A&T High-end Ceiling Speakerphone Function Control Protocol Documentation V1.0
（TCP version）

Content

Version record 3
1、Introduction 3
2、Block Diagram 3
3、Protocol 3
4、Response Message Specification 8
5、Function Control Description 9
5.1 Device Display 9
5.1.1 Device Information Acquisition 9
5.1.2 Local Sound Amplification Acquisition 11
5.1.3 Device Mute State 11
5.1.4 Device Operating Status 12
5.1.5 Local Voice Amplification Status Of The Device 13
5.2 Device Control 13
5.2.1 Device Indicator Blinking 13
5.2.2 Local Sound Amplification Control 13
5.2.3 Device Restart 14
5.2.4 Device Mute Control 14
5.2.5 Device Amplification Control 15
5.3 Device Group Management 16
5.4 Camera Linkage 17
5.4.1 Obtain Camera Binding Information 17
5.4.2 Obtain The Configuration Information About The Linkage Area 18
5.4.3 Get Real-time Speaker Location Information 22

Version record

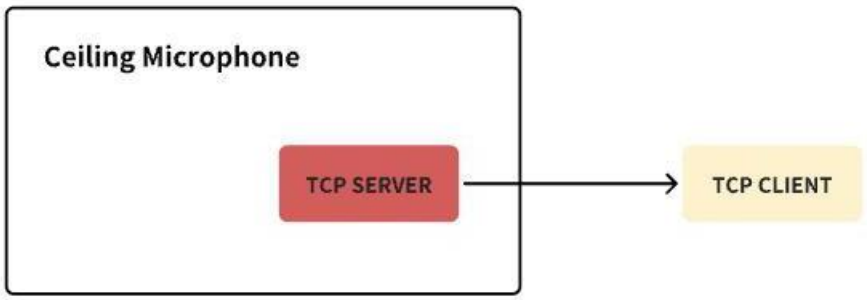
Document version	Revision time	Reviser	Remark
V1.0	2024.06.06	Yongjian.Ma	Support device display (eg: basic information, silent status, and running status) and basic function control (eg:indicator blinking, device restart, and sound amplification control). Note: Ensure that the version of the configuration tool is 0.22.0 or later and the firmware version is 5.5 or later.

1、Introduction

This document is intended for third-party control systems to access the capping device to obtain information and control the device.

2、Block Diagram

A TCP server (see the red box) is built in the ceiling wheat device. The central controller needs to implement a TCP client (see the yellow box) to communicate with each other according to the protocol specified in this document. As shown in the picture below:



The TCP server uses the SSL encryption mode (link encryption only, that is, no certificate is required). Port number: 19998.

3、Protocol

Since TCP itself is a reliable communication based on byte stream, the application layer must define the communication format based on packet in order to solve the sticky packet problem.

Generally, packets are in the form of TLV (T: type, L: length, V: specific data). The current packet form based on TCP adopts the form of data header and data body, which are described as follows:



Data Head description: The header consists of two fields, namely the message type and the remaining length of the message.

Message type: consists of one byte, its specific meaning is shown in the following table:

Message type	**bit7**	**bit6**	**bit5**	**bit4**	**bit3 – bit0 retention**	**Message direction**	**Description**
connect(0)	0	0	0	0	0	client to server	handshake
connack(1)	0	0	0	1	0	server to client	handshake response
request(2)	0	0	1	0	0	client to server	request
response(3)	0	0	1	1	0	server to client	respond
notify(4)	0	1	0	0	0	server to client	asynchronous notification
ping(5)	0	1	0	1	0	client to server	keepalive message
pong(6)	0	1	1	0	0	server to client	keepalive message
disconnect(7)	0	1	1	1	0	client to server	Active disconnection (the server will use this message to determine whether the client is exiting abnormally)

Message remaining length

Use 1-4 bytes encoding, indicating the size of the data body (excluding the message header), as shown in the following table:

Coded bytes	**Length range**	**Format**
1	0 - 127	0x0 - 0x7F
2	128 - 16383	(0x80, 0x01) - (0xFF, 0x7F)
3	16384 - 2097151	(0x80,0x80,0x01) - (0xFF,0xFF, 0x7F)

A&T

4	2097152 - 268435455	（0x80,0x80,0x80,0x01）- (0xFF,0xFF,0xFF,0x7F)

Note: This length encoding is derived from the remaining length encoding in the MQTT protocol.

