ROOBO MQTT 协议文档

v1.3.0

文档状态：

[ ] 草稿 [√] 正式修改 [ ] 正式发布	文件标识：	RB/BJ-SDK-PC-2017
	文件名称：	ROOBO MQTT 协议文档
	文件版本：	v1.3.0
	作　者：	柳青
	最后修改日期：	2017/10/26

文档修订记录：

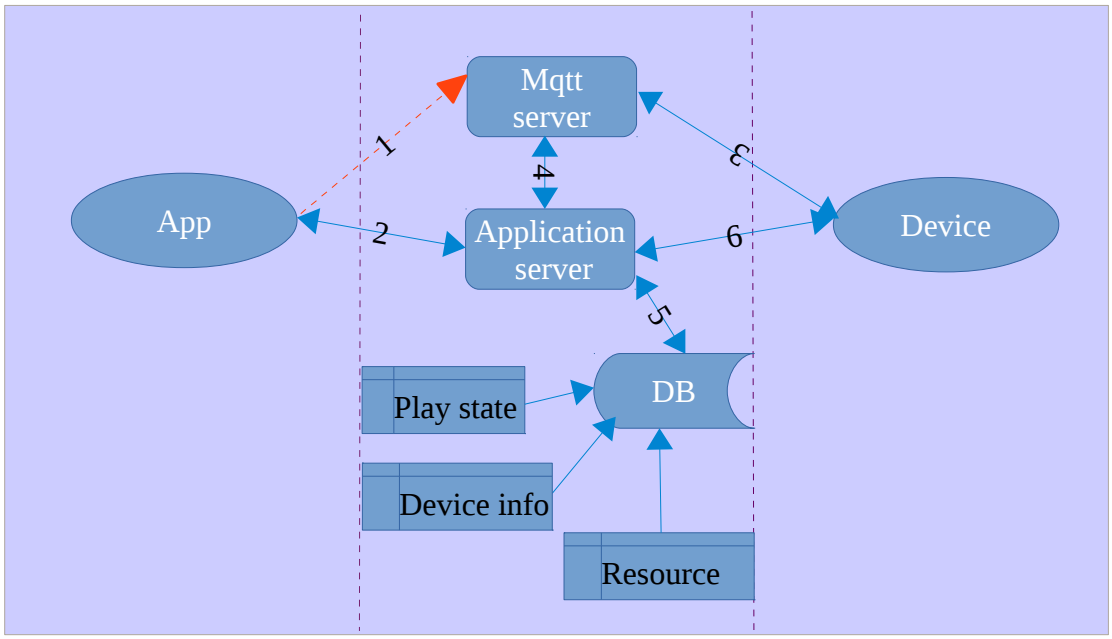
版本号	时间	编写人	审核人	备注
v1.0.0	2017/09/07	柳青	胡晓龙 邱斌	初版
v1.1.0	2017/09/15	柳青	胡晓龙 邱斌	增加资源预置
v1.2.0	2017/09/18	柳青	胡晓龙 邱斌	增加自定义消息和 OTA 升级
v1.2.1	2017/09/21	柳青	胡晓龙 邱斌	增加 MQTT 断网重连机制 增加 MQTT 错误回调机制
v1.2.2	2017/09/26	柳青	胡晓龙 邱斌	增加播放状态获取接口 修改播放状态上报接口
v1.2.3	2017/10/12	柳青	胡晓龙 邱斌	增加收藏资源接口 增加离线资源播放控制
v1.3.0	2017/10/26	柳青	胡晓龙 邱斌	增加闹钟服务 增加定时关机

