WIFI 音箱模块

网络控制接口

2016/6/26

修订记录Revision record

日期 Date	修订版本 Revision version	描述Description	作者 Author
2015-12-29	1.4	修改 SetAudioDSP 名称为 SetDSP 修改 SetAudioEQ 名称为 SetEQ	Jim
2016-04-04	1.5	增加 MCU 升级相关命令 GetMcuCurrentFwVersion PostMcuFwFile	Jim
2016-04-07	1.51	GetMcuUpgradeState 替换 GetMcuCurrentFwVersion 请求 增加 PlayerDoStop 停止播放命令	Jim
2016-06-26	1.6	增加定时播放协议	Jim

目录

1．协议概述

WIFI 音响模块除了支持标准的 DLNA 和 Airplay 协议外，模块还内置了一套私有的控制协议，除了支持播放器的一些基本控制外，还支持推送播放列表，模块网络配置，一键绑定，U 盘播放，U 盘智能下载，设备发现等高级控制协议.

其中设备发现接口和状态更新事情接口采用 UDP 广播外，其它接口都采样 http post 和模块后台服务器通讯。所有的通讯的报文都采用 Json 格式。

基于 http post 接口请求格式

POST /httpapi.html HTTP/1.1

CMD=HTTPAPI&JSONREQ={“Req”:”SetAVTransportURI”,”Body”:{“CurrentURI”:http://2Flive.anyradio.cn/10.163.209.145/radios/100002/index_100002.m3u8}}

POST 请求内容 CMD=HTTPAPI&JSONREQ=固定协议头，下文涉接口报文只描叙 Json。

接口请求 JSON 格式

```
{
    “Req”:”接口名称”, //字段必须存在
    “Body”:{接口参数}   // Body 字段可有可无
}
```

接口响应 JSON 格式

```
{
  "Result":0,   //整型，执行状态码，0 执行成功，否则失败.字段必须存在
  "ErrMsg":"OK!!" //  错误描叙.字段必须存在
  “Body”:{具体信息} //Body 字段可有可无,不同的接口不一样.

}
```

2. 设备发现协议

2.1 扫描 WIFI 音响设备

扫描局域网存在的设备,APP 端通过发送目标端口为 0x1F1F 的 UDP 广播,在同一个局域网内的设备接收到广播后，报告自己的 IP 地址，设备名称，和 MAC 地址等信息。

JSON 请求格式

```
{"Req":"ScanDevice"}
```

设备对 APP 的响应

```
{
 "Body":
 {
     "DeviceIpAddr": "192.168.1.101",
     "DeviceFiremwareVersion":"V3.0.3.11-1412172121-XHK",
     "DeviceName": "WIFI_AUDIO-00010D",”HttpApiPort”:80,
     "DeviceMacAddr": "F0CC025D0F0E"
 },
 "Result":0, "ErrMsg": "OK!!"
}
```

DeviceIpAddr:设备 IP 地址
DeviceFiremwareVersion:设备当前版本号
DeviceName:设备名称
HttpApiPort: httpapi 控制端口
DeviceMacAddr: 设 MAC 地址

由于广播可能丢失，为了正常的扫描，发送端每次发送可以多发送几次(2-5 次)。

2.2 设备状态更新事情接口

当设备状态发生变化的时候，设备会向发送广播(目的端口为 0x1f20)通知区域网内所有的客户端，客服端可以绑定端口为 0x1f20，监听设备事情，当客户端收到这条命令后，可以发送一系列的查询命令，进一步了解设备的当前状态,客服端不需要对这条命令进行响应.

JSON 请求格式

```
{
    "Req":"DeviceEventNotify",
    “Body”:
     {
```

```
        “SeqNo”:1,
        “DeviceName”:” WIFI_AUDIO-00010D”,
        “EventId”:”1”,
    }
}
```

字段	属性	说明
Req	字符串	接口名称固定为” DeviceEventNotify”
SeqNo	整数	序列号，从 1-n. 由于广播在 WIFI 网络中存在丢包的可能性，因此当有时间触发的时候，模块会连续发送 5 个同样的包，客服端在收到包的时候，如果发现序列号和上一次的一样可以忽略。
EventId	字符串整数	事件 ID “1”: 音量更新事情 “2”: 音源更新事情 “3”: 播放状态更新事件
DeviceName	字符串	触发事情的设备名称，也就是事件由哪个设备发送.

3.音乐控制和播放

3.1 设置播放资源地址

设置播放 URI.

JSON 请求
{"Req":"SetAVTransportURI","Body":{"AVTransportURI":"URI"}}

JSON 响应：
{"Result":0,"ErrMsg":"OK!!"}

字段	属性	说明
Req	字符串	接口名称固定为” SetAVTransportURI”
AVTransportURI	字符串	播放资源的地址,当前只支持 http-get 和本地读取（U 盘或者 SD 卡） 资源类型可以为: 1,音频文件(文件后缀可以为 mp3,flac,wav,ape，不支持不带后缀的文件)； 2,列表文件(后缀为 json)，具体格式请参考 3.2; 3,http live stream +aac 的直播，后缀为.m3u8; Example: http://www.fadfd.com/test.mp3 http://www.fadfd.com/1.m3u8 http://www.fadfd.com/playlist.json /udisk/1.mp3 /udisk/1.json

