**WSCmdApp Protocol Interface Description**

**Ver5.1.3**

**Changsha SunVote Ltd.**

**November 2023**

Document Control:

File name: WSCmdApp Interface Description

Drafter: Chen Dingmin

Author of the latest version: Chen Dingmin

| Document version | date | author | Release Notes |
| --- | --- | --- | --- |
|  |  |  | 1. Initial version |

# Contents

[1. System Overview 5](#_Toc153615949)

[2. Design Purpose 6](#_Toc153615950)

[3. Communication Interface Design Outline 7](#_Toc153615951)

[4. Communication Method 8](#_Toc153615952)

[5. Programming Process Description 8](#_Toc153615953)

[6. JSON Command Description 8](#_Toc153615954)

[6.1 Heartbeat 10](#_Toc153615955)

[6.2 Base Station Connection, Disconnection and Reset 10](#_Toc153615956)

[6.2.1 Connect to Base Station 10](#_Toc153615957)

[6.2.2 Disconnect Base Station 11](#_Toc153615958)

[6.2.3 Reset Base Station 12](#_Toc153615959)

[6.3 Answer questions 12](#_Toc153615960)

[6.3.1 Sign-in equipment 15](#_Toc153615961)

[6.3.2 Choice Questions 17](#_Toc153615962)

[6.3.3 True or False Question 20](#_Toc153615963)

[6.3.4 Quick answers to multiple questions 22](#_Toc153615964)

[6.3.5 Voice question 24](#_Toc153615965)

[6.3.6 Rush answer questions 29](#_Toc153615966)

[6.3.7 Number questions 31](#_Toc153615967)

[6.3.8 Test questions 34](#_Toc153615968)

[6.3.9 Fill in the blanks 57](#_Toc153615969)

[6.3.10 Free question type 59](#_Toc153615970)

[6.4 Swipe card 61](#_Toc153615971)

[6.4.1 Write card data 62](#_Toc153615972)

[6.4.2 Swiping feedback 63](#_Toc153615973)

[6.4.3 NFC data clear command 64](#_Toc153615974)

[6.5 Hardware parameters 64](#_Toc153615975)

[6.5.1 Base station 65](#_Toc153615976)

[6.5.3 Keypad 91](#_Toc153615977)

[6.6 Keypad display settings 99](#_Toc153615978)

[6.6.1 Free title 99](#_Toc153615979)

[6.6.2 Custom information 100](#_Toc153615980)

[6.6.3 Username 102](#_Toc153615981)

[6.6.4 Specify user name [C100B+S6 old model] 104](#_Toc153615982)

[6.7 Keypad online information 105](#_Toc153615983)

[6.7.1 Automatically return the online information 105](#_Toc153615984)

[6.7.2 Obtain the online information actively 106](#_Toc153615985)

[6.8 System parameters 107](#_Toc153615986)

[6.8.1 Exit WSCmdApp 107](#_Toc153615987)

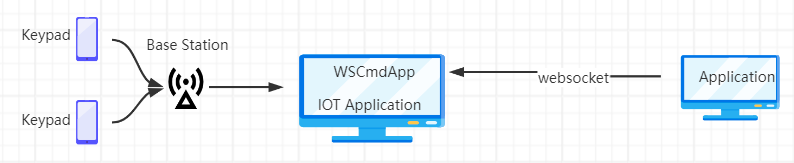
[6.9 Exception handling 108](#_Toc153615988)

[6.9.1 WSCmdApp received data exception 108](#_Toc153615989)

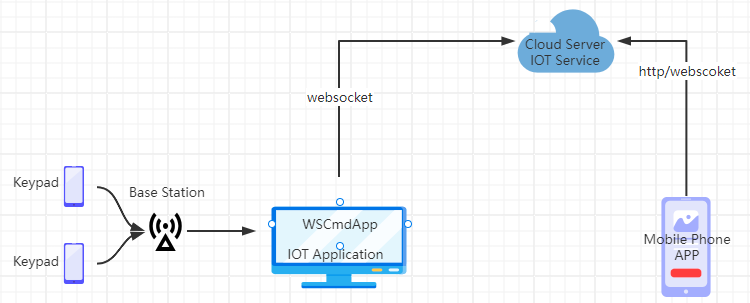
# System Overview

The WSCmdApp   
The wireless feedback system is composed of a handheld keypad, communication base station, WSCmdApp (console program), and control computer. The control computer is connected to the base station through the USB interface. Control instructions are sent from the control computer to the keypad through the base station.

WSCmdApp allows user-developed applications to communicate with the WSCmdApp wireless feedback system. The WSCmdApp console program (hereinafter replaced by WSCmdApp) provides websocket communication and uses the JSON communication format to realize the management of the interactive system and the sending and receiving functions of voting/answering instructions. The call structure diagram is as follows:



If users want to develop Internet applications, they can refer to the following design:



Currently WSCmdApp only supports Windows systems. Users only need to run the WSCmdApp program.

The WebSocket port can be set in the config.ini configuration file in the WSCmdApp program, see the following:

Explanation of keywords in the config.ini file:

Role: Current WebSocket running mode of WSCmdApp, client: client mode, server: server mode.

IP: IP address of the WebSocket server to be accessed; This parameter is valid in client mode.

Port: Port for WebSocket communication.

heartBeatOnOff: Heartbeat judgment switch, this parameter is used in conjunction with the "heartbeat" command in the protocol document.

When the parameter is configured as 0: WSCmdApp will reply with the corresponding parameter and will not enter the heartbeat detection program.

When the parameter is configured as 1: WSCmdApp will reply with the corresponding parameter and will enter the heartbeat detection program.

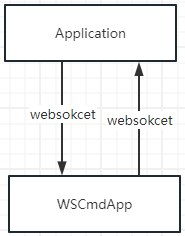
heartBeat: Interval duration of the heartbeat; Range: 10-60 seconds.

When heartBeatOnOff is configured as 1, and the WebSocket running mode is client, if no heartbeat command is received within the given time interval, the WebSocket client will restart and reconnect to the server.

# Design Purpose

1. After the protocol is encapsulated by Websocket, it will have better expansion and cross-application usage capabilities, providing a data communication basis for platform application development.

2. The application program (the subsequent program developed based on WSCmdApp is called an application program) sends instructions to WSCmdApp based on the business. After receiving the instruction, WSCmdApp completes the operation of the base station or keypad and replies to the application program.



# Communication Interface Design Outline

Json data communication is used between the application and WSCmdApp. The basic format of communication is as follows:

Send: Application -> WSCmdApp

{  
 fun: "" // Function definition  
 baseId: 1 // Base station id , numeric type  
 params: { } //Parameter object/array  
 packetTag: "" //Packet tag string   
}

fun is the name corresponding to the function, such as: "readBaseStationID" reads the id of the base station, "startChoices" //Start the choice question, "stopChoices" //Stop the choice question.

baseId: Base station ID, used to distinguish base stations, 0 represents all base stations, 1 represents base station No. 1.

params: Parameters, parameters relative to the function, which can be arrays or fields.

packerTag: This is an extension field which needs to be managed by the application program. The corresponding feedback command at the packerTag position will provide the same string, making it convenient for the application program to track and record.

Receive: WSCmdApp->Application

{  
 fun: "" //Function definition  
 baseId: 1 //Base station id, numeric type  
 infos: {} //Information object/array  
 packetTag: "" // Packet tag string.   
}

fun: The name corresponding to the function, such as: "BaseStationID" reads the base station id, "answerChoices" //Multiple choice keypad feedback value.   
baseId: Base station ID, used to distinguish base stations, 0 represents all base stations, 1 represents base station No. 1.

.   
infos: Data fed back by WSCmdApp to the APP layer. See the specific function definition for details.   
packetTag: This is an extension field which needs to be managed by the application. It returns the received string.

Field value description:

The state field in the feedback infos indicates the status: OK indicates success, FAIL indicates failure.

The values of the hardware parameter fields in the feedback infos: FAIL indicates failure, while other values indicate success.

X~Y: Values within this range specifically refer to integers.

# Communication Method

The data interaction is conducted in a send <--> reply manner. The application sends data to WSCmdApp, and WSCmdApp replies after receiving it and completing the execution.

If the application has no data to send to WSCmdApp, it can send the heartbeat command to WSCmdApp.

When the WSCmdApp is set to WebSocket server mode, the application can confirm the connection status with WSCmdApp by responding to the heartbeats sent by WSCmdApp.

When the WSCmdApp is set to WebSocket client mode, WSCmdApp will automatically determine whether to reconnect based on the settings of the heartbeat judgment switch and heartbeat parameters.

# Programming Process Description

The basic process for users to program using WSCmdApp is as follows:   
Configuration parameters: (configurable, see the configuration file config.ini for details)

Programming:

1. Use websocket communication, and IP is the IP address.   
2. Start the WSCmdApp.exe (WSCmdApp) process.   
3. The business software can communicate directly after connecting to the server. The communication data format is JSON (UTF-8 encoding). Please see the JSON command description section for specific parameters.   
4. The business software sends an exit command, and WSCmdApp.exe (WSCmdApp) exits after receiving the command.

# JSON Command Description

Json data communication is used between the application and WSCmdApp. The basic format of communication is as follows:

Send: Application->WSCmdApp

{  
 fun: "" // Function definition  
 baseId: 1 // Base station id , numeric type  
 params: { } //Parameter object/array  
 packetTag: "" //Packet tag string   
}

fun: defines the function, issue commands, and specify the function to be executed by the command.

baseId: base station ID, numeric type. If you do not specify a specific base station ID, you can pass 0 or not pass the Key value. If you want to specify multiple base station IDs, you need to issue this command multiple times and switch to different base station IDs for sending.

params: Parameters that can be an object or an array, used as an additional data package accompanying the command issued. Please refer to the specific function definition for details.

packerTag:   
data tag string, used to distinguish the current instruction from other instructions. The user can customize the data sent by the application to WSCmdApp (if the user does not have customized data, it can be the packet serial number, so that the reply data can be distinguished as to which data packet the reply is). When WSCmdApp replies to the application, it directly backfills the tag data received by the application.

