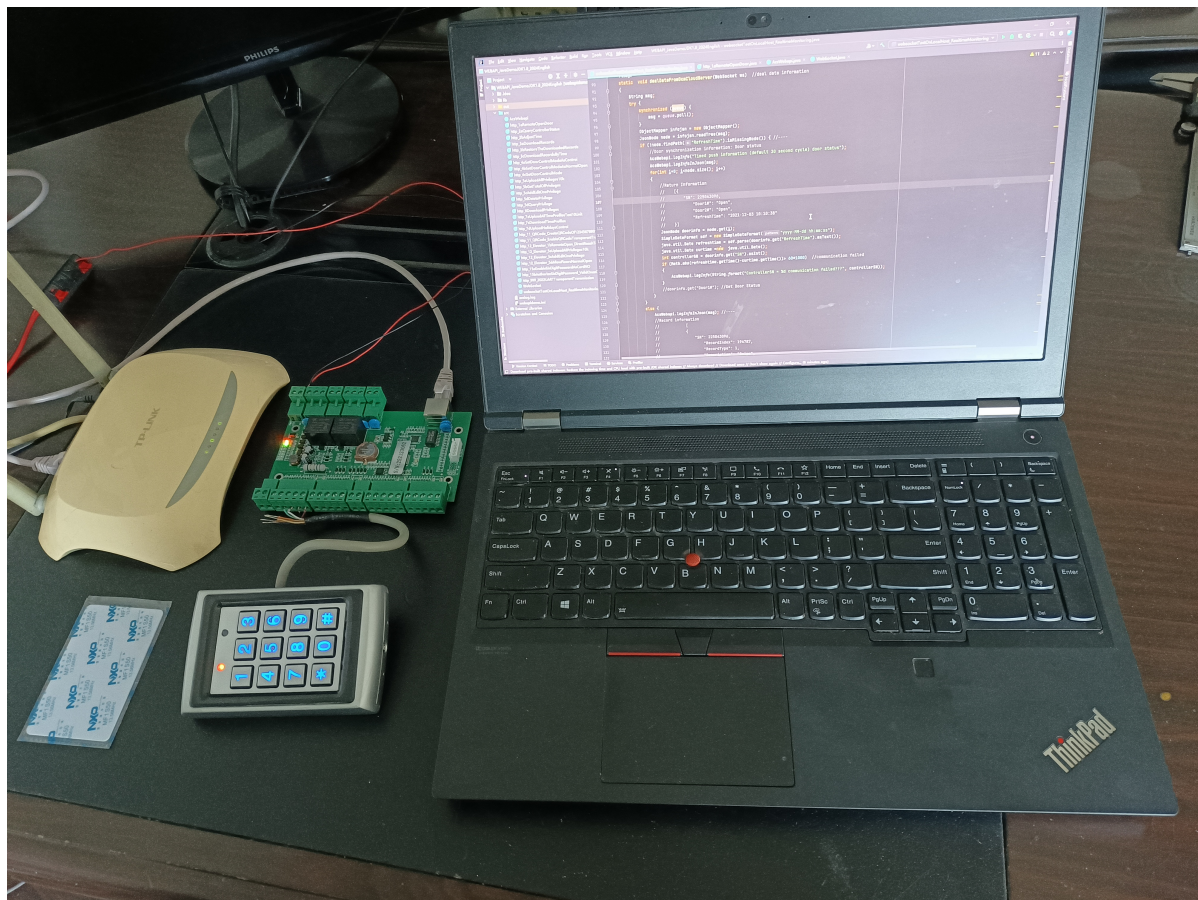


1. Requirements

This document will help you to develop the door access controller step by step.