		4,定时列表文件,文件名固定为 timerlist.json

3.2 播放列表格式

播放列表 JSON 格式如下,用户 APP 可以在播放列表上面扩展一些字段.

```
{
    "FileType":"0",   //播放列表类型
    "classItems":[
     {
        "begin":"0",     //开始播放 index.
        "id":"1430106802143", //列表 ID
        "module":"cycle", //”cycle”循环播放，其它都是顺序播放
        "musics":
        [
         {
             "id":"5290808",   //歌曲 ID,在当前列表中不能重复.
             "name":"环球资讯广播", //名称

    "url":"http://live.anyradio.cn/10.163.209.145/radios/100459/index_100459.m3u8",  //对应的下载地址
             "position":1, //  在列表中的位置，从 1 开始
             "duration":"23:59:59" //播放时间,HH:MM:SS 时分秒
         },
         {
           "id":"6518556",
           "name":"英语综合广播（轻松调频）915",
           "url":"http://live.anyradio.cn/10.163.209.145/radios/100641/index_100641.m3u8",
           "position":2,
           "duration":"23:59:59"
         },
         {
           "id":"6518557",
           "name":"英语资讯广播 846",
           "url":"http://live.anyradio.cn/10.163.209.145/radios/105047/index_105047.mp3",
           "position":3,
           "duration":"23:59:59"
         },
         {
           "id":"6518560",
           "name":"环球旅游广播",
           "url":"http://live.anyradio.cn/10.163.209.145/radios/100069/index_100069.mp3",
           "position":6,
```

```
            "duration":"00:09:00"
          }
        ]
      }
    ]
}
```

字段	属性	说明
FileType	字符串整数(可选)	播放列表类型，默认为”0”
begin	字符串整数(可选)	告诉播放器从列表什么地方开始播放，从 0 到 n-1(n 为播放列表长度);
classItems	数组(强制)	数组的每个对象是一个播放列表，当前数组长度为 1
classItems[].id	字符串(强制)	播放列表 ID 具有唯一性，APP 可以通过 ID 来判断播放列表是否更新
module	字符串(可选)	播放模式,字段可选，如果不存在说明为顺序播放。 只能取下面值: 1,”**cycle”** 循环播放;
musics	数组(强制)	对象为歌曲信息
musics[].id	字符串(强制)	音乐 ID,在当前播放列表中不能重复出现
musics[].type	字符串数字（可选）	**当音响播放器是通过 musics[].url 后缀来判断音乐文件类型的，如果 url 没有后缀，一定要通过这个字段来告诉播放器音乐文件的类型.** 音乐文件类型可以为 1:**MP3** 2:**WAV** 3:**FLAC** 4:**AAC** 5:**M3U8** 6:**APE**
musics[].name	字符串(可选)	音乐名称
musics[].url	字符串(强制)	同 SetAVTransportURI 请求的 AVTransportURI
musics[].position	整数(可选)	可以为 1-n(n 为播放列表长度)指定当前音乐在播放列表中的位置，如果不存在位置信息按照出现顺序叠加。 APP 可以根据位置发送 PlayerDoSeek 切换播放曲目.
musics[].duration	字符串时间(HH:MM:SS)可选	音乐播放时间，如果不存在由播放器自己计算

3.3 播放

播放
JSON 请求
{"Req":"PlayerDoPlay"}

JSON 响应
{“Result”:0,”ErrMsg”:”OK!!”}

3.4 暂停

暂停播放

JSON 请求
{
"Req":"PlayerDoPause"
}

JSON 响应
{
“Result”:0,
”ErrMsg”:”OK!!”
}

3.5 停止

停止播放
JSON 请求
{
"Req":"PlayerDoStop"
}

JSON 响应
{
“Result”:0,
”ErrMsg”:”OK!!”
}

3.6 Seek

JSON 请求
设置播放进度条

```
{
    "Req":"PlayerDoSeek",
    "Body":
    {
        "Unit":"REL_TIME",
        "Target":"00:02:15" //到当前曲目的 00:02:15 开始播放
    }
}
```

切换播放列表歌曲

```
{
    "Req":"PlayerDoSeek",
    "Body":
    {
        "Unit":"TRACK_NR",
        "Target":"2"   //播放当前播放列表中第二首歌曲
    }
}
```

JSON 响应

```
{“Result”:0,”ErrMsg”:”OK!!”}
```

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ”PlayerDoSeek”
Unit	字符串	Seek 方式，可以为 1,” REL_TIME” 按照当前音乐的绝对时间 2,” TRACK_NR” 按照 Track 选择播放,只有列表播放的时候再有效。
Target	字符串	当 Unit 为”REL_TIME” 格式为 HH:MM:SS，告诉播放器到前曲目到这个时间点开始播放； 当 Unit 为”TRACK_NR” 可以为 1-n(n 播放列表长度) 告诉播放器切换到当前播放列表这个位置播放。

3.7 上一曲

上一曲可以根据 GetPlayerState 获取当前曲目的位置，通过发送 PlayerDoSeek 实现,假如当前位置为 2.

JSON 请求：

```
{
    "Req":"PlayerDoSeek",
    "Body":
    {
        "Unit":"TRACK_NR",
        "Target":"1"   //当前播放列表位置为 2
    }
}
```

JSON 响应

```
{
    “Result”:0,
    ”ErrMsg”:”OK!!”
}
```

3.8 下一曲

上一曲可以根据 GetPlayerState 获取当前曲目的位置，通过发送 PlayerDoSeek 实现,假如当前位置为 2.