Receive: WSCmdApp -> Application

{  
 fun: "" //Function definition  
 baseId: 1 //Base station id , numeric type  
 infos: {} //Information object/array  
 packetTag: "" // Packet tag string, used to distinguish the current command from other commands. WSCmdApp sends an auto-incrementing number to the application. WSCmdApp replies to the application by directly backfilling the tags received by WSCmdApp.   
}

fun: the same as the send description above.   
baseId: the same as the send description above.   
infos: the data fed back by WSCmdApp to the application layer. See the specific function definition for details.   
packetTag: A string representing the data tag, which is used to indicate the status of the feedback function as a data package attached to the application. If WSCmdApp sends data actively, it will be a set of incremented data.

Field value description:   
The “state” field in the feedback “infos” indicates the status of the operation. “OK” means success, “FAIL” means failure.

The values of the hardware parameter fields in the feedback “infos” is “FAIL” when failure occurs. For successful operations, any other value may be returned.

X~Y: Refers to integers within the specified range.

## Heartbeat

The application sends a heartbeat, and WSCmdApp responds to determine the connection status.

Send:

fun: "heartBeat" //Heartbeat   
baseId: //No such value   
params:{}//No such value   
packetTag:"1"//The received packetTag can be used to analyze the communication status

Command example:

{  
 "fun" : "heartbeat" ,  
 "packetTag" : "1"   
}

Receive:

fun: "heartBeat" //Heartbeat   
baseId: //No such value   
infos:{}//No such value   
packetTag:"1"//Data packet tag

Command example:

{  
 "fun" : "heartbeat" ,  
 "packetTag" : "1"   
}

## Base Station Connection, Disconnection and Reset

After the application successfully connects to the WSCmdApp network (receives heartbeat), it can send a command to connect to the base station. After receiving a successful connection from the base station, you can communicate with the SunVote wireless feedback system (base station, keypad).

### Connect to Base Station

Read the online base station status   
Send:   
fun: "readConnectBase" //Connect to the base station   
baseId:0 //Base station ID   
params: {} //Empty   
packetTag: "1" //

Command example:

{  
 "fun" : "readConnectBase" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun: "connectBase" //Connect to the base station   
baseId:1 //Base station ID   
infos: {"state":"OK"} //OK means connection successful   
packetTag:"1" //Received packetTag

Command example:

{  
 "fun" : "connectBase" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

The base station actively reports after being connected, and WSCmdApp reports the information.

Receive:

fun: "connectBase" //Connect to the base station   
baseId:1 //Base station ID   
infos: {"state":"OK"} //OK means connection successful   
packetTag:"1" //Received packetTag

Command example:

{  
 "fun" : "connectBase" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

### Disconnect Base Station

WSCmdApp reports information.

Receive:

fun: "disconnectBase" //Disconnect the base station   
baseId:1 //Base station ID   
infos: {} //No such value   
packetTag:"1"

Command example:

{  
 "fun" : "disconnectBase" ,  
 "baseId" : 1 ,  
 "packetTag" : "1"   
}

### Reset Base Station

Send:

fun: "resetBase" //Reset base station   
baseId:1,   
params:{} //No such value

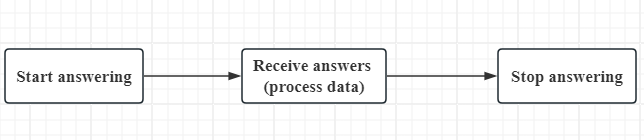
Command example:

{  
 "fun" : "resetBase" ,  
 "baseId" : 1 ,  
 "packetTag" : "1"   
}

Note: B200 supports this command

## Answer questions

Answering process:



When starting to answer questions, you can specify the display title and start answering keypad for each type of questions, or it can be empty.

The fields added to the top-level object are as follows:

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySnList | Specified keypad list | keySn array, empty means all keypads are involved. |

When stopping answering, you can specify the stop answering keypad or leave it empty for each type of questions. The fields added to the top-level object are as follows:

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySnList | Specified keypad list | keySn array, empty means all keypads are involved. |

If a keypad is specified, the receiving status of the specified keypad will be returned one by one. The received infos array object is as follows:

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |
| state | state | OK means success |

Start answering

Command example:

{  
 "keySnList" : [ {  
 "keySn" : "1837454011"          
 } , {              
 "keySn" : "1837454012"          
 } ]   
}

The receiving status of the keypad returns

fun: "startXXKeypad" //Start XX answering keypad (XX is the specific name corresponding to starting answering)   
baseId: 1//The ID of the base station that received the instruction   
infos:{}//Array object   
instruction example:

{      
 "fun" : "startBindKeypad" ,  
 "baseId" : 1 ,  
 "Info" : [ {              
 "keySn" : "1837454011" ,  
 "state" : "OK"          
 } ] ,  
 "packetTag" : "1"   
}

Reply to the keypad's receiving status return

Send:

fun: "startXXKeypad" //Start xx answering keypad   
baseId:"1"//Received base station ID   
params:{} //Empty

Command example:

{      
 "fun" : "startBindKeypad" ,  
 "baseId" : 1 ,  
 "packetTag" : "1"   
}

Stop answering

Command example:

{         
 "keySnList" : [ {              
 "keySn" : "1837454011"          
 } , {              
 "keySn" : "1837454012"          
 } ]   
}

The receiving status of the keypad returns

fun: "stopXXKeypad" //Stop XX answering keypad (XX is the specific name corresponding to stop answering)   
baseId: 1//The ID of the base station that received the instruction   
infos:{}//Array object

Command example:

{  
 "fun": "stopBindKeypad",  
 "baseId": 1,  
 "infos": [{  
 "keySn": "1837454011",  
 "state": "OK"  
 }],  
 "packetTag": "1"  
}

Reply to the keypad's receiving status return

Send:

fun: "stopXXKeypad" //Stop xx answering keypad   
baseId: "1"//Received base station ID   
params: {} //Empty

Command example:

{  
 "fun" : "stopBindKeypad" ,  
 "baseId" : 1 ,  
 "packetTag" : "1"   
}

### Sign-in equipment

#### Start sign-in

Send:

fun: "startBind" //Start sign-in   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| bindMode | Sign-in mode | 1: Sign in with specified key  2: Sign in with PIN code (default value) |
| modifyMode | Modify mode | 0: Cannot be modified  1: Can be modified (default value) |
| lessMode | reserved | 0: (default value) |
| options | reserved | 10: (default value) |
| optionalN | Optional number of digits | 1≤N≤14 |
| keyValue | Specify key value | 0: Not specified [OK key to sign in]  1:1  2:2  3:3  4:4  5:5  6:6  7:7  8:8  9:9  10:0  11: Up key  12: Down key  13: OK  14:ESC  15:Menu |
| tipTitle | prompt title | String, no more than 16 bytes in length |

Command example:

{  
 "fun": "startBind",  
 "baseId": 0,  
 "params": {  
 "bindMode": "1",  
 "lessMode": "0",  
 "modifyMode": "1",  
 "limitNumber": "1",  
 "options": "10",  
 "optionalN": "10",  
 "keyValue": "0"  
 },  
 "packetTag": "1"  
}

#### Receive sign-in information

Receive:

fun: "answerBind" //Sign-in returns   
baseId: "1"//The ID of the base station that received the instruction   
infos:{}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| time | Answer time | Unit second |
| keySn | Keypad Sn |  |
| keyValue | key value | Keypad submitted value |

Command example:

{  
 "fun": "answerBind",  
 "baseId": 1,  
 "infos": {  
 "time": "2.42",  
 "keySn": "1479824643",  
 "keyValue": "1234"  
 },  
 "packetTag": "1"  
}

#### Stop sign-in

Send:

fun: "stopBind" //Stop sign-in   
params:{} //Default is empty

Command example:

{  
 "fun" : "stopBind" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopBind" //Stop sign-in   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "stopBind" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| Base station + keypad | Support/not support | Remark |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

### Choice Questions

#### Start answering

Send:

fun: "startChoices" //Start choice question  
params: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| optionsMode | Option Category | 1: Display letters (default value) 2: Display numbers |
| secrecyMode | Confidential mode | 0: Not confidential (default value) 1: Confidential |
| modifyMode | Modify mode | 0: Cannot be modified 1: Can be modified (default value) |
| lessMode | forced selection mode | 0: Optional (default value) 1: Not optional 2: Allow repeated input 3: Allow repeated input and cannot be optional |
| options | Number of options | 1≤M≤10 (default value 4) |
| optionalN | number of options can be selected | 1≤N≤M (default value 1) |
| tipTitle | prompt title | String, no more than 16 bytes in length |

Command example:

{  
 "fun": "startChoices",  
 "baseId": 0,  
 "params": {  
 "optionsMode": "1",  
 "secrecyMode": "0",  
 "modifyMode": "1",  
 "lessMode": "0",  
 "options": "4",  
 "optionalN": "1"  
 },  
 "packetTag": "1"  
}

Receive:

fun: "startChoices" //Start the choice question   
baseId: "1"//The ID of the base station that received the instruction   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "startChoices" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Receive answer

Receive:

fun: "answerChoices" //Return the answer of the choice questions   
baseId: "1"//The ID of the base station that received the instruction   
infos: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| time | Answer time | Unit second |
| keySn | Keypad Sn |  |
| keyValue | key value | Keypad submitted value |

Command example:

{  
 "fun": "answerChoices",  
 "baseId": 1,  
 "infos": {  
 "time": "2.42",  
 "keySn": "1479824643",  
 "keyValue": "A"  
 },  
 "packetTag": "1"  
}

#### Stop answering

Send:

fun: "stopChoices" //Stop the choice questions   
params:{} //Default empty   
command example:

{  
 "fun" : "stopChoices" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopChoices" //Stop the choice question   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "stopChoices" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| Base station + keypad | Support/not support | Remark |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

### True or False Question

#### Start answering

Send:

fun: "startTrueFalse" //Start the judgment question   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| optionsMode | Option Category | 1:True/False  2:Yes/No  3:√/× |
| secrecyMode | Confidential mode | 0: Not confidential (default value)  1: Confidential |
| modifyMode | Modify mode | 0: Cannot be modified  1: Can be modified (default value) |
| tipTitle | prompt title | String, no more than 16 bytes in length |

Command example:

{  
 "fun" : "startTrueFalse" ,  
 "baseId" : 0 ,  
 "params" : {  
 "optionsMode" : "1" ,  
 "secrecyMode" : "0" ,  
 "modifyMode" : "1"  
 },  
 "packetTag" : "1"   
}

Receive:

fun: "startTrueFalse" //Start the judgment question   
baseId: "1"//The ID of the base station that received the instruction   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "startTrueFalse" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Receive answer