Device	Notes
Router	
Door Controller	connect to the Router by wire
PC or Notebook	connect to the Router by wire or WIFI (OS: win 7, win 10 or win 11)
IC Card Reader	connect to the door access controller



2. Step1. start DcmCloudServer

2.1 run the program On Linux

Open the terminal:

Terminal executes DcmCloudServerEnglish

[if running in the background for a long time, use `nohup ./DcmCloudServerEnglish &`]

Please note that in some operating systems, the file format needs to be changed to executable (`chmod +x DcmCloudServerEnglish` in the terminal)

2.2 run the program On Windows

Double click to run DcmCloudServerEnglish, or configure it to run as a Windows service for a long time (eg: NSSM)

3. Step2. Realtime-monitoring

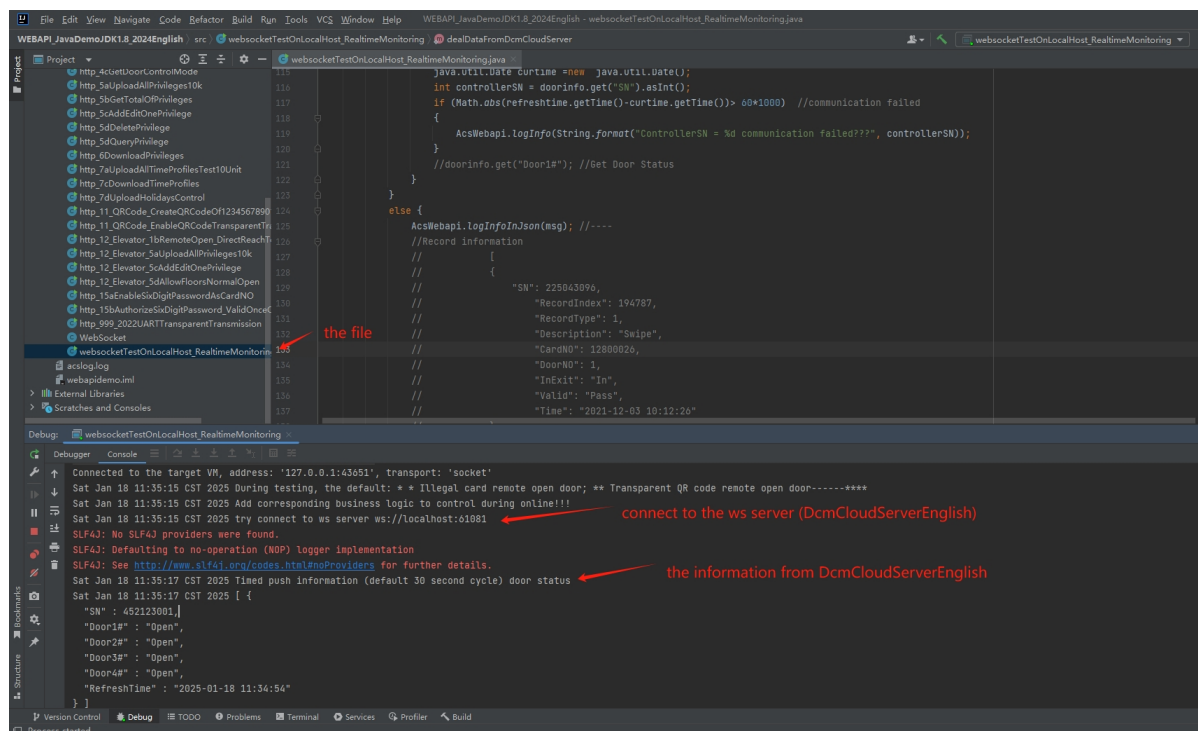
The demo as websocket client connects to the websocket server (port: 61081) on the DcmCloudServer.

```
private static final String SERVER = "ws://localhost:61081"; //Dcm websocket service ip/port
```

run

WEBAPI_JavaDemoJDK1.8_2024English\src\websocketTestOnLocalHost_RealtimeMonitoring.java

in the java ide (eg.: IntelliJ IDEA 2022.3.2(Community Edition))