JSON 请求：

```
{
    "Req":"PlayerDoSeek",
    "Body":
    {
        "Unit":"TRACK_NR",
        "Target":"3"   //当前播放列表位置为 2
    }
}
```

JSON 响应

```
{
    “Result”:0,
    ”ErrMsg”:”OK!!”
}
```

3.9 设置音量

设置音响音量

JSON 请求

```
{
    "Req":"PlayerSetVolume",
    "Body":
    {
        "DesiredVolume":"70"
    }
}
```

DesiredVolume:要设置音量的大小,0-100,0 为静音，100 为最大.

JSON 响应

```
{
    “Result”:0,
    ”ErrMsg”:”OK!!”
}
```

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ” PlayerSetVolume”
DesiredVolume	字符串整数(强制)	目标音量，系统音量为 100 级，取值范围为 0-100。

3.10 获取播放状态

获取播放器当前状态,在播放的时候，APP 通过发送这条请求来判断状态变化。

JSON 请求

```
{
    "Req":"GetPlayerState",
    "Body":
    {
        "Variables":"AVTransportURI TransportState CurrentTrackDuration  NumberOfTracks CurrentTrack  RelativeTimePosition CurrentVolume AudioSource AudioDSP AudioEQ "
    }
}
```

JSON 响应


```
{
      "Result": 0,
      "ErrMsg": "OK!!"
      "Body":
      {
       "AVTransportURI":"/udisk/10kHz0dB.mp3",
       "TransportState":"PLAYING",
       "CurrentTrackURI":"/udisk/10kHz0dB.mp3",
       "CurrentTrackDuration":"00:01:00",
       "NumberOfTracks":"1",
       "CurrentTrack": "1",
       "RelativeTimePosition": "00:00:41" ,
       ” CurrentVolume”:”60”,
       “AudioSource”:”WIFI”,
       “AudioDSP”:”0”,
       “AudioEQ”:”0”
      }
}
```


字段	属性	说明
Req	字符串(强制)	接口名称固定为 ” GetPlayerState”
AVTransportURI	字符串	它的值有 3 种可能 **1,音响在 WIFI 模式下列表播放** “/upnp/playlist.json;xxxxxx”，其中 xxxxxx 为播放列表 id,例如” /upnp/playlist.json;22950387” 说明音响当前播放的播放列表 ID 为”22950387” **2,音响在 WIFI 模式下单曲目播放** 对应的值和”CurrentTrackURI”一样，音乐的下载地址 **3,音响在 USB 模式下播放** 固定为“/upnp/playlist.json;local” 如果这个字段为空说明当前没有加载。 **4,音响在定时播放** “/upnp/playlist.json;xxxxxx;yyyyy” xxxxxx 为定时列表 ID yyyyy:为定时项 ID
Variables	字符串	可以为 下面一个或者多个，以空格分割 AVTransportURI

		TransportState CurrentTrackDuration NumberOfTracks CurrentTrack RelativeTimePosition CurrentVolume AudioSource AudioDSP AudioEQ
CurrentTrackURI	字符串	当前播放曲目的下载地址
TransportState	字符串	当前的播放状态 “STOPPED”停止 “PLAYING”播放中 “PAUSED_PLAYBACK”暂停 “TRANSITIONING”加载中 “NO_MEDIA_PRESENT”没有设置播放资源
CurrentTrackDuration	字符串	当前曲目的播放时间，格式为 HH:MM:SS
NumberOfTracks	字符串整数	当前播放列表有多少曲目。
CurrentTrack	字符串整数	当前曲目在播放列表中的位置,可以为 1-n,n 为播放列表长度.
RelativeTimePosition	字符串	当前曲目当前播放的确定时间,格式为 HH:MM:SS
CurrentVolume	字符串整数	当前音量大小
AudioSource	字符串	当前音源 “WIFI” “AUX” “USB” “HotKey1” “HotKey2” “HotKey3” “HotKey4” “HotKey5” “HotKey6”
AudioDSP	字符串整数	音效 DSP 0:关闭 DSP 1: Music 2: Sub Woofer 3: Radio
AudioEQ	字符串整数	EQ 可以取值 0-255 低 4 位为低音的 EQ 值 高 4 位为高音的 EQ 值

3.11 获取播放列表

获取播放列表.
JSON 请求
{“Req”:”GetPlaylist”,”Body”:{“Type”:0}}

JSON 响应

```
{
    “Result”:0,
    ”ErrMsg”:”OK!!”,
    ”Body”:
    {
        “FileType”:”0”,”id”:”local”,” classItems”:[{},{}]   //播放列表,具体参考播放列表格式
    }
}
```

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ”GetPlaylist”
Type	整数(强制)	0: 获取当前播放列表 1: 获取 U 盘扫描的播放列表 2: 获取当前的定时器播放列表 3: 获取 Hotkey1 播放列表 4: 获取 Hotkey2 播放列表 5: 获取 Hotkey3 播放列表 6: 获取 Hotkey4 播放列表 7: 获取 Hotkey5 播放列表 8: 获取 Hotkey6 播放列表

3.12 音源切换

切换音频源，设备支持音频可以来自于 AUX IN, WIFI,U 盘，Hotkeyn.