Receive:

fun: "answerTrueFalse" //The answer to the true or false question returns   
baseId: "1"//The ID of the base station that received the instruction   
infos: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| time | Answer time | Unit second |
| keySn | Keypad Sn |  |
| keyValue | key value | Values submitted by keypad (1 right, 2 wrong) |

Command example:

{  
 "fun" : "answerTrueFalse" ,  
 "baseId" : 1 ,  
 "Info" : {  
 "time" : "2.42" ,  
 "keySn" : "1479824643" ,  
 "keyValue" : "1"  
 },  
 "packetTag" : "1"   
}

#### Stop answering

Send:

fun: "stopTrueFalse" //Stop judgment question   
params:{} //Default empty

Command example:

{  
 "fun" : "stopTrueFalse" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopTrueFalse" //Stop judgment question   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "stopTrueFalse" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| Base station + keypad | Support/not support | Remark |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

### Quick answers to multiple questions

#### Start answering

Send:

fun: "startMulQuestions" //Start multiple questions   
params: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| questionType | question type | 0: Single choice question (default value)  1: True or False question |
| secrecyMode | Confidential mode | 0: Not confidential (default value)  1: Confidential |
| modifyMode | Modify mode | 0: Cannot be modified  1: Can be modified (default value) |
| lessMode | reserved | 0 |
| questionNumber | Number of questions | 1~16 |
| options | Number of options | 1~10 (valid for single choice questions, default value 4) |
| tipTitle | prompt title | String, no more than 16 bytes in length |

Command example:

{  
 "fun": "startMulQuestions",  
 "baseId": 0,  
 "params": {  
 "questionType": "0",  
 "secrecyMode": "0",  
 "modifyMode": "1",  
 "lessMode": "0",  
 "questionNumber": "10",  
 "options": "4"  
 },  
 "packetTag": "1"  
}

Receive:

fun: "startMulQuestions" //Start multiple questions   
baseId: "1"//The ID of the base station that received the instruction   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "startMulQuestions" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Receive answer

Receive:

fun: "answerMulQuestions" //Answers to multiple questions returns   
baseId: "1"//The ID of the base station that received the instruction   
infos: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| time | Answer time | Unit second |
| keySn | Keypad Sn |  |
| keyValue | key value | Keypad submitted value |

Command example:

{  
 "fun": "answerMulQuestions",  
 "baseId": 1,  
 "infos": {  
 "time": "2.42",  
 "keySn": "1479824643",  
 "keyValue": "AACDBDADBD"  
 },  
 "packetTag": "1"  
}

#### Stop answering

Send:

fun: "stopMulQuestions" //Stop multiple questions   
params:{} //Default empty

Command example:

{  
 "fun" : "stopMulQuestions" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun:"stopMulQuestions" //Stop multiple questions   
baseId: "1" //Received base station ID   
infos:{"state":"OK"} //Return status, success is OK

Command example:

{  
 "fun" : "stopMulQuestions" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| Base station + keypad | Support/not support | Remark |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |



















### Rush answer questions

#### Start answering

Send:

fun: "startRushAnswer" //Start the rush answer question   
params: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| rushAnswerMode | Answer mode | 0: Normal answering (default value) |
| modifyMode | Modify mode | 0: Cannot be modified  1: Can be modified (default value) |

Command example:

{  
 "fun": "startRushAnswer",  
 "baseId": 0,  
 "params": {  
 "rushAnswerMode": "0",  
 "modifyMode": "1"  
 },  
 "packetTag": "1"  
}

Receive:

fun: "startRushAnswer" //Start the rush answer question   
baseId: "1"//The ID of the base station that received the instruction   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "startRushAnswer" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Receive answer

Receive:

fun: "answerRushAnswer" //The answer to the rush answer question returns   
baseId: "1"//The ID of the base station that received the instruction   
infos: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| time | Answer time | Unit second |
| keySn | Keypad Sn |  |
| keyValue | key value | Keypad submitted value |

Command example:

{  
 "fun": "answerRushAnswer",  
 "baseId": 1,  
 "infos": {  
 "time": "1.42",  
 "keySn": "1479824643",  
 "keyValue": "1"  
 },  
 "packetTag": "1"  
}

#### Stop answering

Send:

fun: "stopRushAnswer" //Stop rush answer questions   
params:{} //Default is empty

Command example:

{  
 "fun" : "stopRushAnswer" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopRushAnswer" //Stop rush answer questions   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "stopRushAnswer" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| Base station + keypad | Support/not support | Remark |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

### Number questions

#### Start answering

Send:

fun: "startNumber" //Start number question   
params: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| secrecyMode | Confidential mode | 0: Not confidential (default value)  1: Confidential |
| modifyMode | Modify mode | 0: Cannot be modified  1: Can be modified (default value) |
| aswOption | Button mode | 0: No rules (S6 has a maximum of 16 digits, T2 has a maximum of 14 digits)  1: Reserved  2: Reserved  3: With upper and lower decimal places (not supported by T2) |
| digits | Decimal places | 0: No decimal point  1: One decimal point  2: Two decimal points |
| numMin | lower limit | aswOption=3  digits=0 Range: 0~65535  digits=1 Range: 0~6553  digits=2 Range: 0~655 |
| numMax | upper limit | aswOption=3  digits=0 Range: 0~65535  digits=1 Range: 0~6553  digits=2 Range: 0~655 |
| tipTitle | prompt title | String, no more than 16 bytes in length |

Command example:

{  
 "fun" : "startNumber" ,  
 "baseId" : 0 ,  
 "params" : {  
 "secrecyMode" : "0" ,  
 "modifyMode" : "1" ,  
 "aswOption" : "0" ,  
 "digits" : "0" ,  
 "numMin" : "0" ,  
 "numMax" : "0"  
 },  
 "packetTag" : "1"   
}

Receive:

fun: "startNumber" //Start the number question   
baseId: "1"//The ID of the base station that received the instruction   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "startNumber" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Receive answer

Receive:

fun: "answerNumber" //The answer to the numerical question returns   
baseId: "1"//The ID of the base station that received the instruction   
infos: {}

| Json field | Field meaning | Assignment and meaning |
| --- | --- | --- |
| time | Answer time | Unit second |
| keySn | Keypad Sn |  |
| keyValue | key value | Keypad submitted value |

Command example:

{  
 "fun": "answerNumber",  
 "baseId": 1,  
 "infos": {  
 "time": "2.42",  
 "keySn": "1479824643",  
 "keyValue": "12"  
 },  
 "packetTag": "1"  
}

#### Stop answering

Send:

fun: "stopNumber" //Stop number question   
params:{} //Default empty

Command example:

{  
 "fun" : "stopNumber" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopNumber" //Stop number question   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "stopNumber" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| Base station + keypad | Support/not support | Remark |
| --- | --- | --- |
| B100-2.4G+S6 | support | Key mode (0-no rules up to 16 bits) |
| B200-2.4G+S6 | support | Key mode (0-no rules up to 16 bits) |
| B200-5.8G+T2 | support | Key mode (0-no rules up to 14 digits)  key mode (3-with upper and lower decimal places) (not supported) |

### Test questions

#### Start test questions

##### Online test question interface

Send:

fun: "startKeyPadMultipleQuestionsTest" //Start online test question type   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| examNo | Exam number | 1-9999  Note: 1: A new question starts |
| displayMode | Question number display mode | 0: Do not display the beginning  1: Display the beginning as "Q"  2: Display the beginning as "Question" |
| questions | Question details | Array [max 200 cells] |

Question details array parameter description

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| questionLevel1Symbol | Level 1 question number | 1-200 |
| questionLevel2Symbol | Level 2 question number | 1-99 [valid when displayMode=0] |
| questionLevel3Symbol | Level 3 question number | 1-99 [valid when displayMode=0] |
| questionType | question type | 1: Single choice  2: Multiple choices (sorting)  3: Number  4: Judgment |
| option | Number of options | Valid when questionType = 1.  Valid when questionType = 2.  This item defaults to 4;   Range: 1-10. |

Instruction sample: Level 1 question type + Do not display the beginning  
Sample Purpose: Level 1 question type, do not display the beginning, total number of questions is 4.

{  
 "fun": "startKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction sample: Level 2 question type + Do not display the beginning.  
Sample Purpose: Level 2 question type, do not display the beginning, total number of questions is 4

{  
 "fun": "startKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "2",  
 "questionLevel2Symbol": "1",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "2",  
 "questionLevel2Symbol": "1",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction example: Level 3 question type + Do not display the beginning.  
Sample Purpose: Level 3 question type, do not display the beginning, the total number of questions is 4

{  
 "fun": "startKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionLevel3Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionLevel3Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionLevel3Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionLevel3Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction Example: level 1 question type + Display with “Q” at the beginning

Example Purpose: level 1 question type, display with “Q” at the beginning, the total number of questions is 4.

{  
 "fun": "startKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 1,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction example: Level 1 question type + Display the beginning as "question" Example Purpose: Level 1 question type displays the beginning as "question", total number of questions is 4 questions

{  
 "fun": "startKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 2,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Receive:

fun: "keyPadMultipleQuestionsTest" //   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "keyPadMultipleQuestionsTest" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "59"   
}

##### Multi-subject question interface

Send:

fun: "startKeyPadMultipleQuestionsCourse" //Start multi-subject question type   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| examNo | Exam number | 1-9999  Note: 1: A new question starts |
| displayMode | Question number display mode | 0: Do not display the beginning  1: Display the beginning as “Q”  2: Display the beginning as “题” |
| questions | Question details | Array [max 200 cells] |

Question details array parameter description

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| questionLevel1Symbol | Level 1 question number | 1-200 |
| questionLevel2Symbol | Level 2 question number | 1-99 [valid when displayMode=0] |
| course | suject | 1: Chinese  2: Mathematics  3: English  4: Politics  5: History  6: Geography  7: Biology  8: Physics  9: Chemistry |
| questionType | question type | 1: Single choice  2: Multiple choices (sorting)  3: Number  4: Judgment |
| option | Number of options | Valid when questionType = 1.  Valid when questionType = 2.  This item defaults to 4;  range: 1-10. |

Instruction sample: With subjects + Level 2 question types + Do not display the beginning   
Sample Purpose: Chinese 2 questions + Mathematics 2 questions + English 2 questions

{  
 "fun": "startKeyPadMultipleQuestionsCourse",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "course": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "2",  
 "course": "1",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "course": "2",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "2",  
 "course": "2",  
 "questionType": "4"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "course": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "2",  
 "course": "3",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction sample: with subjects + level 1 question type + do not show the beginning  
Sample Purpose: Chinese 4 questions

