MXC

TPCI command strings

Third-party command strings for Shure Microflex Complete discussion system

Version: 5 (2020)

Table of Contents

1 Introduction

The MXC Microflex Complete Conference System features an ethernet connection with the purpose of providing an interface for controlling and monitoring of the system using TPCI (Third Party Control Interface). By setting up a simple (raw) TCP/IP socket connection to the DIS-CCU Central Unit (CCU), control options are available.

This document describes the TCP/IP raw socket protocol for communicating with the MXC Conference System. This protocol provides a set of commands, enabling a third part control application to monitor and/or control system status of a MXC Conference System.

Some examples of functionally available using the protocol:

- Setting a microphone in speak or in request
- Retrieving a list of seats available in the system
- Starting/stopping a voting session

This “simple to use” interface supports applications developed by customers, so the protocol is deliberately kept simple to avoid complexity.

The protocol offers means for supplementing the control functionality available through the CCU interactive display and browser interface, however some commands and settings available in the browser interface and in the CCU interactive display are not available using the TPCI

Customer applications can include but are not limited to AMX ® or Crestron ® room control systems, PC or micro controller based applications e.g. for button mimics and camera control applications.

2 General Protocol Behavior

2.1 TCP/IP socket connection

A TCP/IP socket connection to the CCU must be established for the TPCI protocol to become available. Configuration of the CCU connection to the Ethernet must be defined from the CCU interactive front plate control/browser interface, and an IP address for the CCU must be assigned in the network.

Choose either a static IP address or an IP address assigned through DHCP. It is convenient to ensure, that the CCU ends up with the same IP address at each start up.

For using *hostname.local* on Windows, the Bonjour protocol distributed by Apple needs to be installed on Windows.

Knowing the IP address, the only additional information required for setting up a TCP/IP connection is the port number:

Port Number = 3142

2.1.1 Test connection to CCU via Putty

If the CCU is assigned IP address 192.168.1.100, the external application must connect the TCP/IP socket to the address 192.168.1.100:3142.

Knowing the IP address of the CCU a connection can be set up using a simple terminal program like Putty ®.

1. Download at www.putty.org
2. Start Putty.

3. Insert IP address and Port Number.
4. Select ‘Raw’ for the Connection Type.
5. Press ‘Open’ to establish connection to the CCU. Control is now possible.
6. Give command ‘help’ to see a list of commands available.

2.2 Cmd structure (Command to CCU)

To control the CCU an External Control sends commands to the CCU as command lines. Commands lines are build up in a very simple manner:

```
<command><SP><data><CR>

<command><SP><data><LF>

<SP>      Space - 0x20 = 32

<CR>      Carriage return - 0x0D = 13

<LF>      Line Feed - 0x0A = 10
```

Command lines are terminated by Carriage Return <CR> or Line Feed <LF> or both. In order to be able to communicate with Windows systems, Linux systems or other systems, the CCU understands both types of command line terminations.

Notice also that there is a space between the command and data. If a command does not carry any data, space is possible but not required.

The commands are not sensitive to upper/lower case.

2.2.1 Example

```
mic_on 212
```

Turn on microphone at seat 212. Command = 'mic_on', data = '212'. The ‘mic_on’ command carries a seat number as data.

2.3 Cmd structure (Message from CCU)

Command lines out of the CCU are just as simple:

```
<command><SP><data><CR><LF>

<SP>      Space - 0x20 = 32

<CR>      Carriage return - 0x0D = 13

<LF>      Line Feed - 0x0A = 10
```

Again, to satisfy most systems, the CCU terminates command lines by including both <CR> and <LF>.

2.4 Seat numbering

Conference units are identified by means of seat numbers. Each Conference Unit is assigned a seat number. This is done automatically for all Conference Units, when they are connected to the CCU. The CCU Browser interface is used to change seat numbering if desired.

Seat numbers must be in the range from 1 to 65535.

2.5 CCU reply to commands

Generally, a command from external application is replied to by the CCU. But, reply to a command is for a number of commands only produced if actions are taken by the CCU due to the command.

When f. ex. a ‘mic_on’ command results in a microphone being turned on, the CCU replies with a ‘mic_on’ command. On the other hand, if a 'mic_on' command does not lead to turning on a microphone, the CCU does not produce any reply.

There can be several reasons for the CCU to reject turning on a microphone:

- The microphone is already turned on
- The microphone is no longer connected to the system.
- Speak list is already full ('max_speakers'), and interrupt is not possible.

2.6 Retrieving system status

The CCU supports streaming of status. When an External Control f. ex. issues a **'mic_status', 'audio_status', 'reply_status'** or **'voting_status'** command, the CCU responds by sending the status. Hereby, it is possible for an External Control to synchronize with the CCU status.

3 Help

help Command to CCU

help <command> Command to CCU

Help information is available listing all commands supported by the CCU. If a command is included in the help command, description on that command is returned.

Note: The help command results in a number of command lines returned from the CCU. The command is intended for use in a simple console.