3.1. WS Server push Information to the Client

3.1.1. Timed push information (default 30 second cycle) door status

		[{ "SN" : 452123001, "Door1#" : "Closed", "Door2#" : "Open", "Door3#" : "Open", "Door4#" : "Open", "RefreshTime" : "2025-01-18 11:34:54" }]	
	Arrays		
SN	Number	452123001	Controller SN(9 digits)
1#Door	String	"Closed"	range: "Open","Closed" No.1 door status: Closed
2#Door	String	"Open"	No.2 door status: Open
3#Door	String	"Open"	No.3 door status: Open
4#Door	String	"Open"	No.4 door status: Open
RefreshTime	String	"2025-01-18 11:34:54"	Information Refresh Time of the PC

3.1.2. Actively pushed record information (RecordType==1)

After swiping ic cards, you can get record information.

```
Sat Jan 18 11:43:12 CST 2025 Timed push information (default 30 second cycle) door status
Sat Jan 18 11:43:12 CST 2025 [ {
  "SN" : 452123001,
  "Door1#" : "Open",
  "Door2#" : "Open",
  "Door3#" : "Open",
  "Door4#" : "Open",
  "RefreshTime" : "2025-01-18 11:43:11"
} ]
Sat Jan 18 11:43:24 CST 2025 [ {
  "SN" : 452123001,
  "RecordIndex" : 599,
  "RecordType" : 1,
  "Description" : "Denied Access:No PRIVILEGE",
  "CardNO" : 2152191830,
  "DoorNO" : 1,
  "InExit" : "In",
  "Valid" : "Forbidden",
  "Time" : "2025-01-18 11:43:25"
} ]
```

 new record information

###

		[{ "SN" : 452123001, "RecordIndex" : 599, "RecordType" : 1, "Description" : "Denied Access:No PRIVILEGE", "CardNO" : 2152191830, "DoorNO" : 1, "InExit" : "In", "Valid" : "Forbidden", "Time" : "2025-01-18 11:43:25" }]	
	Arrays		
SN	Number	452123001	Controller SN(9 digits)
RecordIndex	Number	599	record index
RecordType	Number	1	range: 1,2,3 =1 swipe record =2 Events(remote open, push button, Door Status, Reboot) =3 Warns(Fire etc.)
Description	String	"Denied Access: No PRIVILEGE"	
CardNO	Number	2152191830	
DoorNO	Number	1	range: 1,2,3,4
InExit	String	"In"	range: "In","Exit"
Valid	String	"Forbidden"	range: "Pass","Forbidden"
Time	String	"2025-01-18 11:43:25"	time format: yyyy-MM-dd HH:mm:ss

3.1.3. Dealing record information

3.1.3.1. source code for dealing

the function dealDataFromDcmServer is followed.

You can analyse "long vcard = recinfo.get("CardNO").asLong();" and decide whether to open door.

```
static void dealDataFromDcmCloudServer(WebSocket ws) //deal data information
{
    String msg;
    try {
        synchronized (queue) {
            msg = queue.poll();
        }
    }
}
```

```

    ObjectMapper infojsn = new ObjectMapper();
    JsonNode node = infojsn.readTree(msg);
    if (!node.findPath("RefreshTime").isMissingNode()) { //----
        //Door synchronization information: Door status
        Acswebapi.logInfo("Timed push information (default 30 second
cycle) door status");
        Acswebapi.logInfoInJson(msg);
        for(int i=0; i<node.size(); i++)
        {
            //Return Information
            //    [{
            //        "SN": 225043096,
            //        "Door1#": "Open",
            //        "Door2#": "Open",
            //        "RefreshTime": "2021-12-03 10:10:38"
            //    }]
            JsonNode doorinfo = node.get(i);
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd
hh:mm:ss");
            java.util.Date refreshtime =
sdf.parse(doorinfo.get("RefreshTime").asText());
            java.util.Date curtime =new java.util.Date();
            int controllerSN = doorinfo.get("SN").asInt();
            if (Math.abs(refreshtime.getTime()-curtime.getTime())>
60*1000) //communication failed
            {
                Acswebapi.logInfo(String.format("ControllersN = %d
communication failed???", controllerSN));
            }
            //doorinfo.get("Door1#"); //Get Door Status
        }
    }
    else {
        Acswebapi.logInfoInJson(msg); //----
        //Record information
        //    [
        //    {
        //        "SN": 225043096,
        //        "RecordIndex": 194787,
        //        "RecordType": 1,
        //        "Description": "Swipe",
        //        "CardNO": 12800026,
        //        "DoorNO": 1,
        //        "InExit": "In",
        //        "Valid": "Pass",
        //        "Time": "2021-12-03 10:12:26"
        //    }
        //    ]
        for (int i = 0; i < node.size(); i++) {
            JsonNode recinfo = node.get(i);
            if (!recinfo.findPath("RecordType").isMissingNode()) {
                if (recinfo.get("RecordType").asInt() == 1) { //Only
focus on swipe records
                    //new record information
                    if
(recinfo.get("Valid").asText().equals("Forbidden")) //no pass

