智慧家庭/主机系统 - 接口设计

Revision:

```
2019.4.19 v0.1 scott
  1.创建

2019.4.22 v0.2 scott
1. 保持api接口与云系统主机api一致

2019.6.1  v0.3. scortt
  1. 添加   "1.5 获取设备profile信息" , "1.6 模拟生成端设备状态信息" , " 设备状态流式推送 TCP"
```

名词定义

- `SmartBox` 智能主机，安装在业主家庭，用于控制业务房间的各种智能化传感设备，例如： 空调、新风、地热、灯光等。
- `SmartServer` 云端接入smartbox的通信服务器
- `McuController` 智能主机设备上的外围传感器控制模块，与ARM系统通过UART通信。

说明

系统特点:

1. 基本流程

smartbox 连接到云端服务器， 提供Mcu和云端系统之间数据传输交换。

移动应用App控制家庭设备时，通过云端服务器SmartServer将指令转发到家庭主机 smartbox , smartbox将指令通过UART接口下发到MCU模块.

2. 端设备

端设备以 类型(sensor_type)+编号(sensor_id)做唯一识别。 每个端设备具有多项控制和参数值。业务家庭的智能化设备连接到smartbox控制主机上，每一个物理设备都被定义抽象为一个 `端设备` (sensor) 。

App或外部系统通过Tcp/Http等接入方式发送对 sensor的控制和查询操作。

Mcu主控可以定义为一种特殊sensor ， 其 sensor_type= 99 , sensor_id = 0 。

3. Mcu传输控制

消息类型

```
SensorStatusQuery = 1,  端设备查询
SensorStatusValue = 2,  端设备状态上报
SensorValueSet = 3,     端设备参数值设置
McuStatusQuery = 4,     Mcu设备查询
McuStatusValue = 5,     Mcu设备状态上报
McuValueSet = 6,          Muc设备参数设置
Heartbeat = 20          心跳
```

协议规格

```
Host <-> Mcu 通信格式

   A,B,C,D,E,F,G

   A - 消息类型 SensorMessageType (1..6,20)
   B - sensor_type (1..3)
   C - sensor_id (1..3)
   D - feature_name ( 2,3)
   E - feature_value ( 2,3)
   F - flag 控制符，在一次传递多个连续值时用于控制是否连续的消息包 (0 : 最后一项, 1: 不是最后一项)
   G - 包分隔符 '\n'

编码:
 M = A,B,C,D,E 消息内容
 P = HEX( M + CRC16(M) )
 P' = P + G

解码:
 P = P'[:-1]
 P = BIN(P)
 M = P[:-2]
 if CRC16(M) == P[-2:]:
    pass
 else:
    error

室内机MCU上报防区状态
 A = SensorStatusValue(2)
 B = 0
 C = 1-16 防区编号
 D = io
 E = 0/1 开闭状态

室内机下发MCU开门指令

 A = McuValueSet(6)
 B = 0
 C = 0
 D = 0
 E = 0

室内机上报心跳
 A = Heartbeat(20)
 B = C = D = E = 0
```

主机端口：

1.ssdp server

智能主机(smartbox)接入家庭网络，地址由路由器dhcp获取， 网络内的控制设备(client)通过ssdp协议发现智能主机。 智能主机会将可访问的主机地址和url发送给控制设备(client)。

```

```

错误码

```
#define Error_NoError 0
#define Error_UnknownError 1    // 未定义
#define Error_SystemFault 1001  // 系统错误
#define Error_TokenInvalid 1002 // 令牌错误
#define Error_AccessDenied 1003 // 访问受限
#define Error_PermissionDenied 1004 // 权限受限
#define Error_ParameterInvalid 1005 // 参数无效
#define Error_PasswordError 1006    // 密码错误
#define Error_UserNotExist 1007     // 用户不存在
#define Error_ObjectHasExist 1008   // 对象已存在
#define Error_ObjectNotExist 1009   // 对象不存在
#define Error_ResExpired 1010       // 资源过期
#define Error_ReachLimit 1011       // 达到上限

#define Error_DeviceServerNotFound  2001 //接入服务器未配置
```