Message body description

Different messages have different message body contents, as shown in the following table:

Message type	Message body	Total message length (bytes)	Description
connect	2 bytes big-ended order	1 + 1 + 2 = 4	The message body is the keepalive time
connack	1 byte	1 + 1 + 1 = 3	The message body is the handshake result： 0 indicates success
request	128 bytes	1 + 2 + 128	For example the message body contains 128 bytes of data
response	12 bytes	1 + 1 + 12	For example the message body contains 12 bytes of data
notify	0 byte	1 + 1 = 2	For example the message body contains 0 bytes of data
ping	0	1 + 1 = 2	Message-free body
pong	0	1 + 1 = 2	Message-free body
disconnect	0	1 + 1 = 2	Message-free body



Example: The data in the following screenshot is displayed in hex mode.

1、 Connect message (4 bytes)



- 0x00：Indicates the connect message type
- 0x02：Indicates that the remaining message length is 2 bytes
- 0x00 0x05：Indicates that the ping-pong period is set to 5 seconds

A&T

2、 Connack message（3bytes）

```
10 01 00
```

- 0x10：Indicates the connack message type
- 0x01：Indicates that the remaining message length is 1 byte
- 0x00：Handshake success

3、 Request message

```
20 1c 7b 22 72 65 71 75 65 73 74 22 3a 22 64 65   | .{"request":"de|
76 2e 76 6f 6c 75 6d 65 2e 67 65 74 22 7d         |v.volume.get"}|
```

- 0x20：Indicates the request message type
- 0x1c：Indicates that the remaining message length is 28 bytes

4、Response message

```
30 31 7b 22 72 65 73 70 6f 6e 73 65 22 3a 22 64   |01{"response":"d|
65 76 2e 76 6f 6c 75 6d 65 2e 67 65 74 22 2c 22   |ev.volume.get","|
64 61 74 61 22 3a 7b 22 76 6f 6c 75 6d 65 22 3a   |data":{"volume":|
38 7d 7d                                          |8}}|
```

- 0x30：Indicates the response message type
- 0x31：Indicates that the remaining message length is 49 bytes

5、 Notify message

A&T

```
40 db 02 7b 22 6e 6f 74 69 66 79 22 3a 22 64 65   |@..{"notify":"de|
76 2e 69 6e 66 6f 22 2c 22 64 61 74 61 22 3a 7b   |v.info","data":{|
22 6e 61 6d 65 22 3a 22 e7 a1 ac e4 bb b6 e6 b5   |"name":".......|
8b e8 af 95 5f 23 32 22 2c 22 73 6e 22 3a 22 43   |...._#2","sn":"C|
30 31 46 32 33 34 38 30 30 30 30 32 38 22 2c 22   |01F2348000028","|
6e 65 74 22 3a 7b 22 64 68 63 70 22 3a 30 2c 22   |net":{"dhcp":0,"|
69 70 22 3a 22 31 37 32 2e 31 36 2e 34 30 2e 31   |ip":"172.16.40.1|
32 22 2c 22 6e 65 74 6d 61 73 6b 22 3a 22 32 35   |2","netmask":"25|
35 2e 32 35 35 2e 32 35 35 2e 30 22 2c 22 67 61   |5.255.255.0","ga|
74 65 77 61 79 22 3a 22 31 37 32 2e 31 36 2e 34   |teway":"172.16.4|
30 2e 32 35 34 22 2c 22 6d 61 63 22 3a 22 37 30   |0.254","mac":"70|
3a 34 66 3a 30 38 3a 30 38 3a 61 34 3a 31 39 22   |:4f:08:08:a4:19"|
7d 2c 22 72 75 6e 74 69 6d 65 22 3a 22 34 39 37   |},"runtime":"497|
32 22 2c 22 6e 6f 77 22 3a 22 32 30 32 34 2d 30   |2","now":"2024-0|
36 2d 31 32 20 31 39 3a 30 30 3a 32 39 22 2c 22   |6-12 19:00:29","|
76 65 72 73 69 6f 6e 22 3a 22 35 2e 34 2e 32 22   |version":"5.4.2"|
2c 22 64 65 73 63 22 3a 22 30 30 35 2e 30 30 36   |,"desc":"005.006|
2e 30 30 30 2e 30 32 34 2e 30 31 39 2e 30 30 32   |.000.024.019.002|
2d 28 30 30 30 34 29 22 2c 22 73 73 70 65 56 65   |-(0004)","sspeVe|
72 73 69 6f 6e 22 3a 22 76 30 2e 31 2e 33 35 5f   |rsion":"v0.1.35_|
32 30 32 34 30 35 32 37 22 2c 22 62 6f 61 72 64   |20240527","board|
54 79 70 65 22 3a 22 4d 43 31 30 22 7d 7d          |Type":"MC10"}}|
```