```

```

        {
            //???????????????? Issue VirtualCardNO door
opening command according to customer rules ?????????????????????????????????
            int controllersSN = recinfo.get("SN").asInt();
            long vcard = recinfo.get("CardNO").asLong();
            int doorNO = recinfo.get("DoorNO").asInt();
            String inout = recinfo.get("InExit").asText();

            ObjectMapper mapper = new ObjectMapper();
            ObjectNode webcmd = mapper.createObjectNode();
            webcmd.put("jsonrpc", "2.0");
            webcmd.put("method", "RemoteOpenDoor");
            ArrayNode params = webcmd.putArray("params");
            ObjectNode param = mapper.createObjectNode();
            param.put("ControllersSN", controllersSN);
            param.put("DoorNO", doorNO);
            param.put("InExit", inout.equals("In") ? 1 : 2);
            param.put("VirtualCardNO", vcard);
            params.add(param);
            webcmd.put("id", 1001);
            ws.send(webcmd.toString()); //remote open door

with virtual cardNO

            AcsWebapi.logInfo(String.format("Issue remote
door opening command for invalid card - SN=%d, swipe card number=%d",
controllersSN, vcard));
        }
    } else if (recinfo.get("RecordType").asInt() == 2) {
        //remote open door

        if
(recinfo.get("Description").asText().indexOf("RemoteOpenDoor") > 0) {
            int controllersSN = recinfo.get("SN").asInt();
            long vcard = recinfo.get("CardNO").asLong();
            int doorNO = recinfo.get("DoorNO").asInt();
            AcsWebapi.logInfo(String.format("Remote door
opening successful - card number=%d, SN=%d, door number=%d", vcard,controllersSN,
doorNO));
        }
    }
}
else if (!recinfo.findPath("PrimitHexData").isMissingNode())
//Transparent data transmission
{
    //QRCode
    String qrinfoHex = recinfo.get("PrimitHexData").asText();
    String qrinfo =
recinfo.get("ConvertedToString").asText();

    //????????????????????Perform the door opening operation
after analyzing the QR code
    int controllersSN = recinfo.get("SN").asInt();
    int doorNO = recinfo.get("DoorNO").asInt();
    long vcard = 90010020011; //Virtual CardNO
recinfo.get("CardNO").asLong();
    String inout = recinfo.get("InExit").asText();