{  
 "fun": "startKeyPadMultipleQuestionsCourse",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "course": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "course": "1",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "course": "1",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "course": "1",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction sample: with subjects + level 1 question type + display the beginning as "Q"  
Sample Purpose: Chinese 4 questions

{  
 "fun": "startKeyPadMultipleQuestionsCourse",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 1,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "course": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "course": "1",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "course": "1",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "course": "1",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction sample: With subjects + Level 1 question type + Display the beginning as "题"   
Sample Purpose: Chinese 4 questions

{  
 "fun": "startKeyPadMultipleQuestionsCourse",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 2,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "course": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "course": "1",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "course": "1",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "course": "1",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Receive:

fun: "keyPadMultipleQuestionsCourse" //   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "keyPadMultipleQuestionsCourse" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "59"   
}

##### Interface for Subjective Question Score Input

Send:

fun: "startKeyPadMultipleQuestionsSubjective" //Start subjective question score input   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| examNo | Exam number | 1-9999  Note: 1: A new question starts |
| questions | Question details | Array [max 200 cells] |

Question details array parameter description

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| questionLevel1Symbol | Level 1 question number | 1-200 |
| toplimit | numerical upper limit | 1-600 |
| decimal | Decimal places | 0: Invalid decimal  1: One decimal place  2: Two decimal places  The default value of this item is 0. |

{  
 "fun": "startKeyPadMultipleQuestionsSubjective",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "toplimit": "100",  
 "decimal": "0"  
 }, {  
 "questionLevel1Symbol": "2",  
 "toplimit": "100",  
 "decimal": "1"  
 }, {  
 "questionLevel1Symbol": "3",  
 "toplimit": "100",  
 "decimal": "2"  
 }, {  
 "questionLevel1Symbol": "4",  
 "toplimit": "100"  
 }]  
 },  
 "packetTag": "59"  
}

Receive:

fun: "KeyPadMultipleQuestionsSubjective" //   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "KeyPadMultipleQuestionsSubjective" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "59"   
}

#### Receive answer

Receive:

fun: "answerKeyPadMultipleQuestions" //Return test questions

baseId: "1" //The ID of the base station that received the instruction

infos: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| Time | Answer time | Unit: seconds, valid within 20 minutes |
| keySn | Keypad Sn |  |
| Seq | Question serial number |  |
| keyValue | key value | Keypad submitted value |

Command example:

{  
 "fun": "answerKeyPadMultipleQuestions",  
 "baseId": 1,  
 "infos": {  
 "time": "5.42",  
 "keySn": "1479824643",  
 "seq": "1",  
 "keyValue": "12"  
 },  
  
 "packetTag": "1"  
}

#### Stop answering

Send:

fun: "stopKeyPadMultipleQuestions" //Stop test question type

params:{} //Default empty

Command example:

{  
 "fun" : "stopKeyPadMultipleQuestions" ,  
  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopKeyPadMultipleQuestions" //Stop keypad test questions

baseId: "1" //Received base station ID

infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "stopKeyPadMultipleQuestions" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Pause to answer

Send:

fun: "pauseKeyPadMultipleQuestions" //Pause for test questions

baseId: "1" //Received base station ID

params:{} //Default empty

Command example:

{  
 "fun" : "pauseKeyPadMultipleQuestions" ,  
  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Receive:

fun: "pauseKeyPadMultipleQuestions" //Pause keypad test questions

baseId: "1" //Received base station ID

infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "pauseKeyPadMultipleQuestions" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Continue test questions

##### Online test question interface

Send:

fun: "continueKeyPadMultipleQuestionsTest" //Continue online test question type   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| examNo | Exam number | 1-9999 |
| displayMode | Question number display mode | 0: Do not display the beginning  1: Display the beginning as “Q”  2: Display the beginning as “题” |
| questions | Question details | Array [max 200 cells] |

Question details array parameter description

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| questionLevel1Symbol | Level 1 question number | 1-200 |
| questionLevel2Symbol | Level 2 question number | 1-99 [valid when displayMode=0] |
| questionLevel3Symbol | Level 3 question number | 1-99 [valid when displayMode=0] |
| questionType | question type | 1: Single choice  2: Multiple choices (sorting)  3: Number  4: Judgment |
| option | Number of options | Valid when questionType = 1.  Valid when questionType = 2.  This item defaults to 4;  range: 1-10. |

Instruction example: Level 1 question type + do not display the beginning

{  
 "fun": "continueKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction example: Level 2 question type + do not show the beginning

{  
 "fun": "continueKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction example: Level 3 question type + do not display the beginning

{  
 "fun": "continueKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionLevel3Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionLevel3Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionLevel3Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "questionLevel3Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction example: Level 1 question type + display the beginning as "Q"

{  
 "fun": "continueKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 1,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Command example: Level 1 question type + display the beginning as "题"

{  
 "fun": "continueKeyPadMultipleQuestionsTest",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 2,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Receive:

fun: "keyPadMultipleQuestionsTest" //   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "keyPadMultipleQuestionsTest" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "59"   
}

##### Multi-subject question interface

Send:

fun: "continueKeyPadMultipleQuestionsCourse" //Continue multi-subject question type   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| examNo | Exam number | 1-9999 |
| displayMode | Question number display mode | 0: Do not display the beginning  1: Display the beginning as “Q”  2: Display the beginning as “题” |
| questions | Question details | Array [max 200 cells] |

Question details array parameter description

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| questionLevel1Symbol | Level 1 question number | 1-200 |
| questionLevel2Symbol | Level 2 question number | 1-99 [valid when displayMode=0] |
| course | suject | 1: 语文 2: 数学 3: 英语 4: 政治 5: 历史 6: 地理 7: 生物 8: 物理 9: 化学 |
| questionType | question type | 1: Single choice  2: Multiple choices (sorting)  3: Number  4: Judgment |
| option | Number of options | Valid when questionType = 1.  Valid when questionType = 2.  This item defaults to 4;  range: 1-10. |

Instruction sample: With subjects + Level 2 question types + Do not display the beginning   
Sample Purpose: Chinese 2 questions + Mathematics 2 questions + English 2 questions

{  
 "fun": "continueKeyPadMultipleQuestionsCourse",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "course": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "2",  
 "course": "1",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "course": "2",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "2",  
 "course": "2",  
 "questionType": "4"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "1",  
 "course": "3",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "1",  
 "questionLevel2Symbol": "2",  
 "course": "3",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction sample: with subjects + level 1 question type + do not show the beginning  
Sample Purpose: Chinese 4 questions

{  
 "fun": "continueKeyPadMultipleQuestionsCourse",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 0,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "course": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "course": "1",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "course": "1",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "course": "1",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction sample: with subjects + level 1 question type + display the beginning as "Q"   
Sample Purpose: Chinese 4 questions

{  
 "fun": "continueKeyPadMultipleQuestionsCourse",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 1,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "course": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "course": "1",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "course": "1",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "course": "1",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Instruction sample:With subjects + Level 1 question type + Display the beginning as "题"   
Sample Purpose: Chinese 4 questions

{  
 "fun": "continueKeyPadMultipleQuestionsCourse",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "displayMode": 2,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "course": "1",  
 "questionType": "1"  
 }, {  
 "questionLevel1Symbol": "2",  
 "course": "1",  
 "questionType": "2",  
 "option": "5"  
 }, {  
 "questionLevel1Symbol": "3",  
 "course": "1",  
 "questionType": "3"  
 }, {  
 "questionLevel1Symbol": "4",  
 "course": "1",  
 "questionType": "4"  
 }]  
 },  
 "packetTag": "59"  
}

Receive:

fun: "keyPadMultipleQuestionsCourse" //   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "keyPadMultipleQuestionsCourse" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "59"   
}

##### Interface for subjective question score input

Send:

fun: "continueKeyPadMultipleQuestionsSubjective" //Continue to input subjective question scores   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| examNo | Exam number | 1-9999 |
| questions | Question details | Array [max 200 cells] |

Question details array parameter description

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| questionLevel1Symbol | Level 1 question number | 1-200 |
| toplimit | Numerical upper limit | 1-600 |
| decimal | Decimal places | 0: Invalid decimal  1: One decimal place  2: Two decimal places  The default value of this item is 0. |

{  
 "fun": "continueKeyPadMultipleQuestionsSubjective",  
 "baseId": 0,  
 "params": {  
 "examNo": 1,  
 "questions": [{  
 "questionLevel1Symbol": "1",  
 "toplimit": "100",  
 "decimal": "0"  
 }, {  
 "questionLevel1Symbol": "2",  
 "toplimit": "100",  
 "decimal": "1"  
 }, {  
 "questionLevel1Symbol": "3",  
 "toplimit": "100",  
 "decimal": "2"  
 }, {  
 "questionLevel1Symbol": "4",  
 "toplimit": "100"  
 }]  
 },  
 "packetTag": "59"  
}

Receive:

fun: "keyPadMultipleQuestionsSubjective" //   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "keyPadMultipleQuestionsSubjective" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "59"   
}

#### Specify keypad to transfer answer content

Note: This interface can only operate one keypad at a time.