Command	Description
allow_mic_off	Set "Allow Microphone Off" on/off.
audio_path	Define audio connections (mic/linein to speaker/lineout/floor)
audio_status	Request status of audio settings
auto_floor	Activate or deactivate AutoFloor (floor sound in interpreter channels with no active interpreter)
cancel_attendance_check	Cancel an attendance check session
cancel_voting	Cancel a voting session
custom_button_indication	Provide user feedback on custom function buttons
get_device_info	Get Device Information
help	Help for command interface
input_selection	Setup Audio input for a seat
line_input_gain_1	Adjust line input gain to 0 dB or 10 dB
line_input_gain_2	Adjust line input gain to 0 dB or 10 dB
line_input_level_1	Level of line input 1 signal
line_input_level_2	Level of line input 2 signal
line_output_volume	Volume of line output signal
loudspeaker_volume	Loudspeaker volume of microphone units
max_replies	Set the max number of replies. Possible values
max_requests	Max size of Request Queue
max_speakers	Max number of delegate speakers
max_total_speakers	Max number of speakers (delegates+chairman)
meeting_status	Request current meeting status
start_meeting	Start a meeting
stop_meeting	Stop current meeting
mic_all_delegates_off	Turn all delegate microphone off
mic_all_off	Turn all microphones off
mic_all_replies_off	Turn all microphone replies off
mic_all_requests_off	Turn all microphone requests off
mic_level	Change microphone audio gain level
mic_attenuation	Change microphone attenuation
mic_interrupt	Set microphone interrupt mode
mic_mode	Set microphone operation mode. Reply modes are only possible in MXC/6000 mode
mic_next_on	Turn next requester on
mic_next_request_on	Turn on the next request
mic_off	Turn off microphone with seat number
mic_on	Turn on microphone with seat number

Command	Description
mic_on_from_request	Turn on requesting mic with seat_number
mic_priority	Change microphone priority
mic_reply_on	Set a microphone in reply
mic_reply_off	Remove a microphone from reply
mic_request_off	Turn request off microphone with seat number
mic_request_on	Turn request on microphone with seat number
mic_speaker_attenuation	Change microphone unit speaker attenuation
mic_status	Microphone Status
Mix_minus	Mix_minus setting
sign_status	Get current status of name sign control
sign_mode	Sets current namesign control mode
sign_font_size_change	Define change to NameSign font
sign_seat	Sets/clears seat assignment to namesign
sign_text	Sets/clears text to show on namesign
reply_status	Get reply options info.
speak_button_lock	Set Speak Button Lock on/off.
start_attendance_check	Start an attendance check session
start_voting	Start a voting session
stop_attendance_check	Stop an attendance check session
stop_voting	Stop a voting session
voting_status	Request voting status

4 Microphone Control

4.1 Microphone on

mic_on <seat_no>	Command to CCU
mic_on <seat_no> <name>	Message from CCU

A microphone is turned on.

<seat_no>	The seat number
<name>	The participant name.

If the CCU turns on the microphone, it will reply with a **'mic_on'** command. And, if the microphone appeared in the request list, it is taken out of the request list, which makes the CCU issue a **'mic_request_off'** command as well.

When SW6000 is connected and MXC640 or DC6990 devices are in the system *seat names* are not in use.

4.2 Microphone off

mic_off <seat_no>	Command to CCU and message from CCU

Instruct the CCU to turn off microphone at seat_no. If the CCU turns off the microphone, it replies with a **'mic_off'** command. If the microphone turned off was the last one in the speak list the CUU returns with a **'mic_all_off'** command.

4.3 All off

mic_all_off	Command to CCU

Instruct the CCU to turn off all microphones. The CCU responds to the command by issuing a **'mic_off'** command for each microphone which is turned off and when all microphones are off with a **'mic_all_off'** command.

4.4 All delegate off

mic_all_delegates_off	Command to CCU

Instruct the CCU to turn off all delegate microphones. A Chairman is not turned off. The CCU responds to the command by issuing a **'mic_off'** command for each microphone which is turned off. If the microphone turned off was the last one in the speak list the CUU returns with a **'mic_all_off'** command.

4.5 Mute all

mic_mute_all <state>

Instruct the CCU to mute/un-mute all delegate microphones.

<state>	Values: 'activate', 'deactivate'

When active, no delegate microphones can be set into Speak.

Note: If the socket connection is disconnected during an 'activate' mute state, the CCU will send a 'deactivate' command.

4.6 Reply on

mic_reply_on <seat_no>	Command to CCU

mic_reply_on <seat_no> <reply position> <reply #> <name> Message from CCU

Instruct the CCU to insert a microphone into reply list.

If the CCU inserts a unit into the reply list, it replies with a '**mic_reply_on'** command.

<seat_no>	The seat number
<reply position>	Informs about the position in the reply list.
<reply #>	Informs about the reply number in the reply configuration
<name>	The *seat name* or *delegate name.*

4.7 Reply off

mic_reply_off <seat_no> Command to CCU

Remove microphone from reply list.

If the CCU removes the unit from the reply list, it replies with a **'mic_reply_off'** command.

4.8 All reply off

mic_all_replies_off Command to CCU

Clear the reply list.

The CCU responds by issuing a **'mic_reply_off'** command for each microphone that is removed from the reply list.