```

```

        ObjectMapper mapper = new ObjectMapper();
        ObjectNode webcmd = mapper.createObjectNode();
        webcmd.put("jsonrpc", "2.0");
        webcmd.put("method", "RemoteOpenDoor");
        ArrayNode params = webcmd.putArray("params");
        ObjectNode param = mapper.createObjectNode();
        param.put("ControllerSN", controllerSN);
        param.put("DoorNO", doorNO);
        param.put("InExit", inout.equals("In") ? 1 : 2);
        param.put("VirtualCardNO", vcard);
        params.add(param);
        webcmd.put("id", 1001);
        ws.send(webcmd.toString()); //remote open door with
virtual cardNO

        Acswebapi.logInfo(String.format("Issue remote door
opening command for QR code - QR code information=%s, VirtualCardNO=%d, SN=%d,
doorNO=%d",qrinfo, vcard, controllerSN, doorNO));
    }
}
}
//logInfo((new ObjectMapper()).readTree(info).toPrettyString());
} catch (JsonProcessingException e) {
    //throw new RuntimeException(e);
} catch (ParseException e) {
    //throw new RuntimeException(e);
}
}
}

```

3.1.3.2. remote open door

In the demo, if the property of "Valid" is "DeniedAccess", the door will be remote opened.

"Issue remote door opening command for invalid card - SN=252123001, swipe card number=2152191830"

The RecordType of the new record is 2 (events). It indicates that the operation is completed successfully.

3.1.3.2.1. post (no reply)

		<pre>{ "jsonrpc" : "2.0", "method" : "RemoteOpenDoor", "params" : [{ "ControllerSN" : 252123001, "DoorNO" : 1, "InExit" : 1, "VirtualCardNO" : 2152191830 }], "id" : 1001 }</pre>	
jsonrpc	String	"2.0"	const
method	String	RemoteOpenDoor	
params	Object		
ControllerSN	Number	252123001	Controller SN(9 digits)
DoorNO	Number	1	range: 1,2,3,4
InExit	Number	1	range:1,2 =1 "In", =2 "Exit"
VirtualCardNO	Number	2152191830	
id	Number	1001	Can be used as a serial number [synchronous on return]

3.1.3.2.2. new record(RecordType==2)

if pushed record information (RecordType==2), it indicated that the operation is completed successfully.

		[{ "SN" : 252123001, "Index" : 84, "RecordType" : 2, "Description" : "Remote Open Door", "CardNO" : 2152191830, "DoorNO" : 1, "InExit" : "In", "Valid" : "Pass", "Time" : "2025-01-20 18:15:26" }]	
	Arrays		
SN	Number	252123001	Controller SN(9 digits)
Index	Number	84	record index
RecordType	Number	2	range: 1,2,3 =1 swipe record =2 Events(remote open, push button, Door Status, Reboot) =3 Warns(Fire etc.)
Description	String	"Remote Open Door"	
CardNO	Number	2152191830	
DoorNO	Number	1	range: 1,2,3,4
InExit	String	"In"	range: "In","Exit"
Valid	String	"Pass"	range: "Pass","DeniedAccess"
Time	String	"2025-01-20 18:15:26"	time format: yyyy-MM-dd HH:mm:ss

4. Step3. HTTP -change testControllerSN

change testControllerSN to the controller SN (eg.: 252123001) in
WEBAPI_JavaDemoJDK1.8_2024English\src\AcsWebapi.java

```
public class Acswebapi {
    //localhost
    public static String testUrl = "http://localhost:61080/";
    public static int testControllerSN = 252123001;
```

5. Step4. HTTP RemoteOpenDoor

run WEBAPI_JavaDemoJDK1.8_2024English\src\http_1aRemoteOpenDoor.java. Success as followed.

```
Debug: websocketTestOnLocalHost_RealtimeMonitoring x http_1aRemoteOpenDoor x
Debugger Console
C:\devsoft\Java\corretto-17.0.3\bin\java.exe ...
Connected to the target VM, address: '127.0.0.1:23987', transport: 'socket'
Mon May 22 15:34:03 CST 2023 RemoteOpenDoor
Mon May 22 15:34:03 CST 2023 Send Command:
{
  "jsonrpc" : "2.0",
  "method" : "RemoteOpenDoor",
  "params" : [ {
    "ControllerSN" : 226000340,
    "DoorNO" : 1
  } ],
  "id" : 1001
}
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#noProviders for further details.
Mon May 22 15:34:03 CST 2023 {
  "jsonrpc" : "2.0",
  "result" : {
    "success" : true,
    "msg" : "controller instructions execute successfully ok",
    "code" : 1,
    "ControllerSN" : 226000340
  },
  "id" : 1001
}
Mon May 22 15:34:03 CST 2023 226000340 RemoteOpenDoor OK.
Disconnected from the target VM, address: '127.0.0.1:23987', transport: 'socket'
```