- 0x40：Indicates the notify message type
- 0xdb 0x02：Indicates that the remaining length is 128 * 0x02 + (0xdb & 0x7F) = 347nytes

6、 Ping message（2bytes）

```
50 00
```

- 0x50：Indicates the ping message type
- 0x00：Indicates that the remaining length is 0 bytes

7、 Pong message（2 bytes）

```
60 00
```

- 0x60：Indicates the pong message type
- 0x00：Indicates that the remaining length is 0 bytes

Message model

The connect message is the first one initiated by the client after a successful TCP connection. The server returns a connack message after receiving the connection request to inform the client whether the handshake is successful.

When connack returns a success message, the client can initiate a specific request through the request message (described in the following section). Upon receiving the request, the server will

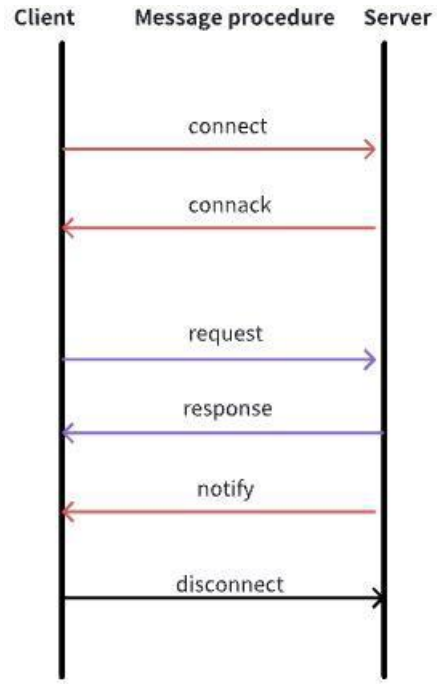
A&T

return the request confirmation or result through a response message. The server will also push the real-time status of the device to the client through the notify message.

When the client does not send a request request for a long time, it needs to send a ping message, and the server returns a pong message after receiving the ping message.

When the client no longer needs to communicate, it can send a disconnect message requesting the server to close the connection.

The specific process is shown in the following figure:



4、Response Message Specification

The request/response/notify message body is in JSON format. The request message contains the request field, which indicates the specific request. The response message contains the response field, and its content is consistent with that of the request. A notification message contains the notify field, which indicates a specific notification.