4.9 Next on

mic_next_on Command to CCU

Turns on first microphone from the reply list. If the reply list is empty the first microphone in the request list is turned on. If a microphone is turned on, the CCU sends a **'mic_on'** command and a **'mic_reply_off'** or **'mic_request_off'** command.

4.10 Next request on

mic_next_request_on Command to CCU

Turns on the first microphone from the request list. If a microphone is turned on, the CCU sends a **'mic_on'** command and a **'mic_request_off'** and a **'mic_all_replies_off'** command.

4.11 Mic on from request list

mic_on_from_request <seat_no> Command to CCU

Turns on microphone **<seat_no>** from the request list. If a microphone is turned on, the CCU sends a '**mic_on'** command and a '**mic_request_off'** and a **'mic_all_replies_off'** command.

4.12 Request on

mic_request_on <seat_no> Command to CCU

mic_request_on <seat_no> <request position> <name> Message from CCU

Instruct the CCU to insert a microphone the request list. If the CCU inserts the unit into the request list, it replies with a **'mic_request_on'** command.

<seat_no>	The seat number
<request position>	Informs about the position in the request list.
<name>	The *seat name* or *delegate name.*

4.13 Request off

`mic_request_off <seat_no>`	Command to CCU

Instruct the CCU to remove microphone from request list. If the CCU removes the unit from the request list, it replies with a '**`mic_request_off`**' command.

4.14 All request off

`mic_all_requests_off`	Command to CCU

Instruct the CCU to clear the request list. The CCU responds by issuing a '**`mic_request_off`**' command for each microphone that is removed from the request list.

4.15 Max total speakers

`max_total_speakers <max total speakers>`	Command to CCU and message from CCU

Maximum number of speakers allowed to speak. The CCU responds by sending a '**`max_total_speakers`**' command ('max_total_speakers' must always be >= 'max_speakers'. The logic will automatically decrease 'max_speakers' if required)

<max total speakers>	Values: '1' to '8'

4.16 Max delegate speakers

`max_speakers <max speakers>`	Command to CCU and message from CCU

Maximum number of delegates allowed to speak. The CCU responds by sending a '**`max_speakers`**' command.

<max speakers>	Values: '1' to '8'

4.17 Max replies

`max_replies <max replies>`	Command to CCU and message from CCU

Maximum number of delegates allowed in the reply list. The CCU responds by sending a '**`max_replies`**' command.

<max replies>	Values: '0' to '250'.

4.18 Max requests

`max_requests <max requests>`	Command to CCU and message from CCU

Maximum number of delegates allowed in the request list. The CCU responds by sending a '**`max_requests`**' command.

<max requests>	Values: '0' to '250'.

4.19 Speak mode

`mic_mode <mode>`	Command to CCU and message from CCU

Set system speak mode. The CCU responds by sending a '**`mic_mode`**' command.

<mode>	Values: 'auto' (Automatic), 'fifo' (First-in-first-out), 'manual' (Manual), 'vox' (Voice Active), 'Auto + reply' (Automatic+Reply), 'Manual + reply' (Manual + reply) and 'vox+reply' (Voice Active + Reply)

4.20 Speak interrupt ability

mic_interrupt <ability> Command to CCU and message from CCU

Set ability to interrupt. Defines, whether microphones should interrupt or not. The CCU responds by sending a '**mic_interrupt**' command.

<ability>	Values: 'same', 'lower' and 'off'

4.21 Mic priority

mic_priority <seat_number> <priority> Command to CCU and message from CCU

This command sets the priority of a microphone. The CCU responds to this command by returning a **'mic_priority'** message.

<seat_number>	The seat number of the microphone to adjust
<priority>	The desired priority. Values: 0 to 5, where 0 is the lowest priority and 5 is the highest priority.

4.22 Speak button lock

speak_button_lock <mode>

Sets the speak button lock mode Command to CCU and message from CCU

The CCU responds to this command by returning a **'speak_button_lock'** message. There is no setting in the TCP/IP protocol for setting time A and B.

<mode>	Values: 'on', 'off'

4.23 Allow mic off mode

allow_mic_off <mode> Command to CCU and message from CCU

Sets the allow mic off mode. The CCU responds to this command by returning a **'allow_mic_off'** message.

<mode>	Values: 'on', 'off'

4.24 Seat state

seat_state <seat number> <seat state> <name> Message from CCU

Information about a seat.

This information is sent from the CCU in the following situations:

- When a delegate logs in
- When a delegate logs out
- When the seat name is modified
- When the external control application requests microphone status (**mic_status**).
- When a microphone unit becomes lost or found

<seat number>	The seat number identification of a conference unit. An integer ranging from 1 to 65535.
<seat state>	The current state of the seat. Values: 'active' or 'passive'
<name>	The *seat name* or *delegate name.* If a *delegate name* is available for the seat number then the *delegate name* is provided. Otherwise the *seat name* is provided.

4.24.1 Example

```
-> seat_number 12 active John Jones
```

4.25 Microphone status

mic_status	Command to CCU

Ask the CCU to deliver status of the system (microphones in speak, and microphones in reply/request list).

