API 说明文档

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **时间** | **起草** | **修订** | **版本** | **说明** |
| 2016/10/9 | Mingao | Lewis | V0.0.1 | 初步完成用户相关操作说明 |
| 2016/10/10 | Lewis |  | V0.0.2 | 加入Websocket机制 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**1． 总述**

客户端通过HTTP发起数据请求，服务端回复JSON数据格式响应。

回复数据结构为：

**type** **MessageInfo struct {**

**ErrorNumber int `json:"error"`**

**Action string `json:"action"`**

**Data interface{} `json:"data"`**

**}**

error 表示为错误号

action 表示为动作 (请求/回复/通知)

data 表示为回复数据（注：根据不同的请求对应不同的数据）

**其中error错误号为：**

**const** (

    SUCCESS = 0x00 //成功

    FAILED = 0x01 //失败

    //LOCAL

    USERNAME\_NIL = 0x10 //用户名为空

    PASSWORD\_NIL = 0x11 //密码为空

    NO\_USERNAME = 0x12 //用户名不存在

    INVALID\_USERNAME = 0x13 //用户名非法

    INVALID\_PASSWORD = 0x14 //密码不正确

    NOSAME\_PASSWORD = 0x15 //两次密码不一致

    REGISTER\_USER = 0x16 //用户已经注册

LONGINNED\_USER = 0x17 //用户已登录

ERROR\_USERINFORMATION = 0X18 //清除了Session

INVALID\_UID = 0x19 //非法的UID

    //SERVER

    SERVER\_NORUNNING = 0x20 //未启动服务器

    SERVER\_NOFOUND = 0x21 //未发现服务器

    SERVER\_ISBUSY = 0x22 //服务器忙

    ACCESS\_TIMEOUT = 0x23 //访问超时

    //OTHER

INVALID\_PARAMETER = 0x30 //参数非法

    ACCESS\_OUTRANGE = 0x31 //超出范围

    //to add

    CONTACT\_ADMIN = 0x40 //联系管理员

    NOTFOUND\_CONFILE = 0x41 //未找到配置文件

    INVALID\_GWID = 0x42 //无效的gwid

)

（注：当错误号为0x00时，表示当前操作成功。

当错误号为0x01时，必须带有回复数据，指明错误原因

当错误号为其它值时，回复数据可有可无， 视具体情况而定）

**action的状态号：**

**const** (

    REQUEST = "request"

REPLY = "reply"

NOTIFY = "notify"

)

（注：状态号表示请求是否能正常执行，回复数据为字符串）

**2. 用户管理**

**2.1****用户请求登录**

请求方试: POST

请求命令：/data/login?username=user&password=pwd

回复数据：

a)：成功登录，回复数据

{

“error” : 0

“action” : REPLY

“data”: {

uid :xxxxxxxx

}

}

实例：

{

"error": 0,

" action ": "reply",

"data": {

“uid”: “1474274034”

}

}

b)： 登录失败，回复数据

{

“error”:errno

“action”: “reply”

}

实例：

{

"error": 20,

"action": "reply",

}

（注1： 错误号20， 即0x14, 表示密码不正确。 ）

（注2： uid为系统自动生成， 当添加用户成功后，将uid直接与用户名绑定，并且保证唯一）

**2.2** **获取用户信息：**

请求方式：GET

请求命令：/data/userinfo?uid=xxxxxxxxx

1. 请求成功，回复数据

{

"error": 0,

"action": **REPLY**,

"data": {

“username”:user

“password”:pwd

….

}

}

实例：

{

"error": 0,

"action": "reply",

"data": {

“username”:"admin"

“password”: "12345678"

}

}

1. 请求失败，回复数据

{

“error”:errno

“action”: REPLY

}

实例：

{

"error": 14,

"action": "reply",

}

（注： 错误号14， 即0x12, 表示用户不存在。 ）

**2.3****添加用户信息**

请求方式：POST

请求命令：/data/adduserinfo?username=user&password=pwd&compassword=confirmpwd[&online=[online|offonline]]

1. 请求成功，回复数据

{

"error": 0,

"action": **REPLY**,

}

b)请求失败，回复数据

{

“error”:errno

“action”: **REPLY**

}

(注1：只要amdin才有添加用户信息权限)

**2.4** **更改用户信息**

请求方式：POST

请求命令：/data/userinfo?uid=xxxxxxxxx[&password=pwd&compassword=compwd][&online=[online|offline]]

**2.4.1 修改用户密码**

/data/userinfo?uid=xxxxxxxxx&password=pwd&compassword=compwd

1. 请求成功，回复数据

{

"error": 0,

"action": **REPLY**,

}

b)请求失败，回复数据

{

“error”:errno

“action”: REPLY

}

(注：如果当前用户拥有在线权限，还需要将云端密码更新)