目录

1. 三端通信

1.1 三端通信主流程

手机端、服务器端、设备端通信流程如下：



通信通道说明：

1. 手机端和 mqtt server 通信，目前该通道未打通，手机端通过 http 请求发送消息 mqtt server，通过 push 服务接受消息；

2. 手机端和 appliaction servrer 通过 http 和 push server 通信；

3. 设备和 mqtt server 通信，发送和订阅 mqtt 消息；

4. mqtt server 和 application server 通信；

5. application server 和数据库通信；

6. 设备端和 application server 通过 http 通信，完成设备注册、获取会话 token、设备绑定等。

DB 存储信息说明：

1. Play state 存放播放状态；

2. Device info 设备信息；

3. Resource 播放资源信息。

1.2 MQTT 断网重连机制

在网络断开和网络抖动时，mqtt client 可能会断开与服务器的连接，断网重连机制是每个周期重试 60 次第一个周期时间间隔是 1s，即 1s 重连 1 次重试 60 次，后面每个周期重连间隔时间增加 1s，第二个周期 2s 一次，重试 60 次，依次类推。每个重连周期结束时会上报一次错误。

1.3 MQTT 错误回调机制

mqtt 连接、订阅、发送消息时发生错误之后通过 message_fail_callback 回调接受错误消息。

2. 播放控制

- 播放状态维护：

  设备端将资源播放的状态同步给服务器，手机端从服务器获取播放状态，服务器根据当前的播放状态响应播放控制指令。

- 播放控制流程：

  1. 通过手机端向设备发送播放控制指令，服务端手收到指令后根据当前的播放状态向设备端发送对应播放控制指令和数据；

  2. 设备端自身控制播放，向服务器同步播放状态和发送播放控制指令。

2.1 在线资源点播、历史点播、收藏点播

在线点播：从分类列表进入专辑，从专辑列表进入资源列表进行点播。

历史点播：从播放历史进行点播。

收藏点播：从收藏列表进行点播。

cmd：demandMusicOnline

```
{
  "cmd":"demandMusicOnline",
  "trackListId":10194,
  "trackId":341280,
  "url":"http://dwn.roo.bo/voices/songs/yiqianlingyiye/huanggongdexiaoairen.mp3",
  "downloadUrl":"http://dwn.roo.bo/voices/songs/yiqianlingyiye/huanggongdexiaoairen.mp3",
```

```
  "title":"皇宫的小矮人"
}
```

参数说明：

字段名称	类型	说明
trackListId	int	专辑 id
trackId	int	资源 id
url	string	播放 url
downloadUrl	string	下载地址
title	string	资源名称、标题

以上是 **TrackInfo** 的数据结构。

2.2 播放预置资源

播放预置专辑列表中的资源。通过 trackId 在设备端预置资源列表中进行搜索，如果设备端已经存在该资源，直接播放本地资源，没有该资源时进行在线播放。

cmd：playPresetTrack

参数参照 TrackInfo。

2.3 上一首

播放当前曲目的上一首，手机端和设备都可以控制上一首。设备端控制上一首请参照**“设备端发送播放控制指令”。**

cmd：backward

返回 TrackInfo。

2.4 下一首

播放当前曲目的下一首,手机端和设备都可以控制下一首。设备端控制下一首请参照**“设备端发送播放控制指令”。**

cmd：forward

返回 TrackInfo。

2.5 暂停播放

停止当前正在播放的资源。

cmd:pause

```
{
   "cmd":"pause"
}
```

2.6 恢复播放

恢复播放被暂停下来的资源。

cmd:resume

```
{
   "cmd":"resume"
}
```

2.7 设置播放模式

手机端设置设备端的播放模式。

cmd:setPlayMode

```
{
   "cmd":"setPlayMode",
   "mode":1  // 1 ：单曲循环 2 ：顺序播放 3 ：随机播放 4 ：全部循环
}
```

播放模式:

```
typedef enum play_mode_type {
   PLAY_MODE_SINGLE=1,  //单曲循环
   PLAY_MODE_ORDER,  //  顺序播放 默认播放模式
   PLAY_MODE_RANDOM,  //  随机播放
   PLAY_MODE_LOOP_ALL, // 全部循环
} play_mode_type;
```

