Yamaha CS-700
Integrators Reference Guide

Revision 5.2
3/24/2020

Yamaha Unified Communications
144 North Road, Suite 3250
Sudbury, MA 01776
978-610-4040

Table of Contents

1 Overview

The Yamaha CS-700 provides standard USB 3.0 and USB 2.0 communication links to a host processor (PC). It also provides an IP interface for management and VoIP calling. This guide describes the interfaces that hosts and external clients can use to manage and monitor the CS-700.

Clients can communicate with the device through these interface mechanisms:

- USB
  - USB HID (Human Interface Device)
  - USB Bulk
- IP
  - Telnet
  - SSH
  - Provisioning per DHCP option 66 or 150 over HTTP, TFTP or FTP
  - SNMP

For the USB interface we provide a C library to enable a host application to easily integrate with the CS-700.
The guide is organized as follows:

- **USB Description** – Details the USB interfaces to the host, including the hub, audio and camera components.
- **IP Interface** – Describes the IP interface that can be used to create applications to manage and control the device over IP.
- **Library Interface** – Describes the C library interface that can be used to create applications to manage and control the device over USB.
- **API Command Reference** – Provide details of the API commands and notifications that can be communicated between the host and device via the library, and the IP interface.
- **Provisioning** – Describes how to provision the device via an IP connection using DHCP options 66 and 150.
- **SNMP** – Describes SNMP support and the corresponding MIB for querying device status and for receiving traps.

2 USB Description

When the CS-700 USB interface is connected to a host PC, four components of the CS-700 will enumerate with the host:

- USB HUB - provides both 2.0 and 3.0 USB connectivity
- USB Audio - speaker and microphone, USB 2.0
- USB Camera - supports USB 3.0 and 2.0

The host processor connects directly to the HUB chip. The HUB chip then connects to the other USB components in the system as shown in the diagram below:

$$\begin{tikzcd}
\text{Host} \arrow[r, leftrightarrow] & \text{HUB} \arrow[r, leftrightarrow] \arrow[dr, leftrightarrow] & \text{Audio} \\
 & & \text{Camera}
\end{tikzcd}$$

The section below lists the CS-700 USB VIDs and PIDs, and the sections that follow describe each of the device's USB interfaces.

2.1 VID and PID

The CS-700 Vendor IDs (VID) and Product IDs (PID) for each component are shown below.

Component	VID	PID	Comment
HUB USB v3.1	0x0499	0x4031	USB v3.1 port
HUB USB v2.0	0x0499	0x4033	USB v2.0 port
Audio	0x0499	0x4030	Audio Class
Camera	0x0499	0x4032	Video Class

2.2 Interface Definitions for the USB HUB Component

The USB HUB component will enumerate to the host as a Universal Serial Bus controller. It will be displayed within the Host's Device Manager as two devices: a USB 3.0 Certified Hub and a USB 2.0 Certified Hub. Once the HUB chip enumerates with the Host Processor, it will initiate the enumeration of the other Cannonball components.

The Hub will enumerate as follows:

- Universal Serial Bus Controller, USB 2.0 MTT Hub
- Universal Serial Bus Controller, USB 3.0 Hub

2.3 Interface Definitions for the Audio Component

The USB Audio chip will enumerate the following interfaces:

- Interface 0; Audio Control: ep0, input type (0x101), output type (0x405), bma-control (3)
- Interface 1; Audio Output: Streaming, ep1 (OUT), ISOC- async, 2 channel, 48 KHz, 16/24 bit, 192 bytes
- Interface 2; Audio Input: Streaming, ep2 (IN), ISOC- sync, 2 channel, 48 KHz, 16/24 bit, 288 bytes
- Interface 3; HID: ep4 (IN), ep5 (OUT), 6 Report IDs:
    - ID 1: Input, Consumer Page, Volume +/-, size 1, count 2
    - ID 2: Input, Custom Call Control Settings Page, size 8, count 1
    - ID 3: Output, Telephony Page, (Mute, Hook, Flash Answer), size 1, count 5
    - ID 4: IN/OUT, Custom Communications Page, size 8, count 31 (IN); 8, count 1 (OUT)
    - ID 5: IN/OUT, Skype Reports, (Mute, Hook, Flash) IN; (Mute, Hook, Ring, Hold) OUT
    - ID 6: OUT, Custom Audio Settings, (Room Properties), size 8, count 7 (OUT)
- Interface 4; BULK: ep7 (IN), ep8 (OUT), 64 bytes (Upgrade)

2.3.1 Audio Report

Description	Endpoint/ Report ID	Report Value	Notes
Mic Mute States			
Mic Mute	ep0 / 0x06	0x00	Control Transfer from Host
Mic Unmute	ep0 / 0x06	0x01	Control Transfer from Host
Speaker Volume Gain			
Speaker volume	ep0 / 0x02	value	To CS-700 from Host
Speaker Volume Control			
Volume Up	ep4 / 0x01	0x01	Host responds with Control Transfer speaker gain value on ep0 (see above)
Volume Down	ep4 / 0x01	0x02	Host responds with Control Transfer speaker gain value on ep0 (see above)
Mic Mute States			
Mute	ep4 / 0x02	0x02	CS-700 VoIP App to Host
Unmute	ep4 / 0x02	0x00	CS-700 VoIP App to Host
Call Control			
Off Hook (answer)	ep4 / 0x02	0x04	
On Hook (hang up)	ep4 / 0x02	0x01	
Hold	ep4 / 0x02	0x05	
Resume	ep4 / 0x02	0x06	
System Control			
Restart System	ep4 / 0x02	0x03	Not supported

Description	Endpoint/ Report ID	Report Value	Notes
Mic Mute States			
Mute	ep5 / 0x04	0x08	Host Application to CS-700
Unmute	ep5 / 0x04	0x09	Host Application to CS-700
Call States			
Ringing (incoming call)	ep5 / 0x04	0x05	
In Progress	ep5 / 0x04	0x04	
On Hold	ep5 / 0x04	0x07	
Ended	ep5 / 0x04	0x06	

Description	Report ID	Report Value	Notes
Telephony Page	0x03	-	(Currently unused in Cannonball)
Skype	0x05	-	See skype specification for details
CS-700 Audio Parameters	0x06	-	Room Property Parameters (4) #People, EQ, MicHP, Reverb (Currently unused in Cannonball)

2.3.2 Speaker Volume Synchronization

The speaker volume can be controlled from the device by using the volume up/down cap-touch buttons on the base unit.

The speaker volume range is 1 to 18. The host range is greater. In order to sync volume with the host, the device range is mapped to the host range. When the volume is changed from the device, the device sends one or more volume up/down reports to the host to achieve the best match based on the mapping. Likewise, when the host sends to the device a speaker volume that the user selected on the host, the device maps that setting to a value in the 1 to 18 range. See diagram below.

Device		OS		Application
Volume Button Press	→	Adjust Volume		
*Map Gain to Index	←	Send Gain		
				Adjust Volume on App
		Adjust Volume	←	Send Gain to OS
*Map Gain to Index	←	Send Gain		

* Appendix A indicates the speaker volume mappings.

2.4 Interface Definitions for the Camera Component

The USB Camera interface will conform to USB Device Class Definition for Video Devices, version 1.5.

The USB Camera chip will enumerate the following interfaces:

USB 2.0
- Interface 0; BULK: ep1 (in/out), 512 bit packet size

USB 3.0
- Interface 0; Video Control: Interrupt, ep2 (IN), 1024 bit packet size
- Interface 1; Video Streaming: BULK, ep3 (IN), 1024 bit packet size

2.4.1 Video Configuration and Status

The video controls for pan, tilt and zoom (PTZ) are implemented digitally in the camera.

The video controls for brightness, contrast, sharpness, saturation, balance, backlight compensation and gain are implemented in the camera image processor.

The host will use GET_CUR and SET_CUR requests to set the desired parameter. When the camera is enumerated, the VC_INPUT_TERMINAL descriptor and VC_PROCESSING_UNIT descriptor will set the appropriate flags for the parameters is supports. It will also define an endpoint for interrupt support to report local changes back to the host.

Description	ID	Report Value	Notes
Video Control ID	**ep0/4**		
Zoom	0x0B	value	Range: 100->400
PanTilt	0x0D	value	Range: +/- 30, Range: +/- 18
Roll	0x0F	value	Not Used
Exposure	0x04	value	Not Used
Privacy	0x11	value	Not Used
Video Processing ID	**ep0/2**		
Backlight	0x01	value	Range: 0->5
Brightness	0x02	value	Range: 0->250
Contrast	0x03	value	Range: 60->140
Saturation	0x07	value	Range: 50->150
Sharpness	0x08	value	Range: 0->255
Hue	0x06	value	Range: 0->180
Gamma	0x09	value	Range: 1->255
Flicker	0x05	value	1-50Hz, 2-60Hz

2.4.2 Video Stream

Skype video specifications* specify the following:

- USB 2.0
  - 640 x 480 , 30 fps, YUY2 color space
  - 640 x 360 , 30 fps, YUY2 color space

- USB 3.0
  - 1920 x 1080 , 30 fps, YUY2/NV12 color space
  - 1280 x 720 , 30 fps, YUY2/NV12 color space
  - 640 x 360 , 30 fps, YUY2/NV12 color space

Note: The CS-700 video camera only supports uncompressed video formats, which limits the resolution in USB 2.0.

* Skype 2.0: Skype and Lync Video Capture Specification (Doc # H100693)
Skype 3.0: Skype for Business Video Capture Specification (Doc # H100693)

3 IP Interface

Integrators wanting to communicate programmatically with the CS-700 for device management or operational control may use the CS-700 IP Management Interface. Use of the Yamaha Dialer Application that runs on an Android based tablet is described in the *CS-700 Operations Guide*.

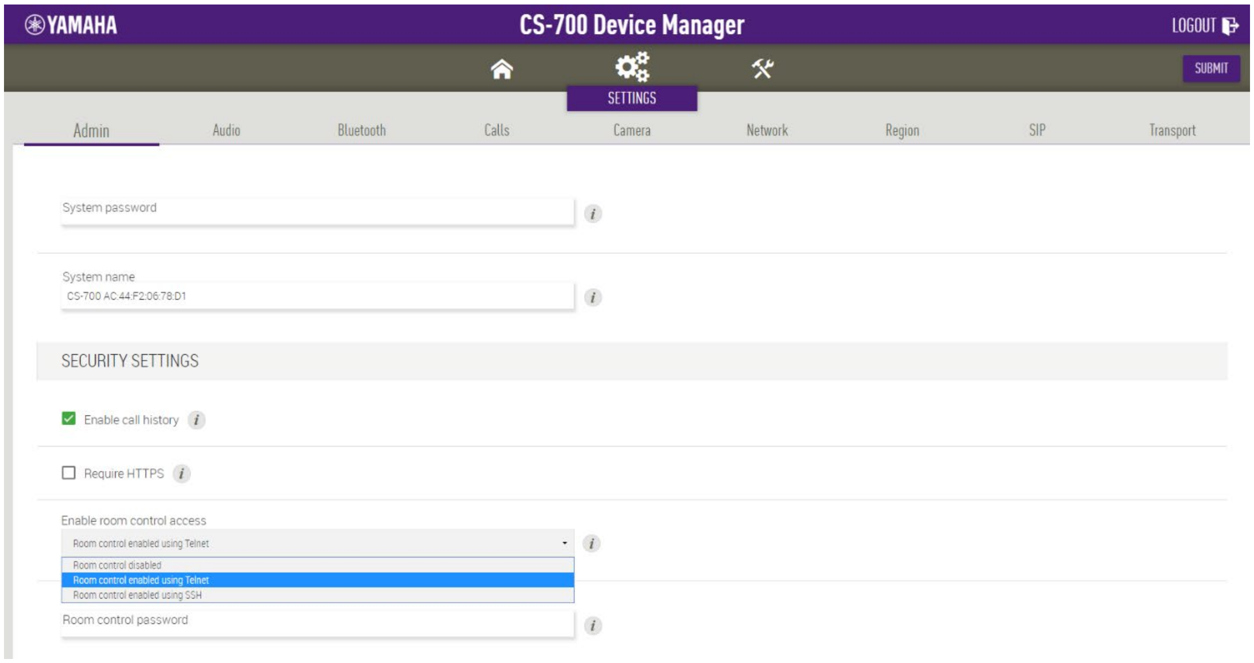
The interface supports these features:

- Connect to a device via IP
- Transmit API commands
- Receive API responses and asynchronous notifications

The API commands are described in section 5, **API Command Reference**.

If the host can access the device via an IP connection, then the client can communicate via the IP-supported commands.

To use an IP connection for the control, with the CS-700 being the server, the "Enable room control access" control on the Admin Settings page has to be set to either "Room control enabled using Telnet" or to "Room control enabled using SSH". This IP connection does not have to be enabled to use the Yamaha Dialer Application that runs on an Android based tablet.



System password

System name
CS-700 AC:44:F2:06:78:D1

SECURITY SETTINGS

☑ Enable call history

☐ Require HTTPS

Enable room control access
Room control enabled using Telnet
- Room control disabled
- Room control enabled using Telnet
- Room control enabled using SSH

Room control password

To start the IP connection, provide the IP address, port number, user name and password in a telnet or a SSH client to establish the connection. The login username is **roomcontrol** and the password is **Yamaha-CS-700**, both case sensitive. The port number is 23 for Telnet, and 22 for SSH. If it is required to see the API commands echoed back to the client, execute "set echo 1" command after each successful login.

echo

Description: This will make sure the users of the CLI can see what they are typing by allowing commands to be echoed back.

Property Actions: set

Command Definition:

Action	Definition
set	`set echo <0|1>`

Parameters:

Parameter	Description
0	Turn off screen echo
1	Turn on screen echo

CLI Format Examples:

```
set echo 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

4 Library Interface

Integrators wanting to communicate programmatically with the CS-700 for device management or operational control will find it most convenient to use the CS-700 Management Library. This is a C library that is available for Windows, and MacOS. It enables developers to access the device using higher-level functions without needing to directly manage the low-level USB HID or bulk interface.

The library supports these features:

- Connect to a device via USB
- Transmit API commands
- Receive API responses and asynchronous notifications
- Upgrade firmware
- Download logs
- Import and export configuration files

The API commands are described in section 5, **API Command Reference**.

If the host is connected directly to the device via USB, then a client application can communicate via the USB interface.

The Library interface is described below.

4.1 USB Library Reference

This section describes the CS-700 Management Library's C interface. The header file is distributed with the CS-700 SDK.

CsDevComm_Mode enum

Description: Enumeration of USB connections type. The client should specify the connection type, Auto, HID, or Bulk, at the start of a session. Auto will select Bulk if available otherwise HID. Bulk is preferable when upgrading firmware.

Syntax:

```
typedef enum {
	CsDevComm_Mode_Auto = 0,
	CsDevComm_Mode_Hid,
	CsDevComm_Mode_Bulk,
	CsDevComm_Mode_Num
} CsDevComm_Mode;
```

devAttach

Description: Establish USB communication with a device.

Syntax:

```
EXPORT int __cdecl devAttach(int mode, void(*debug_callback)(char *msg),
                                  void(*event_callback)(int type));
```

Parameters:

Parameter	Description
Mode	The USB connection mode, Auto, HID or Bulk, chosen from enum CsDevComm_Mode.
debug_callback	Pointer to callback function to handle debug messages.
Msg	Pointer to debug message.
event_callback	Pointer to callback function to handle events.
Type	Type of event as listed in table below.

Return Values:

Value	Description
-1	General error
0	Success
1	No device connected
2	Failed authentication

Event Types:

Type	Description
0xFF	Device disconnected

devDetach

Description: Disconnect session.

Syntax:

```
EXPORT void __cdecl devDetach(void);
```

Return Values:

Value	Description
-1	General error
0	Success

isConnected

Description: Get the USB connection status.

Syntax:

```
EXPORT int __cdecl isConnected(void);
```

Return Values:

Value	Description
-1	General error
0	Not connected
1	Connected HID
2	Conned Bulk

coreCliCmd

Description: Send a CLI-format command to the device. The commands are defined in in section 5, API Command Reference.

Syntax:

```
EXPORT int __cdecl coreCliCmd(char *cmd, char *rsp, int len);
```

Parameters:

Parameter	Description
cmd	CLI command string (see API Command Reference)

rsp	Pointer to response string
len	Bytes available in response string

Return Values:

Value	Description
-1	General failure
0	Success

notifCtrl

Description: Enable/disable notifications and register a callback function to handle CLI-format notifications. The notifications are defined in section 5, API Command Reference. Notifications are automatically disabled if device is disconnected.

Syntax:

```
EXPORT int __cdecl notifCtrl(int mode, void(*notif_callback_t)(char *msg));
```

Parameters:

Parameter	Description
mode	Enable/disable notifications
notif_callback_t	Pointer to callback function to handle CLI notification
msg	Callback function parameter: Notification message string (see API Command Reference)

Return Values:

Value	Description
-1	General failure
0	Success

updateFirmware

Description: Update firmware on the CS-700 device. The host PC must have access to a proper firmware bundle.

Syntax:

```
EXPORT int __cdecl updateFirmware(char *bundle, void(*update_firmware_callback_t)(int
nPercent));
```

Parameters:

Parameter	Description
bundle	Complete file path of upgrade bundle
update_firmware_callback_t	Pointer to callback function to handle upgrade progress notifications

nPercent	Callback function parameter: Percent completion. A value of -1 indicates an upgrade error.

Return Values:

Value	Description
-1	General failure
0	Success

downloadLogs

Description: Download a zipped log file to the specified directory.

Syntax:

```
EXPORT int __cdecl downloadLogs(char *filepath, void(*download_logs_callback_t)(int
nPercent, char *logname));
```

Parameters:

Parameter	Description
filepath	Directory on host to which the log file will be saved
download_logs_callback_t	Pointer to callback function to handle download progress notifications
nPercent	Callback function parameter: Percent completion. A value of -1 indicates a download error.
logname	Returned name of downloaded log file

Return Values:

Value	Description
-1	General failure
0	Success

importConfig

Description: Import a configuration XML file to the CS-700 device. The host PC must have access to a proper configuration file. The output of this is the zipped config file after completion.

Syntax:

```
EXPORT int __cdecl importConfig(char *configFile, void(*import_config_callback_t)(int
nPercent, char *filepath));
```

Parameters:

Parameter	Description
configFile	Complete file path of config file

import_config_ callback_t	Pointer to callback function to handle import progress notifications
nPercent	Callback function parameter: Percent completion. A value of -1 indicates an import error.
filepath	Callback function parameter: Returned name of downloaded config file.

Return Values:

Value	Description
-1	General failure
0	Success

exportConfig

Description: Export the CS-700 configuration settings to an XML file on the host.

Syntax:

```
EXPORT int __cdecl exportConfig(char *filepath,
void(*export_config_callback_t)(int nPercent, char *filepath));
```

Parameters:

Parameter	Description
filepath	Directory on host to which the config file will be saved
export_config_ callback_t	Pointer to callback function to handle export progress notifications
nPercent	Callback function parameter: Percent completion. A value of -1 indicates an export error.
filepath	Callback function parameter: Returned name of downloaded config file.

Return Values:

Value	Description
-1	General failure
0	Success

5 API Command Reference

This section describes the API commands and notifications that can be transmitted between a device and host using the IP interface and the library interface defined in section 3 and section 4 above.

5.1 Overview

The API is organized by functional Category, such as Audio and Camera, and within each category are listed the properties, status items and commands that are available for that category. Notifications are identified with their corresponding property or status item.

The table below describes the command syntax.

<parameter>	Items in angle brackets are parameters
<"call-status">	A quoted item represents the name of a parameter further described in the Parameters section.
<0|1|2>	A list of possible values are separated by vertical bars.
<connected|disconnected>	An unquoted parameter is a literal value.
<0..10>	A range of values is separated by an ellipsis.
<"paired"> [<"mac"> <"name">]	Square brackets designate optional parameters.
[<"mac"> <"name">]+	A list of repeating values is designated by a plus sign.

For properties, the description lists the actions that can be performed on the property, including:

get – property can be read; this is true of all properties
set – property can be set
notify – when the property changes, a notification is sent to registered clients

The Command Definition and Parameters tables describe each supported action and associated parameters and values.

5.2 Categories

The API is organized by these Categories:

- System (sys)
- Audio
- Camera
- Bluetooth (bt)
- Bluetooth Low Energy (ble)
- Call
- Network (net)
- VoIP

The sections below describe the interfaces supported in each category.

Changes to properties in the Network group must be applied by the net-commit command. See Appendix B – Crestron Integration TCP/IP for re-registration sequence with the SIP server.
Changes to properties in the VoIP group must be applied by the voip-commit command. See Appendix B – Crestron Integration TCP/IP for re-registration sequence with the SIP server.

5.2.1 System

5.2.1.1 System Properties

product

Description: Product name. Read-only set at manufacturing.

Property Actions: get

Command Definition:

Action	Definition
get	`get product`
response	`val product <"name">`

Parameters:

Parameter	Description
`name`	String. Product name. Valid values are CS-700 and CS-700-SP

CLI Format Examples:

```
get product
val product CS-700
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

base-sernum

Description: Query product serial number. Read-only set at manufacturing.

Property Actions: get

Command Definition:

Action	Definition
get	`get base-sernum`
response	`val base-sernum <"number">`

Parameters:

Parameter	Description
`number`	String. Device serial number.

CLI Format Examples:

```
get base-sernum
val base-sernum CSS701000062
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

voip-capable

Description: Is VoIP supported on the device. This capability is set at the factory and cannot be changed.

Property Actions: get

Command Definition:

Action	Definition
get	`get voip-capable`
response	`val voip-capable <0|1>`

Parameters:

Parameter	Description
0	Device does not support VoIP
1	Device does support VoIP

CLI Format Examples:

```
get voip-capable
val voip-capable 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

base-ver

Description: Query product version.

Property Actions: get

Command Definition:

Action	Definition
get	`get base-ver`
response	`val base-ver <"version">`

Parameters:

Parameter	Description
`version`	String. Firmware bundle version.

CLI Format Examples:

```
get base-ver
val base-ver 1.1.0.212
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

systemname

Description: Specifies the system name.

Property Actions: set, get, notify

Default Value: Product name concatenated with the MAC address

Command Definition:

Action	Definition
get	`get systemname`
response	`val systemname <"system-name">`
set	`set systemname <"system-name">`
notify	`notify sys.systemname <"system-name">`

Parameters:

Parameter	Description
`systemname`	System name string

CLI Format Examples:

```
get systemname
val systemname CS-700 AC:44:F2:11:22:33
```

```
set systemname CS-700-MAINCONF
```

```
notify sys.systemname CS-700-MAINCONF
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

md5-password

Description: Administrator password for the device, stored as MD5 sum.

Property Actions: set

Default Value: 5735c3a7aa6ffcfe6ab123835584db75 (7386 as MD5 sum)

Command Definition:

Action	Definition
set	`set md5-password <"password">`
notify	`notify sys.md5-password <"password">`

Parameters:

Parameter	Description
`password`	Password string as MD5 sum (32 hexadecimal numbers)

CLI Format Examples:

```
set md5-password 5735c3a7aa6ffcfe6ab123835584db75
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

enable-btn-camera

Description: Enable/disable the camera button on the main unit.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get enable-btn-camera`
response	`val enable-btn-camera <0|1>`
set	`set enable-btn-camera <0|1>`
notify	`notify sys.enable-btn-camera <0|1>`

Parameters:

Parameter	Description
0	Disable button
1	Enable button

CLI Format Examples:

```
get enable-btn-camera
val enable-btn-camera 1
```

```
set enable-btn-camera 1
```

```
notify sys.enable-btn-camera 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

enable-btn-audio

Description: Enable/disable the microphone mute button on the main unit.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get enable-btn-audio`
response	`val enable-btn-audio <0|1>`
set	`set enable-btn-audio <0|1>`
notify	`notify sys.enable-btn-audio <0|1>`

Parameters:

Parameter	Description
0	Disable button
1	Enable button

CLI Format Examples:

```
get enable-btn-audio
val enable-btn-audio 1
```

```
set enable-btn-audio 1
```

```
notify sys.enable-btn-audio 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

enable-btn-volume

Description: Enable/disable the speaker volume buttons on the main unit.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get enable-btn-volume`
response	`val enable-btn-volume <0|1>`
set	`set enable-btn-volume <0|1>`
notify	`notify sys.enable-btn-volume <0|1>`

Parameters:

Parameter	Description
0	Disable button
1	Enable button

CLI Format Examples:

```
get enable-btn-volume
val enable-btn-volume 1
```

```
set enable-btn-volume 1
```

```
notify sys.enable-btn-volume 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

enable-btn-bluetooth

Description: Enable/disable the Bluetooth button on the main unit.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get enable-btn-bluetooth`
response	`val enable-btn-bluetooth <0|1>`
set	`set enable-btn-bluetooth <0|1>`
notify	`notify sys.enable-btn-bluetooth <0|1>`

Parameters:

Parameter	Description
0	Disable button
1	Enable button

CLI Format Examples:

```
get enable-btn-bluetooth
val enable-btn-bluetooth 1
```

```
set enable-btn-bluetooth 1
```

```
notify sys.enable-btn-bluetooth 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

enable-led-call

Description: Enable/disable the call state LED on the main unit.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get enable-led-call`

response	`val enable-led-call <0|1>`
set	`set enable-led-call <0|1>`
notify	`notify sys.enable-led-call <0|1>`

Parameters:

Parameter	Description
0	Disable LED
1	Enable LED

CLI Format Examples:

```
get enable-led-call
val enable-led-call 1
```

```
set enable-led-call 1
```

```
notify sys.enable-led-call 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

region

Description: Region in which device is operating, setting by index.

Property Actions: set, get, notify

Default Value: 23

Command Definition:

Action	Definition
get	`get region`
response	`val region <1..24>`
set	`set region <1..24>`
notify	`notify sys.region <1..24>`

Parameters:

Parameter	Description
1	Argentina
2	Australia
3	Belgium
4	Brazil
5	Canada
6	Chile
7	China
8	Costa Rica
9	France

10	Germany
11	Hong Kong
12	India
13	Israel
14	Italy
15	Japan
16	Malaysia
17	Mexico
18	New Zealand
19	Singapore
20	South Africa
21	Taiwan
22	United Kingdom
23	USA
24	Venezuela

CLI Format Examples:

```
get region
val region 23
```

```
set region 23
```

```
notify sys.region 23
```

Supported Products: CS700-SP
Available in API Version: 1.1

recent-call-enabled

Description: Enables or disable the Recent Calls list.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get recent-call-enabled`
response	`val recent-call-enabled <0|1>`
set	`set recent-call-enabled <0|1>`
notify	`notify sys.recent-call-enabled <0|1>`

Parameters:

Parameter	Description
0	Disable recent call list
1	Enable recent call list

CLI Format Examples:

```
get recent-call-enabled
val recent-call-enabled 1
```

```
set recent-call-enabled 1
```

```
notify sys.recent-call-enabled 1
```

Supported Products: CS700-SP
Available in API Version: 1.1

require-https

Description: –Specify if HTTPS is required on Web UI login.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get require-https`
response	`val require-https <0|1>`
set	`set require-https <0|1>`
notify	`notify sys.require-https <0|1>`

Parameters:

Parameter	Description
0	Do not require HTTPS
1	Require HTTPS

CLI Format Examples:

```
get require-https
val require-https 0
```

```
set require-https 0
```

```
notify sys.require-https 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

power-saving-mode

Description: Configure power saving mode setting.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get power-saving-mode`
response	`val power-saving-mode <0|1>`
set	`set power-saving-mode <0|1>`
notify	`notify sys.power-saving-mode <0|1>`

Parameters:

Parameter	Description
0	Power save mode is off
1	Power save mode is on

CLI Format Examples:

```
get power-saving-mode
val power-saving-mode 1
```

```
set power-saving-mode 1
```

```
notify sys.power-saving-mode 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

power-saving-time

Description: Configure power saving time in minutes.

Property Actions: set, get, notify

Default Value: 20

Command Definition:

Action	Definition
get	`get power-saving-time`
response	`val power-saving-time <0..480>`
set	`set power-saving-time <0..480>`
notify	`notify sys.power-saving-time <0..480>`

Parameters:

Parameter	Description
`timeout`	Power save timeout in minutes

CLI Format Examples:

```
get power-saving-time
val power-saving-time 20
```

```
set power-saving-time 20
```

```
notify sys.power-saving-time 20
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

auto-deploy

Description:	Enable auto provisioning server.
Property Actions:	set, get, notify
Default Value:	1

Command Definition:

Action	Definition
get	`get auto-deploy`
response	`val auto-deploy <0|1>`
set	`set auto-deploy <0|1>`
notify	`notify sys.auto-deploy <0|1>`

Parameters:

Parameter	Description
0	Disable auto-discovery of provisioning server
1	Enable auto-discovery of provisioning server

CLI Format Examples:

```
get auto-deploy
val auto-deploy 1
```

```
set auto-deploy 1
```

```
notify sys.auto-deploy 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

enable-deploy

Description:	Enable provisioning server.
Property Actions:	set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get enable-deploy`
response	`val enable-deploy <0|1>`
set	`set enable-deploy <0|1>`
notify	`notify sys.enable-deploy <0|1>`

Parameters:

Parameter	Description
0	Disable provisioning
1	Enable provisioning

CLI Format Examples:

```
get enable-deploy
val enable-deploy 1
```

```
set enable-deploy 1
```

```
notify sys.enable-deploy 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

provisioning-interval

Description: Specify the provisioning interval for the device, in minutes.

Property Actions: set, get, notify

Default Value: 1440 (1 day)

Command Definition:

Action	Definition
get	`get provisioning-interval`
response	`val provisioning-interval <1..44640>`
set	`set provisioning-interval <1..44640>`
notify	`notify sys.provisioning-interval <1..44640>`

Parameters:

Parameter	Description
1..44640	Provisioning interval in minutes

CLI Format Examples:

```
get provisioning-interval
val provisioning-interval 1440
```

```
set provisioning-interval 1440
```

```
notify sys.provisioning-interval 1440
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

snmp-enable

Description: Enable or disable SNMP support.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get snmp-enable`
response	`val snmp-enable <0|1>`
set	`set snmp-enable <0|1>`
notify	`notify sys.snmp-enable <0|1>`

Parameters:

Parameter	Description
0	Disable SNMP
1	Enable SNMP

CLI Format Examples:

```
get snmp-enable
val snmp-enable 1
```

```
set snmp-enable 1
```

```
notify sys.snmp-enable 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

snmp-community

Description: Specifies the SNMP read-only community string used for queries from the server and transmitted traps. Read-only indicates the authorization level. The device does not support write operations initiated through SNMP.

Property Actions: set, get, notify

Default Value: public

Command Definition:

Action	Definition
get	`get snmp-community`
response	`val snmp-community <"community">`
set	`set snmp-community <"community">`
notify	`notify sys.snmp-community <"community">`

Parameters:

Parameter	Description
`community`	Read-only community string

CLI Format Examples:

```
get snmp-community
val snmp-community it-support
```

```
set snmp-community it-support
```

```
notify sys.snmp-community it-support
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

snmp-contact-name

Description: Specifies the contact name, typically the system administrator. This string is informational and can include an email address. It is not associated with traps.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get snmp-contact-name`
response	`val snmp-contact-name <"contact">`
set	`set snmp-contact-name <"contact">`
notify	`notify sys.snmp-contact-name <"contact">`

Parameters:

Parameter	Description
`contact`	String. Contact name.

CLI Format Examples:

```
get snmp-contact-name
val snmp-contact-name "Jessica Taylor <jtaylor@abc.com>"
```

```
set snmp-contact-name Jessica Taylor <jtaylor@abc.com>
```

```
notify sys.snmp-contact-name Jessica Taylor <jtaylor@abc.com>
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

snmp-device-location

Description: Specifies the location of the device for informational purposes.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get snmp-device-location`
response	`val snmp-device-location <"location">`
set	`set snmp-device-location <"location">`
notify	`notify sys.snmp-device-location <"location">`

Parameters:

Parameter	Description
`location`	String. Device location.

CLI Format Examples:

```
get snmp-device-location
val snmp-device-location "Bldg 12 Conf Room CB"
```

```
set snmp-device-location Bldg 12 Conf Room CB
```

```
notify sys.snmp-device-location Bldg 12 Conf Room CB
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

dialer-connection-mode

Description: Specifies the type of connection the tablet or dialer makes with the base.
Property Actions: set, get, notify

Default Value: disconnected

Command Definition:

Action	Definition
get	`get dialer-connection-mode`
response	`val dialer-connection-mode <ip|rc|disconnected>`
set	`set dialer-connection-mode <rc|disconnected>`

notify	`notify sys.dialer-connection-mode <ip|rc|disconnected>`

Parameters:

Parameter	Description
`ip`	Connected via IP dialer
`rc`	Connected via Room Control
`disconnected`	Dialer is not connected

The following table show the state of dialer-connection-mode after an action from the dialer or from the CLI.

Action	Current state disconnected	Current state ip	Current state rc
set dialer-connection-mode disconnected	No state change	IP dialer disconnection and new state is disconnected	New state is disconnected
set dialer-connection-mode rc	New state is rc	IP dialer disconnection; and new state is rc	No state change
Connect IP dialer	IP dialer becomes connected and new state is ip	No state change	No state change
Disconnect IP dialer	No state change	IP dialer disconnection and new state is disconnected	No state change

CLI Format Examples:

```
get dialer-connection-mode
val dialer-connection-mode ip
```

```
set dialer-connection-mode rc
```

```
notify sys.dialer-connection-mode rc
```

Supported Products: CS700-SP
Available in API Version: 1.1

snmp-address

Description: Specifies SNMP server address to which traps will be sent.

Property Actions: set, get, notify

Command Definition:

Action	Definition

get	`get snmp-address`
response	`val snmp-address <"address">`
set	`set snmp-address <"address">`
notify	`notify sys.snmp-address <"address">`

Parameters:

Parameter	Description
`address`	String. IP address or DNS name. Leave blank to disable traps.

CLI Format Examples:

```
get snmp-address
val snmp-address 200.200.210.152
```

```
set snmp-address 200.200.210.152
```

```
notify sys.snmp-address 200.200.210.152
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

verbose-log-enabled

Description:	Enable verbose logging.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get verbose-log-enabled`
response	`val verbose-log-enabled <0|1>`
set	`set verbose-log-enabled <0|1>`
notify	`notify sys.verbose-log-enabled <0|1>`

Parameters:

Parameter	Description
0	Disable verbose logging
1	Enable verbose logging

CLI Format Examples:

```
get verbose-log-enabled
val verbose-log-enabled 0
```

```
set verbose-log-enabled 0
```

```
notify sys.verbose-log-enabled 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

ui-mask

Description: Enable or disable menus in dialer via a bit mask.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get ui-mask`
response	`val ui-mask <"bit mask">`
set	`set ui-mask <"bit mask">`
notify	`notify sys.ui-mask <"bit mask">`

Parameters:

Parameter	Description
`bit mask`	
0	All menus are enabled in the dialer
1	For future use, value is ignored
2	For future use, value is ignored
4	Do not allow modifications of call forwarding
8	Do not allow modifications of call history
16	Do not allow modifications of contacts
32	Do not allow modifications of DND

CLI Format Examples:

```
get ui-mask
val ui-mask 0
```

// this prevents the dialer from modifying DND
```
set ui-mask 32
```

```
notify sys.ui-mask 32
```

// this prevents the dialer from modifying DND and call forwarding 32 + 4 = 36
```
set ui-mask 36
```

```
notify sys.ui-mask 36
```

Supported Products: CS700-SP
Available in API Version: 1.1

5.2.1.2 System Statuses

usb-conn-status

Description: Query USB connection status.