**2.4.2 修改用户上线权限**

请求方式：POST

请求命令：/data/userinfo?uid=xxxxxxxxx&online=[online|offline]

1. 请求成功，回复数据

{

"error": 0,

"action": **REPLY**,

}

b)请求失败，回复数据

{

“error”:errno

“action”: REPLY

}

(注1：只要amdin才有修改用户上线权限)

(注2： a)如果用户没有上线权限，将权限修改成online后，除了更新权限状态外，还需要将用户信息上传到云端)

b)如果用户有上线权限，将权限修改成offline后，除了更新权限状态外，还需要将用户信息从云端删除

)

**2.4.3 同时修改密码和权限**

请求方式：POST

请求命令：/data/userinfo?uid=xxxxxxxxx&password=pwd&compassword=compwd &online=[online|offline]

1. 请求成功，回复数据

{

"error": 0,

"action": **REPLY**,

}

b)请求失败，回复数据

{

“error”:errno

“action”: REPLY

}

(注1：只要amdin才有修改用户上线权限)

(注2： a)如果用户没有上线权限，将权限修改成online后，除了更新权限状态以及本地密码外，还需要将用户信息上传到云端)

b)如果用户有上线权限，将权限修改成offline后，除了更新权限状态以及本地密码外，还需要将用户信息从云端删除

)

2.4.3

**2.5：****删除用户信息**

请求方式：GET

请求命令：/data/deluserinfo?uid=xxxxxxxxxx

1. 请求成功，回复数据

{

"error": 0,

"action": **REPLY**,

}

b)请求失败，回复数据

{

“error”:errno

“action”: REPLY

}

(注：只要amdin才有删除用户权限，但是管理员不能删除自己)

**2.6** **查看所用用户详细信息(不安全)**

请求方式：GET

请求命令：/data/userlist

1. 请求成功，回复数据

{

"error": 0,

"action": REPLY,

"data": {

"User": {

"userx": {

**….**

**｝**

}

**｝**

}

实例：

{

"error": 0,

"action": "reply",

"data": {

"User": {

"user1": {

"uid": 1474274034,

"username": "admin",

"password": "12345678",

"online": "offine",

"language": "zh-CN"

},

"user2": {

"uid": 1474343828,

"username": "everglow",

"password": "123456",

"online": "online",

"language": "zh-CN"

}

}

}

}

1. 请求失败，回复数据

{

“error”:errno

“action”: REPLY

}

(注1：只要amdin才有查看所有用户信息权限)

(注2：用户详细信息结构如下：

**type** Registry **struct** {

    UId int64 `json:"uid"`

    Username string `json:"username"`

    Password string `json:"password"`

    Online string `json:"online"`

    Language string `json:"language"`

**}**

)

* 1. **查看当前在线用户**

请求方式：POST

请求命令：/data/onlineuserinfo

a)请求成功，回复数据

{

"error": 0,

"action": REPLY,

"data": {

"User": {

"user1": {

username: user

}

….

}

}

}

实例：

{

"error": 0,

"action": "reply",

"data": {

"User": {

"user1": {

“username":admin"

}

"user2": {

"username":"everglow"

}

}

}

}

(注1：只要amdin才有修改用户上线权限)

**2.9** **用户请求退出登录**

请求方式：POST

请求命令1：/data/exit?uid=xxxxxxxxx

请求命令2：/data/exit?username=user

a)请求成功，回复数据

{

"error": 0,

"action": REPLY,

}

b)请求失败，回复数据

{

“error”:errno

“action”: REPLY

}

**2.10** **找回用户密码**

请求方式：POST

请求命令1：/data/findpwd?username=xxxxxx&gwid=yyyyyyyyyy

a)请求成功，回复数据

{

"error": 0,

"action": **REPLY**,

"data": {

“username”:user

“password”:pwd

}

}

实例：

{

"error": 0,

"action": "reply",

"data": {

“username”:"admin"

“password”: "12345678"

}

}

b)请求失败，回复数据

{

“error”:errno

“action”: REPLY

}

**3 设备管理**

（特别说明：当利用HTTP协议请求控制设备时，设备状态发生改变后，其它用户无法得知设备状态已经改变。后期急需调整为利用Websocket协议通信方式，因此后期对于设备管理接口会调整）