2.8 播放文字消息

播放手机端发送给设备端的文字消息。

播放原理：手机端将文字发送到服务器端合成 tts 播放文件，服务器将播放文件的 url 通过 mqtt 消息发送给设备，设备接受到对应消息后通过 url 进行播放。

cmd：showText

```
{
   "cmd":"showText",
   "text":"ok",
   "tts":"http://ros.roobo.net/voice/static/2017-09-
05/reply.jZmZhMzI1NTExZjc.c02a99c290a8fc303beeffea685851c49081.1504648016.3e58594f-
e466-4598-8951-b55c6ea34c47.mp3",
   "nickname":"",
   "avatar":"",
   "timestamp":1504619246
}
```

参数说明：

字段名称	类型	说明
text	string	播放的文本内容
tts	string	文本对应 wave 播放地址
nickname	string	用户昵称【可选】
avatar	string	用户头像【可选】

2.9 播放音频消息

播放 mqtt 发送过来的音频消息。来自手机端，也可能来自服务器。

cmd：playVoice

```
{
   "cmd":"playVoice",
   "url":"http://media-
test.roobo.net/voices/chat/20170905_34911d97d3f87c50bc888103f7b3115a.amr",
   "nickname":"",
   "avatar":"",
```

```
  "timestamp":1504619434
}
```

url 是音频文件对应的播放地址。其他参数可选，其他参数和播放文字消息的意义一致。

2.10 上报播放状态

在播放状态改变后上报资源播放状态给服务器。

接口文件：play_control.h

播放状态列表：

```
typedef enum play_status_type {
   PLAY_STATE_READY, //准备
   PLAY_STATE_LOADING, // 加载中
   PLAY_STATE_PLAY, // 播放
   PLAY_STATE_PAUSE, // 暂停
   PLAY_STATE_COMPLETED, // 播放结束
} play_state_type;
```

状态上报方法：

```
void send_play_state(play_track_info *track_info,  play_state_type state, int offset);
```

参数说明：

字段名称	类型	说明
track_info	play_track_info *	资源信息 TrackInfo
state	play_state_type	播放状态
offset	int	播放进度

```
PLAY_STATE_PLAY, // 播放
PLAY_STATE_PAUSE, // 暂停
```

播放和暂停这两种状态是必须上报的，当切换到下一首时，需要先暂停上一首，并上报状态。

2.11 获取播放状态

获取设备端资源播放状态和进度。设备端收到该指令后向手机端发送播放状态，具体参照**send_play_state。**

cmd:getPlayState

```
{
   "cmd":"getPlayState"
}
```

2.12 播放控制指令

设备端向服务端发送播放控制指令。

接口文件：play_controller.h

指令列表：

```
typedef enum play_command_type {
   PLAY_COMMAND_FORWARD, // 下一首
   PLAY_COMMAND_BACKWARD, // 上一首
} play_command_type;
```

指令发送方法：

```
void send_play_cmd(play_command_type cmd);
```

参数说明：

字段名称	类型	说明
cmd	play_command_type	播放指令

2.13 上报播放模式

向服务端发送当前的播放模式。在播放模式改变后向服务器发送当前的播放模式。默认播放模式是“顺序播放”。

接口文件：play_controller.h

指令列表参照枚举 play_mode。

指令发送方法：

void send_play_mode(play_mode_type mode);

参数说明：

字段名称	类型	说明
mode	play_mode	播放模式，play_mode

2.14 收藏资源

设备端收藏当前正在播放的资源。只支持在线资源的收藏。

接口文件：play_controller.h

收藏方法：

void collect_track(const int track_list_id, const int track_id);

参数说明：

字段名称	类型	说明
track_list_id	int	专辑 id
track_id	int	资源 id

2.15 离线资源播放控制

设备端播放外部资源【sdcard 中的资源】时，手机端如何进行播放控制，设备端如何进行播放控制？

设备端在播放离线资源的时候与播放在线资源一样上传播放状态，专辑 id 和资源 id 都传-1，-1 对于服务器来说代表未知资源【Unknown Resource】。服务器会将播放状态和信息同步给手机端。手机端播放控制逻辑和在线资源一样保持不变，服务器根据特殊的专辑 id 和资源 id 来处理，对于 Unknown Resource 服务器直接透传播放控制指令。上一首和下一首指令将不携带播放资源信息，只有播放控制指令。设备端的离线播放控制播由设备应用层自己处理，但是需要上传播放状态。设备端在上传播放状态信息时如果有 title 将 title 带上。