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get usb-conn-status`
response	`val usb-conn-status <0|1>`
notify	`notify sys.usb-conn-status <0|1>`

Parameters:

Parameter	Description
0	USB is not connected.
1	USB is connected.

CLI Format Examples:

```
get usb-conn-status
val usb-conn-status 1
```

```
notify sys.usb-conn-status 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

upgrade-status

Description: Query upgrade status. .

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get upgrade-status`
response	`val upgrade-status <"status">`
notify	`notify sys.upgrade-status <"status">`

Parameters:

Parameter	Description
`Ready`	Ready for upgrade
`Downloading`	Downloading upgrade file

`Failed`	Upgrade failed.
`Flashing`	Writing upgrade to flash memory.
`WaitingCallEnd`	If any call is active, upgrade suspended until call ends.
`Rebooting`	Upgrade complete, rebooting system.

CLI Format Examples:

```
get upgrade-status
val upgrade-status Flashing
```

```
notify sys.upgrade-status Flashing
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-upgrade-status

Description: Indicates the camera module upgrade status.

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get camera-upgrade-status`
response	`val camera-upgrade-status <"upgrade-status">`
notify	`notify sys.camera-upgrade-status <"upgrade-status">`

Parameters:

Parameter	Description
`Started`	Upgrade has started.
`Downloading`	Downloading upgrade file.
`Rebooting`	Rebooting after upgrade.
`Complete`	Upgrade successful and complete.
`Failed`	Upgrade failed
`pmuterogress`	Progress of update. This string will be followed by percent complete.
`Current`	This is synonymous with complete. Revision of upgrade and current FW match.

CLI Format Examples:

```
get camera-upgrade-status
val camera-upgrade-status Downloading
```

```
notify sys.camera-upgrade-status Downloading
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

power-saving-status

Description: System standby status.

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get power-saving-status`
response	`val power-saving-status <0|1>`
notify	`notify sys.power-saving-status <0|1>`

Parameters:

Parameter	Description
0	Not in standby
1	In standby

CLI Format Examples:

```
get power-saving-status
val power-saving-status 0
```

```
notify sys.power-saving-status 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

usb-aud-rx-status

Description: Query state of USB RX audio stream (RX audio being received, stopped).

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get usb-aud-rx-status`
response	`val usb-aud-rx-status <0|1>`
notify	`notify sys.usb-aud-rx-status <0|1>`

Parameters:

Parameter	Description
0	USB audio RX stream is closed
1	USB audio RX stream is open

CLI Format Examples:

```
get usb-aud-rx-status
val usb-aud-rx-status 1
```

```
notify sys.usb-aud-rx-status  1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

usb-aud-tx-status

Description: Query state of USB TX audio stream (TX audio being sent, stopped).

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get usb-aud-tx-status`
response	`val usb-aud-tx-status <0|1>`
notify	`notify sys.usb-aud-tx-status <0|1>`

Parameters:

Parameter	Description
0	USB audio TX stream is closed
1	USB audio TX stream is open

CLI Format Examples:

```
get usb-aud-tx-status
val usb-aud-rx-status 1
```

```
notify sys.usb-aud-tx-status 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

5.2.1.3 System Commands

regnotify

Description: Register client for notifications.

Command Definition:

Action	Definition
execute	`regnotify`

Parameters: None

CLI Format Examples:

```
regnotify
```

Supported Products: CS700-AV, CS700-SP

Available in API Version: 1.0

restart

Description: Restart system.

Command Definition:

Action	Definition
execute	`restart`

Parameters: None

CLI Format Examples:

```
restart
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

set reset-setting

Description: Reset settings by category.

Command Definition:

Action	Definition
execute	`set reset-setting <"category">`

Parameters:

Parameter	Description
all	Reset all properties
voip	Reset VoIP properties
audio	Reset audio properties
network	Reset network properties
camera	Reset camera properties
bt	Reset Bluetooth properties
contacts	Delete all VoIP contacts
call-history	Delete call history

CLI Format Examples:

```
set reset-setting all
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

5.2.2 Audio

5.2.2.1 Audio Properties

eq

Description: EQ setting used to adjust the speaker frequencies to your preference for the room and the types of calls.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get eq`
response	`val eq <1|2|3>`
set	`set eq <1|2|3>`
notify	`notify audio.eq <1|2|3>`

Parameters:

Parameter	Description
1	Voice
2	Bass boost
3	Treble boost

CLI Format Examples:

```
get eq
val eq 1
```

```
set eq 1
```

```
notify audio.eq 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

high-pass-filter

Description: High-Pass filter setting. High-Pass filters are provided to adjust to room and application requirements. Use the High-Pass filter in rooms that have a high background noise in the low frequencies (air conditioning, lighting fixtures, etc.). All filters are bi-quad filters, reducing the signal by 6dB per octave.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get high-pass-filter`
response	`val high-pass-filter <0..4>`
set	`set high-pass-filter <0..4>`
notify	`notify audio.high-pass-filter <0..4>`

Parameters:

Parameter	Description
0	None
1	110 Hz
2	140 Hz
3	175 Hz
4	225 Hz

CLI Format Examples:

```
get high-pass-filter
val high-pass-filter 0
```

```
set high-pass-filter 0
```

```
notify audio.high-pass-filter 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

analog-audio-in-mode

Description:	For the TV audio in port, select the gain setting mode, either auto or manual.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get analog-audio-in-mode`
response	`val analog-audio-in-mode <0|1>`
set	`set analog-audio-in-mode <0|1>`
notify	`notify audio.analog-audio-in-mode <0|1>`

Parameters:

Parameter	Description
0	Automatic gain setting
1	Manually specify gain setting

CLI Format Examples:

```
get analog-audio-in-mode
val analog-audio-in-mode 0
```

```
set analog-audio-in-mode 0
```

```
notify audio.analog-audio-in-mode 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

analog-audio-in-gain

Description:	For the TV audio in port, if analog-audio-in-mode is manual, then this is the gain value in dB. This value can be increased or decreased by 0.5dB.
Property Actions:	set, get, notify
Default Value:	4.5

Command Definition:

Action	Definition
get	`get analog-audio-in-gain`
response	`val analog-audio-in-gain <-12..40>`
set	`set analog-audio-in-gain <-12..40>`
notify	`notify audio.analog-audio-in-gain <-12..40>`

Parameters:

Parameter	Description
`-12, -11.5..40`	Manual gain setting This value can be increased or decreased by 0.5dB

CLI Format Examples:

```
get analog-audio-in-gain
val analog-audio-in-gain 4.5
```

```
set analog-audio-in-gain 4.5
```

```
notify audio.analog-audio-in-gain 4.5
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

wireless-omni-mic

Description: Enables additional audio processing of extension wireless omni-directional microphones.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get wireless-omni-mic`
response	`val wireless-omni-mic <0|1>`
set	`set wireless-omni-mic <0|1>`
notify	`notify audio.wireless-omni-mic <0|1>`

Parameters:

Parameter	Description
0	Disable wireless omni-directional microphone
1	Enable wireless omni-directional microphone

CLI Format Examples:

```
get wireless-omni-mic
val wireless-omni-mic 0
```

```
set wireless-omni-mic 1
```

```
notify audio. wireless-omni-mic 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.3

wireless-direct-mic

Description: Enables additional audio processing of extension wireless directional microphones.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get wireless-direct-mic`
response	`val wireless-direct-mic <0|1>`
set	`set wireless-direct-mic <0|1>`
notify	`notify audio.wireless-direct-mic <0|1>`

Parameters:

Parameter	Description
0	Disable wireless directional microphone
1	Enable wireless directional microphone

CLI Format Examples:

```
get wireless-direct-mic
val wireless-direct-mic 0
```

```
set wireless-direct-mic 1
```

```
notify audio. wireless-direct-mic 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.3

wireless-lapel-mic

Description: Enables additional audio processing of extension wireless lapel microphones.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get wireless-lapel-mic`
response	`val wireless-lapel-mic <0|1>`
set	`set wireless-lapel-mic <0|1>`
notify	`notify audio.wireless-lapel-mic <0|1>`

Parameters:

Parameter	Description
0	Disable wireless lapel microphone
1	Enable wireless lapel microphone

CLI Format Examples:

```
get wireless-lapel-mic
val wireless-lapel-mic 0
```

```
set wireless-lapel-mic 1
```

```
notify audio. wireless-lapel-mic 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.3

speaker-volume

Description: Configure speaker volume for calls.
Property Actions: set, get, notify

Default Value: 13

Command Definition:

Action	Definition
get	`get speaker-volume`
response	`val speaker-volume <1..18>`
set	`set speaker-volume <1..18>`
notify	`notify audio.speaker-volume <1..18>`

Parameters:

Parameter	Description
`1..18`	Volume setting

CLI Format Examples:

```
get speaker-volume
val speaker-volume 12
```

```
set speaker-volume 12
```

```
notify audio.speaker-volume 12
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

ring-tone

Description: VoIP ring-tone selection.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get ring-tone`
response	`val ring-tone <0..5>`
set	`set ring-tone <0..5>`
notify	`notify audio.ring-tone <0..5>`

Parameters:

Parameter	Description
`0..5`	Ring-tone index

CLI Format Examples:

```
get ring-tone
val ring-tone 0
```

```
set ring-tone 0
```

```
notify audio.ring-tone 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

ringer-volume

Description:	Configure VoIP ringer volume.
Property Actions:	set, get, notify
Default Value:	13

Command Definition:

Action	Definition
get	`get ringer-volume`
response	`val ringer-volume <1..18>`
set	`set ringer-volume <1..18>`
notify	`notify audio.ringer-volume <1..18>`

Parameters:

Parameter	Description
`1..18`	Ringer volume level

CLI Format Examples:

```
get ringer-volume
val ringer-volume 7
```

```
set ringer-volume 7
```

```
notify audio.ringer-volume 7
```

Supported Products: CS700-SP
Available in API Version: 1.1

5.2.2.2 Audio Statuses

speaker-mute

Description:	Mute speaker.
Property Actions:	set, get, notify

Command Definition:

Action	Definition
get	`get speaker-mute`
response	`val speaker-mute <0|1>`
set	`Set speaker-mute <0|1>`
notify	`notify audio.speaker-mute <0|1>`

Parameters:

Parameter	Description
0	Speaker is not muted
1	Speaker is muted

CLI Format Examples:

```
get speaker-mute
val speaker-mute 1
```

```
set speaker-mute 1
```

```
notify audio.speaker-mute 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.3

mute

Description: Mute microphones.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get mute`
response	`val mute <0|1>`
set	`set mute <0|1>`
notify	`notify audio.mute <0|1>`

Parameters:

Parameter	Description
0	Unmute mics
1	Mute mics

CLI Format Examples:

```
get mute
val mute 0
```

```
set mute 0
```

```
notify audio.mute 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

mic-conn-status

Description:	Query external mic connection status.
Property Actions:	get, notify

Command Definition:

Action	Definition
get	`get mic-conn-status`
response	`val mic-conn-status <"mic_number"> <connected|disconnected>`
notify	`notify audio.mic-conn-status <connected|disconnected>`

Parameters:

Parameter	Description
`connected`	External mic is connected
`disconnected`	External mic is disconnected

CLI Format Examples:

```
get mic-conn-status
val mic-conn-status 1 connected
```

```
notify audio.mic-conn-status connected
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

5.2.3 Camera

5.2.3.1 Camera Properties

camera-ptz-home

Description:	Default PTZ settings for home position. When the device detects that the upstream USB connection has been established, either at startup or after a USB disconnection, it will revert to the default PTZ settings. Calling set on this property will move the camera to the newly set PTZ position. See cam-save-as-defaults to save current PTZ to home setting.
Property Actions:	set, get, notify
Default Value:	0 0 1

Command Definition:

Action	Definition
get	`get camera-ptz-home`

response	`val camera-ptz-home <"pan"> <"tilt"> <"zoom">`
set	`set camera-ptz-home <"pan"> <"tilt"> <"zoom">`
notify	`notify camera.camera-ptz-home <"pan"> <"tilt"> <"zoom">`

Parameters:

Parameter	Description
-30..30	Pan setting (default value is 0)
-18..18	Tilt setting (default value is 0)
100..400	Zoom setting (default value is 100)

CLI Format Examples:

```
get camera-ptz-home
val camera-ptz-home 0 0 100
```

```
set camera-ptz-home 0 0 100
```

```
notify camera.camera-ptz-home 0 0 100
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-flicker

Description:	Camera's flicker setting.
Property Actions:	set, get, notify
Default Value:	2

Command Definition:

Action	Definition
get	`get camera-flicker`
response	`val camera-flicker <1|2>`
set	`set camera-flicker <1|2>`
notify	`notify camera.camera-flicker <1|2>`

Parameters:

Parameter	Description
1	50Hz
2	60Hz

CLI Format Examples:

```
get camera-flicker
val camera-flicker 2
```

```
set camera-flicker 2
```

```
notify camera.camera-flicker 2
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-image-defaults

Description: Camera image default settings. These are Backlight, Brightness, Contrast, Saturation, Sharpness, Hue, and Gamma.

Property Actions: set, get, notify

Default Values: 0 125 110 100 50 90 255

Command Definition:

Action	Definition
get	`get camera-image-defaults`
response	`val camera-image-defaults <"backlight"> <"brightness"> <"contrast"> <"saturation"> <"sharpness"> <"hue"> <"gamma">`
set	`set camera-image-defaults <"backlight"> <"brightness"> <"contrast"> <"saturation"> <"sharpness"> <"hue"> <"gamma">`
notify	`notify camera.camera-image-defaults <"backlight"> <"brightness"> <"contrast"> <"saturation"> <"sharpness"> <"hue"> <"gamma">`

Parameters:

Parameter	Description
`0..5`	Backlight setting
`0..250`	Brightness setting
`60..140`	Contrast setting
`50..150`	Saturation setting
`0..255`	Sharpness setting
`0..180`	Hue setting
`1..255`	Gamma setting

CLI Format Examples:

```
get camera-image-defaults
val camera-image-defaults 1 125 110 100 85 90 220
```

```
set camera-image-defaults 1 125 110 100 85 90 220
```

```
notify camera.camera-image-defaults 1 125 110 100 85 90 220
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

5.2.3.2 Camera Statuses

camera-status

Description: Query camera state.

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get camera-status`
response	`val camera-status <off|initializing|reconnecting|unconnected|power_save|resync|on>`
notify	`notify camera.camera-status <off|initializing|reconnecting|unconnected|power_save|resync|on>`

Parameters:

Parameter	Description
`off`	Camera is off
`initializing`	Camera subsystem is initializing
`upgrading`	Camera module's firmware is being upgraded
`reconnecting`	Camera is coming online
`unconnected`	Camera is not connected to USB
`power_save`	Camera is in power save mode
`resync`	Camera is updating user parameters
`on`	Camera is powered on and available for streaming

CLI Format Examples:

```
get camera-status
val camera-status on
```

```
notify camera.camera-status on
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-stream

Description: Query state of camera stream (video being sent, stopped)

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get camera-stream`
response	`val camera-stream <open|closed>`
notify	`notify camera.camera-stream <open|closed>`

Parameters:

Parameter	Description

`open`	USB video stream is open
`closed`	USB video stream is closed

CLI Format Examples:

```
get camera-stream
val camera-stream open
```

```
notify camera.camera-stream open
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-door

Description: Query state of camera door.

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get camera-door`
response	`val camera-door <open|closed>`
notify	`notify camera.camera-door <open|closed>`

Parameters:

Parameter	Description
`open`	Camera privacy door is open
`closed`	Camera privacy door is closed

CLI Format Examples:

```
get camera-door
val camera-door open
```

```
notify camera.camera-door open
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-mute

Description: Camera video "mute" or stop state.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get camera-mute`

response	`val camera-mute <0|1>`
set	`set camera-mute <0|1>`
notify	`notify camera.camera-mute <0|1>`

Parameters:

Parameter	Description
0	Camera video is not muted
1	Camera video is muted

CLI Format Examples:

```
get camera-mute
val camera-mute 0
```

```
set camera-mute 0
```

```
notify camera.camera-mute 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-pan

Description: Camera's digital pan setting. Camera can only be panned when zoomed in.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get camera-pan`
response	`val camera-pan <-30..30>`
set	`set camera-pan <-30..30>`
notify	`notify camera.camera-pan <-30..30>`

Parameters:

Parameter	Description
-30..30	Pan setting

CLI Format Examples:

```
get camera-pan
val camera-pan 0
```

```
set camera-pan 0
```

```
notify camera.camera-pan 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-tilt

Description: Camera's digital tilt setting. Camera can only be tilted when zoomed in.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get camera-tilt`
response	`val camera-tilt <-18..18>`
set	`set camera-tilt <-18..18>`
notify	`notify camera.camera-tilt <-18..18>`

Parameters:

Parameter	Description
`-18..18`	Tilt setting

CLI Format Examples:

```
get camera-tilt
val camera-tilt 0
```

```
set camera-tilt 0
```

```
notify camera.camera-tilt 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-zoom

Description: Camera's digital zoom setting. When zoom is set to 100, the pan and tilt settings will be forced to zero.

Property Actions: set, get, notify

Default Value: 100

Command Definition:

Action	Definition
get	`get camera-zoom`
response	`val camera-zoom <100..400>`
set	`set camera-zoom <100..400>`

notify	`notify camera.camera-zoom <100..400>`

Parameters:

Parameter	Description
`100..400`	Zoom setting

CLI Format Examples:

```
get camera-zoom
val camera-zoom 100
```

```
set camera-zoom 100
```

```
notify camera.camera-zoom 100
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

camera-backlight

Description: Backlight compensation for camera.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
Get	`get camera-backlight`
response	`val camera-backlight <0..5>`
Set	`set camera-backlight <0..5>`
Notify	`notify camera.camera-backlight <0..5>`

Parameters:

Parameter	Description
`0`	Off
`1..5`	Camera Backlight level.