The CCU responds by sending microphone system status. The status is a list of commands from the CCU:

seat_state	(for all units in the system)
mic_priority	(for all units in the system)
mic_mode	
mic_interrupt	
speak_button_lock	
allow_mic_off	
max_total_speakers	
max_speakers	
max_requests	
max_replies	
auto_floor	
ch_off	
mic_on	(for all units in speakers list)
mic_request_on	(for all units in request list)
mic_reply_on	(for all units in reply list)
mic_status_done	

4.26 Reply status

reply_status

Ask the CCU to deliver status of the reply configuration.

The CCU responds by sending reply system status. The status is a list of commands from the CCU:

```
reply_options_count <count>
reply_configuration <reply#> <priority> <color> <label>
…
reply_status_done
```

4.27 Command error

command_error <error text>	Message from CCU

The CCU has received an unknown command.

<**error text**> is a text explaining the fault case.

4.27.1 Example

```
-> command_error  unknown command
-> command_error  syntax error
```

5 Interpretation Control

5.1 Interpreter Channel on

ch_on <ch_no> <language> Message from CCU

Response when an interpreter channel is set active (interpretation is taking place).

<ch_no>	The channel number (currently 1-31)
<language>	The language assigned to the channel. (Language name in English)

5.2 Interpreter Channel off

ch_off <ch_no> <language> Message from CCU

Response when an interpreter channel is set not active (interpretation is not taking place). The message is also send if the language assigned to the channel is changed. Status of all channels are included in the response to the **'mic_status'** command. If the number of channels are increased, the status of the channels are informed

<ch_no>	The channel number (currently 1-31)
<language>	The language assigned to the channel. (Language name in English)

6 Audio Control

6.1 Loudspeaker volume

loudspeaker_volume <volume> Command to CCU and message from CCU

Set the volume of loudspeakers for all conference units. The CCU responds to this command by returning a **'loudspeaker_volume'** command.

<volume>	The volume of the loudspeakers ranging from -41 to 0. The value -41 indicates Off, whereas values from -40 to 0 indicates attenuation in dB.

6.2 Line input 1 level

line_input_level_1 <level> Command to CCU and message from CCU

Adjust the level of line input signal. The CCU responds to this command by returning a **'line_input_level_1'** command.

<level>	The level of line input 1 ranges from -41 to 0. The value -41 indicates Off, whereas values from -40 to 0 indicates attenuation in dB.

6.3 Line input 2 level

line_input_level_2 <level> Command to CCU and message from CCU

Adjust the level of line input signal. The CCU responds to this command by returning a **‘line_input_level_2’** command.

<level>	The level of line input 2 ranges from -41 to 0. The value -41 indicates Off, whereas values from -40 to 0 indicates attenuation in dB.

6.4 Line input 1 gain

line_input_gain_1 <gain> Command to CCU and message from CCU

Adjust input gain of the line in 1 input. The CCU responds to this command by a **‘line_input_gain_1’** message.

<gain>	Values: 0 or 10. 0 dB will not add any gain to the line input whereas 10 dB will add 10 dB gain to the line input.

6.5 Line input 2 gain

line_input_gain_2 <gain> Command to CCU and message from CCU

Adjust input gain of the line in 2 input. The CCU responds to this command by a **‘line_input_gain_2’** message.

<gain> Values: 0 or 10. 0 dB will not add any gain to the line input whereas 10 dB will add 10 dB gain to the line input.

6.6 Line output volume (A-H)

line_output_volume <output> <volume> Command to CCU and message from CCU

Adjust the level of line output signal. The CCU responds to this command by returning a **'line_output_volume'** command.

<output>	Indicates which output is being controlled. Values: ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’, ‘G’, ‘H’.
<volume>	The volume of line output. Values: ‘-41’ to ‘0’. The value -41 indicates Off, whereas values from -40 to 0 indicates attenuation in dB.

6.7 Audio path

audio_path <path> <on_off> Command to CCU and message from CCU

This command defines audio connections in the system. The CU responds to this command by returning an **‘audio_path’** command.

<path>	Indicates which connection is being controlled. Values: 'mic_to_speaker', 'mic_to_floor', 'linein_1_to_speaker', 'linein_1_to_lineout_A', ‘linein_1_to_floor’, 'linein_2_to_speaker', 'linein_2_to_lineout_A', ‘linein_2_to_floor’.
<on_off>	Indicates, whether the audio is being routed from microphones/line-input_1 to speakers/line-output/floor. Values: ‘on’ and ‘off’.

6.8 Individual speaker attenuation

mic_speaker_attenuation <seat_number> <attenuation> Command to CCU and message from CCU

This command sets the speaker attenuation of a microphone. The CCU responds to this command by returning a **‘mic_speaker_attenuation’** message.

<seat_number>	The microphone to adjust
<attenuation>	The desired speaker attenuation. Values: ‘0’ to ‘7’. 0 to 6 will attenuate 0 to 6 dB. 7 is speaker off.

6.9 Individual microphone attenuation

`mic_attenuation <seat_number> <attenuation>` Command to CCU and message from CCU

This command sets the attenuation of a microphone.