3. 设备控制

3.1 设置音量

cmd:setVolume

```
{
   "cmd":"setVolume",
   "value":88
}
```

参数说明：

字段名称	类型	说明
value	int	音量值【0~100】

3.2 重启设备

cmd: reboot

```
{
   "cmd":"reboot"
}
```

3.3 关闭设备

cmd: powerOff

```
{
   "cmd":"powerOff"
}
```

3.4 上报、获取设备信息

上报设备基本信息，设备信息上传方法请参照 ***device_info_manager.h***。该接口提供了基本信息的上传方法和部分设备特有信息的上传方法。

定义了 wifi 名称、sdcard 剩余空间、sdcard 数量、电量、音量、mac 地址、ip 地址、电源状态、固件版本的上传方法和字段【这些字段不可自定义】，同时提供特定设备信息的上传入口。

设备信息上传时机：

开机连接上 wifi 之后或者 wifi 重新连接上之后全量上报设备基本信息

设备信息改变之后同步对应的改变信息，无需全量同步，只需要同步改变项。**不建议定时同步，建议在设备状态改变后上报变化后的状态。**

设备特有信息上传方法：

void set_device_info(const char *key, const void *value, const int type/* 1 字符串 2 数字 */);

参数说明：

字段名称	类型	说明
key	string	设备信息项对应的 key
value	void *value	设备信息项对应的状态值
type	int	value 类型，1 字符串 2 数字

void send_device_info();
设置完需要上报的设备信息后调用上面的方法上报设备信息。

手机端如何获得设备信息？

服务器会保存设备信息，手机和设备绑定之后可以从服务器获得与之绑定设备的相关信息和状态。

3.5 定时关机

手机端在云端设置定时关机的闹钟提醒，到了关机时间后云端向设备端推送闹钟消息，设备端接受到 type 为**-1000** 的闹钟消息后执行关机操作。

闹钟消息请参照 AlarmInfo

4. 资源预置

在没有连接 WIFI 的情况下，通过播放预置资源，用户也能够使用设备。

4.1. 资源预置方法

资源预置方法如下：

- 服务器预置自定义专辑列表，例如：儿歌、国学、英语、故事四个专辑。每个专辑预置 2 到 5 首音频资源；

- 将预置的专辑和资源列表给设备端预置到 ROM 里面去；

  预置资源模板如下：

专辑	资源 id	资源名称
儿歌	269135811	数鸭子
	269117305	小燕子
国学	268918209	游子吟
	268918197	静夜思
英语	268923767	Buckle My Shoe
	268947040	Sing A Song Of Sixpence
故事	269599968	美猴王出世
	269588372	木偶奇遇记

- 设备端以专辑为单位保存资源列表信息，通过资源名称和资源文件关联，信息保存文件名称为对应专辑名称；
- 设备端保存资源文件。

4.2. 同步专辑列表

开机连接上 wifi 之后同步一次资源列表，使本地资源列表和服务器保持一致。

接口文件： play_controller.h

```
void sync_track_list(void);
```

4.3. 接受专辑列表

cmd：onReceiveSyncTrackList

```
{
   "cmd":"onReceiveSyncTrackList",
   "size":4,
   "listIds":[61287,61286,61285,61284]
}
```

参数说明：

字段名称	类型	说明
size	int	专辑数量
listId	int *listIds	专辑列表 id

4.4. 查询资源列表

根据专辑 id 查询对应的专辑名称、资源 id 信息。

接口文件： play_controller.h

```
void query_track_list_info(const int track_list_id);
```

参数说明：

字段名称	类型	说明
track_list_id	int	专辑 id

4.5. 接受资源列表

接受到资源列表 id 后和本地预置资源进行同步对比处理。**手机端添加新的资源或者删除资源都会向设备端发送该消息。**

cmd：onReceiveTrackListInfo

```
{
   "cmd":"onReceiveTrackListInfo",
   "size":2,
   "title":"test",
   "trackListId":61287,
   "trackIds":[ 269961160,269961161]
}
```