CLI Format Examples:

```
get camera-backlight
val camera-backlight 2
```

```
set camera-backlight 2
```

```
notify camera.camera-backlight 2
```

Supported Products: CS700-AV, CS700-SP

Available in API Version: 1.1

camera-brightness

Description: Camera Brightness level

Property Actions: set, get, notify

Default Value: 125

Command Definition:

Action	Definition
get	`get camera-brightness`
response	`val camera-brightness <0..250>`
set	`set camera-brightness <0..250>`
notify	`notify camera.camera-brightness <0..250>`

Parameters:

Parameter	Description
`0..250`	Camera Brightness level.

CLI Format Examples:

```
get camera-brightness
val camera-brightness 125
```

```
set camera-brightness 125
```

```
notify camera.camera-brightness 125
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

camera-contrast

Description: Camera Contrast level

Property Actions: set, get, notify

Default Value: 110

Command Definition:

Action	Definition
get	`get camera-contrast`
response	`val camera-contrast <60..140>`

set	`set camera-contrast <60..140>`
notify	`notify camera.camera-contrast <60..140>`

Parameters:

Parameter	Description
`60..140`	Camera Contrast level.

CLI Format Examples:

```
get camera-contrast
val camera-contrast 110
```

```
set camera-contrast 110
```

```
notify camera.camera-contrast 110
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

camera-saturation

Description: Camera Saturation level

Property Actions: set, get, notify

Default Value: 100

Command Definition:

Action	Definition
get	`get camera-saturation`
response	`val camera-saturation <50..150>`
set	`set camera-saturation <50..150>`
notify	`notify camera.camera-saturation <50..150>`

Parameters:

Parameter	Description
`50..150`	Camera Saturation level.

CLI Format Examples:

```
get camera-saturation
val camera-saturation 100
```

```
set camera-saturation 100
```

```
notify camera.camera-saturation 100
```

Supported Products: CS700-AV, CS700-SP

Available in API Version: 1.1

camera-sharpness

Description: Camera Sharpness level

Property Actions: set, get, notify

Default Value: 85

Command Definition:

Action	Definition
get	`get camera-sharpness`
response	`val camera-sharpness <0..255>`
set	`set camera-sharpness <0..255>`
notify	`notify camera.camera-sharpness <0..255>`

Parameters:

Parameter	Description
0..255	Camera Sharpness level.

CLI Format Examples:

```
get camera-sharpness
val camera-sharpness 100
```

```
set camera-sharpness 100
```

```
notify camera.camera-sharpness 100
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

camera-hue

Description: Camera Hue level

Property Actions: set, get, notify

Default Value: 90

Command Definition:

Action	Definition
get	`get camera-hue`
response	`val camera-hue <0..180>`
set	`set camera-hue <0..180>`
notify	`notify camera.camera-hue <0..180>`

Parameters:

Parameter	Description
0..180	Camera Hue level.

CLI Format Examples:

```
get camera-hue
val camera-hue 100
```

```
set camera-hue 100
```

```
notify camera.camera-hue 100
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

camera-gamma

Description: Camera Gamma level

Property Actions: set, get, notify

Default Value: 255

Command Definition:

Action	Definition
Get	`get camera-gamma`
Response	`val camera-gamma <1..255>`
Set	`set camera-gamma <1..255>`
Notify	`notify camera.camera-gamma <1..255>`

Parameters:

Parameter	Description
1..255	Camera Gamma level.

CLI Format Examples:

```
get camera-gamma
val camera-gamma 100
```

```
set camera-gamma 100
```

```
notify camera.camera-gamma 100
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

5.2.3.3 Camera Commands

cam-save-as-default

Description: Save the camera's current PTZ settings as the default values.

Command Definition:

Action	Definition
execute	`cam-save-as-default`

Parameters: None

CLI Format Examples:

```
cam-save-as-default
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

cam-apply-defaults

Description: Apply the camera's default PTZ settings. These settings are also automatically applied when the device detects that the upstream USB connection has been established, either at startup or after a USB disconnection.

Command Definition:

Action	Definition
execute	`cam-apply-defaults`

Parameters: None

CLI Format Examples:

```
cam-apply-defaults
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

cam-image-save-as-default

Description: Save the camera's current Backlight, Brightness, Contrast, Saturation, Sharpness, Hue, and Gamma settings as the default values.

Command Definition:

Action	Definition
execute	`cam-image-save-as-default`

Parameters: None

CLI Format Examples:

```
cam-image-save-as-default
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

cam-image-apply-defaults

Description: Apply the camera's default Backlight, Brightness, Contrast, Saturation, Sharpness, Hue, and Gamma settings. These settings are also automatically applied when the device detects that the upstream USB connection has been established, either at startup or after a USB disconnection.

Command Definition:

Action	Definition
execute	`cam-image-apply-defaults`

Parameters: None

CLI Format Examples:

```
cam-image-apply-defaults
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

cam-zoom-in

Description: Zoom in the camera by one level.

Command Definition:

Action	Definition
Execute	`cam-zoom-in`
Notify	`notify camera.zoom <100..400>`

Parameters: None

CLI Format Examples:

```
cam-zoom-in
notify camera.zoom 200
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

cam-zoom-out

Description: Zoom out the camera by one level.

Command Definition:

Action	Definition
Execute	`cam-zoom-out`
Notify	`notify camera.zoom <100..400>`

Parameters: None

CLI Format Examples:

```
cam-zoom-out
notify camera.zoom 100
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

cam-pan-left

Description: Pan the camera left by one level.

Command Definition:

Action	Definition
Execute	`cam-pan-left`
Notify	`notify camera.pan <-30..30>`

Parameters: None

CLI Format Examples:

```
cam-pan-left
notify camera.pan 10
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

cam-pan-right

Description: Pan the camera right by one level.

Command Definition:

Action	Definition
Execute	`cam-pan-right`
Notify	`notify camera.pan <-30..30>`

Parameters: None

CLI Format Examples:

```
cam-pan-right
notify camera.pan 8
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

cam-tilt-up

Description: Tilt the camera up by one level.

Command Definition:

Action	Definition
Execute	`cam-tilt-up`
Notify	`notify camera.tilt <-18..18>`

Parameters: None

CLI Format Examples:

```
cam-tilt-up
notify camera.tilt 8
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

cam-tilt-down

Description: Tilt the camera down by one level.

Command Definition:

Action	Definition
Execute	`cam-tilt-down`
Notify	`notify camera.tilt <-18..18>`

Parameters: None

CLI Format Examples:

```
cam-tilt-down
notify camera.tilt 5
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

5.2.4 Bluetooth

5.2.4.1 Bluetooth Properties

bt-enable

Description: Enable or disable Bluetooth.
Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get bt-enable`
response	`val bt-enable <0|1>`
set	`set bt-enable <0|1>`
notify	`notify bt.bt-enable <0|1>`

Parameters:

Parameter	Description
0	Disable
1	Enable

CLI Format Examples:

```
get bt-enable
val bt-enable 0
```

```
set bt-enable 0
```

```
notify bt.bt-enable 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

bt-mac

Description: Get the Bluetooth device's MAC address advertised over the air. This is read-only and set in manufacturing.

Property Actions: get

Command Definition:

Action	Definition
get	`get bt-mac`
response	`val bt-mac <"mac">`

Parameters:

Parameter	Description
`mac`	MAC address of connected device

CLI Format Examples:

```
get bt-mac
val bt-mac AC:14:22:01:23:45
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

bt-name

Description: The Bluetooth basic-rate device name advertised over the air.

Property Actions: set, get, notify

Default Value: The product name concatenated with the Bluethooth's MAC address - i.e., "Yamaha CS-700 " + last 2 bytes of the Bluetooth's MAC address.

Command Definition:

Action	Definition
get	`get bt-name`
response	`val bt-name <"name">`
set	`set bt-name <"name">`
notify	`notify bt.bt-name <"name">`

Parameters:

Parameter	Description
`name`	String. Name of device.

CLI Format Examples:

```
get bt-name
val bt-name Yamaha CS-700:22:33
```

```
set bt-name CS700-MAINCONF
```

```
notify bt.bt-name CS700-MAINCONF
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

bt-pin

Description: The Bluetooth basic-rate pin for pairing. Pins are 4 digits.

Property Actions: set, get, notify

Default Value: 0000

Command Definition:

Action	Definition
get	`get bt-pin`
response	`val bt-pin <"pin">`
set	`set bt-pin <"pin">`

notify	`notify bt.bt-pin <"pin">`

Parameters:

Parameter	Description
`pin`	4-digit string

CLI Format Examples:

```
get bt-pin
val bt-pin 0000
```

```
set bt-pin 0000
```

```
notify bt.bt-pin 0000
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

bt-pair-timeout

Description: Bluetooth basic-rate pairing timeout in seconds. After this time, the pairing mode will stop. A value of 0 indicates no timeout.

Property Actions: set, get, notify

Default Value: 60

Command Definition:

Action	Definition
get	`get bt-pair-timeout`
response	`val bt-pair-timeout <0|30|60|90|120>`
set	`set bt-pair-timeout <0|30|60|90|120>`
notify	`notify bt.bt-pair-timeout <0|30|60|90|120>`

Parameters:

Parameter	Description
`0`	No timeout
`30,60,90,120`	Pairing timeout in seconds

CLI Format Examples:

```
get bt-pair-timeout
val bt-pair-timeout 60
```

```
set bt-pair-timeout 60
```

```
notify bt.bt-pair-timeout 60
```

Supported Products: CS700-AV, CS700-SP

Available in API Version: 1.4

bt-call-autojoin

Description:	Allow a Bluetooth call on the paired and linked Bluetooth device to automatically join a USB based conference if there is no dialer control app in use.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get bt-call-autojoin`
response	`val bt-call-autojoin <0|1>`
set	`set bt-call-autojoin <0|1>`
notify	`notify bt.bt-call-autojoin <0|1>`

Parameters:

Parameter	Description
0	Bluetooth call auto-join to USB disabled
1	Bluetooth call auto-join to USB enabled

CLI Format Examples:

```
get bt-call-autojoin
val bt-call-autojoin 0
```

```
set bt-call-autojoin 0
```

```
notify bt.bt-call-autojoin 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.3

bt-auto-unpair-on-disconnect

Description:	If enabled, when the Bluetooth device disconnects, it will automatically be unpaired from the CS-700. The Bluetooth device will need to be re-paired to the CS-700 to use it with the CS-700.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get bt-auto-unpair-on-disconnect`

response	`val bt-auto-unpair-on-disconnect <0|1>`
set	`set bt-auto-unpair-on-disconnect <0|1>`
notify	`notify bt. bt-auto-unpair-on-disconnect <0|1>`

Parameters:

Parameter	Description
0	Bluetooth auto-unpair on disconnect disabled
1	Bluetooth auto-unpair on disconnect enabled

CLI Format Examples:

```
get bt-auto-unpair-on-disconnect
val bt-auto-unpair-on-disconnect 0
```

```
set bt-auto-unpair-on-disconnect 0
```

```
notify bt.bt-auto-unpair-on-disconnect 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.4

5.2.4.2 Bluetooth Statuses

bt-status

Description: Get the Bluetooth device status.

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get bt-status`
response	`val bt-status <Off|Initializing|Updating|Pairing|Enabled|Connecting|Connected>`
notify	`notify bt.bt-status <Off|Initializing|Updating|Pairing|Enabled|Connecting|Connected>`

Parameters:

Parameter	Description
`Off`	Bluetooth is off
`Initializing`	Bluetooth subsystem is initializing
`Updating`	Bluetooth processor firmware is being upgraded
`Pairing`	Bluetooth is in pairing mode
`Enabled`	Bluetooth is enabled and on
`Connecting`	Bluetooth is on and connecting
`Connected`	A device is connected

CLI Format Examples:

```
get bt-status
val bt-status Connected
```

```
notify bt.bt-status Connected
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

bt-paired-list

Description:	Get a list of paired Bluetooth basic-rate devices. Return space-separated MAC address and name pairs.
Property Actions:	get

Command Definition:

Action	Definition
get	`get bt-paired-list`
response	`val bt-paired-list <"mac"> <"name"> [<"mac"> <"name">]+`

Parameters:

Parameter	Description
`Mac`	MAC address of paired device.
`name`	String. Name of paired device.

CLI Format Examples:

```
get bt-paired-list
val bt-paired-list D4:B3:77:EF:31:94 ADR6301 D4:B3:77:EF:31:95 ADR6302
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

bt-connected

Description:	Get Bluetooth basic-rate device connected status. If a device is connected, returns its MAC address, name, and profiles.
Property Actions:	get, notify

Command Definition:

Action	Definition
get	`get bt-connected`
response	`val bt-connected [<"mac"> <"name">]`
notify	`notify bt.bt-connected [<"mac"> <"name">]`

Parameters:

Parameter	Description
mac	MAC address of connected device.
name	String. Name of connected device.

CLI Format Examples:

```
get bt-connected
val bt-connected D4:B3:77:EF:31:94 ADR6301 a2dp,hsp
```

```
notify bt.bt-connected D4:B3:77:EF:31:94 SAMSUNG-G925V
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

5.2.4.3 Bluetooth Commands

bt-pair

Description: Initiate Bluetooth pairing mode from the device in order to pair a phone.

Command Definition:

Action	Definition
execute	`bt-pair <0|1>`

Parameters:

Parameter	Description
0	Cancel pairing
1	Initiate pairing

CLI Format Examples:

```
bt-pair 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

bt-unpair

Description: Remove all Bluetooth devices in the paired devices list. If a device is connected, it will be disconnected.

Command Definition:

Action	Definition
Execute	`bt-unpair <all>`

Parameters:

Parameter	Description

All	Remove all paired devices

CLI Format Examples:

```
bt-unpair all
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

bt-disconnect

Description:	Disconnect the connected Bluetooth device.

Command Definition:

Action	Definition
Execute	`bt-disconnect`

Parameters:	None

CLI Format Examples:

```
bt-disconnect
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

nfc-enable

Description:	Enable/disable NFC and the NFC logo LED on the main unit.

Property Actions:	set, get, notify

Default Value:	1

Command Definition:

Action	Definition
get	`get nfc-enable`
response	`val nfc-enable <0|1>`
set	`set nfc-enable <0|1>`
notify	`notify bt.nfc-enable <0|1>`

Parameters:

Parameter	Description
0	Disable NFC and NFC LED
1	Enable NFC and NFC LED

CLI Format Examples:

```
get nfc-enable
```

```
val nfc-enable 1
```

```
set nfc-enable 1
```

```
notify bt.nfc-enable 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.1

5.2.5 Call

5.2.5.1 Call Statuses

status

Description: Query a call status.

Property Actions: get, notify

Command Definition:

Action	Definition
Get	`get status <1..3|usb|bt>`
Response	`val status <1..3|usb|bt> <"call-status">`
Notify	`notify call.status <1..3|usb|bt> <"call-status">`

Parameters:

Parameter	Description
`1`	VoIP line 1
`2`	VoIP line 2
`3`	VoIP line 3 (used for transfer)
`usb`	USB line
`bt`	Bluetooth line
`call-status`	String as shown below for each call type.
`VoIP Calls`	
`idle`	Initialization state.
`incoming`	Receiving incoming VoIP call.
`calling`	Initiating outgoing VoIP call.
`failed`	Outgoing call attempt failed; it is followed by "disconnected" when the attempt is disconnected.
`connected`	Call is connected.
`onhold`	Call is on hold.
`connected-in-conf`	Call is in a conference.
`disconnected`	Not in a call or phone is not registered.
`update`	The call is transferred by the far end (with some call managers the call state will remain "connected" after the transfer).
`missed`	Incoming VoIP call missed due to Do-Not-Disturb.
`USB Calls`	

active	USB audio is active.
idle	Initialization state.
incoming	Incoming USB call signal is received from host.
inactive	USB audio is not active.
connected	Call is connected.
onhold	USB audio is on hold.
connected-in-conf	Call is in a conference.
BT Calls	
active	BT audio is active.
idle	Initialization state.
incoming	Incoming BT call signal is received from connected phone.
inactive	BT audio is not active.
connected	Call is connected.
onhold	BT audio is on hold.
connected-in-conf	Call is in a conference.

CLI Format Examples:

```
get status 1
val status 1 connected
```

```
notify call.status 1 connected
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0,   VoIP Available in API Version: 1.1

status-all

Description: Query all call status. See "call-status" property for description of status types.

Property Actions: get

Command Definition:

Action	Definition
get	`get status-all`
response	`val status-all line1:<"status1"> line2:<"status2"> line3:<"status3"> bt:<"statusbt"> usb:<"statususb">`

Parameters:

Parameter	Description
status1	VoIP line 1 call status
status2	VoIP line 2 call status
status3	VoIP line 3 call status
statusbt	Bluetooth call status
statususb	USB call status

CLI Format Examples:

```
get status-all
val status-all line1:disabled line2:disabled line3:disabled bt:idle usb:idle
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0,   VoIP Available in API Version: 1.1

call-info

Description: Query call information with given call id.

Property Actions: get

Command Definition:

Action	Definition
get	`get call-info <1..3, bt, usb>`
response	`val call-info <1..3> <"name"> <"number"> <"call-status">`

Parameters:

Parameter	Description
1	VoIP line 1
2	VoIP line 2
3	VoIP transfer line
name	String. Name of caller.
number	String. Phone number.
call-status	Call status as described in the "get status" parameter description.