JSON 请求

```
{
    "Req":"SwitchAudioSource",
    "Body":
    {
```

"AudioSource":"USB"
}
}
JSON 响应
{ "Result": 0, "ErrMsg": "OK!!" }

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ”SwitchAudioSource”
AudioSource	字符串整数(强制)	“AUX”: 切换到 AUX 模式 “WIFI”: 切换到 WIFI 模式 “USB”: 切换到 U 盘或者 SD 卡模式，播放 U 盘，SD 卡中的音频文件 “HOTKEYn”: n 为 1-6,切换到数字键（1-6）所绑定的播放列表播放

3.13 获取当前音源

获取当前音频源。

JSON 请求
{"Req":”GetCurrentAudioSource"}

JSON 响应
{ "Result": 0, "ErrMsg": "OK!!" ,”Body”:{“AudioSource”:”USB”}}

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ”GetCurrentAudioSource”
AudioSource	字符串(强制)	“AUX”: AUX 模式 “WIFI”: WIFI 模式 “USB”: U 盘或者 SD 卡模式，播放 U 盘，SD 卡中的音频文件 “HOTKEYn”: n 为 1-6,正在播放数字键（1-6）所绑定的播放列表

3.14 数字键绑定

设置热键

JSON 请求格式

```
{
    "Req":"SetupHotKey",
    "Body":
    {
        "KeyId":1,
        "BindAudioUrl":"http://192.168.1.100/hotkey1.json",
        "Type":0
    }
}
```

JSON 响应

{“Result”:0,”ErrMsg”:”OK!!” }

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ” SetupHotKey”
KeyId	整数(强制)	1-6，对应 6 个数字键
Type	整数(强制)	BindAudioUrl 对应资源类型 0: BindAudioUrl 对应的资源为播放列表。当前只支持类型为”0”,其它值保留以后使用。
BindAudioUrl	字符串（强制）	绑定资源对应的 http 下载地址

3.15 获取 U 盘信息

获取 U 盘信息

JSON 请求：

{“Req”:”GetUdiskInfo”}

JSON 响应

{“Result”:0,”ErrMsg”:”OK!!”,”Body”:{“Size”:1000000,”Used”:1024,”State”:0,”TempSize”:0}}

字段	属性	说明
Req	字 符 串	接口名称固定为 ” GetUdiskInfo”

	（强制）	
Size	整数(强制)	U 盘或者 SD 卡 容量(KB)
Used	整数（强制）	U 盘或者 SD 卡已经使用的大小（KB）
TempSize	整数（强制）	U 盘或者 SD 卡 temp 文件夹大小(KB)
State	整数（强制）	U 盘或者 SD 卡运行状态 0：运行正常 1：U 盘或者 SD 卡没有插入 2: 不能识别

3.16 设置 DSP

设置 DSP

JSON 请求：
{“Req”:”SetDSP”,”Body”:{“AudioDSP”:”1”}}

JSON 响应
{“Result”:0,”ErrMsg”:”OK!!” }

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ” SetDSP”
AudioDSP	字符串整数(强制)	0:关闭 DSP 1: Music 2: Sub Woofer 3: Radio

3.17 设置 EQ

设置 EQ

JSON 请求：
{“Req”:”SetEQ”,”Body”:{“AudioEQ”:”0”}}

JSON 响应
{“Result”:0,”ErrMsg”:”OK!!” }

字段	属性	说明
Req	字 符 串（强制）	接口名称固定为 ” SetEQ”
AudioEQ	字 符 串 整 数 (强制)	可以取值 0-255 低 4 位为低音的 EQ 值 高 4 位为高音的 EQ 值

3.18 定时播放

推送播放列表请参考”设置播放资源地址” AVTransportURI 请求
获取当前的定时列表请参考”获取播放列表” GetPlaylist 请求
获取当前定时播放内容请参考”获取播放状态” GetPlayerState 请求

定时播放列表 JSON 格式如下，是对普通的播放列表的扩展

```
{
    "FileType":"1",   //表示为定时播放列表
    "id":"1467168329212", //定时列表 ID,APP 可以根据这个和自己缓存对比，判断是否更新
    "classItems":[
    {
    "id":"1467168329216", //定时项 ID.
    "module":"one",
    "musics":[
    {"duration":"00:03:34","name":"004","url":"http://004.mp3","position":1},
    {"duration":"00:02:50","name":"001","url":"http://v29d66fd94457.mp3","position":3},
    {"duration":"00:08:34","name":"006","url":"http:// 529d6713697b2.mp3","position":4},
    ],
    "timers":[{"bgTime":"00:00:00","endTime":"1:00:00","weekDay":"1"}]
    },
```

```
    {
    "id":"1467168329222","module":"one",
    "musics":[
    {"duration":"00:10:35","name":"tips","url":"http:// XCM916.mp3","position":1},
    {"duration":"00:06:33","name":"","url":"http:// Fo371.mp3","position":2},
    {"duration":"00:06:11","name":"008","url":"http:// bY7I131.mp3","position":3}
    ],
    "timers":[{"bgTime":"2:01:00","endTime":"3:00:00","weekDay":"1"}]
   }
]
}
```


字段	属性	说明
FileType	字符串整数(可选)	播放列表类型，”1”为定时列表
classItems	数组(强制)	数组的每个对象是一个播放列表，每个对象为一个定时项，定时项又一个播放列表组成
classItems[].id	字符串(强制)	播放列表 ID 具有唯一性，APP 可以通过 ID 来判断播放列表是否更新
module	字符串(可选)	播放模式,字段可选，如果不存在说明为顺序播放。只能取下面值: 1,”**cycle”** 循环播放;
musics	数组(强制)	对象为歌曲信息
musics[].id	字符串(强制)	音乐 ID,在当前播放列表中不能重复出现
musics[].type	字符串数字（可选）	**当音响播放器是通过 musics[].url 后缀来判断音乐文件类型的，如果 url 没有后缀，一定要通过这个字段来告诉播放器音乐文件的类型.** 音乐文件类型可以为 1:**MP3** 2:**WAV** 3:**FLAC** 4:**AAC** 5:**M3U8** 6:**APE**
musics[].name	字符串(可选)	音乐名称
musics[].url	字符串(强制)	同 SetAVTransportURI 请求的 AVTransportURI
musics[].position	整数(可选)	可以为 1-n(n 为播放列表长度)指定当前音乐在播放列表中的位置，如果不存在位置信息按照出现顺序叠加。 APP 可以根据位置发送 PlayerDoSeek 切换播放曲目.