<**seat_number**>	The seat number for microphone to adjust
<**attenuation**>	The desired microphone attenuation. Values: 0 to 6 dB, where 0 is the lowest attenuation and 6 is most attenuation.

A ‘**mic_attenuation**’ command to the CCU is responded as follow:

- If the audio level options in the unit, is within the 'mic_attenuation' range of 0 to -6 dB: ‘**mic_attenuation**’ message
- If the unit does not exist: **`'Command_error syntax error'`**
- If the audio level options in the unit is outside the 'mic_attenuation' range of 0 to -6 dB: No response

A (successful) **‘mic_attenuation’** command is also responded with a **‘mic_level’** response.

Important: The **‘mic_attenuation’** command will be obsolete in the future but is still maintained for backward compatibility. It is adviced to use the **‘mic_level’** command instead.

6.10 Individual microphone level

`mic_level <seat_no> <level>` Command to CCU

`mic_level <seat_no> <level> <supported range low> <supported range high>` Message from CCU

This is a command which in addition to the mic_attenuation command will set the microphone audio level in a conference unit.

<seat_no>	The seat number
<level>	The level
<supported range low>	The lowest value in the range (can be negative)
<supported range high>	The highest value in the range (can be negative)

A (successful) mic_level command is also responded with a **‘mic_attenuation’** response if the level options in the unit is within the 'mic_attenuation' range of 0 to -6 dB.

Note: The **‘mic_attenuation’** command is maintained for backward compatibility. The **‘mic_level’** supports an extended level range compared to the **‘mic_attenuation’** in order to support units with extended level range.

6.11 Input selection

`input_selection <seat_no> <input>` Command to CCU

This is a command, which will set the XLR input or port A on a MXCMIU unit. The CCU responds by sending a **‘input_selection’** command. If the seat is not configured to use XLR, the response is: **`'not_configurable'`**.

<seat_no>	The seat number
<input>	The input source values: <xlr_mic>, <xlr_line> or <port_a>

If the unit needs to reconfigure it will go off-line for 10 seconds. TPCI will inform about this as shown in this example for seat 4:

```
<input_selection 4 port_a
>input_selection 4 port_a
```

```
>seat_state 4 passive Seat 4
>seat_state 4 active Seat 4
```

6.12 Audio status

audio_status Command to CCU

System audio status.

The CCU responds to this command by returning a list of commands for the audio settings:

loudspeaker_volume	
line_output_volume	(for all outputs)
audio_path	(for all paths)
line_input_level	(for all inputs)
line_input_gain	(for all inputs)
mic_speaker_attenuation	(for all units)
mic_attenuation	(for all units) 1)
mic_level	(for all units)
input_selection	(for all units)
mix_minus	
audio_status_done	

1) The 'mic_attenuation' is only reported if the audio level option in the unit is within the 'mic_attenuation' range of 0 to -6 dB.

6.13 Mix-minus

mix_minus <on_off> Command to CCU and message from CCU

This command turns on and off the mix-minus setting for the system. The state of mic-minus is include in the **’audio_status’** command.

7 Voting Control

The external control interface features control of voting sessions and attendance check sessions in the Central Unit.

7.1 Voting configurations

Two different sets of configurations are available:

- CCU controlled by SW6000
- CCU not controlled by SW6000 (standalone)

7.1.1 Retrieve voting configurations

No matter which configurations applies, it is possible for an external controller to request a list of voting configurations - using the command **`'voting_status'`**. The CCU will reply by returning the list of voting sessions currently applicable (either SW6000 defined voting configurations or build-in voting configurations).

7.1.2 Standalone

The CCU features the following voting configurations in standalone mode:

- 2-button voting
- 2-button secret voting
- 3-button voting
- 3-button secret voting
- 5-button voting
- 5-button secret voting

7.1.3 SW6000 controlled

SW6000 supports a number of voting configurations.

Via the external control protocol, it is possible to make two requests:

- Start one of the SW6000 defined voting configurations
- Start SW6000 default voting configuration

7.2 Voting results

During voting sessions the CCU delivers interim voting results, unless the configuration is secret.

At completion of a voting configuration, the CCU delivers final voting results. Also at completion of an attendance check the CCU delivers final attendance check result.

7.2.1 Standalone

The CCU supports 2-, 3- and 5-button voting.

2-button voting configurations the following alternatives apply:

1 ‘Yes’
2 ‘No’

3-button voting configurations the following alternatives apply:

1 ‘Yes’
2 ‘Abstain‘
3 ‘No’

5-button voting configurations the following alternatives apply:

1	‘+ +’
2	‘+’
3	‘0’
4	‘-‘
5	‘- -‘

7.2.2 SW6000 controlled

With SW6000 attached to the CCU, voting results are defined in SW6000. Up to 9 voting results calculations can be defined.

7.3 Start voting session

start_voting <voting_configuration_id> Command to CCU

Starts a voting session in the CCU.

<voting_configuration_id> Identification of the voting configuration to start.

The CCU replies with **‘voting_started’**, if a voting session is started.