CLI Format Examples:

```
get call-info 1
val call-info 1 Blake 7823 connected
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0,   VoIP Available in API Version: 1.1

number

Description: Query a caller number.

Property Actions: get

Command Definition:

Action	Definition
get	`get number <1..3>`
response	`val number <1..3> <"number">`

Parameters:

Parameter	Description
1	VoIP line 1
2	VoIP line 2
3	VoIP line 3 (used for transfer)
number	Caller's phone number

CLI Format Examples:

```
get number 1
val number 1 7823
```

Supported Products: CS700-SP
Available in API Version: 1.1

caller

Description: Query a caller name.

Property Actions: get

Command Definition:

Action	Definition
get	`get caller <1..3>`
response	`val caller <1..3> <"name">`

Parameters:

Parameter	Description
1	VoIP line 1
2	VoIP line 2
3	VoIP line 3 (used for transfer)
name	Caller's name

CLI Format Examples:

```
get caller 1
val caller 1 Blake
```

Supported Products: CS700-SP
Available in API Version: 1.1

start-time

Description: Query call start time. Return value is in format HH:MM:SS in 24-hour time.

Property Actions: get

Command Definition:

Action	Definition
get	`get start-time <1..3>`
response	`val start-time <1..3> <"time">`

Parameters:

Parameter	Description
1	VoIP line 1
2	VoIP line 2
3	VoIP line 3 (used for transfer)
`time`	HH:MM:SS in 24-hour time

CLI Format Examples:

```
get start-time 1
val start-time 1 13:22:41
```

Supported Products: CS700-SP
Available in API Version: 1.1

codec

Description: Query call codec.

Property Actions: get

Command Definition:

Action	Definition
get	`get codec <1..3>`
response	`val codec <1..3> <codec-index>`

Parameters:

Parameter	Description
1	VoIP line 1
2	VoIP line 2
3	VoIP line 3 (used for transfer)
`codec-index <0..5>`	Codec ID as listed below
0	Disabled
1	G.722
2	G.711U
3	G.711A
4	G729
5	G729

CLI Format Examples:

```
get codec 1
val codec 1 2
```

Supported Products: CS700-SP
Available in API Version: 1.1

call-quality

Description: Query call quality.
Note: The response is a comma separate list of values.

Property Actions: get

Command Definition:

Action	Definition
get	`get call-quality <1..3>`
response	`val call-quality <1..3> <list>`

Parameters:

Parameter	Description
`1`	VoIP line 1
`2`	VoIP line 2
`3`	VoIP line 3 (used for transfer)
`tx_packets`	Number of transmitted packets
`rx_packets`	Number of received packets
`tx_octets`	Number of transmitted octets
`rx-octets`	Number of received octets
`tx-packet-size`	Transmit packet size
`rx-packet-size`	Receive packet size
`lost-packets`	Number of lost packets
`discarded-packets`	Number of discarded packets
`jitter-ms`	Detected jitter in ms
`max-jitter-ms`	Max detected jitter in ms
`min-jitter-ms`	Min detected jitter in ms
`buffer-level-ms`	Jitter buffer size in ms
`local-ip`	Local IP address
`local-port`	Local port number
`remote-ip`	Remote IP address
`remote-port`	Remote port number

CLI Format Examples:

```
get call-quality 1
val call-quality 1
224,227,35840,36320,20,10,12,11,0,20,0,20,200.200.210.131,4000,200.200.210.190,4000
```

Supported Products: CS700-SP
Available in API Version: 1.1

5.2.5.2 Call Commands

dial

Description: Dial of VoIP call with the given number.

Command Definition:

Action	Definition
execute	`dial <"voip-line-id"> <"number">`

Parameters:

Parameter	Description
1	VoIP line 1
2	VoIP line 2
3	VoIP line 3 (used for transfer)
Number	VoIP number to dial

CLI Format Examples:

```
dial 1 7823
```

Supported Products: CS700-SP
Available in API Version: 1.1

answer

Description: Answer the given call.

Command Definition:

Action	Definition
execute	`answer <1..2|usb|bt>`

Parameters:

Parameter	Description
line-id <1..2|usb|bt>	Line ID of call to be answered.

CLI Format Examples:

```
answer  1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0, VoIP(Line ID) Available in API Version: 1.1

hangup

Description: Hang up the given call.

Command Definition:

Action	Definition
execute	`hangup <1..3|usb|bt>`

Parameters:

Parameter	Description
`line-id <1..3|usb|bt>`	Line ID of call to be hung up

CLI Format Examples:

```
hangup  1
```

Supported Products: CS700-AV, CS700-SP,
Available in API Version: 1.0, VoIP(Line ID) Available in API Version: 1.1

hold

Description: Hold the given call.

Command Definition:

Action	Definition
execute	`hold <1..2|usb|bt|all>`

Parameters:

Parameter	Description
`line-id <1..2|usb|bt|all>`	Line ID of call to be held, or hold all lines

CLI Format Examples:

```
hold  1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0, VoIP(Line ID) Available in API Version: 1.1

resume

Description: Resume the given call.

Command Definition:

Action	Definition
execute	`resume <1..2|usb|bt>`

Parameters:

Parameter	Description
`line-id <1..2|usb|bt>`	Line ID of call to be resumed

CLI Format Examples:

```
resume  1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0, VoIP(Line ID) Available in API Version: 1.1

swap

Description: Swap the source call with the target call, used for swapping held calls.

Command Definition:

Action	Definition
execute	`swap  <"held-line-id"> <"active-line-id">`

Parameters:

Parameter	Description
`held-line-id <1..2|usb|bt>`	Line ID of call that is currently on hold; this call will be resumed
`active-line-id <1..2|usb|bt>`	Line ID of call that is currently active; this call will be placed on hold

CLI Format Examples:

```
swap  1 2
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0, VoIP(Line ID) Available in API Version: 1.1

join

Description: Join allows the individual lines to be mixed (joined) or not mixed (separated). Used for conferencing existing calls. Any “line-id” not specified will default to a mix state of 0.

Command Definition:

Action	Definition
execute	`join  <"line-id1" “mix-state”> <"line-id2" “mix_state”> [<"line-id3" “mix_state”>] [<"line-id4" “mix_state”>]`

Parameters:

Parameter	Description
`line-id1 <1..2|usb|bt>`	Line ID1 of call to be mixed or not mixed

line-id2 <1..2|usb|bt>	Line ID2 of call to be mixed or not mixed
line-id3 <1..2|usb|bt>	Line ID3 of call to be mixed or not mixed [optional]
line-id4 <1..2|usb|bt>	Line ID4 of call to be mixed or not mixed [optional]
mix_state	
0	Separate lines (not mixed)
1	Join lines (mix)

CLI Format Examples:

Join (mix) only VoIP lines 1 and 2
```
join 1 1 2 1 usb 0 bt 0
join 1 1 2 1
```

Separate (not mixed) all calls from a conference
```
join  1 0 2 0 usb 0 bt 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

transfer

Description: Transfer the source call to target call. The target call was previously initiated and can either be in an “establishing” state (blind transfer) or a “connected” state (acknowledged transfer). After the call has been successfully transferred, both lines are released.

Command Definition:

Action	Definition
execute	`transfer  <"source-line-id"> <"target-line-id">`

Parameters:

Parameter	Description
source-line-id <1..2>	Line ID of call being transferred
target-line-id <3>	Line ID of call to which the source will be transferred

CLI Format Examples:

```
hold 1
dial 3 6175551212
transfer 1 3
```

Supported Products: CS700-SP
Available in API Version: 1.1

play ring-tone

Description: Play ring tone by index.

Command Definition:

Action	Definition
execute	`play ring-tone <0..5>`

Parameters:

Parameter	Description
`0..5`	Ring-tone index

CLI Format Examples:

```
play ring-tone 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

5.2.6 Network

5.2.6.1 Network Properties

mac

Description: Main unit's MAC address. Read-only value set at manufacturing.

Property Actions: get

Command Definition:

Action	Definition
get	`get mac`
response	`val mac <"mac-address">`

Parameters:

Parameter	Description
`mac-address`	MAC address

CLI Format Examples:

```
get mac
val mac AC:14:22:01:23:46
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

dhcp

Description: Enables or disables DHCP. If DHCP is disabled, the Static IP Address, Subnet Mask, and Default Gateway must be specified.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get dhcp`
response	`val dhcp <0|1>`
set	`set dhcp <0|1>`
notify	`notify net.dhcp <0|1>`

Parameters:

Parameter	Description
0	Disable DHCP
1	Enable DHCP

CLI Format Examples:

```
get dhcp
val dhcp 1
```

```
set dhcp 1
```

```
notify net.dhcp 1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

ip

Description: Current IP Address assigned to the device.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get ip`
response	`val ip <"ip-address">`
set	`set ip <"ip-address">`
notify	`notify net.ip <"ip-address">`

Parameters:

Parameter	Description
`ip-address`	IP address

CLI Format Examples:

```
get ip
val ip 192.168.1.103
```

```
set ip 192.168.1.103
```

```
notify net.ip 192.168.1.103
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

subnet

Description: Subnet Mask to determine the subnet to which the device belongs.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get subnet`
response	`val subnet <"subnet-address">`
set	`set subnet <"subnet-address">`
notify	`notify net.subnet <"subnet-address">`

Parameters:

Parameter	Description
`subnet-address`	String. IP address.

CLI Format Examples:

```
get subnet
val subnet 255.255.255.0
```

```
set subnet 255.255.255.0
```

```
notify net.subnet 255.255.255.0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

gateway

Description: The Default Gateway is the device's default router on the IP network.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get gateway`

response	`val gateway <"gateway-address">`
set	`set gateway <"gateway-address">`
notify	`notify net.gateway <"gateway-address">`

Parameters:

Parameter	Description
`gateway-address`	String. IP address.

CLI Format Examples:

```
get gateway
val gateway 192.168.3.1
```

```
set gateway 192.168.3.1
```

```
notify net.gateway 192.168.3.1
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

dns1

Description: Address of the primary Domain Name System (DNS) server.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get dns1`
response	`val dns1 <"dns-address">`
set	`set dns1 <"dns-address">`
notify	`notify net.dns1 <"dns-address">`

Parameters:

Parameter	Description
`dns-address`	String. IP address.

CLI Format Examples:

```
get dns1
val dns1 192.168.3.45
```

```
set dns1 192.168.3.45
```

```
notify net.dns1 192.168.3.45
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

dns2

Description: Address of the secondary Domain Name System (DNS) server.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get dns2`
response	`val dns2 <"dns-address">`
set	`set dns2 <"dns-address">`
notify	`notify net.dns2 <"dns-address">`

Parameters:

Parameter	Description
`dns-address`	String. IP address.

CLI Format Examples:

```
get dns2
val dns2 192.168.3.46
```

```
set dns2 192.168.3.46
```

```
notify net.dns2 192.168.3.46
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

vlan

Description: Specifies VLAN behavior and support for the device. .
Property Actions: set, get, notify

Default Value: 2

Command Definition:

Action	Definition
get	`get vlan`
response	`val vlan <0|1|2>`
set	`set vlan <0|1|2>`
notify	`notify net.vlan <0|1|2>`

Parameters:

Parameter	Description
0	Manually specify VLAN ID. If the environment uses Voice VLAN but the Voice VLAN ID cannot be detected, choose this, and specify the ID in the VLAN ID field, net.vlan-id

1	Disable VLAN. Switches off VLAN capabilities
2	Automatically detect VLAN ID. should be used in environments that provide a Voice VLAN with automatic detection, in which case the device will determine the VLAN identifier and register in that network

CLI Format Examples:

```
get vlan
val vlan 2
```

```
set vlan 2
```

```
notify net.vlan 2
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

vlan-id

Description: The manually specified VLAN ID. If the VLAN mode is automatic, this is the detected VLAN ID if VLAN is active.

Property Actions: set, get, notify

Default Value: 3

Command Definition:

Action	Definition
get	`get vlan-id`
response	`val vlan-id <1..4094>`
set	`set vlan-id <1..4094>`
notify	`notify net.vlan-id <1..4094>`

Parameters:

Parameter	Description
1..4094	VLAN ID

CLI Format Examples:

```
get vlan-id
val vlan-id 3
```

```
set vlan-id 3
```

```
notify net.vlan-id 3
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

ntp1

Description: Configure first NTP (Network Time Protocol) server address.

Property Actions: set, get, notify

Default Value: 0.pool.ntp.org

Command Definition:

Action	Definition
get	`get ntp1`
response	`val ntp1 <"ntp-address">`
set	`set ntp1 <"ntp-address">`
notify	`notify net.ntp1 <"ntp-address">`

Parameters:

Parameter	Description
`ntp-address`	String. IP address or URL or DNS name.

CLI Format Examples:

```
get ntp1
val ntp1 0.pool.ntp.org
```

```
set ntp1 0.pool.ntp.org
```

```
notify net.ntp1 0.pool.ntp.org
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

ntp2

Description: Configure second NTP (Network Time Protocol) server address.

Property Actions: set, get, notify

Default Value: 1.pool.ntp.org

Command Definition:

Action	Definition
get	`get ntp2`
response	`val ntp2 <"ntp-address">`
set	`set ntp2 <"ntp-address">`
notify	`notify net.ntp2 <"ntp-address">`

Parameters:

Parameter	Description

`ntp-address`	String. IP address or URL or DNS name.

CLI Format Examples:

```
get ntp2
val ntp2 1.pool.ntp.org
```

```
set ntp2 1.pool.ntp.org
```

```
notify net.ntp2 1.pool.ntp.org
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

ntp3

Description: Configure third NTP (Network Time Protocol) server address.

Property Actions: set, get, notify

Default Value: 2.pool.ntp.org

Command Definition:

Action	Definition
get	`get ntp3`
response	`val ntp3 <"ntp-address">`
set	`set ntp3 <"ntp-address">`
notify	`notify net.ntp3 <"ntp-address">`

Parameters:

Parameter	Description
`ntp-address`	String. IP address or URL or DNS name.

CLI Format Examples:

```
get ntp3
val ntp3 2.pool.ntp.org
```

```
set ntp3 2.pool.ntp.org
```

```
notify net.ntp3 2.pool.ntp.org
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

ntp4

Description: Configure fourth NTP (Network Time Protocol) server address.

Property Actions: set, get, notify

Default Value: none

Command Definition:

Action	Definition
get	`get ntp4`
response	`val ntp4 <"ntp-address">`
set	`set ntp4 <"ntp-address">`
notify	`notify net.ntp4 <"ntp-address">`

Parameters:

Parameter	Description
`ntp-address`	String. IP address or URL or DNS name.

CLI Format Examples:

```
get ntp4
val ntp4 3.pool.ntp.org
```

```
set ntp4 3.pool.ntp.org
```

```
notify net.ntp4 3.pool.ntp.org
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

dot1x-enabled

Description: Enables 802.1x authentication to access the network. When enabled, the 802.1x authentication type and the required credentials also need to be specified. MD5 authentication requires a username and password.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get dot1x-enabled`
response	`val dot1x-enabled <0|1>`
set	`set dot1x-enabled <0|1>`
notify	`notify net.dot1x-enabled <0|1>`

Parameters:

Parameter	Description
0	Disable 802.1x authentication
1	Enable 802.1x authentication

CLI Format Examples:

```
get dot1x-enabled
val dot1x-enabled 0
```

```
set dot1x-enabled 0
```

```
notify net.dot1x-enabled 0
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

dot1x-identity

Description: Set a 802.1x authentication identity – required for MD5.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get dot1x-identity`
response	`val dot1x-identity <"user-name">`
set	`set dot1x-identity <"user-name">`
notify	`notify net.dot1x-identity <"user-name">`

Parameters:

Parameter	Description
`username`	String.

CLI Format Examples:

```
get dot1x-identity
val dot1x-identity testuser
```

```
set dot1x-identity testuser
```

```
notify net.dot1x-identity testuser
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

dot1x-password

Description: Set a 802.1x authentication password – required for MD5.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get dot1x-password`
response	`val dot1x-password <"password">`
set	`set dot1x-password <"password">`
notify	`notify net.dot1x-password <"password">`

Parameters:

Parameter	Description
`password`	String.

CLI Format Examples:

```
get dot1x-password
val dot1x-password 5321
```

```
set dot1x-password 5321
```

```
notify net.dot1x-password 5321
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

5.2.6.2 Network Commands

set net-commit

Description: Apply network configuration changes. You must send the "net-commit" command to apply any changes to the network settings.

Command Definition:

Action	Definition
execute	`set net-commit`

Parameters: None

CLI Format Examples:

```
set net-commit
```

Supported Products: CS700-AV, CS700-SP
Available in API Version: 1.0

5.2.7 VoIP

5.2.7.1 VoIP Properties

registrar

Description: The IP address or DNS name of the SIP registrar server. Required for VoIP call support.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get registrar`
response	`val registrar <"address">`
set	`set registrar <"address">`
notify	`notify voip.registrar <"address">`

Parameters:

Parameter	Description
`address`	String. IP address or DNS name.

CLI Format Examples:

```
get registrar
val registrar 200.200.210.150
```

```
set registrar 200.200.210.150
```

```
notify voip.registrar 200.200.210.150
```

Supported Products: CS700-SP
Available in API Version: 1.1

registrar-backup

Description: The IP address or DNS name of a failover SIP registrar. It should be configured with the failover or secondary SIP registrar IP address or domain name if applicable. If no failover or secondary SIP registrar is present in the VoIP infrastructure, this field should be left blank. When this field is specified, the phone will register with the primary SIP registrar (the 'registrar' property) if it is accessible. If the primary SIP registrar becomes inaccessible via UDP or TCP, the phone will attempt to register with the backup registrar. If the phone successfully registered with the backup registrar, it will switch to the backup registrar to perform outgoing calls and receive incoming calls. While the phone is registered with the backup registrar, it will monitor the primary SIP registrar connection. Once the primary SIP registrar becomes available again, the phone will roll back to register with the primary registrar and route SIP traffic from/to the primary registrar.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get registrar-backup`
response	`val registrar-backup <"address">`
set	`set registrar-backup <"address">`
notify	`notify voip.registrar-backup <"address">`

Parameters:

Parameter	Description
address	String. IP address or DNS name.

CLI Format Examples:

```
get registrar-backup
val registrar-backup 200.200.210.151
```

```
set registrar-backup 200.200.210.151
```

```
notify voip.registrar-backup 200.200.210.151
```

Supported Products: CS700-SP
Available in API Version: 1.1

realm

Description: Realm of the credential to authenticate against the VoIP server. The value here must match the realm sent by the server in the WWW-Authenticate or Proxy-Authenticate header in the 401/407 response. An asterisk ('*') causes the endpoint to respond to any realms.

Property Actions: set, get, notify

Default Value: *

Command Definition:

Action	Definition
get	`get realm`
response	`val realm <"realm">`
set	`set realm <"realm">`
notify	`notify voip.realm <"realm">`

Parameters:

Parameter	Description
realm	Realm string

CLI Format Examples:

```
get realm
val realm *
```

```
set realm *
```

```
notify voip.realm *
```

Supported Products: CS700-SP
Available in API Version: 1.1

proxy

Description: The outbound SIP proxy server's IP address or name. If there are multiple SIP proxies, separate the addresses by a comma. Also note that if the allow strict routing option is set and you have a SIP proxy that is configured for loose routing, add the designation after the proxy's address, for example, '10.134.129.101;lr'.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get proxy`
response	`val proxy <"sip-proxy">`
set	`set proxy <"sip-proxy">`
notify	`notify voip.proxy <"sip-proxy">`

Parameters:

Parameter	Description
`sip-proxy`	String. IP address or name. Separate multiple addresses by a comma. Add ";lr" suffix for loose routing.

CLI Format Examples:

```
get proxy
val proxy 10.123.234.14;lr
```

```
set proxy 10.123.234.14;lr
```

```
notify voip.proxy 10.123.234.14;lr
```

Supported Products: CS700-SP
Available in API Version: 1.1

reg-use-proxy

Description: Indicates whether the SIP proxy server(s) specified in "proxy" property should be used when registering. Selecting this option will add the listed proxy server(s) to the route headers of the SIP REGISTER request.
Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get reg-use-proxy`
response	`val reg-use-proxy <0..3>`
set	`set reg-use-proxy <0..3>`
notify	`notify voip.reg-use-proxy <0..3>`

Parameters:

Parameter	Description
0	No proxy
1	Outbound only
2	Acc only
3	All

CLI Format Examples:

```
get reg-use-proxy
val reg-use-proxy 0
```

```
set reg-use-proxy 0
```

```
notify voip.reg-use-proxy 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

user

Description: SIP username for the account used to authenticate with the SIP registrar and proxies. Required for VoIP support.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get user`
response	`val user <"user-name">`
set	`set user <"user-name">`
notify	`notify voip.user <"user-name">`

Parameters:

Parameter	Description
`user-name`	String. SIP username.