Http 返回消息格式

```
    {
      status: 0 ,   // 0 : 正常  ,  1 :异常
      errcode: 0,  // 错误码
      errmsg: '' , // 错误信息
      result: {}   // 返回的数据对象，dict/array/object/简单数据类型(string,int,float)
    }
```

1. 设备功能接口

1.1 主机状态查询

名称:

getDeviceStatus(id)

描述

查询设备的当前运行状态信息，包括当前所有连接主机的家庭智能设备的状态值。

Request

```
URL: /api/smartbox/status
Medthod:    GET
Headers:
Character Encoding: utf-8
Content-Type: x-www-form-urlencoded
Query Parameters:
    - id    设备编号
```

Response

```
Headers:
Character Encoding: utf-8
Content-Type: application/json
Data:
  - status  状态码 0 : succ; others : error
  - errcode 错误码
  - errmsg  错误信息
  - result
    - id                    设备硬件编号
    - time              当前时间
    - host_ver          软件版本号
    - mcu_ver               软件版本号
    - boot_time         启动时间
    - ips                当前设备ip地址 多个ip以 , 分隔
    - loginserver_url        登录服务器地址
    - alarm_enable          启用报警
    - watchdog_enable       启用看门狗
    - sensors       (array)
        - sensor_type           端设备类型
        - sensor_id         端设备编号
        - p(1..n)               端设备当前运行参数值
```

Examples

```
{
  status:0,
  result:{
    id: 'FB00027800000000001',
    time: 1500092322,
    host_ver: 1.0.1,
    mcu_ver: 1.0.0 ,
    boot_time: 150000765,
    ips: '192.168.0.101,11.0.0.11 ',
    loginserver_url : 'svr1.smarthome.com',
    alarm_enable : 1
    watchdog_enable : 1
    sensors : [
        { sensor_type: 5 ,  sensor_id: 1, s:'on' , p: 1 },
        { sensor_type: 9 ,  sensor_id: 1, x:'on' , y: 1 },
        ....
    ]
  }
}
```

Remarks

1.2 室内主机参数设置

名称:

setSmartBoxParams()

描述

设置主机运行参数

Request

```
URL: /api/smartbox/params
Medthod:    POST
Headers:
Character Encoding: utf-8
Content-Type: x-www-form-urlencoded
Query Parameters:
Body:
   - id                  设备编号
   - name                参数名称
   - value               参数值
   - module              设备模块编号    0 : arm , 1:mcu

** 支持参数(name) **
   watchdog_enable    是否启用看门狗 0 :close , 1: open
   alarm_enable       是否启用报警功能 0: close , 1: open
   reboot             设备重启等待时间 0: 即刻 ,   n : 推迟秒数
   save               设置参数保存 0:不保存（默认） , 1:保存
```

Response

```
Headers:
Character Encoding: utf-8
Content-Type: application/json
Data:
  - status  状态码 0 : succ; others : error
  - errcode 错误码
  - errmsg  错误信息
  - result
```

Remarks

1.3 端设备状态值查询

名称:

getSensorStatusValue()

描述

查询某一个端设备的当前运行值。 端设备以 类型(sensor*type)+编号(sensor*id)做唯一识别。 每个端设备具有多项控制和参数值。

Request

```
URL: /api/smartbox/sensor/status
Medthod:    GET
Headers:
Character Encoding: utf-8
Content-Type: x-www-form-urlencoded
Query Parameters:
    - sensor_type  端设备类型
    - sensor_id       端设备编号
Body:
```

Response

```
Headers:
Character Encoding: utf-8
Content-Type: application/json
Data:
  - status  状态码 0 : succ; others : error
  - errcode 错误码
  - errmsg  错误信息
  - result
    - sensor_type
    - sensor_id
    - p (1..n) 多个端设备的状态值
```

Examples

```
    {
      status:0,
      result:{
         sensor_type: 5 ,
         sensor_id: 1,
         s:'on' ,
         p: 1
      }
    }
```

Remarks

1.4 端设备参数设置(控制)

名称:

setSensorValue()