In standalone the ‘voting_configuration_id’ is defined as:

'1'	2-button voting
'2'	2-button secret voting
'3'	3-button voting
'4'	3-button secret voting
'5'	5-button voting
'6'	5-button secret voting

Used with SW6000 the configurations can be requested with the command **‘voting_status’**.

If no ‘voting_configuration_id’ is specified, configuration ‘3’ is used in standalone and the meeting default in SW6000.

7.3.1 Example

```
<- voting_status
-> voting_configuration 1 3-button voting
-> voting_configuration 2 3-button secret voting
-> voting_configuration 3 5-button voting
-> voting_configuration 4 5-button secret voting
-> voting_status_done
<- start_voting 1
-> voting_started 1
-> interim_voting_result 1 0 Yes
-> interim_voting_result 2 0 Abstain
-> interim_voting_result 3 0 No
-> individual_vote 3 Carsten 1
-> individual_vote 8 Bill Jones 1
-> individual_vote 7 John 1
-> individual_vote 6 Peter 3
-> interim_voting_result 1 3 Yes
-> interim_voting_result 2 0 Abstain
-> interim_voting_result 3 1 No
-> stop_voting
-> voting_stopped
-> final_voting_result 1 3 Yes
-> final_voting_result 2 0 Abstain
-> final_voting_result 3 1 No
```

7.4 Voting session started

voting_started <voting_configuration_id> Message from CCU

Message from the CCU, that a voting session is started

<voting_configuration_id>	Identification of the voting configuration to start.

7.5 Stop voting session

stop_voting Command to CCU

Used to stop an ongoing voting session in the CCU. If the voting session is stopped, the CCU replies with **'voting_stopped'**.

7.6 Voting session stopped

voting_stopped Message from CCU

Message from the CCU, that a voting session is stopped

7.7 Cancel voting session

cancel_voting Command to CCU

Used to cancel an ongoing voting session in the CCU. If the voting session is cancelled, the CCU replies with '**voting_cancelled'**.

7.8 Voting session cancelled

voting_cancelled Message from CCU

Message from the CCU, that a voting session is canceled.

7.9 Interim voting results

interim_voting_result <result_id><interim_result><result_text> Message from CCU

During a voting session the CCU informs about interim voting results. When new votes are cast, the CCU distributes interim voting results. This command informs about one of the interim voting results.

<result_id>	With SW6000 connected values [1 to 9] corresponding to the 9 result columns in the SW6000 'Voting Configurations'. For a standalone CCU this is the button numbers [1 to 5].
<interim_result>	Interim voting result. With SW6000 this is the result for the 9 result columns. For a standalone CCU this is the number of votes given on the specified button.
<result_text>	Text related to the result. With SW6000 this is the labels for the 9 result columns. For a standalone CCU this is the voting button labels.

7.10 Interim individual voting results

individual_vote <seat_no> <name> <button_id> Message from CCU

During a voting session the CCU informs about individual votes. When new votes are cast, the CCU distributes individual votes unless the voting session is secret. This command informs about one vote.

NOTE: Only votes cast by pressing a button on a conference unit are delivered on TPCI. Votes cast in voting control software such as SW6000 are not provided on TPCI, however the **'interim-voting_result'** and **'final_voting_result'** does include votes cast on SW6000.

<seat_no>	The seat number

<name>	The seat name or participant name
<button_id>	The voting button number: [1 to 5]

7.10.1 Example

```
> individual_vote 3 “Carsten” 1
> individual_vote 8 “Bill Jones” 1
> individual_vote 7 “John” 1
> individual_vote 6 “Peter” 3
> individual_vote 8 “Bill Jones” 2
```

Important: Individual results are not visible in TCPI for votes cast from the CUA application in SW6000

7.11 Final voting results

final_voting_result <result_id><final_result><result_text> Message from CCU

At completion of a voting session the CCU distributes final voting results. This command informs about final voting result for one of the voting alternatives.

<result_id>	Identification of result. With SW6000 connected values [1 to 9] corresponding to the 9 result columns in the SW6000 ‘Voting Configurations’. For a standalone CCU this is the button numbers [1 to 5].
<final_result>	With SW6000 this is the result for the 9 result columns. For a standalone CCU this is the number of votes given on the specified button.
<result_text>	Text related to the result.

7.12 Start attendance check session

start_attendance_check Command to CCU

Used to start an attendance check session in the CCU. If an attendance check session is started, the CCU replies with **‘attendance_check_started’**.

7.13 Attendance check session started

attendance_check_started Message from CCU

Message from the CCU, that a attendance check session is started

7.14 Stop attendance check session

stop_attendance_check Command to CCU

This command is used to stop an ongoing attendance check session in the CCU. If the attendance check session is stopped, the CCU replies with **‘attendance_check_stopped’**.

7.15 Attendance check session stopped

attendance_check_stopped Message from CCU

Message from the CCU, that a attendance check session is stopped

7.16 Cancel attendance check session

cancel_attendance_check Command to CCU

Used to cancel an ongoing attendance check session in the CCU. If the attendance check session is cancelled, the CCU replies with **‘attendance_check_cancelled’**.