CLI Format Examples:

```
get user
val user testuser
```

```
set user testuser
```

```
notify voip.user testuser
```

Supported Products: CS700-SP
Available in API Version: 1.1

password

Description: SIP password for the account used to authenticate with the SIP registrar and proxies. Required for VoIP call support.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get password`
response	`val password <"password">`
set	`set password <"password">`
notify	`notify voip.password <"password">`

Parameters:

Parameter	Description
`password`	String. SIP user password.

CLI Format Examples:

```
get password
val password 1234
```

```
set password 1234
```

```
notify voip.password 1234
```

Supported Products: CS700-SP
Available in API Version: 1.1

id

Description: SIP ID used for SIP registration. If this field is left blank, the voip.user (User name) field will be used as the ID.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get id`
response	`val id <"id">`
set	`set id <"id">`
notify	`notify voip.id <"id">`

Parameters:

Parameter	Description

id	String. SIP ID.

CLI Format Examples:

```
get id
val id 7824
```

```
set id 7824
```

```
notify voip.id 7824
```

Supported Products: CS700-SP
Available in API Version: 1.1

name

Description: VoIP Display Name. The Display Name is shown when an outbound call is made. If no Display Name is provided, the User name will be used. Please note that the IP PBX might override the display name sent by the device and replace it with names configured in the PBX.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get name`
response	`val name <"name">`
set	`set name <"name">`
notify	`notify voip.name <"name">`

Parameters:

Parameter	Description
name	String. VoIP Display Name.

CLI Format Examples:

```
get name
val name Blake
```

```
set name Blake
```

```
notify voip.name Blake
```

Supported Products: CS700-SP
Available in API Version: 1.1

reg-timeout

Description: Registration Timeout is the optional timeout for SIP account registration, in seconds.
Property Actions: set, get, notify

Default Value: 60

Command Definition:

Action	Definition
get	`get reg-timeout`
response	`val reg-timeout <1..604800>`
set	`set reg-timeout <1..604800>`
notify	`notify voip.reg-timeout <1..604800>`

Parameters:

Parameter	Description
`1..604800`	Registration timeout in seconds

CLI Format Examples:

```
get reg-timeout
val reg-timeout 60
```

```
set reg-timeout 60
```

```
notify voip.reg-timeout 60
```

Supported Products: CS700-SP
Available in API Version: 1.1

rereg-delay

Description: Configure auto re-registration retry interval in seconds. If SIP registration is unsuccessful, this property specified the time duration between retry attempts in seconds.

Property Actions: set, get, notify

Default Value: 300

Command Definition:

Action	Definition
get	`get rereg-delay`
response	`val rereg-delay <1..604800>`
set	`set rereg-delay <1..604800>`
notify	`notify voip.rereg-delay <1..604800>`

Parameters:

Parameter	Description
`1..604800`	Auto re-registration retry interval in seconds

CLI Format Examples:

```
get rereg-delay
val rereg-delay 300
```

```
set rereg-delay 300
```

```
notify voip.rereg-delay 300
```

Supported Products: CS700-SP
Available in API Version: 1.1

use-timer

Description: Specify the preference for using SIP session keep-alive timers. During a SIP session, if SIP session timers are active, the SIP User Agent (UA) periodically sends INVITE or UPDATE requests (also called refresh requests) to keep the SIP session alive. The interval and use of the keep-alive is determined at call negotiation. If one of the UAs in a call does not receive the refresh request within the expiration timeout period, it will terminate the session

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get use-timer`
response	`val use-timer <0|1|2|3>`
set	`set use-timer <0|1|2|3>`
notify	`notify voip.use-timer <0|1|2|3>`

Parameters:

Parameter	Description
0	Inactive. Session Timers will not be used in any session, except if explicitly required in the remote request
1	Optional. Session Timers will be used in all sessions whenever the remote supports and uses it
2	Mandatory. Session Timers support will be a requirement for the remote to be able to establish a session
3	Always. Session Timers will always be used in all sessions, regardless whether the remote supports or uses it or not

CLI Format Examples:

```
get use-timer
val use-timer 1
```

```
set use-timer 1
```

```
notify voip.use-timer 1
```

Supported Products: CS700-SP

Available in API Version: 1.1

timer-se

Description: The expiration period (seconds) is the interval at which the phone will consider the SIP session timed out if it does not receive a refresh message from the remote phone. At call negotiation, the nodes will negotiate the expiration period to be used for the session. If the negotiated value is less than the session timer's minimum expiration, then the session timer minimum expiration is used instead. It is measured in seconds.

Property Actions: set, get, notify

Default Value: 1800 (30 minutes)

Command Definition:

Action	Definition
get	`get timer-se`
response	`val timer-se <90..604800>`
set	`set timer-se <90..604800>`
notify	`notify voip.timer-se <90..604800>`

Parameters:

Parameter	Description
`90..604800`	Session timer in seconds

CLI Format Examples:

```
get timer-se
val timer-se 1800
```

```
set timer-se 1800
```

```
notify voip.timer-se 1800
```

Supported Products: CS700-SP
Available in API Version: 1.1

timer-min-se

Description: SIP session timer minimal expiration period (seconds). This is the minimum period that the device will accept when negotiating the expiration period with the remote phone. If the session timer expiration duration is less than this value, this value is used instead. It is measured in seconds.

Property Actions: set, get, notify

Default Value: 90

Command Definition:

Action	Definition

get	`get timer-min-se`
response	`val timer-min-se <1..604800>`
set	`set timer-min-se <1..604800>`
notify	`notify voip.timer-min-se <1..604800>`

Parameters:

Parameter	Description
`1..604800`	Minimum session timer value in seconds

CLI Format Examples:

```
get timer-min-se
val timer-min-se 90
```

```
set timer-min-se 90
```

```
notify voip.timer-min-se 90
```

Supported Products: CS700-SP
Available in API Version: 1.1

use-100rel

Description: Implements reliable SIP provisional responses. SIP is a request-response type of protocol with two types of responses: provisional and final. Final responses are sent reliably, using an ACK to ensure receipt. Provisional responses by default are not sent reliably and do not require an ACK; however, in some cases, such as for PSTN interoperability support, reliability of provisional types of responses is needed. Choose this option to add the PRACK (provisional ACK) message support for reliability.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get use-100rel`
response	`val use-100rel <0|1>`
set	`set use-100rel <0|1>`
notify	`notify voip.use-100rel <0|1>`

Parameters:

Parameter	Description
0	Disable reliable provisional responses
1	Enable reliable provisional responses

CLI Format Examples:

```
get use-100rel
```

```
val use-100rel 0
```

```
set use-100rel 0
```

```
notify voip.use-100rel 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

auto-update-nat

Description: Configure whether SIP traversal behind NAT is disabled. Used for device behind a symmetric NAT (Network Address Translation). When enabled, the device will keep track of the public IP address from the response of the REGISTER request. If it detects that the address has changed, it will unregister the current Contact, update the Contact with the transport address obtained from the Via header, and register a new Contact to the SIP registrar. This option will also update the public name of the UDP transport if STUN is configured.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
get	`get auto-update-nat`
response	`val auto-update-nat <0|1>`
set	`set auto-update-nat <0|1>`
notify	`notify voip.auto-update-nat <0|1>`

Parameters:

Parameter	Description
0	Disable NAT
1	Enable NAT

CLI Format Examples:

```
get auto-update-nat
val auto-update-nat 1
```

```
set auto-update-nat 1
```

```
notify voip.auto-update-nat 1
```

Supported Products: CS700-SP
Available in API Version: 1.1

dtmf-method

Description: DTMF signaling method.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get dtmf-method`
response	`val dtmf-method <0|1|2>`
set	`set dtmf-method <0|1|2>`
notify	`notify voip.dtmf-method <0|1|2>`

Parameters:

Parameter	Description
0	RTP -- RFC2833
1	SIP INFO
2	In-band

CLI Format Examples:

```
get dtmf-method
val dtmf-method 0
```

```
set dtmf-method 0
```

```
notify voip.dtmf-method 0
```

Supported Products: CS700-SPed to specify the dynamic RTP payload type for DTMF signaling via RTP.
Property Actions: set, get, notify

Default Value: 96

Command Definition:

Action	Definition
get	`get dtmf-rtp-payload-type`
response	`val dtmf-rtp-payload-type <96..127>`
set	`set dtmf-rtp-payload-type <96..127>`
notify	`notify voip.dtmf-rtp-payload-type <96..127>`

Parameters:

Parameter	Description
96..127	RTP packet type

CLI Format Examples:

```
get dtmf-rtp-payload-type
val dtmf-rtp-payload-type 96
```

```
set dtmf-rtp-payload-type 0
```

```
notify voip. dtmf-rtp-payload-type 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

media-onhold-method

Description:	The Media on Hold method setting allows switching the Media on Hold behavior between the different RFC definitions.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get media-onhold-method`
response	`val media-onhold-method <0|1>`
set	`set media-onhold-method <0|1>`
notify	`notify voip.media-onhold-method <0|1>`

Parameters:

Parameter	Description
0	M line only -- RFC3264. Implements the functionality according to RFC3264. In the INVITE message SDP body, the attribute 'a=sendonly' is set to a designated media stream to put media on-hold
1	M and C line -- RFC2543. Implements the functionality according to RFC2543. In the INVITE message SDP body, the connection line ip is set to '0.0.0.0' (e.g. 'c= IN IP4 0.0.0.0'), and the attribute 'a=inactive' is added

CLI Format Examples:

```
get media-onhold-method
val media-onhold-method 0
```

```
set media-onhold-method 0
```

```
notify voip.media-onhold-method 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

use-srtp

Description:	Controls Secure Real-time Transport Protocol (SRTP) usage.
Property Actions:	set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get use-srtp`
response	`val use-srtp <0|1|2>`
set	`set use-srtp <0|1|2>`
notify	`notify voip.use-srtp <0|1|2>`

Parameters:

Parameter	Description
0	Disabled -- Do not use SRTP; always use RTP
1	Optional -- Use the optional disposition for SRTP in SDP; if the remote end supports SRTP then use SRTP; otherwise use RTP
2	Mandatory -- Force use of SRTP; if the remote end does not support SRTP the call does not connect

CLI Format Examples:

```
get use-srtp
val use-srtp 0
```

```
set use-srtp 0
```

```
notify voip.use-srtp 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

rtp-port

Description: Base port number for VoIP RTP. RTP is originated and received on even port numbers, and the associated RTCP uses the next higher odd port number.
Property Actions: set, get, notify

Default Value: 4000

Command Definition:

Action	Definition
get	`get rtp-port`
response	`val rtp-port <1024..65535>`
set	`set rtp-port <1024..65535>`
notify	`notify voip.rtp-port <1024..65535>`

Parameters:

Parameter	Description
`1024..65535`	Port number

CLI Format Examples:

```
get rtp-port
val rtp-port 4000
```

```
set rtp-port 4000
```

```
notify voip.rtp-port 4000
```

Supported Products: CS700-SP
Available in API Version: 1.1

set-qos

Description: Option to enable QoS (Quality of Service) tagging for SIP and media. For layer 3, at the Internet layer, the DiffServ (Differentiated Services) precedence level is Class 3. The Differentiated Services Code Point (DSCP) in the IP header is set to 24 (0x18). For layer 2, IEEE 802.1p tagging is supported.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get set-qos`
response	`val set-qos <0|1>`
set	`set set-qos <0|1>`
notify	`notify voip.set-qos <0|1>`

Parameters:

Parameter	Description
0	Disable QoS
1	Enable QoS

CLI Format Examples:

```
get set-qos
val set-qos 0
```

```
set set-qos 0
```

```
notify voip.set-qos 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

udp-tcp-selection

Description: Transport that will be used for SIP messages.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get udp-tcp-selection`
response	`val udp-tcp-selection <0|1>`
set	`set udp-tcp-selection <0|1>`
notify	`notify voip.udp-tcp-selection <0|1>`

Parameters:

Parameter	Description
0	UDP
1	TCP

CLI Format Examples:

```
get udp-tcp-selection
val udp-tcp-selection 0
```

```
set udp-tcp-selection 0
```

```
notify voip.udp-tcp-selection 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

local-port

Description: Specifies the local port for SIP transport.

Property Actions: set, get, notify

Default Value: 5060

Command Definition:

Action	Definition
get	`get local-port`
response	`val local-port <1024..65535>`
set	`set local-port <1024..65535>`
notify	`notify voip.local-port <1024..65535>`

Parameters:

Parameter	Description

`1024..65535`	Port number

CLI Format Examples:

```
get local-port
val local-port 5060
```

```
set local-port 5060
```

```
notify voip.local-port 5060
```

Supported Products: CS700-SP
Available in API Version: 1.1

ip-addr

Description: Configure optional address to advertise as the address of this transport. Can specify any address or hostname for this field. For example, it can point to the public address of a NAT router where port mappings have been configured for SIP..

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get ip-addr`
response	`val ip-addr <"ip-address">`
set	`set ip-addr <"ip-address">`
notify	`notify voip.ip-addr <"ip-address">`

Parameters:

Parameter	Description
`ip-address`	IP address

CLI Format Examples:

```
get ip-addr
val ip-addr 192.168.1.103
```

```
set ip-addr 192.168.1.103
```

```
notify voip.ip-addr 192.168.1.103
```

Supported Products: CS700-SP
Available in API Version: 1.1

bound-addr

Description: Optional address where the socket should be bound to. This option should only be used to selectively bind the socket to a particular interface, and should not be used to set the published address of a transport – the *ip-addr* attribute should be used for that purpose.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get bound-addr`
response	`val bound-addr <"ip-address">`
set	`set bound-addr <"ip-address">`
notify	`notify voip.bound-addr <"ip-address">`

Parameters:

Parameter	Description
`ip-address`	IP address

CLI Format Examples:

```
get bound-addr
val bound-addr 192.168.1.103
```

```
set bound-addr 192.168.1.103
```

```
notify voip.bound-addr 192.168.1.103
```

Supported Products: CS700-SP
Available in API Version: 1.1

no-refer-sub

Description: Enable or disable suppress subscription during transfer. When transferring a SIP call, the SIP REFER process automatically establishes a temporary event subscription to notify the party initiating the transfer about the receiver's status in handling the REFER. In some cases these event subscriptions and notifications are not needed, such as when forking is not used. Enable this option to suppress the automatic event subscriptions when transferring calls.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get no-refer-sub`
response	`val no-refer-sub <0|1>`
set	`set no-refer-sub <0|1>`
notify	`notify voip.no-refer-sub <0|1>`

Parameters:

Parameter	Description
0	Disable suppressing subscription
1	Enable suppressing subscription

CLI Format Examples:

```
get no-refer-sub
val no-refer-sub 0
```

```
set no-refer-sub 0
```

```
notify voip.no-refer-sub 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

min-size

Description:	Use compact SIP message format. The SIP protocol specifies that header field names can be in the full name form or in the abbreviated form. Abbreviated form is useful when messages might be too large to be carried on the available transport, for example when exceeding UDP's Maximum Transmission Unit (MTU). Enable this option to encode SIP headers in their short forms to reduce size. By default, the option is not enabled and SIP headers in outgoing messages will be encoded in their full names. (See SIP protocol standard, IETF RFC 3261.)
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get min-size`
response	`val min-size <0|1>`
set	`set min-size <0|1>`
notify	`notify voip.min-size <0|1>`

Parameters:

Parameter	Description
0	Do not use compact format
1	Use compact format.

CLI Format Examples:

```
get min-size
val min-size 0
```

```
set min-size 0
```

```
notify voip.min-size 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

allow-strict

Description: Allow strict routing for SIP registration proxies. By default, proxies specified for SIP registration will be configured as loose-routing proxies. The loose-routing designation will be automatically appended to each proxy address when the proxy is added to the SIP Route header. Older proxies may be strict-routing (see IETF RFC 2543), not supporting loose routing (see IETF RFC 3261). Enable this option if you are using strict-routing proxies. If this option is enabled and you are specifying one or more loose-routing proxies in the Proxy field, then you must manually add the suffix to each loose-routing proxy address. For example, "10.134.123.101;lr".

