IoT 平台设备定时器协议

1. 术语

标准时间格式

本文中说的`标准时间格式`是指 ISO 8601 规范中如下例这样的格式：

2016-11-29T02:49:00.000Z

2. 概述

IoT 平台支持定时器有三类：

- 单次定时：在具体的一个时间点(使用标准时间格式表示)执行一个动作，例如：在`2016-11-29T02:49:00.000Z`执行`"switch":"off"`动作；
- 重复定时：通过 CRON 格式指定一个时间点集合(只支持分，时和星期)，例如：在`0 3 * * 1,2,3,4,5`（周一至周五的 3 点 0 分）执行`"switch":"off"`动作；
- 循环定时：（旧称：每隔类定时）从标准时间点 A 开始，每隔 B 分钟执行一个动作；或者，从标准时间点 A 开始，每个 B 分钟执行一个开始动作，C 分钟(以开始动作执行时为基准)后执行一个结束动作；其中，C 必须小于 B。

3. 协议

3.1 定时器定义

在 IoT 协议中的`params`中包含定时器字段：`timers`，类型为数组，数组元素为定义：

```
{
        "enabled":<整数，表示是否启用>,
        "type":<字符串，表示定时器类型>,
        "at":<字符串，表示时间>,
```

```
        "do":<对象，包含定时器动作>,
        "startDo":<对象，包含定时器动作>,
        "endDo":<对象，包含定时器动作>
}
```

3.2 取值范围

- enabled：0 表示禁用，非 0 表示启用；
- type：可取值为 once、repeat 和 duration，分别表示单次定时、重复定时和循环定时；
- at：可取值如下：
    - 当 type 为 once 时，取值为标准时间，例如：`2016-11-29T02:49:00.000Z`；
    - 当 type 为 repeat 时，取值为 CRON 计划时间，例如：`0 3 * * 1,2,3,4,5`；
    - 当 type 为 duration 时，取值为 `标准时间 A 间隔分钟数 B` 或者 `标准时间 A 间隔分钟数 B 持续分钟数 C`，示例：`2016-11-29T03:00:00.000Z 10`，`2016-11-29T03:00:00.000Z 90 5`；
- do：根据具体设备功能，包含相应的动作。例如，对于单通道开关而言可以为{"switch":"off"};
- startDo 与 endDo：取值意义与 do 相同。

4. 定时器样例(以单通道开关为例)

单次定时：

```
    "timers":[
                {
                    "enabled":1,
                    "type":"once",
                    "at":"2016-11-29T02:49:00.000Z",
                    "do":{"switch":"off"}
                }
        ]
```

重复定时：

```
        "timers":[
```

```
                {
                    "enabled":1,
                    "type":"repeat",
                    "at":"0 3 * * 1,2,3,4,5",
                    "do":{"switch":"off"}
                }
        ]
```

循环定时（此类型定时器目前只能设置一个）：

```
        "timers":[
                {
                    "enabled":1,
                    "type":"duration",
                    "at":"2016-11-29T03:00:00.000Z 90 5",
                    "startDo":{"switch":"on"},
                    "endDo":{"switch":"off"}
                }
          ]
```

多个定时器：

```
        "timers":[
                {
                    "enabled":1,
                    "type":"once",
                    "at":"2016-11-29T02:49:00.000Z",
                    "do":{"switch":"off"}
                },
                {
                    "enabled":1,
                    "type":"repeat",
                    "at":"0 3 * * 1,2,3,4,5",
                    "do":{"switch":"off"}
                },
                {
                    "enabled":1,
                    "type":"duration",
                    "at":"2016-11-29T03:00:00.000Z 90 5",
                    "startDo":{"switch":"on"},
                    "endDo":{"switch":"off"}
                }
```

```
]
```