7.17 Attendance check session cancelled

attendance_check_cancelled Message from CCU

Message from the CCU, that a attendance check session is canceled

7.18 Interim attendance check result

interim_attendance_check_result <interim_result> Message from CCU

Used by the CCU to inform about the interim attendance check result.

<interim_result>	Contains the interim attendance check result.

7.19 Final attendance check result

final_attendance_check_result <final_result> Message from CCU

Used by the CCU to inform about the final attendance check result after the voting session is stopped. For a standalone system it indicates how many delegates have pressed the 'attendance' button.

<final_result>	Contains the interim attendance check result.

7.20 Voting status

voting_status Command to CCU

Used to request voting status. The result is a list of available voting configurations and info if a voting session is started or stopped. The CCU returns the commands:

```
voting_configuration <voting_configuration_id> <voting_configuration_name>
...
voting_configuration <voting_configuration_id> <voting_configuration_name>
voting_stopped or voting_started <voting_configuration_id>
voting_status_done
```

<voting_configuration_id>	Is an integer identifying the voting configuration.
<voting_configuration_name>	Is a name for the configuration.

7.20.1 Example

```
//Voting ongoing
<- voting_status
-> voting_configuration 1
-> voting_configuration 2
-> voting_configuration 3
-> voting_configuration 4
-> voting_started 1
-> voting_status_done

//No voting ongoing
<- voting_status
-> voting_configuration 1
-> voting_configuration 2
-> voting_configuration 3
-> voting_configuration 4
-> voting_stopped
-> voting_status_done
```

8 Meeting

8.1 Start meeting

start_meeting <meeting_id>	Command to CCU

Start a meeting in the CCU. Responded with **’meeting_started’** if a meeting is started.

<meeting_id>	The meeting id can be requested with the command **‘meeting_status’**

8.1.1 Example

```
//Start meeting without an ID:
< start_meeting
> command_error syntax error

//Start a meeting with an invalid ID
< start_meeting 29

//Start a meeting without an active meeting controller
< start_meeting 37
> meeting_stopped
> meeting_status_done

//Start a valid meeting
< start_meeting 17
> meeting_started 17
```

8.2 Meeting started

meeting_started <meeting_id>	Message from CCU

A meeting has been started in the CCU.

<meeting_id>	The meeting id can be requested with the command **‘meeting_status’**

8.2.1 Example

```
//Start a valid meeting from TPCI
< start_meeting 17
> meeting_started 17

//A meeting is started from some other source
> meeting_started 17

//Meeting status is requested
> meeting_status
 …
< meeting_started 17
< meeting_status_done
```

8.3 Stop meeting

stop_meeting	Command to CCU

Stop a meeting in the CCU. Responded with **’meeting_stopped’** if a meeting was running when the command was received by the CCU

8.3.1 Example

```
< stop_meeting
> meeting_stopped
```

8.4 Meeting stopped

meeting_stopped Message from CCU

A meeting has been stopped in the CCU

8.4.1 Example

```
//Meeting stopped
> meeting_stopped

//Request meeting status
> meeting_status
 …
< meeting_stopped
< meeting_status_done
```

8.5 Meeting status

meeting_status Command to CCU

Request the meeting status. The result is the meeting run state and a list of available meetings. The response is completed with ***‘meeting_status_done’***.

```
meeting <meeting_id> <meeting_title>
meeting <meeting_id> <meeting_title>
meeting <meeting_id> <meeting_title>
…
meeting <meeting_id> <meeting_title>
meeting_stopped or meeting_started <meeting_id>
meeting_status_done
```

<meeting_id>	Is an integer identifying the meeting
<meeting_title>	Is a name for the meeting

Important: Meeting commands are only usable in connection with SW6000

9 Speech Time Alarm

9.1 Speech time alarm

`mic_speech_time_alarm <seat_no> <alarm_status>` Message from CCU

Sent from CCU to indicate current alarm status of speech time for an open microphone. When a microphone is closed, alarm state is always reverted to value 0. The mic speech time alarm status is included in the **`‘mic_status’`**.

<seat_no>		The seat number
<alarm_status>	0	No alarm or expiration; issued only after another value was indicated in a previous ‘mic_speech_time’ for this seat, and then the microphone was switched off
	1	Alarm (if there was one set, otherwise this value is skipped).
	2	Time expired.

9.1.1 Example

```
-> mic_on 3                       //Speech time alarm state implicitly becomes 0
-> mic_speech_time_alarm 3 1      //Alarm
-> mic_speech_time_alarm 3 0      //No alarm or expiration
-> mic_speech_time_alarm 3 2      //Time expired
-> mic_off 3                      //Speech time alarm state implicitly becomes 0
```

Important: Speech time alarm commands are only usable in connection with SW6000

10 Namesign Content Control

When changes in the Name sign screen are made in the CGUI, the corresponding TCPI commands are reported in the protocol when 'Apply content' is activated.

When commands are executed externally the command are executed in the CCU immediately.

The **'sign_status'** command lists the actual settings which are applied and not what is selected in the CGUI.