musics[].duration	字符串时间(HH:MM:SS)可选	音乐播放时间，如果不存在由播放器自己计算
classItems[].timer	数组(强制)	数组的每个对象代表一个定时器，只要数组中任何一个定时器触发，播放器都会开始播放对应的播放列表
classItems[].timer[].bgTime	字符串(强制)	开始时间 HH:MM:SS 时分秒 24 小时格式
classItems[].timer[].endTime	字符串(强制)	结束时间 HH:MM:SS 时分秒 24 小时格式
classItems[].timer[].weekDay	字符串(强制)	星期几 1:星期一 2:星期二 ……. 7:星期天

4. 设备配置与管理

4.1 获取设备基本信息

JSON 请求

```
{"Req":"GetDeviceBasicConfig"}
```

JSON 响应

```
{
    “Result”:0,
    ”ErrMsg”:”OK!!”,
    "Body":
    {
        "DeviceName":"WIFI_AUDIO_XXX",
        ”DeviceFiremwareVersion”:” V3.0.3.11-1412172121-XHK”
    }
}
```

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ” GetDeviceBasicConfig”
DeviceName	字符串(强制)	设备名称 长度可以为 1-32 的字符串

DeviceFiremwareVersion	字 符 串 (强制)	硬件固件版本号

4.2 设置设备基本信息

JSON 请求

```
{
    "Req":"SetDeviceBasicConfig",
    "Body":
    {
        "DeviceName":"WIFI_AUDIO_XXX",
        "AccessPassword":"12345678"
    }
}
```

JSON 响应

```
{
    “Result”:0,
    ”ErrMsg”:”OK!!”
}
```

字段	属性	说明
Req	字 符 串 （强制）	接口名称固定为 ” SetDeviceBasicConfig”
DeviceName	字 符 串 (强制)	设备名称 长度可以为 1-32 的字符串
AccessPassword	字 符 串 (可选)	音响 WIFI 的密码密码长度为 8-63. 如果为空或者字段不存在，音响 WIFI 不加密.

4.3 获取设备网络配置

获取当前网络状态

JSON 请求

```
{
    “Req”:”GetNetworkState”
}
```

JSON 响应

```
{
    “Result”:0,
    ”ErrMsg”:”OK!!”,
    "Body":
    {
        “DeviceWanConnect”:”1”,
        "DeviceWanIp":"192.168.0.100",
        ”DeviceWanGw”:”192.168.0.1”,
        ”DeviceWanMask”:”255.255.255.0”,
        ”DeviceWanDNS”:”192.168.0.1”
    }
}
```

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ”GetNetworkState”
DeviceWanConnect	字符串整数(强制)	“1”: 音响成功连接到路由器 “0”: 网络断开
DeviceWanIp	字符串(强制)	路由器分配给音响的 IP 地址
DeviceWanGw	字符串(强制)	网关地址
DeviceWanMask	字符串(强制)	子网掩码
DeviceWanDNS	字符串(强制)	域名解析服务器

4.4 设置设备网络配置

```
{
    "Req":"SetNetworkConfig",
    "Body":
    {
      ”NetworkMode“:”1“,
      ”EthMode“:”1“,
      ”WlanHotspot”:”ON”,
      ”WanMode”:”DHCP”
    }
}
```

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ”SetNetworkConfig”
WlanHotspot	字符串整数(强制)	“ON”: 打开音箱设备自己的 WIFI “OFF”: **关闭音箱设备自己的 WIFI,不建议使用，如果音箱设备与连接的路由器断开，APP 讲无法再控制设备。** “AUTO”: 自动，在无线桥接模式下，如果设备连接成功关闭热点，否则打开热点，用户可以连接到热点来重新配置设备;
NetworkMode	字符串整数(强制)	网络模式 “0”:路由器模式，这个时候音箱设备就是一个普通的 WIFI 路由器，手机可以直接连接到音箱控制；在此模式，以太网一般做 WAN 口，通过网线连接到家庭路由器; “1”:无线桥接模式，音箱设备通过 WIFI 连接到家庭路由器，手机通过连接到家庭路由器控制音箱；此模式下，以太网只能做 LAN 口，或者关闭。
EthMode	字符串整数(强制)	以太网模式 “0”:禁用，如果音箱没有以太网口，关闭以太网省电； “1”:LAN 口，PC 可以通过网线连接到音箱; “2”:WAN 口，通过网线连接到路由器的 LAN 口。
WanMode	字符串(强制)	WAN 口模式，只有以太网为 WAN 口时候或者无线桥接模式下才有效 “DHCP” “STATIC”

4.5 获取设备周围 WIFI 路由器列表

扫描设备周围的热点列表

JSON 请求：

{"Req":"WiFiScan"}

JSON 响应

{“Result”:0,”ErrMsg”:”OK!!”,”Body”:{ “ApList”:[{“SSID”:”TP-LINK_xxxX”,”BSSID”:”F0010203040D”,” Rssi”:100,”Channel”:1,”Auth”:””,”Encry”:””}]}}

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ” WiFiScan” WiFiScan 大概需要 3-5 秒钟，因此设备要 3-5 秒才响应
ApList	数组(强制)	设备周围的 WIFI 路由器列表
SSID	字符串(强制)	WIFI 路由器名称
BSSID	字符串(强制)	WIFI 路由器的 MAC 地址
Rssi	字符串(强制)	信号强度 0-100 100 信号最好，0 信号最差
Auth	字符串(强制)	WIFI 路由器认证方式 “WPAPSK” “WPA2PSK” “WEP” “OPEN”
Encry	字符串(强制)	WIFI 路由器加密方式 “AES”: “TKIP” “NONE” “TKIPAES”
Channel	整数(强制)	WIFI 路由器工作的无线信道

4.6 设备连接 WIFI 路由器

设备连接到指定的 WIFI 网络.