5.1. post

		<pre>{ "jsonrpc" : "2.0", "method" : "RemoteOpenDoor", "params" : [{ "ControllerSN" : 252123001, "DoorNO" : 1 }], "id" : 1001 }</pre>	
jsonrpc	String	"2.0"	const
method	String	RemoteOpenDoor	
params	Arrays		

		<pre>{ "jsonrpc" : "2.0", "method" : "RemoteOpenDoor", "params" : [{ "ControllerSN" : 252123001, "DoorNO" : 1 }], "id" : 1001 }</pre>	
		<pre>{ "jsonrpc" : "2.0", "method" : "RemoteOpenDoor", "params" : [{ "ControllerSN" : 252123001, "DoorNO" : 1 }], "id" : 1001 }</pre>	
ControllerSN	Number	252123001	Controller SN(9 digits)
DoorNO	Number	1	range: 1,2,3,4
id	Number	1001	Can be used as a serial number [synchronous on return]

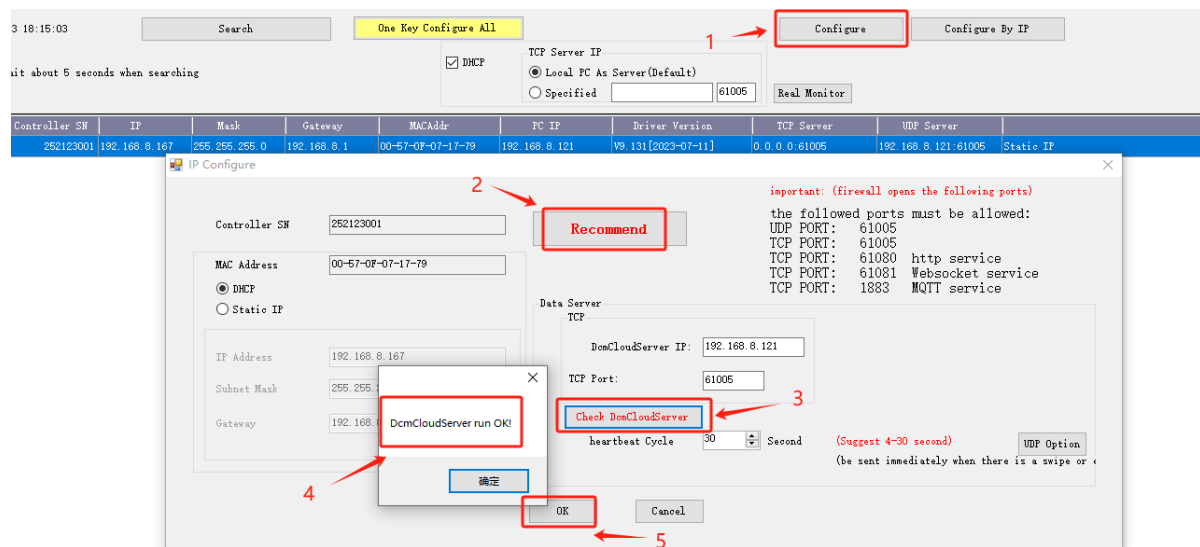
5.2. reply

		<pre>{ "jsonrpc" : "2.0", "result" : { "success" : true, "msg" : "controller instructions execute successfully ok", "code" : 1, "ControllerSN" : 252123001 }, "id" : 1001 }</pre>	
jsonrpc	String	"2.0"	const
result	Object		
success	Boolean	true	range: true, false