Send:

fun: "copyKeyPadMultipleQuestions" //Transfer keypad answer content   
params:{} //

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| baseId | ID of the base station to which the keypad is connected | 1-32 |
| keySn | Target keypad SN |  |
| ID | Question number | 1-200 |
| questionType | question type | 1: Single choice  2: Multiple choices (sorting)  3: Number  4: Judgment |
| value | Question answer | String 16 bytes |

Command example:

{  
 "fun": "copyKeyPadMultipleQuestions",  
 "baseId": 1,  
 "params": {  
 "keySn": "0007831203",  
 "questions": [{  
 "id": "1",  
 "questionType": "1",  
 "value": "A"  
 }, {  
 "id": "2",  
 "questionType": "2",  
 "value": "AB"  
 }]  
 },  
 "packetTag": "19"  
}

Receive:

fun: "copyKeyPadMultipleQuestions" //   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "copyKeyPadMultipleQuestions" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Broadcast the correct answer to the keypad

Send:

fun: "okKeyPadMultipleQuestions" //Broadcast the correct answer to the keypad   
baseId: "1"//Received base station ID   
params:{} //Default empty

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| baseId | Base station ID | 1-32 |
| ID | Question number | 1-200 |
| questionType | Question type | 1: Single choice  2: Multiple choices (sorting)  3: Number  4: Judgment |
| value | Question answer | String 16 bytes |

Command example:

{  
 "fun": "okKeyPadMultipleQuestions",  
 "baseId": 0,  
 "params": [{  
 "id": "1",  
 "questionType": "1",  
 "value": "A"  
 }, {  
 "id": "2",  
 "questionType": "2",  
 "value": "AB"  
 }],  
 "packetTag": "20"  
}

Receive:

fun: "okKeyPadMultipleQuestions" //   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "okKeyPadMultipleQuestions" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

### Fill in the blanks

#### Start answering

Send:

fun: "startTextAnswer" //Start filling in the blanks   
params: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| optionsMode | fill-in-the-blank type | 1: Any input (default value) |
| modifyMode | reserved | 1: (default value) |
| secrecyMode | reserved | 0: (default value) |

Command example:

{  
 "fun": "startTextAnswer",  
 "baseId": 0,  
 "params": {  
 "optionsMode": "1",  
 "modifyMode": "1",  
 "secrecyMode": "0"  
 },  
 "packetTag": "1"  
}

Receive:

fun: "startTextAnswer " //Start filling in the blanks   
baseId: "1" //The ID of the base station that received the instruction   
infos:{"state":"OK"} //Return status, success is OK

Command example:

{  
 "fun" : "startTextAnswer" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Receive answer

Receive:

fun: "answerTextAnswer" // Fill-in-the-blank question response   
baseId: "1" //The ID of the base station that received the instruction   
infos: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |
| keyValue | key value | string |

Command example:

{  
 "fun" : "answerTextAnswer" ,  
 "baseId" : 1 ,  
 "Info" : {  
 "keySn" : "1479824643" ,  
 "keyValue" : "abcdef"  
 },  
 "packetTag" : "1"   
}

#### Stop answering

Send:

fun: "stopTextAnswer" //Stop filling in the blanks   
params:{} //Default is empty

Command example:

{  
 "fun" : "stopTextAnswer" ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopTextAnswer" //Stop filling in the blanks   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "stopTextAnswer" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

### Free question type

#### Start answering

Send:

fun: "startFreeQuestion" //Start filling in the blanks   
params: {}

Command example:

{  
 "fun" : "startFreeQuestion" ,  
 "baseId" : 0 ,  
 "params" : {},  
 "packetTag" : "1"   
}

Receive:

fun: "startFreeQuestion " //Start free question type   
baseId: "1" //The ID of the base station that received the instruction   
infos:{"state":"OK"} //Return status, success is OK

Command example:

{  
 "fun" : "startFreeQuestion" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

#### Receive answer

Receiving:   
It is divided into three types: choice questions, true/false questions and numerical questions.

Choice questions:   
fun: "answerChoices" //Free question type answer return   
baseId: "1"//The ID of the base station that received the instruction   
infos: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |
| keyValue | key value | ABCD/1234 |

Command example:

{  
 "fun": "answerChoices",  
 "baseId": 1,  
 "infos": {  
 "keySn": "1479824643",  
 "keyValue": "ABC"  
 },  
 "packetTag": "0"  
}

True or false question:   
fun: "answerTrueFalse" //Free question type answer returns   
baseId: "1"//The ID of the base station that received the instruction   
infos: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |
| keyValue | key value | 1~2 |

Command example:

{  
 "fun" : "answerTrueFalse" ,  
 "baseId" : 1 ,  
 "Info" : {  
 "keySn" : "1479824643" ,  
 "keyValue" : "1"  
 },  
 "packetTag" : "0"   
}

Number question:   
fun: "answerNumber" //Free question type answer returns   
baseId: "1"//The ID of the base station that received the instruction   
infos: {}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |
| keyValue | key value | Keypad submitted value |

Command example:

{  
 "fun" : "answerNumber" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "keySn" : "1479824643" ,  
 "keyValue" : "99.99"  
 },  
 "packetTag" : "0"   
}

#### Stop answering

Send:   
fun: "stopFreeQuestion" //Stop free question type   
params:{} //Default is empty

Command example:

{  
 "fun" : "stopFreeQuestion" ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopFreeQuestion" //Stop free question   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

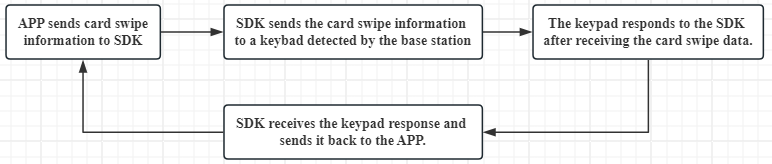
{  
 "fun" : "stopFreeQuestion" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4G+S6 | support | Only customized keypad support |

## Swipe card

Card swiping process:



After swiping the card, the keypad and base station can be paired automatically. When it is necessary to swipe the card to bind, the application software first sends the name of the keypad to the base station. When the keypad is close to the base station sensing area, it will directly bind the current data and return the status to the application. After the binding is successful, the application will send the binding information of the next keypad, and bind multiple keypads in sequence.

### Write card data

Send:

fun: "writeNFCInfo" //Write NFC binding data   
baseId: 1//1~32 (ID of the base station connected to the keypad)   
params:{}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| userName | Name | Up to 24 Chinese characters (48 English characters), displayed in the upper left corner of the keypad |
| matchCode | pairing code | Four-byte hexadecimal display |

Command example:

{  
 "fun": "writeNFCInfo",  
 "baseId": 1,  
 "params": {  
 "userName": "John",  
 "matchCode": "22120007"  
 },  
 "packetTag": "1"  
}

Receive:   
fun: "writeNFCInfo" //   
baseId: "1"//Received base station ID   
infos:{"value":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "writeNFCInfo" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4GS+S6 | support | Effective with NFC keypad |

### Swiping feedback

Receive:   
fun: "NFCInfo" //Swiping card feedback   
baseId: "1"//ID of the base station connected to the keypad   
infos:{}

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |
| state | state | 1: Reserved  2: The base station reports the keypad SN of the NFC card swiping  3: Reserved  4: The message of card swiping to write the specified pairing code  5: The base station reports the keypad SN of the M1 card swiping |
| number | quantity | reserved |

Command example:

{  
 "fun": "NFCInfo",  
 "baseId": 1,  
 "infos": {  
 "keySn": "1479824643",  
 "state": "4",  
 "number": "0"  
 },  
 "packetTag": "1"  
}

Support Device Description

| Base station + keypad | Support/not support | Remark |
| --- | --- | --- |
| B200-2.4GS+S6 | support |  |
| B200-2.4GS+M1 | support |  |

### NFC data clear command

Send:   
fun: "writeNFCEmpty" //Clear NFC data   
baseId: 1//1~32 (ID of the base station connected to the keypad)   
params:{}//

Command example:

{  
 "fun" : "writeNFCEmpty" ,  
 "baseId" : 1 ,  
 "params" : {},  
 "packetTag" : "1"   
}

Receive:   
fun: "writeNFCEmpty" //   
baseId: "1"//Received base station ID   
infos:{"value":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "writeNFCEmpty" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4GS+S6 | support | Only with NFC keypad |

## Hardware parameters

The data returned by hardware reading and writing operations are the same. For specific values, please refer to the value of the write command.

### Base station

#### Base station number

Read

Send:   
fun: "readBaseStationID" //Read base station number   
baseId: 1//1~32   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationID" ,  
 "baseId" : 2 ,  
 "packetTag" : "1"   
}

Return   
Receive:   
fun: "baseStationID"   
baseId: "2"//Base station ID   
infos:{"value":"2"}//Return base station ID

Command example:

{  
 "fun" : "baseStationID" ,  
 "baseId" : 2 ,  
 "infos" : {  
 "pathAdd" : "12345" ,  
 "value" : "2"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Base station frequency point

Read

Send:   
fun: "readBaseStationChannel" //Read base station frequency   
baseId: 0//0~32 (the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationChannel" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Write

Send:   
fun: "writeBaseStationChannel" //Write base station frequency point   
baseId: 1//1~32 (base station must be specified)   
params: {"value":"1"}//1~12 (frequency point)

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| value | Base station frequency point | Valid for single frequency point base station |
| values | Base station frequency point | Valid for multi-frequency base station |

Command example:

{  
 "fun": "writeBaseStationChannel",  
 "baseId": 1,  
 "params": {  
 "value": "3",  
 "values": ["3", "6", "9", "12"]  
 },  
 "packetTag": "1"  
}

Return

Receive:   
fun: "baseStationChannel"   
baseId: 1//Base station ID   
infos:{"value":"3"}//Return the base station frequency value

Command example:

{  
 "fun": "baseStationChannel",  
 "baseId": 1,  
 "infos": {  
 "value": "3",  
 "values": ["3", "6", "9", "12"]  
 },  
 "packetTag": "1"  
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4GS+S6 | Support single frequency point | Frequency range 1~80 |
| B200-2.4GS+S6 | Support single frequency point | Frequency range 1~80 |
| B200-2.4GM+S6 | Support multiple frequency points | Frequency range 1~80 |

#### Keypad login menu display

Read

Send:   
fun: "readKeyboardLoginMenu" //Read the keypad login menu display   
baseId: 0//0~32 (the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readKeyboardLoginMenu" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Write

Send:   
fun: "writeKeyboardLoginMenu" //Write keypad login menu display   
baseId: 1 //1~32 (base station must be specified)   
params: {"value":"1"}//0~1 (0 does not allow login; 1 Login allowed)

Command example:

{  
 "fun" : "writeKeyboardLoginMenu" ,  
 "baseId" : 1 ,  
 "params" : {  
 "value" : "1"  
 },  
 "packetTag" : "1"   
}

Return

Receive:

fun: "baseStationKeyboardLoginMenu"   
baseId: 1//Base station ID   
infos:{"value":"1"}//Return to the keypad login menu display value

Command example:

{  
 "fun" : "baseStationKeyboardLoginMenu" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "1"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Base station model (read only)

Read

Send:

fun: "readBaseStationModel" //Read base station model   
baseId: 0//0~32 (the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationModel" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

return

Receive:

fun: "baseStationModel"   
baseId: 1//Base station ID   
infos:{"value":"218"}//Return the base station model

Command example:

{  
 "fun" : "baseStationModel" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "218"  
 },  
 "packetTag" : "1"   
}

