Interface Specification

<Server>

# Introduction

This document is aimed to introduce server’s interface specification to other groups and customers. Meanwhile, this document should nail down the border of function implements between server and other parts.

By checking Section 2, you will know functions that the server could provide and the server need other parts to provide for server.

The hardware part should check Section 4 in order to know how to communicate with server by socket.

The Section 5 is aimed to describe how to test all interfaces.

# Services

## Services Provided

|  |  |  |  |
| --- | --- | --- | --- |
| # | Service | Provided By | Tested By |
| 1 | Hardware want to build stable connections with server. | server\_register | Server - T1 -5.1 |
| 2 | Hardware report data to server when it changes. | server\_report | Server - T3 -TC 5.3 |
| 3 | Hardware send heartbeat package to server. | server\_report |  |
| 4 | End user want to query hardware’s information. | server\_query, DB\_checkAuthority | Server - T2 -TC 5.2 |
| 5 | End user want to send command to hardware. | server\_command, DB\_queryHardware, DB\_checkAuthority | Server - T2 -TC 5.2 |
| 6 | End user want to communicate with DB.(When it comes to something about user, building, room and so on.) | server\_DB, DB’s other functions. | Server - T4 -TC 5.4 |

## Access Method

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| server\_report | Hardware use the socket which built in register stage to send data. | data | Text in JSON format | The content of data could be different when hardware want to report its data or send heartbeat package. |  | 2, 3 |
| server\_register | Hardware built a socket with server on ‘IP:443’. And then send data. | data | Text in JSON format | The JSON format data should contains the following fields:  ‘hid’: Hardware’s unique ID.  ‘type’: The type of this socket. (Report / Receive)  ‘auth’: The authenticating key which will be confirmed by server. | Wrong hardware’s ID or authenticating key. | 1 |
| server\_query | Send post/get request to ‘IP:80/api/hardware’ | data | HTTP request parameters package | The JSON format data should contains the following fields:  ‘uid’: The user’s unique ID.  ‘sid’: The user’s security ID.  ‘hid’: The target hardware’s unique ID. | The user don’t have authority to access the hardware. | 4 |
| server\_command | Send post/get request to ‘IP:80/api/command’ | data | HTTP request parameters package | The JSON format data should contains the following fields:  ‘uid’: The user’s unique ID.  ‘sid’: The user’s security ID.  ‘hid’: The target hardware’s unique ID.  ‘cmd’: The user command. | The user don’t have authority to access the hardware. | 5 |
| server\_DB | Send post/get request to ‘IP:80/interface/<type>/<task>’ | data | HTTP request parameters package | The data should contain everything that the interface of DB need. | According to the DB’s response. | 6 |

## Access Method Effects

|  |  |
| --- | --- |
| **Access Method** | **Description** |
| server\_report | If the received JSON data is empty which indicating that hardware sent a heartbeat package, the server will update the hardware’s last updating time to keep it in a online state.  If the received JSON only contains ‘data’ filed indicating that the sensor want to report it latest data, the server will record the data in the RAM or Redis.  If the received JSON data contains ‘data’ and ‘cmd’ filed which indicating that the actuator’s state is change by a command, the server will record it’s latest data and the latest efficient command in RAM or Redis. |
| server\_register | The server will check whether the authenticating key is correct or not.  After, the server will check the hardware’s identification according to the the information which got from DB.  The server will build a report / receive socket according to the filed ‘type’. |
| server\_query | The server will check whether the user has authority to access this hardware.  Next, the server will request hardware’s persistent information like it’s nickname, type and so on from DB.Combining the real-time data which stored in server, the server will return this to clients. |
| server\_command | The server will check whether the user has authority to access this hardware.  Next, server will ask the IC for command which need to be sent to hardware.  Finally, the server will send the command to hardware. |
| server\_DB | The server will redirect this request to DB’s API and return what DB response to client. |

