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 requests UID from Server. | server\_allocate | Server - T1 -5.1 |
| 2 | Hardware syncs GPIO with Server. | server\_report | Server - T3 -TC 5.3 |
| 3 | End User send command to server. | server\_command |  |
| 4 | End User updates Server’s data. | server\_update | Server - T2 -TC 5.2 |
| 5 | End user queries Server’s data. | server\_query | Server - T2 -TC 5.2 |

## Access Method

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| server\_allocate | http://SERVER\_ADDR/hardware/allocate | Nothing | Nothing | The Raspberry requests Server to allocate it a unique ID. |  | 2, 3 |
| server\_report | http://SERVER\_ADDR/command/report | data | Text in JSON format | The JSON format data should contains the following fields:   1. uid-String-Raspi Unique ID 2. content-String-Sensor Data | Wrong hardware UID. | 1 |
| server\_command | http://SERVER\_ADDR/command/command | data | HTTP request parameters package | The JSON format data should contains the following fields:   1. uid-Integer-User ID 2. token-String-User Token 3. command-String-Json String | The user don’t have permission to do that. | 4 |
| server\_update | http://SERVER\_ADDR/interface/<task> | data | HTTP request parameters package | The JSON format data should contains the following fields:   1. user-String-User Account (Authentication) 2. password-String-User Password (Authentication) 3. Other text filed according to task | The user don’t have permission to do that. | 5 |
| server\_query | http://SERVER\_ADDR/interface/<task> | data | HTTP request parameters package | The JSON format data should contains the following fields:  1. user-String-User Account (Authentication)  2. password-String-User Password (Authentication)  3. Other text filed according to task | The user don’t have permission to do that. | 6 |

## Access Method Effects

|  |  |
| --- | --- |
| **Access Method** | **Description** |
| server\_allocate | The Raspberry PI got an UID from server. And then Raspbery PI could communicate with Server by using this UID. |
| server\_report | Raspberry PI sync the data with server. To be more specifically, Raspberry PI report the sensors’ GPIO value to server and then got the latest GPIO setting and GPIO value of device (light and alarm). |
| server\_command | The server check the permission of user and then apply command to hardware. |
| server\_update | The server check the permission of user and then apply modification. |
| server\_query | The server check the permission of user and then answer the queries. |

## Services Required

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Access Method** | **Parameter name** | **Parameter type** | **Description** | **Exceptions** | **Map to services** |
| DB\_Interface | Sqlite Connection | Nothing | Nothing | Server communicate with DB file by sqlite3 connection. |  | 4, 5 |

# Local Types

|  |  |
| --- | --- |
| **Type** | **Value Space** |
| Raspi | Raspberry PI |
| JSON | A text-format dict which contains some field. |

# Interface Design Issues

Nothing.

# Test Cases

### 5.1 Server - T1 - Test Register Function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Step** | **Description** | **Input Type/Value** | **Expected Results** | **Service** | **Preamble** |
| 1 | Raspi requests server to allocate an UID to it. | Nothing |  |  | 1 |
| 2 | Raspi got the UID and save it in local configuration file. | HTTP response. | Return {"status":0, "info":"UID"} |  | 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 |