5. 使用 CoLink 接口解析

在初始化 CoLink 时，可以注册一个回调函数 colinkRecvUpdateCb，通过此函数来接收“timer”字段的内容，以下是对”timer”字段解析的简单代码

```
static void colinkRecvUpdateCb(char* data)
{
    cJSON *json_root = NULL;
    cJSON *timers_p = NULL;
    cJSON *one_timer_p = NULL;
    cJSON *timer_do_p = NULL;
    cJSON *timer_startDo_p = NULL;
    cJSON *timer_endDo_p = NULL;
    cJSON *json_temp_p = NULL;
    uint8_t timer_num = 0;
    uint8_t i = 0;

    colinkPrintf("colinkRecvUpdate [%s]\r\n", data);

    json_root = cJSON_Parse(data);

    if (!json_root)
    {
        colinkPrintf("parse json failed\r\n");
        return;
    }

    timers_p = cJSON_GetObjectItem(json_root, "timers");

    if (timers_p)
    {
        timer_num = cJSON_GetArraySize(timers_p);
        for (i = 0; i < timer_num; i++)
        {
            one_timer_p = cJSON_GetArrayItem(timers_p, i);
            json_temp_p = cJSON_GetObjectItem(one_timer_p, "type");
            if (!colinkStrcmp(json_temp_p->valuestring, "once"))
            {
                colinkPrintf("timer type: once\r\n");
                json_temp_p = cJSON_GetObjectItem(one_timer_p, "enabled");
                colinkPrintf("timer enable: %d\r\n", json_temp_p->valueint);
```

```
                    json_temp_p = cJSON_GetObjectItem(one_timer_p, "at");
                    colinkPrintf("timer at: %s\r\n", json_temp_p->valuestring);
                    timer_do_p = cJSON_GetObjectItem(one_timer_p, "do");
                    json_temp_p = cJSON_GetObjectItem(timer_do_p, "switch");
                    colinkPrintf("timer do: switch %s\r\n", json_temp_p->valuestring);
                }
                else if (!colinkStrcmp(json_temp_p->valuestring, "repeat"))
                {
                    colinkPrintf("timer type: repeat\r\n");
                    json_temp_p = cJSON_GetObjectItem(one_timer_p, "enabled");
                    colinkPrintf("timer enable: %d\r\n", json_temp_p->valueint);
                    json_temp_p = cJSON_GetObjectItem(one_timer_p, "at");
                    colinkPrintf("timer at: %s\r\n", json_temp_p->valuestring);
                    timer_do_p = cJSON_GetObjectItem(one_timer_p, "do");
                    json_temp_p = cJSON_GetObjectItem(timer_do_p, "switch");
                    colinkPrintf("timer do: switch %s\r\n", json_temp_p->valuestring);
                }
                else if (!colinkStrcmp(json_temp_p->valuestring, "duration"))
                {
                    colinkPrintf("timer type: duration\r\n");
                    json_temp_p = cJSON_GetObjectItem(one_timer_p, "enabled");
                    colinkPrintf("timer enable: %d\r\n", json_temp_p->valueint);
                    json_temp_p = cJSON_GetObjectItem(one_timer_p, "at");
                    colinkPrintf("timer at: %s\r\n", json_temp_p->valuestring);
                    timer_startDo_p = cJSON_GetObjectItem(one_timer_p, "startDo");
                    json_temp_p = cJSON_GetObjectItem(timer_startDo_p, "switch");
                    colinkPrintf("timer start do: switch %s\r\n", json_temp_p->valuestring);
                    timer_endDo_p = cJSON_GetObjectItem(one_timer_p, "endDo");
                    json_temp_p = cJSON_GetObjectItem(timer_endDo_p, "switch");
                    colinkPrintf("timer end do: switch %s\r\n", json_temp_p->valuestring);
                }
            }
        }

ExitErr1:
    cJSON_Delete(json_root);
    return;
}
```

免责申明和版权公告

本文中的信息，包括供参考的 URL 地址，如有变更，恕不另行通知。文档“按现状”提供，不负任何担保责任，包括对适销性、适用于特定用途或非侵权性的任何担保，和任何提案、规格或样品在他处提到的任何担保。本文档不负任何责任，包括使用本文档内信息产生的侵犯任何专利权行为的责任。本文档在此未以禁止反言或其他方式授予任何知识产权使用许可，不管是明示许可还是暗示许可。Wi-Fi 联盟成员标志归 Wi-Fi 联盟所有。ESP 商标为乐鑫公司注册商标文中提到的所有商标名称、商标和注册商标属其各自自所有者的财产，特此声明。