描述

下发对端设备的命令控制

Request

```
URL: /api/smartbox/sensor/params
Medthod:    POST
Headers:
Character Encoding: utf-8
Content-Type: x-www-form-urlencoded
Query Parameters:
Body:
    - sensor_type   端设备类型
    - sensor_id     端设备编号
    - name      参数名称
    - value     参数值
```

Response

```
Headers:
Character Encoding: utf-8
Content-Type: application/json
Data:
  - status  状态码 0 : succ; others : error
  - errcode 错误码
  - errmsg  错误信息
  - result
```

Remarks

1.5 获取设备profile信息

名称:

getProfile()

描述

App第一次连接到智能设备需查询获取设备profile信息来呈现房间智能设备的拓扑

Request

```
URL: /api/smartbox/profile
Medthod:    GET
Headers:
Character Encoding: utf-8
Content-Type: x-www-form-urlencoded
Query Parameters:
```

Response

```
Headers:
Character Encoding: utf-8
Content-Type: application/json
Data:
  - status  状态码 0 : succ; others : error
  - errcode 错误码
  - errmsg  错误信息
  - result
    -           参见 [ 附录.profile ] 定义细节
```

Remarks

1.6 模拟生成端设备状态信息

名称:

generateSensorStatus()

描述

生成设备状态信息，检查数据是否通过tcp接口传递到App，并检查是否上报状态数据到云系统服务器

Request

```
URL: /api/smartbox/sensor/status/generate
Medthod:    GET
Headers:
Character Encoding: utf-8
Content-Type: x-www-form-urlencoded
Query Parameters:
    - sensor_type  int. 设备类型
    - sensor_id       int. 设备编号
    - name              string/int. 参数名称
    - value           string/int. 参数值
```

Response

```
Headers:
Character Encoding: utf-8
Content-Type: application/json
Data:
  - status  状态码 0 : succ; others : error
  - errcode 错误码
  - errmsg  错误信息
  - result
    -           参见 [ 附录.profile ] 定义细节
```

Remarks

附录

1. 设备状态流式推送

App程序主动连接到设备主机的 TCP `7002` 端口，等待设备主机推送设备状态消息到App。

状态编码格式:

```
{
   "id" : "",
   "name" : "sensor_status",
   "values" : {
      "params" : {
         "1" : "1"
      },
      "sensor_id" : 1,
      "sensor_type" : 2
   }
}
```

2. 智能设备Profile定义规格

```
{
  "id": "9901",
  "name": "house-template-1",
  "version": "1.0.0",
  "author": "scott",
  "project_id": "9999",
  "garden_id" : "home",
  "manufacture": "fanbei-smart",

  "house": {"name": "我的家"},
  "rooms": {
    "A":{"name": "客厅" },
    "B":{"name": "主卧" },
    "C":{"name": "卫生间" }
  },

  "sensor_defines": [
    {
      "name": "灯",
      "type": 2 ,
      "vendor": "Panasonic",
      "model": "YJ-911",
      "features": {
        "switch": {
          "name": "开关",
          "id": 1,
          "value_type": {"name": "enum", "items": ["on","off"] ,"default":"off"},
          "commands": {
            "on": {"name": "开", "value": 1},
            "off": {"name": "关", "value": 0}
          }
        },
        "brightness": {
```

```
          "name": "亮度",
          "id": 2,
          "value_type": {"name": "num", "start": 1,"end": 10,"step": 1,"default":
1},
          "commands": {
            "max": {"name": "最亮","value": 10},
            "min": {"name":"最暗", "value": 1},
            "up": {"name": "增亮","value": "u"},
            "down": {"name": "减暗","value": "d"}
          }
        },
        "color": {
          "name": "颜色",
          "id": 3,
          "value_type": {"name": "num","start": 1,"end": 3,"step": 1,"default": 1}
,
          "commands": {
            "red": {"name": "红","value":1},
            "green": {"name": "绿","value":2},
            "blue": {"name": "蓝","value":3},
            "next": {"name": "下一种","value":"n"},
            "back": {"name": "上一种","value":"b"}
          }
        }

      }
    }
  ],

  "sensors": [
    {"type": 2,"id": 1,"room_id": "A","name": "吊灯"},
    {"type": 2,"id": 2,"room_id": "A","name": "床头灯"}
  ]

}
```