When the client makes a request, if the current device is not ready, or the parameters of the request are invalid or other reasons cause the request to fail, the device will return the following message as a response:

{"response":"dev.request.state","data":{"request":"current request","reason":0}}

Specific instructions are shown in the following table:

reason	Instructions
-1	The equipment is not ready
0	Success (will not appear)
1	Parameter invalid
2	Request not supported
3	Device exception
Other	Subsequent addition

5、Function Control Description

5.1 Device Display

5.1.1 Device Information Acquisition

Request	{
	"request": "dev.info.get"
	}

Respond	{
	"response": "dev.info.get"
	}
	The delegate request has been received and the result will be notified via notification

Notification	{
	"notify": "dev.info",
	"data": {
	"name": "Ceiling microphone",

A&T

A&T

<table>
<tr>
<td></td>
<td>
<pre>
"sn": "C01F2348000028",
"net": {
"dhcp": 0,
"ip": "172.16.40.12",
"netmask": "255.255.255.0",
"gateway": "172.16.40.254",
"mac": "xx:xx:xx:xx:xx:xx"
},
"runtime": "77808",
"now": "2024-06-13 15:14:25",
"version": "5.4.2",
"desc": "005.006.000.024.019.002-(0004)",
"sspeVersion": "v0.1.35_20240527",
"boardType": "MC10"
}
}
</pre>
name：Device name sn：
Device string number
net：Device network information
runtime：Device running time (seconds)
</td>
</tr>
</table>

	now：Device current time version： Device firmware version desc：Internal device firmware version sspeVersion：Device algorithm firmware version boardType：Device type If the name and net fields are modified, they are updated every time the device information is obtained, and the runtime and now fields are updated every time the device information is obtained.
Description	Obtain device information and version

5.1.2 Local Sound Amplification Acquisition

Request	{
	"request": "dev.volume.get"
	}

Respond	{
	"response": "dev.volume.get",
	"data": {
	"volume": 7
	}
	}
	volume value range：[0, 8]

5.1.3 Device Mute State

Request	{
	"request": "dev.mute.get"
	}

Respond	{
	"response": "dev.mute.get",
	"data": {

	"mute": 0 } } mute description of value： 1： Device mute 0：Device normal recording

5.1.4 Device Operating Status

Request	{
	"request": "dev.state.get"
	}

Respond	{
	"response": "dev.state.get",
	"data": {
	"state": 1
	}
	}
	state value description： 1：
	leisure
	2：recording
	3：production test (this value will only be available with production test software) 4
	：upgrade

Notification	{
	"notify": "dev.state.info",
	"data": {
	"state": 1,
	"megaphone": 1
	}
	}

Description	Automatically triggers a notification when the device status changes

5.1.5 Local Voice Amplification Status Of The Device

Request	{
	"request": "dev.megaphone.get"
	}

Respond	{
	"response": "dev.megaphone.get",
	"data": {
	"megaphone": 0
	}
	}

5.2 Device Control

5.2.1 Device Indicator Blinking

Request	{
	"request": "dev.led.blink"
	}

Respond	{
	"response": "dev.led.blink"
	}

Description	Device indicator blinking

5.2.2 Local Sound Amplification Control

Request	{
	"request": "dev.volume.set",
	"data": {
	"volume": 7

A&T

	} } volume value range[0, 8]

Respond	{
	"response": "dev.volume.set"
	}

Notification	{
	"notify": "dev.volume.info",
	"data": {
	"volume": 7,
	"mute": 0
	}
	}

Description	Trigger notifications when the user changes the volume (including remote control, group synchronization control)

5.2.3 Device Restart

Request	{
	"request": "dev.reboot"
	}

Respond	{
	"response": "dev.reboot"
	}

Description	The client does not have to wait for a response message

5.2.4 Device Mute Control

Request	{
	"request": "dev.mute.set",
	"data": {

	"mute": 1 } }

Respond	{
	"response": "dev.mute.set"
	}

Notification	{
	"notify": "dev.volume.info",
	"data": {
	"volume": 7,
	"mute": 0
	}
	}

Description	Triggers notifications when the user changes the mute Settings (including remote control, group synchronization control)