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get allow-strict`
response	`val allow-strict <0|1>`
set	`set allow-strict <0|1>`
notify	`notify voip.allow-strict <0|1>`

Parameters:

Parameter	Description
0	Enable strict routing
1	Disable strict routing

CLI Format Examples:

```
get allow-strict
val allow-strict 0
```

```
set allow-strict 0
```

```
notify voip.allow-strict 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

stun-srv

Description: Specifies the STUN (Session Traversal Utilities for NAT) server IP address or name to use to determine if the phone is behind a NAT, the type of NAT, and the public address of the phone. The field can contain a comma separated list of servers. Each server can be a domain name, host name, or IP address, and it may contain an optional port number. (For STUN see IETF RFC 5389.)

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get stun-srv`
response	`val stun-srv <"stun-address">`
set	`set stun-srv <"stun-address">`
notify	`notify voip.stun-srv <"stun-address">`

Parameters:

Parameter	Description
`stun-address`	String. IP address, domain name, or host name, and optional port number. Separate multiple addresses by comma.

CLI Format Examples:

```
get stun-srv
val stun-srv 10.123.145.15
```

```
set stun-srv 10.123.145.15
```

```
notify voip.stun-srv 10.123.145.15
```

Supported Products: CS700-SP
Available in API Version: 1.1

use-ice

Description: Enables the use of the ICE (Interactive Connectivity Establishment) protocol for NAT traversal. ICE takes advantage of STUN and TURN to identify candidates (IP addresses and ports) for communication, evaluating and prioritizing the candidate pairs to select the best route. Expensive candidates, such as using a media relay, are selected only as a last resort. (For ICE see IETF RFC 5245.)

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get use-ice`
response	`val use-ice <0|1>`
set	`set use-ice <0|1>`
notify	`notify voip.use-ice <0|1>`

Parameters:

Parameter	Description
0	Disable ICE
1	Enable ICE

CLI Format Examples:

```
get use-ice
val use-ice 0
```

```
set use-ice 0
```

```
notify voip.use-ice 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

ice-regular

Description:	ICE nomination method. When using ICE, select the preferred ICE Nomination Method. To validate candidate pairs (IP addresses and ports for the local and remote nodes), CS-700 sends STUN binding requests as part of the media connectivity tests. When a candidate is nominated for use, a STUN binding request is sent with a flag indicating that the candidate pair is nominated.
Property Actions:	set, get, notify
Default Value:	1

Command Definition:

Action	Definition
get	`get ice-regular`
response	`val ice-regular <0|1>`
set	`set ice-regular <0|1>`
notify	`notify voip.ice-regular <0|1>`

Parameters:

Parameter	Description
0	Regular nomination method. CS-700 validates candidate pairs with initial STUN binding requests, and then selects a valid candidate pair by sending another STUN binding request with a flag indicating that the pair is nominated.
1	Aggressive nomination method. CS-700 does not wait to set the nominated flag in a second STUN binding request, but rather sets the flag in the STUN binding requests for all of the candidate pairs. The ICE processing completes when the first pair successfully passes connectivity checks. The aggressive method is faster but does not always result in the optimal path being selected.

CLI Format Examples:

```
get ice-regular
val ice-regular 1
```

```
set ice-regular 1
```

```
notify voip.ice-regular 1
```

Supported Products: CS700-SP
Available in API Version: 1.1

ice-max-hosts

Description:	Maximum number of ICE host candidate. An ICE host candidate is an actual local transport address in the host. Host transport addresses are obtained by binding to attached network interfaces. These interfaces include both physical interfaces and virtual interfaces such as VPN. This option specifies the maximum number of local ICE host candidates that may be used in evaluating candidate pairs when determining the best route.
Property Actions:	set, get, notify
Default Value:	5

Command Definition:

Action	Definition
Get	`get ice-max-hosts`
Response	`val ice-max-hosts <0..10>`
Set	`set ice-max-hosts <0..10>`
Notify	`notify voip.ice-max-hosts <0..10>`

Parameters:

Parameter	Description
0	No maximum
1..10	Maximum number of host candidates

CLI Format Examples:

```
get ice-max-hosts
val ice-max-hosts 5
```

```
set ice-max-hosts 5
```

```
notify voip.ice-max-hosts 5
```

Supported Products: CS700-SP
Available in API Version: 1.1

ice-no-rtcp

Description:	Option to not disable the RTCP component in ICE.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
Get	`get ice-no-rtcp`
Response	`val ice-no-rtcp <0|1>`
Set	`set ice-no-rtcp <0|1>`
Notify	`notify voip.ice-no-rtcp <0|1>`

Parameters:

Parameter	Description
0	Do not disable RTCP
1	Keep RTCP disabled

CLI Format Examples:

```
get ice-no-rtcp
val ice-no-rtcp 0
```

```
set ice-no-rtcp 0
```

```
notify voip.ice-no-rtcp 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

use-turn

Description:	Enables the use of a TURN (Traversal Using Relay NAT) relay when using ICE. A TURN relay is a media relay server residing on the public internet which can relay media data packet between clients. TURN relays are used when other preferred mechanisms are not available, such as STUN or direct connectivity. If TURN is enabled, the other TURN settings (server, username and password) must also be specified.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
Get	`get use-turn`
Response	`val use-turn <0|1>`
Set	`set use-turn <0|1>`
Notify	`notify voip.use-turn <0|1>`

Parameters:

Parameter	Description
0	Disable TURN
1	Enable TURN

CLI Format Examples:

```
get use-turn
val use-turn 0
```

```
set use-turn 0
```

```
notify voip.use-turn 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

turn-srv

Description: TURN server domain name or hostname, and port. The format is either 'DOMAIN:PORT' or 'HOST:PORT'

Property Actions: set, get, notify

Command Definition:

Action	Definition
Get	`get turn-srv`
Response	`val turn-srv <"address">`
Set	`set turn-srv <"address">`
Notify	`notify voip.turn-srv <"address">`

Parameters:

Parameter	Description
`Address`	String. The format is either 'DOMAIN:PORT' or 'HOST:PORT'.

CLI Format Examples:

```
get turn-srv
val turn-srv 200.200.210.153:5349
```

```
set turn-srv 200.200.210.153:5349
```

```
notify voip.turn-srv 200.200.210.153:5349
```

Supported Products: CS700-SP
Available in API Version: 1.1

turn-user

Description: Username to authenticate against the TURN server.

Property Actions: set, get, notify

Command Definition:

Action	Definition
Get	`get turn-user`
Response	`val turn-user <"user-name">`
Set	`set turn-user <"user-name">`
Notify	`notify voip.turn-user <"user-name">`

Parameters:

Parameter	Description
`user-name`	String.

CLI Format Examples:

```
get turn-user
val turn-user testuser
```

```
set turn-user testuser
```

```
notify voip.turn-user testuser
```

Supported Products: CS700-SP
Available in API Version: 1.1

turn-passwd

Description: Password to authenticate against the TURN server.

Property Actions: set, get, notify

Command Definition:

Action	Definition
Get	`get turn-passwd`
Response	`val turn-passwd <"password">`
Set	`set turn-passwd <"password">`
Notify	`notify voip.turn-passwd <"password">`

Parameters:

Parameter	Description
`Password`	String.

CLI Format Examples:

```
get turn-passwd
val turn-passwd 1234
```

```
set turn-passwd 1234
```

```
notify voip.turn-passwd 1234
```

Supported Products: CS700-SP
Available in API Version: 1.1

turn-tcp

Description: Configure whether to use TCP on TURN relay; otherwise use UDP.
Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
Get	`get turn-tcp`
Response	`val turn-tcp <0|1>`
Set	`set turn-tcp <0|1>`
Notify	`notify voip.turn-tcp <0|1>`

Parameters:

Parameter	Description
0	Do not use TCP; use UDP
1	Use TCP

CLI Format Examples:

```
get turn-tcp
val turn-tcp 0
```

```
set turn-tcp 0
```

```
notify voip.turn-tcp 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

codec1

Description: Highest prioritized codec. At least one codec different from “None” has to be selected in voip.codec1-voip.codec5.

Property Actions: set, get, notify

Default Value: 1

Command Definition:

Action	Definition
Get	`get codec1`
Response	`val codec1 <0..5>`
Set	`set codec1 <0..5>`

Notify	`notify voip.codec1 <0..5>`

Parameters:

Parameter	Description
0	None
1	G.722
2	G.711 u-law (PCMU)
3	G.711 A-law (PCMA)
4	G.726
5	G.729

CLI Format Examples:

```
get codec1
val codec1 1
```

```
set codec1 1
```

```
notify voip.codec1 1
```

Supported Products: CS700-SP
Available in API Version: 1.1

codec2

Description: Second highest prioritized codec. At least one codec different from “None” has to be selected in voip.codec1-voip.codec5.

Property Actions: set, get, notify

Default Value: 2

Command Definition:

Action	Definition
Get	`get codec2`
Response	`val codec2 <0..5>`
Set	`set codec2 <0..5>`
Notify	`notify voip.codec2 <0..5>`

Parameters:

Parameter	Description
0	None
1	G.722
2	G.711 u-law (PCMU)
3	G.711 A-law (PCMA)
4	G.726
5	G.729

CLI Format Examples:

```
get codec2
val codec2 2
```

```
set codec2 2
```

```
notify voip.codec2 2
```

Supported Products: CS700-SP
Available in API Version: 1.1

codec3

Description: Third highest prioritized codec. At least one codec different from “None” has to be selected in voip.codec1-voip.codec5.

Property Actions: set, get, notify

Default Value: 3

Command Definition:

Action	Definition
get	`get codec3`
response	`val codec3 <0..5>`
set	`set codec3 <0..5>`
notify	`notify voip.codec3 <0..5>`

Parameters:

Parameter	Description
0	None
1	G.722
2	G.711 u-law (PCMU)
3	G.711 A-law (PCMA)
4	G.726
5	G.729

CLI Format Examples:

```
get codec3
val codec3 3
```

```
set codec3 3
```

```
notify voip.codec3 3
```

Supported Products: CS700-SP
Available in API Version: 1.1

codec4

Description: Fourth highest prioritized codec. At least one codec different from "None" has to be selected in voip.codec1-voip.codec5.

Property Actions: set, get, notify

Default Value: 4

Command Definition:

Action	Definition
get	`get codec4`
response	`val codec4 <0..5>`
set	`set codec4 <0..5>`
notify	`notify voip.codec4 <0..5>`

Parameters:

Parameter	Description
0	None
1	G.722
2	G.711 u-law (PCMU)
3	G.711 A-law (PCMA)
4	G.726
5	G.729

CLI Format Examples:

```
get codec4
val codec4 4
```

```
set codec4 4
```

```
notify voip.codec4 4
```

Supported Products: CS700-SP
Available in API Version: 1.1

codec5

Description: Lowest prioritized codec. At least one codec different from "None" has to be selected in voip.codec1-voip.codec5.

Property Actions: set, get, notify

Default Value: 5

Command Definition:

Action	Definition
get	`get codec5`

response	`val codec5 <0..5>`
set	`set codec5 <0..5>`
notify	`notify voip.codec5 <0..5>`

Parameters:

Parameter	Description
0	None
1	G.722
2	G.711 u-law (PCMU)
3	G.711 A-law (PCMA)
4	G.726
5	G.729

CLI Format Examples:

```
get codec5
val codec5 5
```

```
set codec5 5
```

```
notify voip.codec5 5
```

Supported Products: CS700-SP
Available in API Version: 1.1

ptime

Description: The ptime (packetization interval) value for a codec determines the length of time in milliseconds represented by the media in an RTP packet which is used to transmit audio traffic.
Property Actions: set, get, notify

Default Value: 20

Command Definition:

Action	Definition
get	`get ptime`
response	`val ptime <10..60>`
set	`set ptime <10..60>`
notify	`notify voip.ptime <10..60>`

Parameters:

Parameter	Description
`10..60`	ptime interval in 10 ms increments

CLI Format Examples:

```
get ptime
```

```
val ptime 20
```

```
set ptime 20
```

```
notify voip.ptime 20
```

Supported Products: CS700-SP
Available in API Version: 1.1

mwi

Description:	Enable displaying the message waiting indicator (MWI) on the device and enable receiving message waiting notifications from the PBX. The PBX must be configured to support voice mail for the registered user in order for this feature to work properly.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get mwi`
response	`val mwi <0|1>`
set	`set mwi <0|1>`
notify	`notify voip.mwi <0|1>`

Parameters:

Parameter	Description
0	Disable MWI signaling
1	Enable MWI signaling

CLI Format Examples:

```
get mwi
val mwi 0
```

```
set mwi 0
```

```
notify voip.mwi 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

vm-number

Description:	The number that is dialed when voicemail is called from the UI.
Property Actions:	set, get, notify

Command Definition:

Action	Definition
get	`get vm-number`
response	`val vm-number <"vm-number">`
set	`set vm-number <"vm-number">`
notify	`notify voip.vm-number <"vm-number">`

Parameters:

Parameter	Description
`vm-number`	String. VoIP voicemail number.

CLI Format Examples:

```
get vm-number
val vm-number 7243
```

```
set vm-number 7243
```

```
notify voip.vm-number 7243
```

Supported Products: CS700-SP
Available in API Version: 1.1

vm-count

Description: Query VoIP voice mail count.

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get vm-count`
response	`val vm-count <"vm-count">`
notify	`notify voip.vm-count <"vm-count">`

Parameters:

Parameter	Description
`vm-count`	Number of VoIP voice mail messages.

CLI Format Examples:

```
get vm-count
val vm-count 2
```

```
notify voip.vm-count 2
```

Supported Products: CS700-SP
Available in API Version: 1.1

do-not-disturb

Description:	Configure do-not-disturb (DND) setting.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get do-not-disturb`
response	`val do-not-disturb <0|1>`
set	`set do-not-disturb <0|1>`
notify	`notify voip.do-not-disturb <0|1>`

Parameters:

Parameter	Description
0	Disable DND
1	Enable DND

CLI Format Examples:

```
get do-not-disturb
val do-not-disturb 0
```

```
set do-not-disturb 0
```

```
notify voip.do-not-disturb 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

auto-answer

Description:	Auto answer incoming VoIP calls. We recommend enabling this feature only for test purposes. If the phone is set to Do Not Disturb or if there are no available lines, the Forward rules will apply. If there are no Forward rules specified, the incoming call will be sent to voice mail. If voice mail is not supported, the call will be rejected.
Property Actions:	set, get, notify
Default Value:	0

Command Definition:

Action	Definition
get	`get auto-answer`
response	`val auto-answer <0|100..699>`
set	`set auto-answer <0|100..699>`
notify	`notify voip.auto-answer <0|100..699>`

Parameters:

Parameter	Description
0	Disable auto-answer
100..699	RESPONSE to send when answering

CLI Format Examples:

```
get auto-answer
val auto-answer 0
```

```
set auto-answer 0
```

```
notify voip.auto-answer 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

duration

Description: Specifies the maximum VoIP call duration in minutes. When the call duration reaches the maximum duration, the call will be automatically terminated.
Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get duration`
response	`val duration <0..10080>`
set	`set duration <0..10080>`
notify	`notify voip.duration <0..10080>`

Parameters:

Parameter	Description
0	No maximum
1..10080	Maximum VoIP call duration in minutes

CLI Format Examples:

```
get duration
val duration 0
```

```
set duration 0
```

```
notify voip.duration 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

dial-plan

Description: Specifies the VoIP dial plan string. See the User's Guide for a detailed description of dial plan settings.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get dial-plan`
response	`val dial-plan <"dial-plan">`
set	`set dial-plan <"dial-plan">`
notify	`notify voip.dial-plan <"dial-plan">`

Parameters:

Parameter	Description
`dialplan`	Dial plan string.

CLI Format Examples:

```
get dial-plan
val dial-plan 0|[49]xx|1[2-9]xx[2-9]xxxxxx
```

//The following allows only US–style 1 + area–code + local–number, but disallows area codes and local numbers
//starting with 0 or 1. It also allows 411, 911, and operator calls (0)

```
set dial-plan 0|[49]xx|1[2-9]xx[2-9]xxxxxx
```

```
notify voip.dial-plan 0|[49]xx|1[2-9]xx[2-9]xxxxxx
```

Supported Products: CS700-SP
Available in API Version: 1.1

always-forwarding

Description: Enable or disable forwarding all incoming VoIP calls to the specified number.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get always-forwarding`
response	`val always-forwarding <0|1>`
set	`set always-forwarding <0|1>`
notify	`notify voip.always-forwarding <0|1>`

Parameters:

Parameter	Description
0	Disable
1	Enable

CLI Format Examples:

```
get always-forwarding
val always-forwarding 0
```

```
set always-forwarding 0
```

```
notify voip.always-forwarding 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

always-forwarding-num

Description: Forward all incoming VoIP calls to the specified number.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get always-forwarding-num`
response	`val always-forwarding-num <"fwd-number">`
set	`set always-forwarding-num <"fwd-number">`
notify	`notify voip.always-forwarding-num <"fwd-number">`

Parameters:

Parameter	Description
`fwd-number`	String. VoIP dialing number.

