"code 4" 客户已超期

\* 0,"code 6" 选择运营商超出数量限制

\* 0,"code 7" 无访问权限

\* 0,"code 8" 试用数超出限制

\* 0,"code 9" 设备数超出限制

\* 0,"code 10" 单台设备下载红外超出限制

\*/

"irDataList": [{

"rid": 2,

"fre": 38000,

"type": 2,

"exts": {

"305": "67",

"1008": "0401000D0D",

"1010": "03010C0503020C0603030C0003040C0103050C0403060C0203070C02030C0C1603130C0703250C0B030B0C1503090C10030A0C10",

"1011": "03080C01",

"1015": "030C100C030C1002030C100A030C1000",

"1001": "00020400",

"1511": "if(extKeyStatus == void) {return;} int timing\_on = extKeyStatus.get(9, 0); int timing\_off = extKeyStatus.get(10, 0); if(timing\_on>0&&timing\_on<=1440){ bytes[3]=(byte)(bytes[3]+0x40); bytes[7]=(byte)(bytes[7]+timing\_on/60); bytes[8]=(byte)(bytes[8]+timing\_on`); bytes[13]=(byte)0; for(int i=0;i<13;i++){ bytes[13]=(byte)(bytes[13]+bytes[i]); }} else if(timing\_off>0&&timing\_off<=1440){ bytes[3]=(byte)(bytes[3]+0x20); bytes[5]=(byte)(bytes[5]+timing\_off/60); bytes[6]=(byte)(timing\_off`); bytes[13]=(byte)0; for(int i=0;i<13;i++){ bytes[13]=(byte)(bytes[13]+bytes[i]); } }",

"1002": "0EA682000040A0002000000000052D",

"1005": "0205A0020560020540020520",

"1004": "04010C072004010C078004010C070004010C07C004010C0740",

"1506": "0,1,2,3",

"1518": "if(exts~=nil) then if(power==1) then timing\_on=exts[9]; if((timing\_on==0)or(timing\_on==nil))then return; end if((timing\_on>0) and (timing\_on<=1440)) then bytes[4]=bytes[4]+0x40; bytes[8]=bytes[8]+math.floor(timing\_on/60); bytes[9]=bytes[9]+timing\_on`; bytes[14]=0; for i=1,13,1 do bytes[14]=(bytes[14]+bytes[i]);end end elseif(power==0) then timing\_off=exts[10];if((timing\_off==0)or(timing\_off==nil))then return; end if((timing\_off>0) and (timing\_off<=1440)) then bytes[4]=bytes[4]+0x20; bytes[6]=bytes[6]+math.floor(timing\_off/60); bytes[7]=timing\_off`; bytes[14]=0; for i=1,13,1 do bytes[14]=(bytes[14]+bytes[i]);end end end end if(functionId==1) thenif(power==1)then bytes[5]=0;bytes[14]=0;for i=1,13,1 do bytes[14]=(bytes[14]+bytes[i]);end end end",

"300": "3078,3078,3078,4538",

"301": "548,548",

"302": "548,1686"

},

"keys": null,

"extJSON": ""

}]

}]

**离线XML数据请参考IrRemoteController.xml中的exts元素**。

第三方厂商只需要把这些数据添加到一个map中即可:

//注意 数据里如果字符串有换行可以用空格代替，不能去掉换行符之类间隔符号。

HashMap<Integer, String> ext = new HashMap<Integer, String>();

ext.put(305, "197");

ext.put(306, "1");

ext.put(1008, "0605100C01030A");

ext.put(1503, "T");

ext.put(1013, "03020400030204010302040203020403");

ext.put(1015, "060002012C3001060002002C3000");

ext.put(1509, "4&3");

ext.put(1001, "00020001");

ext.put(1003, "020101");

ext.put(1002, "090908205002002000D0");

ext.put(1505, "T|S&0,2,3");

ext.put(1004, "040009010004000C0100040008010004000B010004000A0100");

ext.put(300, "8930,4500");

ext.put(301, "645,541");

ext.put(302, "645,1680");

ext.put(303, "4&645,20700");