JSON 请求

```
{
    "Req":"WiFiStaConnect",
    "Body":
    {
        "WiFiStaSSID":"TP-LINK_XXXX",
```

```
        "WiFiStaKey":"12345678"
    }
}
```

JSON 响应

```
{
    “Result”:0,
    ”ErrMsg”:”OK!!”
}
```

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ” WiFiStaConnect”
WiFiStaSSID	字符串(强制)	设备要连接 WIFI 路由器名称。 长度为（1-32）
WiFiStaKey	字符串(可选)	设备要连接 WIFI 路由器的密码密码 长度为 8-63. 如果为空或者字段不存在说明要连接路由器没有加密.

4.7 重启设备

重启设备

JSON 请求

```
{"Req":"RestartDevice"}
```

JSON 响应

```
{“Result”:0,”ErrMsg”:”OK!!”}
```

4.8 恢复出厂设置

恢复设备出厂设置

JSON 请求：

```
{
    “Req”:”RestoreDeviceFactorySettings”
}
```

JSON 响应

{
“Result”:0,
”ErrMsg”:”OK!!”
}

调用这条命令后，应该马上调用 RestartDevice 重启设备.

4.9 固件升级

升级设备固件，设备采用双备份，不需要当心升级过程中断电，在升级的时候断电，程序会自动恢复到之前版本。

JSON 请求
{
“Req”:”UpgradeFirmware”,
”Body”:
{
“UpgradeUrl”:”http://upgradeserver/firmware”
}
}
JSON 请求获取升级状态
{
“Req”:”UpgradeFirmware”
}

UpgradeUrl:新版本的下载地址,当前只是支持 http 下载，由于升级是一个异步过程，请求会马上响应，当”Body”不存在的时候，系统认为是在请求当前的升级状态。

JSON 响应
{“Result”:2003,”ErrMsg”:” Firmware Upgrading....”}

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ” UpgradeFirmware” **由于升级需要大概 1 分钟，当设备收到这个请求的时候，马上返回”Result”:2003，代表升级正在进行中，APP 端想要获取当前升级的状态只能再次发送 UpgradeFirmware(“UpgradeUrl”不能用)，如果返回”0”说明升级成功,”2004”升级失败,”2003”升级进行中.**

		当升级成功后应该马上调用 RestartDevice 命令重启设备，如果不重启设备，系统依旧以旧版本软件运行。
UpgradeUrl	字符串(可选)	固件的下载地址，当前只支持 http 下载。升级，当字段不存在的时候，设备返回当前的升级状态
Result	整型	0: 升级成功 2003：正在升级 2004: 升级失败

5. MCU 固件升级

MCU 固件版本号：2 字节整数，高字节代表设备类型，低字节代表对应设备的版本号，只有在设备类型一样，设备版本号比之前版本号大才进行升级。

固件升级分两步:
1 首先 APP 检查当前的版本号，如果要升级上传升级文件给 WIFI 模组；WIFI 模组把升级文件的信息和内容保存到自己的 flash 中;
2 通知 MCU 升级，MCU 通过比较 flash 中的固件，确定是否升级，如果要升级，下载 Flash 中的固件到自己的 Flash 中。

5.1 获取 MCU 升级状态

JSON 请求

```
{
    “Req”:” GetMcuUpgradeState”
}
```

JSON 响应：

```
{
    “Result”:0,”ErrMsg”:”OK!!”,”Body”:
    {
```

“McuCurrentFwVersion”:”1” ,
”CurrentUpgFwVersion”:”1”,
“CurrentUpgFwMD5”:”122d5d984589346739122d5d984589346739”,
“CurrentUpgradeState”:0,
}
}

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ”GetMcuUpgradeState”
McuCurrentFwVersion	字符串整数（强制)	MCU 当前运行的版本号
CurrentUpgFwVersion	字符串整型(强制)	系统保存的 MCU 升级文件的版本号
CurrentUpgFwMD5	字符串数字	系统保存 MCU 升级文件的 MD5 值
CurrentUpgradeState	整数	升级状态 0:等待 MCU 和主机连接 1:MCU 与主机已经连接 2:正在下载固件到 Flash 3:升级准备就绪等待 MCU 执行升级 4-104:MCU 升级中.升级进度为(CurrentUpgradeState-4)% 只要在升级状态为 2 的时候，APP 才能够下发 PostMcuFwFile 请求，其它状态 WIFI 模组一概不接受。

5.2 上传最新固件升级文件

JSON 请求
{
“Req”:” PostMcuFwFile”,
”Body”:
{
“FwDownloadUrl”:”http://upgradeserver/firmware”,
“FwVersion”:”2”,
“FwFileMD5”:” 122d5d984589346739122d5d984589346739”
}
}