5.2.5 Device Amplification Control

Request	{
	"request": "dev.megaphone.set",
	"data": {
	"megaphone": 0
	}
	}

Respond	{
	"response": "dev.megaphone.set"
	}

Notification	{
	"notify": "dev.state.info",
	"data": {

	"state": 1, "megaphone": 1 } }

5.3 Device Group Management

Request	{
	"request": "dev.group.settings.get"
	}

Respond	{
	"response": "dev.group.settings.get",
	"data": {
	"master": 1,
	"deviceTable": [
	"xx:xx:xx:xx:xx:xx",
	"xx:xx:xx:xx:xx:xx"
	],
	"masterAddress": "192.168.1.121",
	"macAddress": "xx:xx:xx:xx:xx:xx"
	}
	}
	master value description： 1：
	Primary device 0：Secondary
	device deviceTable array
	description：
	mac addresses of all secondary devices in the current group
	masterAddress Secondary device Specifies the IP address of the primary device to be connected

	macAddress The mac address of the master device that the secondary device needs to connect to

Notification	{
	"notify": "dev.group.settings",
	"data": {
	"master": 1,
	"deviceTable": [
	"xx:xx:xx:xx:xx:xx",
	"xx:xx:xx:xx:xx:xx"
	],
	"masterAddress": "192.168.1.121",
	"macAddress": "11:22:22:22:33:44"
	}
	}
	It is also automatically triggered when the device group status changes

Description	Obtain device group management information

5.4 Camera Linkage

5.4.1 Obtain Camera Binding Information

Request	{
	"request": "dev.camera.bindings.get"
	}

Respond	{
	"response": "dev.camera.bindings.get",
	"data": {
	"enable": 1,
	"mac": "xx:xx:xx:xx:xx:xx"
	}

	} The mac address is the mac address of the camera device

Notification	{
	"notify": "dev.camera.bindings",
	"data": {
	"enable": 1,
	"mac": "xx:xx:xx:xx:xx:xx"
	}
	}
	enable field description：
	1：The current ceiling mic is bound to the camera with the specified mac address 0
	：unbind
	It is also automatically triggered when the camera binding information changes

5.4.2 seat

Request	{
	"request": "dev.camera.settings.get"
	}

Respond	{
	"response": "dev.camera.settings.get",
	"data": {
	"enable": 1,
	"settings": {
	"seats": [
	{
	"id": 1,
	"label": "Linkage area 1",
	"pos": [

```
                [
                    4.19,
                    5.79
                ],
                [
                    4.99,
                    5.79
                ],
                [
                    4.99,
                    6.59
                ],
                [
                    4.19,
                    6.59
                ]
            ],
            "enable": 1
        },
        {
            "id": 2,
            "label": "Linkage area 2",
            "pos": [
                [
                    3.63,
                    4.45
                ],
                [
                    4.43,
```

	4.45], [4.43, 5.25], [3.63, 5.25]], "enable": 1 }] } } }

Notification	{
	"notify": "dev.camera.settings",
	"data": {
	"enable": 1,
	"settings": {
	"seats": [
	{
	"id": 1,
	"label": "Linkage area 1",
	"pos": [
	[
	4.19,

```
                5.79
            ],
            [
                4.99,
                5.79
            ],
            [
                4.99,
                6.59
            ],
            [
                4.19,
                6.59
            ]
        ],
        "enable": 1
    },
    {
        "id": 2,
        "label": "Linkage area 2",
        "pos": [
            [
                3.63,
                4.45
            ],
            [
                4.43,
                4.45
            ],
```

	[4.43, 5.25], [3.63, 5.25]], "enable": 1 }] } } }

5.4.3 Get Real-time Speaker Location Information

Notification	{
	"notify": "dev.speaker.seat",
	"data": {
	"seat": 2
	}
	}
	seat value description: The algorithm will calculate the speaker's position in real time and calculate whether the speaker is in the preset linkage area according to the actual configured linkage area information. When it determines that the speaker is in the preset linkage area, it will take the initiative to send this notification.