10.1 Sign mode

sign_mode <mode> Command to CCU and message from CCU

Command to CCU (set's and applies the mode) and indication from CCU when 'Apply content' in the CCU is activated. The 'sign_mode' command is included in response to **'sign_status'.**

<mode>	The CCU sign mode. Values: 'participants', 'text', 'identify' and 'clear'

10.2 Sign font size change

sign_font_size_change <pt_change> Command to CCU and message from CCU

Set/retrieve the 'Font size change' value to be used/is used in modes: Participants and texts.

The 'Font size change' value will be rounded to the nearest valid value. Command to CCU (set's and applies the 'Font size change'). Indication from CCU when 'Apply content' in the CCU is activated. The 'pt_change' command is also included in response to **'sign_status'**.

<pt_change>	Pt size change values for the text in the sign. Values: 190 to -70 in 10 step.

10.2.1 Example

```
<- sign_font_size_change 22
-> sign_font_size_change 20
<- sign_font_size_change 117
-> sign_font_size_change 120
<- sign_font_size_change 47
-> sign_font_size_change 50

<- sign_status
-> ...
-> sign_font_size_change 50
-> ...
-> sign_status_done
```

10.3 Sign seat

sign_seat <serial_number> <primary_seat> <secondary_seat> Command to CCU and message from CCU

Assign seat(s) to a sign or clear seat assignment. Command to CCU (set's and applies the seat assignment). Indication from CCU when 'Apply content' in the CCU is activated. The 'sign_seat' command is included in response to **'sign_status'**.

<serial_number>	The serial number of the sign to assign seats to
<primary_seat>	The primary seat assignment. Values: 1 to 65551 (optional)[1]
<secondary_seat>	The secondary seat assignment. Values: 1 to 65551 (optional)[2]

1. If no seats are given in this command, the seat assignments are cleared for the specified sign
2. It is only possible to specify a secondary seat if a primary seat is specified

10.3.1 Example

```
<- sign_seat 123.456.789
-> sign_seat 123.456.789
<- sign_seat 123.456.789 2
-> sign_seat 123.456.789 2
<- sign_seat 123.456.789 4 2
-> sign_seat 123.456.789 4 2

<- sign_status
-> ...
-> sign_seat 123.456.789 4 2
-> - ...
-> sign_status_done
```

10.4 Sign text

sign_text <serial_number> <first_line> <second_line> Command to CCU and message from CCU

Assign texts to a sign or clear text assignment. Command to CCU (set's and applies the sign text) and indication from CCU when 'Apply content' in the CCU is activated. The 'sign_text' command is included in response to **'sign_status'**.

<serial_number>	The serial number of the sign to assign seats to
<first_line	The first text line. Values: Alpha-numerical. The text must be surrounded with " (double quote). (Optional)[1]
<second_line	The second text line. Values: Alpha-numerical. The text must be surrounded with " (double quote). (Optional)[2]

1. If no texts are given in this command, the text assignments are cleared for the specified sign
2. It is only possible to specify a second text line if a primary text line is specified

10.4.1 Example

```
<- sign_text 000.112.911
-> sign_text 000.112.911
<- sign_text 123.456.789 "John Jones " "William Francis"
-> sign_text 123.456.789 "John Jones " "William Francis"
<- sign_text 125.012.345 "Peter Wake" "Wake Inc."
-> sign_text 125.012.345 "Peter Wake" "Wake Inc."
```

10.5 Sign status

sign_status Command to CCU

Request the status for sign settings/selections in the CCU-CGUI. The result is sign mode state and a list of sign settings. The response is completed with **'meeting_status_done'**.

```
sign_mode <mode>
sign_font_size_change <pt_change>
sign_seat <serial_number> <primary_seat> <secondary_seat>
sign_text <serial_number> <first_line> <second_line>
…
```

sign_status_done

10.5.1 Example

```
<- sign_status
-> sign_mode text
-> sign_font_size_change 50
-> sign_seat 123.456.789 1 2
-> sign_text 123.456.789 "John Jones " "William Francis"
-> ...
-> sign_seat 125.012.345 7
-> sign_text 125.012.345 "Peter Wake" "Wake Inc."
-> ...
-> sign_status_done
```

Important: Name sign commands are only usable in a stand-alone system, where SW6000 in not in use as the Name sign screen in the DIS-CCU web application is disables when SW6000 is in use.

Any sign command given when SW6000 is connected, will be executed, when SW6000 disconnect.

United States, Canada, Latin America, Caribbean:
Shure Incorporated
5800 West Touhy Avenue
Niles, IL 60714-4608
USA

Phone: +1 847 600 2000
Fax: +1 847 600 1212 (USA)
Fax: +1 847 600 6446
Email: info@shure.com

Europe, Middle East, Africa:

Shure Europe Gmbh
Jakob-Dieffenbacher-Str. 12
75031 Eppingen
Germany

Phone: +49 (0) 7262-9249-100
Fax: +49 (0) 7262-9249-114
Email: info@shure.de

Asia, Pacific:

Shure Asia Limited
22/F, 625 King's Road
North Point, Island East,
Hong Kong

Phone: (+852) 2893-4290
Fax: (+852) 2893-4055
Email: info@shure.com.hk

www.shure.com