JSON 响应

{“Result”:0,”ErrMsg”:” OK!!”}

字段	属性	说明
Req	字符串（强制）	接口名称固定为 ” PostMcuFwFile”
FwDownloadUrl	字符串 (强制)	固件的下载地址，当前只支持 http 下载。升级
FwVersion	字符串整型	要上传固件的版本号 可以为 1-65535
FwFileMD5	字符串数字(强制固定长度为 32)	要上传文件的 MD5 值

6. FTP 服务器

设备内置一个精简的 ftp 服务器，可以对 U 盘或者 SD 卡进行访问,FTP 服务器的地址为 ftp://设备的地址，端口为默认端口；FTP 服务器为主动 FTP 服务器，不支持 PASV，用户 ID 为 anonymous。

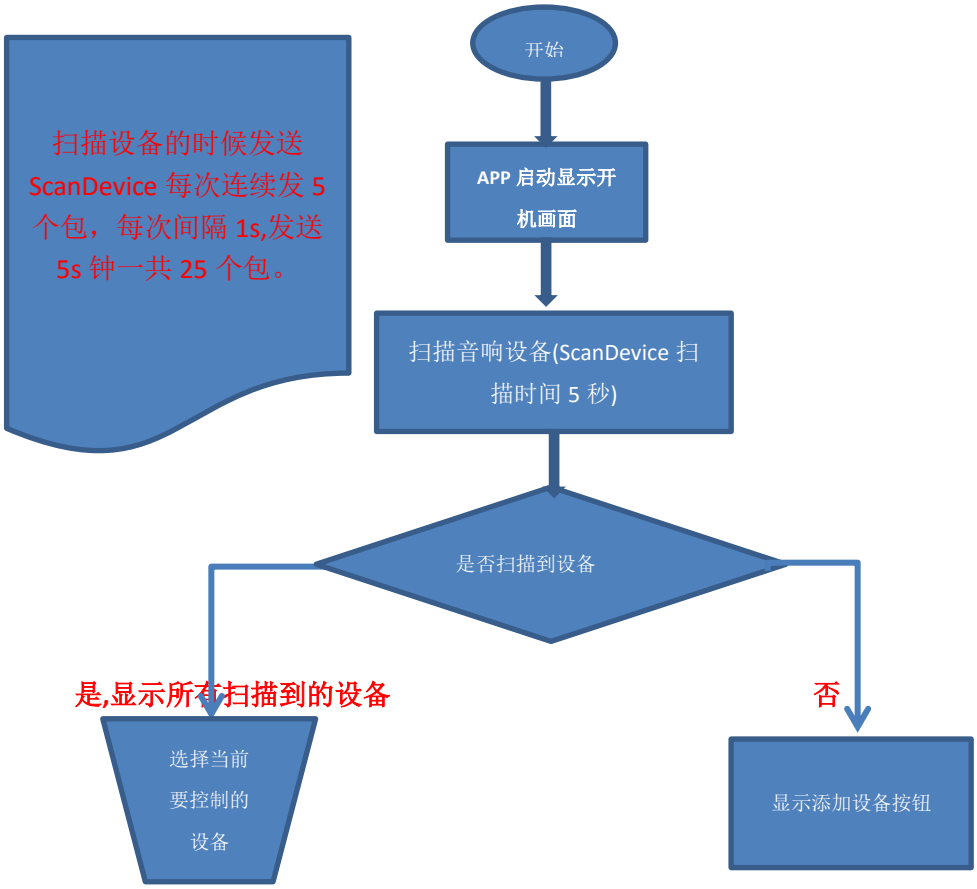
访问 sd 卡或者 U 盘 ftp://xx.xx.xx.xx/udisk/

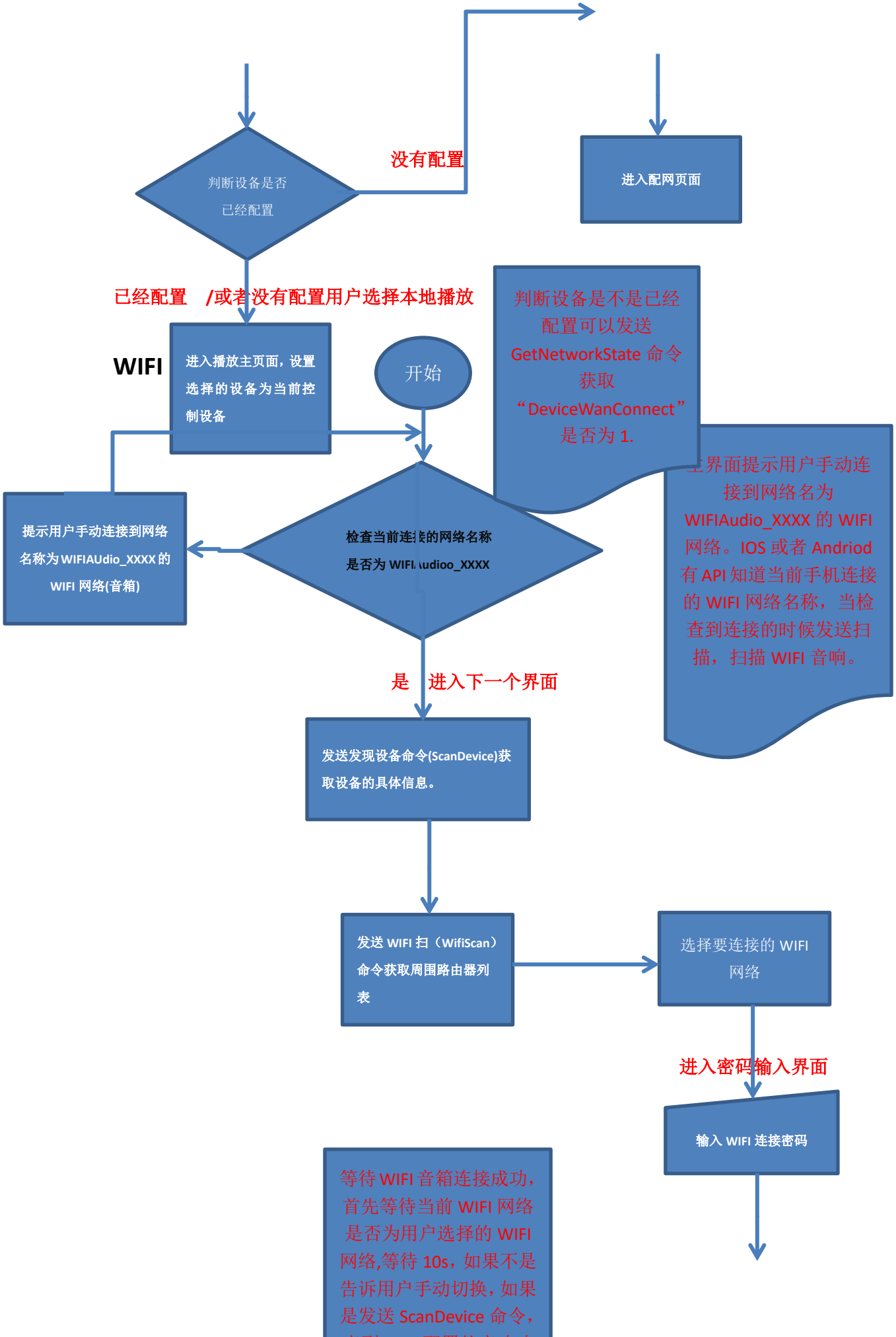
7. 附录

7.1 客户端播放器组件

1,http 服务器组件，APP 必须有一个内置 http 服务器，音箱通过 http-get 获取播放 列表和本地播放资源
2,设备扫描组件，APP 定时发送 ScanDevice 请求，扫描音箱设备;
3,命令发送组件，所有播放器控制命令和设备管理配置都是采用 http post 请求

7.2 客户端 WIFI 配网流程




判断设备是否已经配置
没有配置
进入配网页面
已经配置 /或者没有配置用户选择本地播放
WIFI
进入播放主页面，设置选择的设备为当前控制设备
开始
判断设备是不是已经配置可以发送 GetNetworkState 命令获取“DeviceWanConnect”是否为 1.
界面提示用户手动连接到网络名为 WIFIAudio_XXXX 的 WIFI 网络。IOS 或者 Andriod 有 API 知道当前手机连接的 WIFI 网络名称，当检查到连接的时候发送扫描，扫描 WIFI 音响。
提示用户手动连接到网络名称为 WIFIAUdio_XXXX 的 WIFI 网络(音箱)
检查当前连接的网络名称是否为 WIFI Audioo_XXXX
是 进入下一个界面
发送发现设备命令(ScanDevice)获取设备的具体信息。
发送 WIFI 扫（WifiScan）命令获取周围路由器列表
选择要连接的 WIFI 网络
进入密码输入界面
输入 WIFI 连接密码
等待 WIFI 音箱连接成功，首先等待当前 WIFI 网络是否为用户选择的 WIFI 网络,等待 10s，如果不是告诉用户手动切换，如果是发送 ScanDevice 命令，