**3.1 获得所有网关信息**

请求方式：**GET**

请求命令：/data/gatewaylist

1. 请求成功，回复数据

{

"error": 0,

"statue": "reply",

"data": [

{

"id": 0,

"category": "gateway",

"gatewayName": "gateway1",

"address": "192.168.10.99",

"protocol": "SIMULATE",

~~"nodeData": [~~

~~{~~

~~"id": 0,~~

~~"category": "node1",~~

~~"nodename": "test1",~~

~~"address": "fe80::19c0:d694:85c1:b81",~~

~~"protocol": "SIMULATE"~~

~~},~~

~~{~~

~~"id": 1,~~

~~"category": "node2",~~

~~"nodename": "test2",~~

~~"address": "fe80::19c0:d694:85c1:b82",~~

~~"protocol": "ZIBGEE"~~

~~},~~

~~{~~

~~"id": 2,~~

~~"category": "node3",~~

~~"nodename": "test3",~~

~~"address": "fe80::19c0:d694:85c1:b83",~~

~~"protocol": "ZWAVE"~~

~~},~~

~~{~~

~~"id": 3,~~

~~"category": "node4",~~

~~"nodename": "test4",~~

~~"address": "fe80::19c0:d694:85c1:b84",~~

~~"protocol": "ZWAVE"~~

~~},~~

~~{~~

~~"id": 4,~~

~~"category": "node5",~~

~~"nodename": "test5",~~

~~"address": "fe80::19c0:d694:85c1:b85",~~

~~"protocol": "ZWAVE"~~

~~}~~

~~]~~

},

{

"id": 1,

"category": "getway",

"gatewayName": "gateway2",

"address": "192.168.10.227",

"protocol": "SIMULATE",

~~"nodeData": [~~

~~{~~

~~"id": 0,~~

~~"category": "switch",~~

~~"nodename": "switch1",~~

~~"address": "fe80::19c0:d694:85c1:b81",~~

~~"protocol": "SIMULATE"~~

~~},~~

~~{~~

~~"id": 1,~~

~~"category": "switch",~~

~~"nodename": "switch2",~~

~~"address": "fe80::19c0:d694:85c1:b81",~~

~~"protocol": "SIMULATE"~~

~~},~~

~~{~~

~~"id": 2,~~

~~"category": "test",~~

~~"nodename": "test3",~~

~~"address": "fe80::19c0:d694:85c1:b81",~~

~~"protocol": "SIMULATE"~~

~~}~~

~~]~~

},

{

"id": 2,

"category": "getway",

"gatewayName": "gateway3",

"address": "192.168.10.1",

"protocol": "SIMULATE",

~~"nodeData": [~~

~~{~~

~~"id": 0,~~

~~"category": "switch",~~

~~"nodename": "switch1",~~

~~"address": "fe80::19c0:d694:85c1:b81",~~

~~"protocol": "SIMULATE"~~

~~},~~

~~{~~

~~"id": 1,~~

~~"category": "switch",~~

~~"nodename": "switch2",~~

~~"address": "fe80::19c0:d694:85c1:b81",~~

~~"protocol": "SIMULATE"~~

~~}~~

~~]~~

}

]

}

**｝**

（注： 网关与节点数据如下，仅作参数）

**type** GateWayInfo **struct** {

    Id int64 `json:"id,int"`

    Category string `json:"category"`

    GatewayName string `json:"gatewayName"`

    Address string `json:"address"`

Protocol string `json:"protocol"`

    NodeData []NodeInfo `json:"nodeData"`

}

**type** NodeInfo **struct** {

    Id int64 `json:"id,int"`

    Category string `json:"category"`

    Nodename string `json:"nodename"`

    Address string `json:"address"`

    Protocol string `json:"protocol"`

}

data：即返回一个GateWayInfo数组，这个数组NodeData字段又包含一个NodeInfo数组。

**3.2 请求结点信息**

请求方式：**POST**

请求命令：/data/gatewaylist?gid=x

1. 请求成功，回复数据

{

"error": 0,

"statue": "reply",

"data": [

{

"id": 0,

"category": "switch",

"nodename": "switch1",

"address": "fe80::19c0:d694:85c1:b81",

"protocol": "SIMULATE"

},

{

"id": 1,

"category": "switch",

"nodename": "switch2",

"address": "fe80::19c0:d694:85c1:b81",

"protocol": "SIMULATE"

},

{

"id": 2,

"category": "test",

"nodename": "test3",

"address": "fe80::19c0:d694:85c1:b81",

"protocol": "SIMULATE"

}

]

}

1. 请求失败， 回复数据

{

"error": 1,

"statue": "reply",

“data”: “xxxxxxxxxx”

}

**3.3 请求控制节点 (待完善)**

请求方式：**POST**

请求命令：/data/gatewaylist?gid=x&uid=y&cmd=zzzzz