Base station model code query table

| **Base station model code** | **Base station model** | **Remark** |
| --- | --- | --- |
| 217 | B100-2.4G |  |
| 218 | B200-2.4G |  |
|  |  |  |

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Base station firmware version (read-only)

read

Send:

fun: "readBaseStationSVersion" //Read the base station firmware version   
baseId: 0 //0~32 (the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationSVersion" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Return

Receive:

fun: "baseStationSVersion"   
baseId: 1//Base station ID   
infos:{"value":"V2.0.2"}//Return to base station firmware version

Command example:

{  
 "fun" : "baseStationSVersion" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "V2.0.2"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Base station serial number (read only)

Read

Send:

fun: "readBaseStationSN" //Read the base station serial number   
baseId: 0 //0~32 (the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationSN" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Return

Receive:

fun: "baseStationSN"   
baseId: 1//Base station ID   
infos:{"value":"FFFFFFFFFF"}//Return the base station serial number

Command example:

{  
 "fun" : "baseStationSN" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "FFFFFFFFFF"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Base station name

Read

Send:

fun: "readBaseStationTitle" //Read the base station name   
baseId: 0 //0~32 (the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationTitle" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Write

Send:

fun: "writeBaseStationTitle" //Write the base station name   
baseId: 1//1~32 (the base station must be specified)   
params: {"value":" A2021"}//The base station name, up to 12 bytes

Command example:

{  
 "fun" : "writeBaseStationTitle" ,  
 "baseId" : 1 ,  
 "params" : {  
 "value" : "A2021"  
 },  
 "packetTag" : "1"   
}

Return

Receive:

fun: "baseStationTitle"   
baseId: 1//Base station ID   
infos:{"value":"A2021"}//Return the base station name

Command example:

{  
 "fun" : "baseStationTitle" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "A2021"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Base station match mode

Read   
Send:   
fun: "readBaseStationPairMode" //Read base station match (pairing) mode   
baseId: 0// 0~32 (the command does not send baseID, the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationPairMode" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Note: Please use with caution when writing.

Write   
Send:   
fun: "writeBaseStationPairMode" //Write base station match (pairing) mode   
baseId: 1//1~32 (base station must be specified)   
params: {"value":"1"}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| value | model | 1: Pairing SN number mode  2: Whitelist mode |

Command example:

{  
 "fun" : "writeBaseStationPairMode" ,  
 "baseId" : 1 ,  
 "params" : {  
 "value" : "1"  
 },  
 "packetTag" : "1"   
}

Return   
Receive:   
fun: "baseStationPairMode"   
baseId: 1//Base station ID   
infos:{"value":"1"}//Return the base station pairing mode value

Command example:

{  
 "fun" : "baseStationPairMode" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "1"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4G+S6 | support |  |

#### Base station match code

Read

Send:

fun: "readBaseStationMatchCode" //Read the base station match&pairing code   
baseId: 0 //0~32 (the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationMatchCode" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

return

Receive:

fun: "baseStationMatchCode"   
baseId: 1//Base station ID   
infos:{"value":"21072333"}//Return base station match&pairing code

Command example:

{  
 "fun" : "baseStationMatchCode" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "21072333"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Base station frequency conflict detection

If a base station frequency conflict is detected, a prompt will be returned to the application, and the application can change the frequency according to the situation.   
Receive:   
fun: "baseStationChannelInterference" // Base station frequency conflict   
baseId: // Base station ID conflict occurs   
infos: { "value": "5"} // Frequency conflict point 5  
packetTag: "0" // Data packet tag

Command example:

{  
 "fun" : "baseStationChannelInterference" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "5"  
 },  
 "packetTag" : "0"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |



#### Quick match

##### Start quick match

Send:

fun: "startQuickMatch" //Start quick matching   
baseId:1 // 1~32 (required parameters, range (1~32))   
params: {}

Command example:

{  
 "fun" : "startQuickMatch" ,  
 "baseId" : 1 ,  
 "packetTag" : "1"   
}

Receive:

fun: "startQuickMatch" //Start quick match   
baseId: "1"//The base station ID that received the instruction   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "startQuickMatch" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

##### Stop quick match

Send:

fun: "stopQuickMatch" //Stop quick matching   
params:{} //Default empty

Command example:

{  
 "fun" : "stopQuickMatch" ,  
 "baseId" : 1 ,  
 "packetTag" : "1"   
}

Receive:

fun: "stopQuickMatch" //Stop quick match   
baseId: "1"//Received base station ID   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "stopQuickMatch" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Keypad login password

Read

Send:

fun: "readKeyboardLoginPW" //Read keypad login password   
baseId: 0// 0~32 (the command does not send baseID, the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readKeyboardLoginPW" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Write

Send:

fun: "writeKeyboardLoginPW" //Write keypad login password   
baseId: 1 //1~32 (base station must be specified)   
params: {"value":"0"}//0~9999 (0 does not need to be checked)

Command example:

{  
 "fun" : "writeKeyboardLoginPW" ,  
 "baseId" : 1 ,  
 "params" : {  
 "value" : "0"  
 },  
 "packetTag" : "1"   
}

Return

Receive:

fun: "baseStationKeyboardLoginPW"   
baseId: 1//Base station ID   
infos:{"value":"1"}//Return the keypad login password value

Command example:

{  
 "fun" : "baseStationKeyboardLoginPW" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "0"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Base station name hidden

Read

Send:

fun: "readBaseStationNameHidden" //Read base station name and hidden   
baseId: 0// 0~32 (the command does not send baseID, the default is 0, which means reading all base stations)   
params: {}//can be left blank

Command example:

{  
 "fun" : "readBaseStationNameHidden" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Write

Send:

fun: "writeBaseStationNameHidden" //Write base station name hidden   
baseId: 1 //1~32 (base station must be specified)   
params: {"value":"0"}//0~1 (0 hides, 1 does not hide)

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| value | model | 0: Hide  1: Not hide |

Command example:

{  
 "fun" : "writeBaseStationNameHidden" ,  
 "baseId" : 1 ,  
 "params" : {  
 "value" : "0"  
 },  
 "packetTag" : "1"   
}

Return

Receive:

fun: "baseStationNameHidden"   
baseId: 1//Base station ID   
infos:{"value":"1"}//Return the hidden value of the base station name (0 is hidden, 1 is not hidden)

Command example:

{  
 "fun" : "baseStationNameHidden" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "0"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |



#### Base station keypad whitelist management

Read

Send:

fun: "readWhiteList" //Read the whitelist list   
baseId: 0//

Command example:

{  
 "fun" : "readWhiteList" ,  
 "baseId" : 0 ,  
 "params" : {},  
 "packetTag" : "1"   
}

Read return   
and receive:   
fun: "whiteList"   
baseId: 1//Base station ID   
infos:{}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |
|  |  |  |

Command example:

{  
 "fun": "whiteList",  
 "baseId": 1,  
 "infos": [{  
 "keySn": "2111120140",  
 "state": "0"  
 },  
 {  
 "keySn": "0005243136",  
 "state": "0"  
 },  
 {  
 "keySn": "2111120102",  
 "state": "0"  
 }  
 ],  
 "packetTag": "1"  
}

Write

Note: Please switch the base station pairing mode to whitelist mode before writing.

Send:

fun: "writeWhiteList" //Write whitelist list   
baseId: 1 //1~32 (base station must be specified)   
keySnList: []//Keypad SN array can contain up to 200 keypad SNs

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySnList | keypad list | B200-2.4GS+S6: The maximum number of keypads is 200.  EA3100-2.4G+S62: The maximum number of keypads is 200. |

Command example:

{  
 "fun": "writeWhiteList",  
 "baseId": 0,  
 "keySnList": [{  
 "keySn": "2111120140"  
 },  
 {  
 "keySn": "0005243136"  
 },  
 {  
 "keySn": "2111120102"  
 }  
 ],  
 "packetTag": "10"  
}

Return

Receive:

fun: "whiteList"   
baseId: 1//Base station ID   
infos:{}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |
| state | keypad status | OK: Success |

Command example:

{  
 "fun": "whiteList",  
 "baseId": 1,  
 "infos": [{  
 "keySn": "2111120140",  
 "state": "OK"  
 },  
 {  
 "keySn": "0005243136",  
 "state": "OK"  
 },  
 {  
 "keySn": "2111120102",  
 "state": "OK"  
 }  
 ],  
 "packetTag": "1"  
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4G+S6 | support |  |
| EA3100-2.4G+S62 | support | After writing the whitelist, EA3100 returns no STATE status. |















#### Base station adds keypad

Send:   
fun: "addWhiteList" //Add keypad   
params:{} //Keypad sn   
keySnList:[ "keySn":"2111120140"] //The list cannot be empty

Command example:

{  
 "fun": "addWhiteList",  
 "baseId": 1,  
 "keySnList": [{  
 "keySn": "2111120140"  
 }],  
 "params": {},  
 "packetTag": "1"  
}

Receive:   
fun: "whiteList"   
baseId: 1//Base station ID   
infos:{}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | KeypadSN |  |
| state | keypad status | OK: Success |

{  
 "fun" : "whiteList" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "keySn" : "1837454012" ,  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4G+S6 | support |  |
|  |  |  |

#### Base station deletes keypad

Send:   
fun: "delWhiteList" //Kick out the keypad, delete the keypad   
params:{} //Keypad sn   
keySnList:[ "keySn":"2111120140"] //The list cannot be empty

Command example:

{  
 "fun": "delWhiteList",  
 "baseId": 0,  
 "keySnList": [{  
 "keySn": "2111120140"  
 }],  
 "params": {},  
 "packetTag": "1"  
}

Receive:   
fun: "delWhiteList"   
baseId: 1//Base station ID   
infos:{}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | KeypadSN |  |
| state | keypad status | OK: Success |

{  
 "fun" : "delWhiteList" ,  
 "baseId" : 1 ,  
 "packetTag" : "1" ,  
 "Info" : {  
 "keySn" : "1837454012" ,  
 "state" : "OK"  
 }   
}

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4G+S6 | support |  |

#### Base station clears all keypads

Send:   
fun: "clearWhiteList" //Clear all keypads   
params:{} //None

Command example:

{  
 "fun" : "clearWhiteList" ,  
 "baseId" : 1 ,  
 "packetTag" : "1"   
}

Receive:   
fun: "clearWhiteList"   
baseId: 1//Base station ID   
infos:{}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| state | keypad status | OK: Success |