7.3 固件升级流程

1，升级配置文件格式，APP 通过这个文件来判断是否要升级

```
{
    "Version":"0", //配置文件版本
    "CustItems":[
     {
        "Id":"XHK", //客户 ID
        "Version":"V3.0.3.30-1506031216-XHK-T015",//最新版本号
            "Url":"http://xxxx/xxx/V3.0.3.30-1506031216-XHK-T015", //最新固件下载地址
      },
     {
        "Id":"CLOUD",
        "Version":"V3.0.3.31-1506302126-CLOUD-004",
            "Url":"http://xxxx/xxx/V3.0.3.31-1506302126-CLOUD-004",
      }
     ]
}
```

2，固件版本号说明

固件版本号由内核版本，生成日期，客户 ID,客户版本组成，以"-"分隔.

例如 V3.0.3.30-1506031216-XHK-T015

内核版本:V3.0.3.30

发布日期:201506031216

客户名称:XHK

客户版本号 T015

3,升级流程

判断版本号是否为最新，只是需要对比版本中的日期和客户 ID，客户 ID 一定要一致。

1，先从服务器下载配置文件 http://7xjwm8.com2.z0.glb.qiniucdn.com/upgrade.json 文件;

2,获取音响当前的版本号;

3, 根据音响版本号中的客户 ID 到配置文件中查找是否有对应的升级信息，如果有,对比当前音响版本是否最新，如果不是最新进到下一步，否则退出;

4,下载最新固件到手机，提醒用户升级.

5 把从服务器下载固件推送给音箱

6, 发送 {“Req”:”UpgradeFirmware”,”Body”:{“UpgradeUrl”:http://upgradeserver/firmware}}; UpgradeUrl，因为升级固件事先已经下载到手机,这个地址一般指向手机的，APP 一般有内置的 http 下载服务器

7, 发送{“Req”:”UpgradeFirmware”},获取升级状态，或者升级状态的时候请求一定不能有 UpgradeUrl 字段，有的话音响将会重新升级。升级大概需要 1 分钟左右成功，一般 1-2 秒发这个请求来获取升级状态。

8，如果升级成功，发送”RestartDevice”重启.