参数说明：

字段名称	类型	说明
size	int	资源数量
title	string	专辑名称
trackListId	int	专辑 id
trackIds	int *trackIds	资源 id

4.6. 查询资源信息

根据资源 id 查询对应的资源信息。返回 **TrackInfo**。

接口文件： play_controller.h

```
void query_track_info(const int track_list_id, const int track_id);
```

参数说明：

字段名称	类型	说明
track_list_id	int	专辑 id
track_id	int	资源 id

4.7. 接受资源信息

cmd：onReceiveTrackInfo

```
{
   "cmd":"onReceiveTrackInfo",
   "trackListId":61287,
   "trackId":269961160,
   "url":"http://dwn.roo.bo/resource/data/2017-08-
22/group15/M07/20/37/8b8fad3ac76328b22b8424c54a2ce85c.mp3",
   "downloadUrl":"http://dwn.roo.bo/resource/data/2017-08-
22/group15/M07/20/37/8b8fad3ac76328b22b8424c54a2ce85c.mp3",
   "title":"竹席为啥凉快"
}
```

参数说明： 参照 **TrackInfo**

5. 自定义消息

设备端和手机端相互发送自定义消息，消息格式是 json 对象，消息内容是设备端和手机端根据具体业务功能自定义。

5.1. 接受自定义信息

cmd：customCommand

```
{
   "cmd":"customCommand",
   "params":{
      "isChildLockOn":true,
      "isEarLightOn":1,
   }
}
```

prams 是自定义消息 json 对象，里面的内容是自定义消息。

5.2. 发送自定义信息

设备端向手机端发送自定义消息，消息内容字符串。

接口文件： custom_message.h

```
int send_custom_message(const char *msg);
```

msg 消息格式

```
{
   "isChildLockOn":true,
   "isEarLightOn":1,
}
```

参数说明：

字段名称	类型	说明
msg	string	消息 JSON 对象

6. OTA 升级

手机端或者服务端向设备端发送系统升级指令，由设备端向服务器发送升级检查请求，如果存在新版本服务器将返回新版本的下载地址，设备端下载 OTA 升级包并进行升级。

cmd：upgrade

```
{
   "cmd":"upgrade",
   "extras":{
      "from":"sr:a01581484e602a9f69e769cbaf8ea206",
      "skipFilter":true
```

```
    }
}
```

该命令将触发设备端进行升级检查进入升级检查。

参数说明：

字段名称	类型	说明
extras	json 对象	该对象可选，不一定存在，该对象用于升级检查，将该对象序列化成字符串后传给升级检查接口。

7. 闹钟服务

用户通过手机端等在云端设置一个闹钟，在闹钟提醒时间到之后，云端向设备端推送相关消息。设备端收到消息后根据提示音提醒用户。也可以根据闹钟类型执行预先定义的业务逻辑。

cmd：alarm

```
{
    "cmd":"alarm",
    "id":8,
    "name":"测试",
    "sound":"http://ros.roobo.net/voice/static/2017-10-
26/reply.MTM1NmVZamcwWlRsaU9HVTBNekZoMTczYTI=.9d2e52a3d06ba38a65afc903c67f2e
92732.1509047078.7073b290-ba55-4202-a418-29da353f5578.mp3",
    "type":1,
    "extra":"",
    "timer":1509020500,
    "timezone":"GMT+8"
}
```

参数说明：

字段名称	类型	说明
id	int	闹钟 ID
name	string	闹钟名称
sound	string	闹钟提示音 url
type	int	闹钟类型
timer	int	闹钟提醒时间

timezone	string	时区 GMT
extra	string	补充信息
repeat	string	-1：不重复，1-7 分别代表周一到周日重复，eg:1,3,4 代表周一、周三、周四重复

以上是 **AlarmInfo** 的数据结构。