{  
 "fun" : "clearWhiteList" ,  
 "baseId" : 1 ,  
 "packetTag" : "1" ,  
 "infos" : {  
 "state" : "OK"  
 }   
}

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4G+S6 | support |  |
| EA3100-2.4G+S62 | support |  |

#### Keypad shutdown time

Read   
and send:   
fun: "readKeypadAutoPowerOffLevel" //Read keypad shutdown time level   
baseId: 0//0~32 (the command does not send baseID, the default is 0, which means reading all base stations)   
params: {"value":0}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| value | Keypad shutdown time level | 0: 30 seconds  1: 2 minutes  2: 10 minutes  3: 45 minutes  255: No automatic shutdown |

Command example:

{  
 "fun" : "readKeypadAutoPowerOffLevel" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Write   
and send:   
fun: "writeKeypadAutoPowerOffLevel" //Write keypad shutdown time level   
baseId: 1//1~32 (base station must be specified)   
params: {"value":"0"}//

Command example:

{  
 "fun": "writeKeypadAutoPowerOffLevel",  
 "baseId": 1,  
 "params": {  
 "value": "0"  
 },  
 "packetTag": "1"  
}

Return   
Receive:   
fun: "keypadAutoPowerOffLevel"   
baseId: 1//Base station ID   
infos:{"value":"0"}//Return keypad shutdown time level   
command example:

{  
 "fun" : "keypadAutoPowerOffLevel" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "value" : "0"  
 },  
 "packetTag" : "1"   
}

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| All series | support |  |







### Keypad

The keypad writing parameter does not need to specify the keypad SN. Then the writing operation will be performed on all online keypads, but no keypad status will be returned. If you need to specify the keypad information to send, you need to specify the keypad SN and it will return all keypad status.

Note: If a keySN in a specified keypad command is empty, the command will be executed without specifying a keypad.







#### Keypad feedback parameters

Read   
Send:   
fun: "readKeypadFeedbackMode" //Read keypad feedback mode parameter   
baseId: 0//0~32 (the command does not send baseID, the default is 0, which means reading all base stations)   
params: {}//Not required when reading

Command example:

{  
 "fun" : "readKeypadFeedbackMode" ,  
 "baseId" : 0 ,  
 "packetTag" : "1"   
}

Write

Send:   
fun: "writeKeypadFeedbackMode" //Write keypad feedback mode parameters   
baseId: 1//1~32 (base station must be specified)   
params: {"mode":"1", "value":"4"}//

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| mode | feedback mode | 0: Not enabled (default 0 is not enabled)  1: Universal key code |
| value | Feedback maximum effective button | 1~10 |

Command example:

{  
 "fun": "writeKeypadFeedbackMode",  
 "baseId": 0,  
 "params": {  
 "mode": "1",  
 "value": "4"  
 },  
 "packetTag": "1"  
}

Return

Receive:   
fun: "keypadFeedbackMode"   
baseId: 1//Base station ID   
infos:{ "mode":"1", "value":"4"}//Return the keypad feedback mode parameter value

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| mode | feedback mode | 0: Not enabled (default 0 is not enabled)  1: Universal key code, not displayed |
| value | Feedback maximum effective button | 1~10 |

Command example:

{  
 "fun" : "keypadFeedbackMode" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "mode" : "1" ,  
 "value" : "4"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Instant keypad feedback [active reporting]

Receive:

fun: "immediateFeedback"   
baseId: 1//Base station ID   
infos:{ "keySn":"2307269906", "keyValue":"A"}//Return keypad feedback mode parameter value

Command example:

{  
 "baseId": 1,  
 "fun": "immediateFeedback",  
 "infos": {  
 "keySn": "2307269906",  
 "keyValue": "A"  
 },  
 "packetTag": "0"  
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

#### Keypad online notification [report]

After the S6 keypad goes online, the keypad SN information is reported.   
Receive:   
fun: "keyboardOnlineOne"   
baseId: 1//Base station ID   
infos:{ "keySn": "1479824643"}//Return keypad feedback mode parameter value

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn |  |

Command example:

{  
 "fun" : "keyboardOnlineOne" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "keySn" : "1479824643"  
 },  
 "packetTag" : "0"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B200-2.4G+S6 | support |  |

#### Keypad heartbeat feedback [report]

Receive:   
fun: "KeyboardHeartbeat"   
baseId: 1//Base station ID   
infos:{ "hModel": "67","keySn":"1479824643","keyValue":"0","ver":"1.0.0" ,"volt":"50"}//Return keypad feedback mode parameter value

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| hModel | Keypad model | 75: T2-5.8G  67: S6-2.4G |
| keySn | Keypad Sn |  |
| keyVer | keypad version |  |
| Volt | Keypad battery | T2 percentage value, S6 is invalid |

Command example:

{  
 "fun": "KeyboardHeartbeat",  
 "baseId": 1,  
 "infos": {  
 "hModel": "67",  
 "keySn": "1479824643",  
 "keyValue": "0",  
 "ver": "1.0.0",  
 "volt": "50"  
 },  
 "packetTag": "0"  
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support | It will give feedback when there is heartbeat configured in factory. It will not give feedback when there is no heartbeat configured in factory. |
| B200-2.4G+S6 | support | It will give feedback when there is heartbeat configured in factory. It will not give feedback when there is no heartbeat configured in factory. |

#### Remote control of keypad shutdown

Write   
Send:   
fun: "remoteKeyShutdown" //Remote control of keypad shutdown   
params: {}//   
Command example:

{  
 "fun": "remoteKeyShutdown",  
 "baseId": 1,  
 "params": [{  
 "keySn": "1837454011"  
 }],  
 "packetTag": "1"  
}

Return   
Receive:   
fun: "remoteKeyShutdown"   
baseId: 1//base station ID   
infos:{"keySn":"1837454011","state":"OK"}// Remote control of keypad shutdown

Command example:

{  
 "fun": "remoteKeyShutdown",  
 "baseId": 1,  
 "infos": {  
 "keySn": "1837454011",  
 "state": "OK"  
 },  
 "packetTag": "1"  
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |
|  |  |  |

#### Keypad short message

Write   
fun: "writeKeyMessage" //Write keypad short message   
params: {} //Array object, the default value in the object is the empty string

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | KeypadSN | Empty string for all online keypads |
| txt | Short message content | The maximum string length is 40 bytes [20 Chinese characters] |

Command example:

All keypads

{  
 "fun" : "writeKeyMessage" ,  
 "baseId" : 0 ,  
 "params" : {  
 "txt" : "latest news"  
 },  
 "packetTag" : "1"   
}

Return   
fun: "writeKeyMessage"   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "writeKeyMessage" ,  
 "baseId" : 0 ,  
 "infos" : {  
 "keySn" : "0000000000" ,  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Specify keypad

{  
 "fun" : "writeKeyMessage" ,  
 "keySnList" : [ {  
 "keySn" : "0007899478"  
 } ] ,  
 "params" : {  
 "txt" : "latest news"  
 },  
 "packetTag" : "1"   
}

Return

fun: "writeKeyMessage"   
infos:{}//Array object, same as sent

Command example:

{  
 "baseId": 3,  
 "fun": "writeKeyMessage",  
 "infos": {  
 "keySn": "0007899478",  
 "state": "OK"  
 },  
 "packetTag": "1"  
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |

## Keypad display settings

Note: "baseId":0 in the return command can be ignored and not processed.











### Custom information

When the PIN code sign-in binding mode is enabled, the keypad can be bound.   
Send:   
fun: "writeKeypadCustomInfo" //Write the custom information of the keypad   
params: {} //Array object, if the field is not filled in, the last value will be retained.

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | Keypad Sn | If empty, it means all online keypads, other array values are invalid |
| scoreMode | Fraction mode | 1: Clear  2: Character display (default value)  3: Five-pointed star display  4: Heart display [valid for S6] |
| scoreValue | score value | scoreMode == 1, invalid here.  scoreMode == 2, 8 characters in length.  scoreMode == 3, 0~6, 2 scores represent 1 five-pointed star, up to 3.  scoreMode == 4, 0~6, 2 points represent 1 heart, up to 3 [S6 valid] |
| tipText | Prompt text | Up to 8 Chinese characters (16 English characters), displayed on the second line of the keypad |

Command sample:   
Sample purpose: Set the keypad integral position with SN "1479824643" to "99", and set the second line of text position to "Bound".

{  
 "fun" : "writeKeypadCustomInfo" ,  
 "baseId" : 1 ,  
 "params" : [ {  
 "keySn" : "1479824643" ,  
 "scoreMode" : "2" ,  
 "scoreValue" : "99" ,  
 "tipText" : "Bound"  
 } ] ,  
 "packetTag" : "1"   
}

Receive:

Specify keypad

fun: "writeKeypadCustomInfo" //Write the user information of the keypad   
baseId:0 //The ID of the base station that received the instruction   
infos:{}//Array object

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | KeypadSN |  |
| state | state | OK(success) |

Command example:

{  
 "fun": "writeKeypadCustomInfo",  
 "baseId": 0,  
 "infos": [{  
 "keySn": "1479824643",  
 "state": "OK"  
 }],  
 "packetTag": "1"  
}

All keypads

fun: "writeKeypadCustomInfo" //Write the user information of the keypad   
baseId:0 //The ID of the base station that received the instruction   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "writeKeypadCustomInfo" ,  
 "baseId" : 0 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |
|  |  |  |

### Username

When PIN code sign-in binding mode is enabled, the keypad can be bound.

Send:   
fun: "writeKeypadUserName" // Write the name of user into keypad   
params: {}//Array object, if the field is not filled in, the last value will be retained.

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | KeypadSN | If empty, it means all online keypads, other array values are invalid |
| userName | Name | Up to 24 Chinese characters (48 English characters), displayed in the upper left corner of the keypad |

Command example:

{  
 "fun": "writeKeypadUserName",  
 "baseId": 1,  
 "params": [{  
 "keySn": "1479824643",  
 "userName": "John"  
 }],  
 "packetTag": "1"  
}

Receive:

Specify the keypad   
fun: "writeKeypadUserName" // Write the name of user into keypad   
baseId:0 //The ID of the base station that received the command   
infos:{}//Array object

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | KeypadSN |  |
| state | state | OK(success) |

Command example:

{  
 "fun" : "writeKeypadUserName" ,  
 "baseId" : 0 ,  
 "infos" : {  
 "keySn" : "1479824643" ,  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

All keypads

fun: "writeKeypadUserName" //Write the user information into the keypad   
baseId:0 //The ID of the base station that received the instruction   
infos:{"state":"OK"}//Return status, success is OK

Command example:

{  
 "fun" : "writeKeypadUserName" ,  
 "baseId" : 0 ,  
 "infos" : {  
 "state" : "OK"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |
|  |  |  |

### Specify user name [C100B+S6 old model]

When PIN code sign-in binding mode is enabled, the keypad can be bound.   
Send   
fun: "writeOneKeypadUserName" //Write the user name into keypad   
params: {}//Array object, if the field is not filled in, the last value will be retained.

