1. **clientCreate**

**功能**

在页面上创建一个mqtt client

**参数**

onConnectCallback 创建成功后的回调函数名

**实例**

$(document).ready(function() {

clientCreate(onConnectCallback);

});

function onConnectCallback(){

console.log(" on call back");

}

1. **clientPublish**

**功能**

发布自定义消息，设备端也要增加解析过程

**参数**

obj 一个对象

**实例**

var obj={“field1”:”aa”,”field2”:33};

clientPublish(obj);

1. **playTrack**

**功能**

播放指定歌曲

**参数**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| trackListId | Int | 列表id |
| trackId | Int | 歌曲id |
| url | String | 播放地址 |
| downloadUrl | String | 下载地址 |

**实例**

playTrack(1,5,”xxx”,”xxx”);

1. **forwardTrack**

**功能**

下一首

**参数**

无

**实例**

forwardTrack();

1. **backwardTrack**

**功能**

上一首

**参数**

无

**实例**

backwardTrack();

1. **pauseTrack**

**功能**

暂停

**参数**

无

**实例**

pauseTrack();

1. **resumeTrack**

**功能**

继续播放

**参数**

无

**实例**

resumeTrack();

1. **queryTrack**

**功能**

询问当前播放歌曲

**参数**

无

**实例**

queryTrack();

1. **queryPlayStatus**

**功能**

询问当前播放状态，是播放还是暂停

**参数**

无

**实例**

queryPlayStatus ();

1. **queryMode**

**功能**

询问当前播放模式，是列表循环还是单曲循环

**参数**

无

**实例**

queryMode();

1. **setRepeatall**

**功能**

设置播放模式为列表循环

**参数**

无

**实例**

setRepeatall ();

1. **setRepeatone**

**功能**

设置播放模式为单曲循环

**参数**

无

**实例**

setRepeatone ();

1. **cmd\_poweroff**

**功能**

远程关机

**参数**

无

**实例**

cmd\_poweroff ();

1. **cmd\_setvolume**

**功能**

音量设置

**参数**

value 音量值

**实例**

cmd\_setvolume(20);

1. **getOnlineStatus**

**功能**

询问设备是否在线

**参数**

无

**实例**

getOnlineStatus ();

1. **getBoxInfo**

**功能**

获取设备信息

**参数**

无

**实例**

getBoxInfo ();

1. **boxUpgrade**

**功能**

升级指令

**参数**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| versionName | String | 固件版本 |
| firmwareUrl | String | 固件地址 |

**实例**

boxUpgrade (“xxx”,”xxx”);

以上是页面主动发布消息，对于无需回应类型的，比如“上一首”，设备收到消息后，完成上一首的操作就可以了；对于需要回应类型的，比如“询问设备信息”，设备收到消息后，把自己的信息作为消息体发布出去，页面收到消息后，通讯完成。

页面收到消息的处理是在mqttclient.js中的onMessageArrived函数，它会把消息体的内容转化成json对象，再根据对象中的字段来做对应处理。例如：

function onMessageArrived(message){

console.log("onMessageArrived:"+message.payloadString);

try{

var obj = JSON.parse(message.payloadString);

console.log(obj);

if(obj.hasOwnProperty("onlineStatus")){

if(onlineStatus\_change)

onlineStatus\_change(obj.onlineStatus);

}

else if(obj.hasOwnProperty("boxInfo")){

if(showBoxInfo)

showBoxInfo(obj.boxInfo);

}

.....

}

catch(e){

console.log(e);

}

}

解析obj是否含有onlineStatus字段，如果有则调用onlineStatus\_change()函数；如果没有则继续解析obj是否含有boxInfo字段，如果有则调用showBoxInfo ()函数；等等

以onlineStatus\_change为例，各页面需要定义自己的onlineStatus\_change，但接口都必须统一；页面A定义了

function onlineStatus\_change(onlineStatus){

**if**(onlineStatus == "on" || onlineStatus == "online" || onlineStatus == "on-upgrade-failed"){

console.log(“this is page a online”);

}

**else**

console.log(“this is page a offline”);

}

页面B可以这样定义

function onlineStatus\_change(onlineStatus){

**if**(onlineStatus == "on" || onlineStatus == "online" || onlineStatus == "on-upgrade-failed"){

console.log(“this is page b online”);

}

**else**

console.log(“this is page b offline”);

}

2个页面中的传入参数相同，但执行的打印可以不同。

以下是onMessageArrived内部调用函数的接口说明：

1. **onlineStatus\_change**

**功能**

当收到设备上下线通知时

**参数**

onlineStatus 字符，可以为on online on-upgrade-failed，代表上线，其他代表下线

**实例**

见上面

1. **showBoxInfo**

**功能**

当收到设备发布设备信息消息后

**参数**

boxInfo 设备信息实体

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| net | String | 连接wifi名称 |
| cardAvailable | Int | 卡的可用容量 |
| cardTotal | Int | 卡的全部容量 |
| electricity | Int | 电量 |
| firmwareVersion | String | 固件版本 |

**实例**

**function** showBoxInfo(boxInfo){

**try**{

**var** net = boxInfo.net;

**var** cardAvailable = boxInfo.cardAvailable;

**var** cardTotal = boxInfo.cardTotal;

**var** electricity = boxInfo.electricity;

**var** firmwareVersion = boxInfo.firmwareVersion;

}

**catch**(e){

console.log(e);

}

}

1. **currentVolumeDisplay**

**功能**

当收到设备发布音量消息后

**参数**

volume音量值

**实例**

**function** currentVolumeDisplay (volume){

console.log(volume);

}

1. **playTrack\_change**

**功能**

当收到设备播放歌曲变化的消息后

**参数**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| trackListId | Int | 列表id |
| trackId | Int | 歌曲id |
| type | Int | 播放类型0普通列表播放 1点播 2留言 |

**实例**

**function** playTrack\_change(trackListId,trackId,type){

console.log(trackListId);

console.log(trackId);

console.log(type);

}

1. **playStatus\_change**

**功能**

当收到设备发布播放状态消息后

**参数**

playStatus播放状态 playing/ pause

**实例**

**function** playStatus\_change (playStatus){

console.log(playStatus);

}

1. **mode\_change**

**功能**

当收到设备发布播放模式消息后

**参数**

mode播放模式

**实例**

**function** mode\_change (mode){

console.log(mode);

}

1. **after\_download**

**功能**

当收到设备发送下载成功消息后

**参数**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| downloadStatus | String | 下载状态 success |
| trackId | Int | 歌曲id |

**实例**

**function** after\_download (downloadStatus,trackId){

console.log(downloadStatus);

console.log(trackId);

}

1. **playSinVoiceUrl**

**功能**

当收到web服务器发送播放声波文件的消息后

**参数**

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 备注 |
| playSinVoiceUrl | String | 声波文件地址 |
| openId | String | 微信用户id |

**实例**

**function** playSinVoiceUrl(playSinVoiceUrl, openId){

console.log(playSinVoiceUrl);

console.log(openId);

}

1. **onConfigWifi**

**功能**

当收到设备通过声波配网后连上wifi后发送消息后

**参数**

configWifi是否连上的描述，正常为success

**实例**

**function** onConfigWifi(configWifi){

**if**(configWifi == "success"){

console.log(“true”);

}

}