## Services Required

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| DB\_getUser | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘uid’: The user’s unique ID.  ‘sid’: The user’s security ID.  ‘hid’: The target hardware’s unique ID. |  | 4, 5 |
| DB\_queryHardware | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘uid’: The user’s unique ID.  ‘sid’: The user’s security ID.  ‘hid’: The target hardware’s unique ID. |  | 4 |
| DB\_getAllRoom | Send post/get request to ‘...’ | data | HTTP request parameters package | None |  |  |
| DB\_getRoomByHID | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘hid’: The target hardware’s unique ID. |  |  |
| DB\_getSensorOfRoom | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘rid’: The target room’s unique ID. |  |  |
| DB\_getDeviceOfRoom | Send post/get request to ‘...’ | data | HTTP request parameters package | The data should contains the following fields:  ‘rid’: The target room’s unique ID. |  |  |
| DB’s other functions. | Send post/get request to ‘...’ | data | HTTP request parameters package | According to the functions of DB. |  | 6 |

# Local Types

|  |  |
| --- | --- |
| **Type** | **Value Space** |
| hardware\_data | A dict which contains hardware’s real-time data. (Stored in RAM or a build-in Redis) |
| JSON | A text-format dict which contains some field. |

# Interface Design Issues

### **4.1 The hardware build socket connection with server.**

When hardware is running, it should try to build sockets with server, in the following steps:

1. Try to build a reporting socket with server.(If the hardware is a actuator, it should build another receiving socket for receiving command.)
2. When socket is built, the hardware should sent a package to server using this socket for authenticating it’s identification.
3. The hardware will received a package in form of ‘{‘status’:0, ‘msg’:’...’}’ indicating whether the socket is admitted by the sever.

### **4.2 When actuator received a command.**

When actuator received a command from server and this command let the actuator’s state changed. The actuator should report this command with its up-to-date state to server using the ‘server\_report’ method through the socket.

# Test Cases

### 5.1 Server - T1 - Test Register Function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Description** | **Input Type/Value** | **Expected Results** | **Service** | **Preamble** |
| 1 | A new hardware call ‘server\_register’ to build connections. | JSON | 1. When socket is admitted by the server : {"status":0, "msg":"Hello Device."} 2. When hardware wants to build a receiving socket but it is not a actuator : {"status":-2, "msg":"Not An Registered Device."} 3. When hardware’s ID is not recorded in DB : {"status":-3, "msg":"Not An Registered Hardware."} 4. When authenticating key is incorrect : {"status":-1, "msg":"Authenticate Failed."} |  | 1 |
| 2 | End User call ‘server\_query’ to query this hardware’s information. | HTTP request parameters package | If this hardware is registered, then the user will get its online state. |  | 2 |

### 5.2 Server - T2 -Test Command Function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Description** | **Input Type/Value** | **Expected Results** | **Service** | **Preamble** |
| 1 | End User call ‘server\_command’ to change hardware’s state. | HTTP request parameters package | 1. When that actuator is offline : { "status": -1, "msg": "You can not operate a sensor."} 2. When user don’t have permission to operate this hardware : {"status": -2, "msg": "Device is offline."} 3. When IC don’t agree this command : {"status": -3, "msg": "..."}   4. When this command is sent to hardware : {"status": 0, "msg": "Message sent."} |  | 1 |
| 2 | End User call ‘server\_query’ to query this hardware’s information. | HTTP request parameters package | If this hardware is registered, then the user will get its up-to-date state. |  | 2 |

### 5.3 Server - T3 -Test Report Function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Description** | **Input Type/Value** | **Expected Results** | **Service** | **Preamble** |
| 1 | Change the environment of sensor. |  |  |  | 1 |
| 2 | End User call ‘server\_query’ to query this hardware’s information. | HTTP request parameters package | If this hardware is registered, then the user will get its up-to-date state. |  | 2 |

### 5.4 Server - T4 -Test Interface Function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Description** | **Input Type/Value** | **Expected Results** | **Service** | **Preamble** |
| 1 | Call DB’s function through Server’s interface | HTTP request parameters package | The DB received correct request and the client received correct response. |  | 1 |