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | KeypadSN | If empty, it means all online keypads, other array values are invalid |
| userName | Name | Up to 5 Chinese characters (10 English characters), displayed in the upper left corner of the keypad |

Command example:

{  
 "fun": "writeOneKeypadUserName",  
 "baseId": 0,  
 "params": [{  
 "keySn": "1479824643",  
 "value": "John"  
 }],  
 "packetTag": "1"  
}

Receive:

Specify the keypad   
fun: "oneKeypadUserName" // Write the user name into keypad   
baseId:1 //The ID of the base station that received the command   
infos:{}//Array object

| **Json field** | **Field meaning** | **Assignment and meaning** |
| --- | --- | --- |
| keySn | KeypadSN | If empty, it means all online keypads, other array values are invalid |
| userName | Name | Up to 5 Chinese characters (10 English characters), displayed in the upper left corner of the keypad |

Command example:

{  
 "fun" : "oneKeypadUserName" ,  
 "baseId" : 1 ,  
 "infos" : {  
 "keySn" : "1479824643" ,  
 "value" : "John"  
 },  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| C100B-2.4G+S6 | support | The old S6 model supports it, but does not support rare characters. |

## Keypad online information







### Obtain the online information actively

The application can also actively obtain all online keypad information when needed.

注意：请不要在开启投票业务时使用。

Send:   
fun: "getKeypadOnLine" //Get the keypad online status   
params:{   
"times":1 //How long the base station scans the keypad, unit seconds, range 1-10 seconds   
} //

Command example:

{  
 "fun" : "getKeypadOnLine" ,  
 "baseId" : 0 ,  
 "params" : {  
 "times" : 2  
 },  
 "packetTag" : "1"   
}

Receive:   
fun: "keypadOnLine" //Keypad online status   
infos:{} //Array object

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| baseId | Base station id | base station id that keySn keypad is connected |
| keySn | Keypad Sn |  |
| keyVer | keypad version |  |
| Volt | Keypad battery | T2 is in percentage, invalid for S6 |

Command example:

{  
 "fun" : "keypadOnLine" ,  
 "baseId" : 0 ,  
 "info" : [ {  
 "baseId" : "1" ,  
 "keySn" : "1837454011" ,  
 "ver" : "1.0.6" ,  
 "hModel" : "67" ,  
 "volt" : "100"  
 } ,  
 {  
 "baseId" : "1" ,  
 "keySn" : "1837454012" ,  
 "ver" : "1.0.6" ,  
 "hModel" : "67" ,  
 "volt" : "30"  
 }  
 ] ,  
 "packetTag" : "1"   
}

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support | Keypad battery: invalid |
| B200-2.4G+S6 | support | Keypad battery: invalid |
|  |  |  |

## System parameters

### Exit WSCmdApp

WSCmdApp, WSCmdApp will end the WSCmdApp.exe process after responding.   
Send:   
fun: "exit" //Exit WSCmdApp   
params:{}//No such value   
packetTag:"1"//The received packetTag can be used to analyze the communication status

Command example:

{  
 "fun" : "exit" ,  
 "packetTag" : "1"   
}

Receive:

fun: "exit" //Exit WSCmdApp   
infos:{}//No such value

Command example:

{  
 "fun" : "exit" ,  
 "packetTag" : "1"   
}

Note: The command is only valid if it is enabled in the configuration file.

Support Device Description

| **Base station + keypad** | **support** | **Remark** |
| --- | --- | --- |
| B100-2.4G+S6 | support |  |
| B200-2.4G+S6 | support |  |
|  |  |  |

## Exception handling

### WSCmdApp received data exception

When WSCmdApp receives abnormal data, it will not handle it and return the following error message description.

Receive:   
fun: "error" //Function error   
baseId: //No such value   
infos:{ "funId":"F0001","code":"10002"}//

Command example:

{  
 "fun" : "error" ,  
 "infos" : {  
 "funId" : "F0001" ,  
 "code" : "10002"  
 },  
 "packetTag" : "1"   
}

| **code** | **meaning** |
| --- | --- |
| 10001 | JSON data that cannot be parsed correctly |
| 10002 | Parsing JSON errors |
| 20001 | Base station not connected |
| 20002 | Function not supported |
| 20003 | The base station ID is a negative number. The application does not need to pass the ID. The default is 0. |
| 20004 | Parameter error |
| 20005 | Base station busy |
| 30001 | The corresponding SN cannot be found |
| 40001 | The upgrade function can only support one base station online |
| 50001 | Base station ID must be specified for this function |
| 60001 | Keypad SN must be specified for this function |

| **funId** | **meaning** |
| --- | --- |
| F0001 | Base station not connected |
| F0002 | Read base station channel |
| F0003 | Write base station channel |
| F0004 | Read base station baseId, channel, RF power level |
| F0005 | Write base station baseId |
| F0006 | Read base station name |
| F0007 | Write base station name |
| F0008 | Read keypad shutdown time level |
| F0009 | Write keypad shutdown time level |
| F0010 | Read base station match&pairing code |
| F0011 | Write base station match&pairing code |
| F0012 | Read base station serial number |
| F0013 | Write base station serial number |
| F0014 | Read the base station model, hardware version, software version, and device SN number |
| F0015 | Write base station hardware version |
| F0016 | Read base station type |
| F0017 | Write base station type |
| F0018 | Read the base station keypad capacity and Lock |
| F0019 | Write base station keypad capacity and Lock |
| F0020 | Read manual access to return online information |
| F0021 | Write to manually obtain and return online information |
| F0022 | Read base station equipment RF power |
| F0023 | Write base station equipment RF power |
| F0024 | Base station device pairing mode or base station name (customized) |
| F0025 | Base station equipment firmware upgrade |
| F0026 | Read base station fast pairing low power mode |
| F0027 | Write base station fast pairing low power mode |
| F0028 | Read keypad login menu display allowed [S6 applicable] |
| F0029 | Write keypad login menu display allowed [S6 applicable] |
| F0030 | Read keypad login password [S6 applicable] |
| F0031 | Write keypad login password [applicable to S6] |
| F0032 | Read base station name hidden switch |
| F0033 | Write base station name hidden switch |
| F0034 | Read the heartbeat switch between base station and keypad |
| F0035 | Write a heartbeat switch between base station and keypad |
| F0036 | Read base station language |
| F0037 | Write base station language |
| F0038 | Read base station CRC |
| F0039 | Write base station CRC |
| F0040 | Writing Broadcast Poll Titles |
| F0041 | Check voting opening password |
| F0042 | Read the base station to obtain the voting activation password |
| F0043 | Write to the base station to obtain the voting activation password |
| F0044 | Read base station user-defined string |
| F0045 | Write base station user-defined string |
| F0046 | Read base station UID |
| F0047 | Write base station UID |
| F0048 | Read keypad feedback mode parameters |
| F0049 | Write keypad feedback mode parameters |
| F0050 | Read base station currently enabled upgrade state. |
| F0051 | Write base station currently enabled upgrade state. |
| F0052 | Manually stop keypad upgrade state |
| F0053 | Write keypad currently enabled upgrade state |
| F0054 | Write to kick out a single keypad |
| F0055 | Clear all keypads |
| F0056 | write to lock keypad |
| F0057 | Write to unlock keypad |
| F0058 | Write the second line of text on T2 keypad [idle mode] |
| F0059 | Read attendance base station clock settings |
| F0060 | Write attendance base station clock settings |
| F0061 | Read the number of attendance signals sent by the attendance keypad per second |
| F0062 | Write the number of times the attendance keypad sends attendance signals per second |
| F0063 | Read attendance keypad sending attendance signal power |
| F0064 | Write attendance keypad sending attendance signal power |
| F0065 | Write and modify keypad pairing code |
| F0066 | Write keypad function prompts preset string feedback |
| F0067 | write idle mode |
| F0068 | Single choice multiple choice with timing mode |
| F0069 | True or False Questions with Timing Mode |
| F0070 | Rush answer questions with timing mode |
| F0071 | Number question with timing mode |
| F0072 | Binding with timer mode |
| F0073 | Base station quick match |
| F0074 | Read base station characteristic data |
| F0075 | Remote shutdown |
| F0076 | Read keypad Online SN |
| F0077 | Write base station reset |
| F0078 | Write keypad peripheral instructions |
| F0079 | Write keypad pairing code |
| F0080 | write lock keypad,unlock keypad |
| F0081 | Fill in the blanks mode |
| F0082 | Read NFC information name |
| F0083 | Write NFC information name |
| F0084 | Write NFC pairing code |
| F0085 | NFC clear command |
| F0086 | Start multiple questions |
| F0087 | Pause multiple questions |
| F0088 | Continue multiple questions from the previous time |
| F0089 | Send correct answers to multiple question types |
| F0090 | Transfer of answers to multiple question types |
| F0091 | Write whitelist |
| F0092 | Read whitelist |
| F0093 | Write base station blacklist |
| F0094 | Read base station blacklist |
| F0095 | C100B+S6 [old model] write username on keypad |
| F0096 | C100B+S6 [old model] read username on keypad |
| F0097 | Keypad pauses sleep operation |
| F0098 | Asynchronous test mode |
| F0099 | Tcp parameter reading |
| F0100 | Tcp parameter writing |
| F0101 | Customed question type parameters |
| F0102 | Write S6 SMS |
| F0103 | Write S6 keypad FSN |
| F0104 | Free question mode |
| F0105 | Base station AES |
| F0106 | Base station AES clone |
| F0107 | Basic parameter configuration |
| F0108 | Delete the specified whitelist SN |
| F0109 | Add specified SN to whitelist |
| F0110 | Voice question mode |
| F0111 | Multiple question quick answer mode |
| F0112 | Quick match mode |
| F0113 | Keypad test mode |
| F0114 | Write points and the second line of text |
| F0115 | Write keypad name |
| F0116 | Keypad shutdown time level |