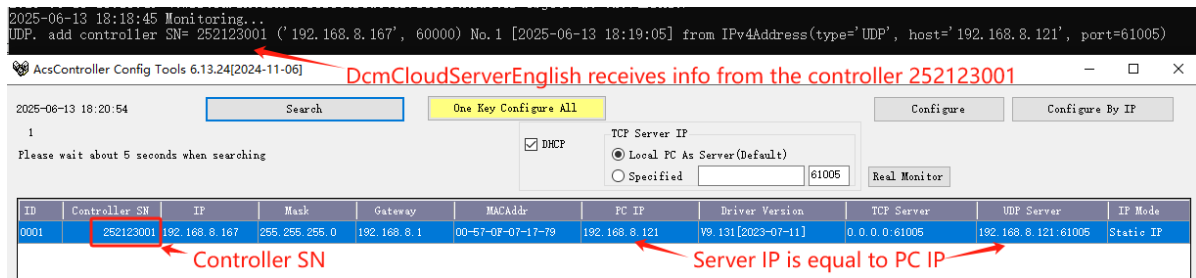
msg	String	"controller instructions execute successfully ok"	
code	Number	1	range: 0,1 =0 false =1 true
ControllerSN	Number	252123001	Controller SN(9 digits)
		{ "jsonrpc" : "2.0", "result" : { "success" : true, "msg" : "controller instructions execute successfully ok", "code" : 1, "ControllerSN" : 252123001 }, "id" : 1001 }	
id	Number	1001	Can be used as a serial number [synchronous on reply]

6. Step5. Configure the Door Controller

run the program AcsControllerConfig\AcsControllerConfig.exe



The operation 3 is checking whether DcmCloudServer is running. If it is not running, please run it. After 5 operations, try Search again, should get the followed new configuration:



The controller and DcmCloudServer can communicate each other.

7. APPENDIX

7.1. java Demo Description

Directory: WEBAPI_JavaDemoJDK1.8_2024English\src

7.1.1. A. HTTP

AcsWebapi.java **Interface**

1,2-Basic Function (RemoteOpen, QueryControllerStatus, AdjustTime)

http_1aRemoteOpenDoor.java

http_2aQueryControllerStatus.java

http_2bAdjustTime.java

3-DownloadRecords

http_3aDownloadRecords.java

http_3bRestoreTheDownloadedRecords.java

http_3cDownloadRecordsByTime.java

4-DoorControlMode NormalOpen

http_4aSetDoorControlModeAsControl.java

http_4bSetDoorControlModeAsNormalOpen.java

http_4cGetDoorControlMode.java

5,6-Privilege

http_5aUploadAllPrivileges10k.java

http_5bGetTotalOfPrivileges.java

http_5cAddEditOnePrivilege.java

http_5dDeletePrivilege.java

http_5dQueryPrivilege.java

http_6DownloadPrivileges.java

7-Time Profiles and Holidays

http_7aUploadAllTimeProfilesTest10Unit.java

http_7cDownloadTimeProfiles.java

http_7dUploadHolidaysControl.java

11-QR Code

http_11_QRCode_CreateQRCodeOf12345678901.java

http_11_QRCode_EnableQRCodeTransparentTransmission.java

12-Elevator

http_12_Elevator_1bRemoteOpen_DirectReachToDesignatedFloor.java

http_12_Elevator_5aUploadAllPrivileges10k.java

http_12_Elevator_5cAddEditOnePrivilege.java

http_12_Elevator_5dAllowFloorsNormalOpen.java

http_13_Locker_5cAddEditOnePrivilege.java

15-Open Door By Six-Digit Pwd

http_15aEnableSixDigitPasswordAsCardNO.java

http_15bAuthorizeSixDigitPassword_ValidOnceOntheSameDay.java

999-Enable Function

http_999_2022UARTTransparentTransmission.java

7.1.2. B. Websocket

WebSocket.java **Interface**

websocketTestOnLocalHost_RealtimeMonitoring.java