CLI Format Examples:

```
get always-forwarding-num
val always-forwarding-num 7823
```

```
set always-forwarding-num 7823
```

```
notify voip.always-forwarding-num 7823
```

Supported Products: CS700-SP
Available in API Version: 1.1

busy-forwarding

Description: Enable or disable forwarding incoming calls to the specified number if the local phone is in 'Do Not Disturb' mode or if both lines are busy.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get busy-forwarding`
response	`val busy-forwarding <0|1>`
set	`set busy-forwarding <0|1>`
notify	`notify voip.busy-forwarding <0|1>`

Parameters:

Parameter	Description
0	Disable
1	Enable

CLI Format Examples:

```
get busy-forwarding
val busy-forwarding 0
```

```
set busy-forwarding 0
```

```
notify voip.busy-forwarding 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

busy-forwarding-num

Description: Forward incoming calls to the specified number if the local phone is in 'Do Not Disturb' mode or if both lines are busy.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get busy-forwarding-num`
response	`val busy-forwarding-num <"fwd-number">`
set	`set busy-forwarding-num <"fwd-number">`
notify	`notify voip.busy-forwarding-num <"fwd-number">`

Parameters:

Parameter	Description

`fwd-number`	String. VoIP dialing number.

CLI Format Examples:

```
get busy-forwarding-num
val busy-forwarding-num 7823
```

```
set busy-forwarding-num 7823
```

```
notify voip.busy-forwarding-num 7823
```

Supported Products: CS700-SP
Available in API Version: 1.1

noanswer-forwarding

Description: Enable or disable forwarding incoming VoIP calls to the specified number if the call is not answered within the duration specified in the 'noanswer-delay' attribute.

Property Actions: set, get, notify

Default Value: 0

Command Definition:

Action	Definition
get	`get noanswer-forwarding`
response	`val noanswer-forwarding <0|1>`
set	`set noanswer-forwarding <0|1>`
notify	`notify voip.noanswer-forwarding <0|1>`

Parameters:

Parameter	Description
0	Disable
1	Enable

CLI Format Examples:

```
get noanswer-forwarding
val noanswer-forwarding 0
```

```
set noanswer-forwarding 0
```

```
notify voip.noanswer-forwarding 0
```

Supported Products: CS700-SP
Available in API Version: 1.1

noanswer-forwarding-num

Description: Forward incoming VoIP calls to the specified number if the call is not answered within the duration specified in the 'noanswer-delay' attribute.

Property Actions: set, get, notify

Command Definition:

Action	Definition
get	`get noanswer-forwarding-num`
response	`val noanswer-forwarding-num <"fwd-number">`
set	`set noanswer-forwarding-num <"fwd-number">`
notify	`notify voip.noanswer-forwarding-num <"fwd-number">`

Parameters:

Parameter	Description
`fwd-number`	String. VoIP dialing number.

CLI Format Examples:

```
get noanswer-forwarding-num
val noanswer-forwarding-num 7823
```

```
set noanswer-forwarding-num 7823
```

```
notify voip.noanswer-forwarding-num 7823
```

Supported Products: CS700-SP
Available in API Version: 1.1

noanswer-delay

Description: Number of seconds to wait before forwarding an unanswered incoming call to the 'noanswer-forwarding-num' number.

Property Actions: set, get, notify

Default Value: 10

Command Definition:

Action	Definition
get	`get noanswer-delay`
response	`val noanswer-delay <2..30>`
set	`set noanswer-delay <2..30>`
notify	`notify voip.noanswer-delay <2..30>`

Parameters:

Parameter	Description
`2..30`	Seconds to wait before forwarding a call

CLI Format Examples:

```
get noanswer-delay
val noanswer-delay 10
```

```
set noanswer-delay 10
```

```
notify voip.noanswer-delay 10
```

Supported Products: CS700-SP
Available in API Version: 1.1

5.2.7.2 VoIP Statuses

registration

Description: Query SIP registration status.

Property Actions: get, notify

Command Definition:

Action	Definition
get	`get registration`
response	`val registration <0..999>`
notify	`notify voip.registration <0..999>`

Parameters:

Parameter	Description
0	Unregistered to SIP server
999	Registration is in process
200	Registered to SIP server
Code other than above	Registration attempt failed

CLI Format Examples:

```
get registration
val registration 999
```

```
notify voip.registration 200
```

Supported Products: CS700-SP
Available in API Version: 1.1

5.2.7.3 VoIP Commands

set dtmf

Description: Send DTMF digit on given VoIP line during an active call.

Command Definition:

Action	Definition
execute	`set dtmf <"voip-line-id"> <"digit">`

Parameters:

Parameter	Description
1	VoIP line 1
2	VoIP line 2
3	VoIP line 3 (used for transfer)
digit	dtmf digit, use - to stop previous DTMF digit

CLI Format Examples:

```
set dtmf 1 5
set dtmf 1 -
```

Supported Products: CS700-SP
Available in API Version: 1.1

set voip-commit

Description: Apply voip configuration changes. You must send the "voip-commit" command to apply any changes to the VoIP settings.

Command Definition:

Action	Definition
execute	`set voip-commit`

Parameters: None

CLI Format Examples:

```
set voip-commit
```

Supported Products: CS700-SP
Available in API Version: 1.1

6 Provisioning

Devices that have an IP connection can be provisioned with configuration settings and can be upgraded through the use of DHCP options 66 or 150. These DHCP options are used to specify an HTTP, TFTP or FTP server from which a device can download configuration files. These files can also specify a firmware upgrade bundle and version information.

When a device starts, it sends a request to the DHCP server for an IP address. In addition to the IP address, the device can query the DHCP server for additional information configured on the DHCP server. Options 66 and 150 provide the address or hostname of one or more HTTP, TFTP or FTP servers.

A TFTP server needs to be configured as:

```
tftp://<server address>
```

An HTTP server needs to be configured as:

```
http://<server address>
```

An FTP server needs to be configured as:

```
ftp://<server address>/user=”<username>”&pass=”<password>”
```

where <username> and <password> are the username and password required to access the FTP server.

Once the CS-700 receives address information for a provisioning server, it tries to access configuration files on that server to auto-configure the device. The configuration file for a specific device needs to be named as follows:

```
<MAC-address of device>.xml.
```

A device provisioning file can include other provisioning files, enabling an organization to construct a hierarchy of configuration settings. For example, you typically create small device-specific files that contain the few configuration settings unique to a device. And you create group-specific files that are shared by devices in a group, and an enterprise-wide file with settings that a shared by the entire enterprise.

Any configuration data provided from the configuration files will overwrite configuration data previously stored on the device.

A list of the available parameters with a short description is provided below in the section titled, “Provisioning File Parameters.”

6.1 Provisioning Files

The provisioning file is in xml format. The syntax of the file is:

```
<provisioning [include=”{comma separated list of include files}”]>
        [<config {list of config parameters} />]
        [<firmware version=”{firmware version}”>{firmware file name}</firmware>]
</provisioning>
```

where [] indicates an optional parameter and {} indicate a description of the content to be provided.

The “config” tag contains all configuration settings that pertain to the device, while the “firmware” tag includes the latest firmware version and firmware file name.

Sample device configuration file

Filename: F0DEF1A064E6.xml for the unit with the MAC address f0:DE:F1:A0:64:E6

```
<provisioning include="enterprise.xml, department.xml">
	<config voip.id="test" voip.name="test" voip.user="test" voip.password="test" />
</provisioning>
```

Sample enterprise configuration file

Filename: enterprise.xml

```
<provisioning>
	<firmware version="2.6.0.294">CS-700-2-6-0-294.bundle</firmware>
	<config
		voip.registrar="200.200.210.240"
		voip.registrar-backup=""
		voip.realm="*"
		voip.reg-use-proxy="0"
		voip.proxy=""
	/>
</provisioning>
```

Sample department configuration file

Filename: department.xml

```
<provisioning>
	<config
		net.ntp1="0.pool.ntp.org"
		net.ntp2="1.pool.ntp.org"
		sys.provisioning-interval="1440"
		voip.ptime="20"
		voip.codec1="1"
		voip.codec2="2"
		voip.codec3="3"
		voip.codec4="4"
		voip.codec5="5"
	/>
</provisioning>
```

6.2 Provisioning File Parameters

Property	**Type**	**Values**	**Default Value**	**Description**
sys.systemname	TEXT	System name string. Default value is the product name concatenated with the MAC address	Product name and MAC Address	Specifies the system name.
sys.md5-password	TEXT	Password string.	7386 (as MD5 sum) see command	Administrator password for the device, stored as MD5 sum.

sys.enable-btn-camera	BOOLEAN	0 : Disable button 1 : Enable button	1	Enable/disable the camera button on the main unit.
sys.enable-btn-audio	BOOLEAN	0 : Disable button 1 : Enable button	1	Enable/disable the microphone mute button on the main unit.
sys.enable-btn-volume	BOOLEAN	0 : Disable button 1 : Enable button	1	Enable/disable the speaker volume buttons on the main unit.
sys.enable-btn-bluetooth	BOOLEAN	0 : Disable button 1 : Enable button	1	Enable/disable the Bluetooth button on the main unit.
sys.enable-led-call	BOOLEAN	0 : Disable LED 1 : Enable LED	1	Enable/disable the call state LED on the main unit.
sys.region	INDEXED_OPTION	1 : Argentina 2 : Australia 3 : Belgium 4 : Brazil 5 : Canada 6 : Chile 7 : China 8 : Costa Rica 9 : France 10 : Germany 11 : Hong Kong 12 : India 13 : Israel 14 : Italy 15 : Japan 16 : Malaysia 17 : Mexico 18 : New Zealand 19 : Singapore 20 : South Africa 21 : Taiwan 22 : United Kingdom 23 : USA 24 : Venezuela	23	Region in which device is operating, setting by index.
sys.require-https	BOOLEAN	0 : Do not require HTTPS 1 : Require HTTPS	0	Configure require https setting
sys.power-saving-mode	BOOLEAN	0 : Power save mode is off 1 : Power save mode is on	1	Configure power saving mode setting.
sys.power-saving-time	NUMBER	timeout : Power save timeout in minutes	20	Configure power saving time in minutes.

sys.provisioning-interval	NUMBER	1..44640 : Provisioning interval in minutes	1440 (1 day)	Specify the provisioning interval for the device, in minutes..
sys.snmp-enable	BOOLEAN	0 : Disable SNMP 1 : Enable SNMP	0	Enable or disable SNMP support..
sys.snmp-community	TEXT	Read-only community string.	public	Specifies the SNMP read-only community string used for queries from the server and transmitted traps. Read-only indicates the authorization level. The device does not support write operations initiated through SNMP.
sys.snmp-contact-name	TEXT	Contact name.		Specifies the contact name, typically the system administrator. This string is informational and can include an email address. It is not associated with traps.
sys.snmp-device-location	TEXT	Device location.		Specifies the location of the device for informational purposes.
sys.snmp-address	TEXT	IP address or DNS name.		Specifies SNMP server address to which traps will be sent. Leave blank to disable traps.
sys.recent-call-enabled	BOOLEAN	0 : Disable recent call list 1 : Enable recent call list	1	Enables or disable the Recent Calls list.
sys.ui-mask	NUMBER	0 : All menus are enabled in the dialer 4 : Do not allow modifications of call forwarding 8 : Do not allow modifications of call history 16 : Do not allow modifications of contacts 32 : Do not allow modifications of DND	0	Enable or disable menus in dialer via a bit mask

audio.eq	NUMBER	1 : Voice 2 : Bass boost 3 : Treble boost	1	EQ setting used to adjust the speaker frequencies to your preference for the room and the types of calls.
audio.high-pass-filter	INDEXED_OPTION	0 : None 1 : 110 Hz 2 : 140 Hz 3 : 175 Hz 4 : 225 Hz	0	High-Pass filter setting. High-Pass filters are provided to adjust to room and application requirements. Use the High-Pass filter in rooms that have a high background noise in the low frequencies (air conditioning, lighting fixtures, etc.). All filters are bi-quad filters, reducing the signal by 6dB per octave.
audio.analog-audio-in-mode	BOOLEAN	0 : Automatic gain setting 1 : Manually specify gain setting	0	For the TV audio in port, select the gain setting mode, either auto or manual.
audio.analog-audio-in-gain	NUMBER	-12.0..40.0 : Manual gain setting	4.5	For the TV audio in port, if analog-audio-in-mode is manual, then this is the gain value in dB. This value can be increased or decreased by 0.5dB.
audio.wireless-omni-mic	BOOLEAN	0 : Disable wireless omni mic 1 : Enable wireless omni mic	0	Enable/disable additional audio processing for wireless omni-directional microphone.
audio.wireless-direct-mic	BOOLEAN	0 : Disable wireless direct mic 1 : Enable wireless direct mic	0	Enable/disable additional audio processing for wireless directional microphone.
audio.wireless-lapel-mic	BOOLEAN	0 : Disable wireless lapel mic 1 : Enable wireless lapel mic	0	Enable/disable additional audio processing for wireless lapel microphone.
audio.ring-tone	INDEXED_OPTION	0..5 : Ring-tone index	0	VoIP ring-tone selection.
audio.ringer-volume	TEXT	1..18 : Ringer volume level	13	Configure VoIP ringer volume.
audio.speaker-volume	TEXT	1..18 : Speak volume level	13	Configure speaker volume for calls.

camera.camera-ptz-home	TEXT	String format: <"pan"> <"tilt"> <"zoom"> pan: <-30..30> tilt: <-18..18> zoom: <100..400>	0 0 100	Default PTZ settings for home position. When the device detects that the upstream USB connection has been established, either at startup or after a USB disconnection, it will revert to the default PTZ settings.
camera.camera-flicker	NUMBER	1 : 50Hz 2 : 60Hz	2	Camera's flicker setting.
camera.camera-image-defaults	TEXT	String format: <"backlight"> <"brightness"> <"contrast"> <"saturation"> <"sharpness"> <"hue"> <"gamma"> backlight: <0..5> brightness: <0..250> contrast: <60..140> saturation: <50..150> sharpness: <0..255> hue: <0..180> gamma: <1..255>	0 125 110 100 50 90 255	Camera image default settings. These are Backlight, Brightness, Contrast, Saturation, Sharpness, Hue, and Gamma.
bt.bt-enable	BOOLEAN	0 : disable 1 : enable	1	Enable or disable Bluetooth basic-rate.
bt.bt-name	TEXT	Name of device. Default value is the product name concatenated with the system's BT MAC address	"Yamaha CS-700 " + BT MAC address	The Bluetooth basic-rate device name advertised over the air.
bt.bt-pin	TEXT	pin : 4-digit string	0000	The Bluetooth basic-rate pin for pairing. Pins are 4 digits.
bt.bt-pair-timeout	NUMBER	0 : No timeout 30, 60, 90, 120 : Pairing timeout in seconds	60	Bluetooth basic-rate pairing timeout in seconds. After this time, the pairing mode will stop.
bt.nfc- enable	BOOLEAN	0 : Disable NFC and NFC LED 1 : Enable NFC and NFC LED	1	Enable/disable NFC and the NFC logo LED on the main unit.

bt.bt-call-autojoin	BOOLEAN	0 : Disable BT call auto join 1 : Enable BT call auto join	0	Enable/disable a Bluetooth call to auto join a USB based conference when there is no dialer app in use.
bt.bt-auto-unpair-on-disconnect	BOOLEAN	0 : Disable BT auto unpair on disconnect 1 : Enable BT auto unpair on disconnect	0	Enable/disable whether a BT device is automatically unpaired when it disconnects from the CS-700.
net.dhcp	BOOLEAN	0 : Disable DHCP 1 : Enable DHCP	1	Enables or disables DHCP. If DHCP is disabled, the Static IP Address, Subnet Mask, and Default Gateway must be specified.
net.ip	TEXT	IP address		Current IP Address assigned to the device.
net.subnet	TEXT	IP address		Subnet Mask to determine the subnet to which the device belongs.
net.gateway	TEXT	IP address.		The Default Gateway is the device's default router on the IP network.
net.dns1	TEXT	IP address.		Address of the primary Domain Name System (DNS) server.
net.dns2	TEXT	IP address.		Address of the secondary Domain Name System (DNS) server.
net.vlan	INDEXED_ OPTION	0 : Manually specify VLAN ID 1 : Disable VLAN 2 : Automatically detect VLAN ID	2	Specifies VLAN behavior and support for the device.
net.vlan-id	NUMBER	1..4094 : VLAN ID	3	The manually specified LAN ID. If the VLAN mode is automatic, this is the detected VLAN ID if VLAN is active.

net.ntp1	TEXT	IP address or URL or DNS name.	0.pool.ntp.org	Configure first NTP (Network Time Protocol) server address.
net.ntp2	TEXT	IP address or URL or DNS name.	1.pool.ntp.org	Configure second NTP (Network Time Protocol) server address.
net.ntp3	TEXT	IP address or URL or DNS name.	2.pool.ntp.org	Configure third NTP (Network Time Protocol) server address.
net.ntp4	TEXT	IP address or URL or DNS name.		Configure fourth NTP (Network Time Protocol) server address.
net.dot1x-enabled	BOOLEAN	0 : Disable 802.1x authentication 1 : Enable 802.1x authentication	0	Enables 802.1x authentication to access the network. When enabled, the 802.1x authentication type and the required credentials also need to be specified. MD5 authentication requires a username and password.
net.dot1x-identity	TEXT	username : String.		Set a 802.1x authentication identity – required for MD5.
net.dot1x-password	TEXT	password : String.		Set a 802.1x authentication password – required for MD5.
voip.registrar	TEXT	IP address or DNS name.		The IP address or DNS name of the SIP registrar server. Required for VoIP call support.
voip.registrar-backup	TEXT	IP address or DNS name.		The IP address or DNS name of a failover SIP registrar.
voip.realm	TEXT	Realm string.	*	Realm of the credential to authenticate against the VoIP server. The value here must match the realm sent by the server in the WWW-Authenticate or Proxy-Authenticate header in the 401/407 response. An asterisk ('*') causes the endpoint to respond to any realms.

voip.proxy	TEXT	IP address or name. Separate multiple addresses by a comma. Add ";lr" suffix for loose routing.		The outbound SIP proxy server's IP address or name. If there are multiple SIP proxies, separate the addresses by a comma. Also note that if the allow strict routing option is set and you have a SIP proxy that is configured for loose routing, add the designation after the proxy's address, for example, '10.134.129.101;lr'.
voip.reg-use-proxy	INDEXED_ OPTION	0 : No proxy 1 : Outbound only 2 : Acc only 3 : All	0	Indicates whether the SIP proxy server(s) specified in "proxy" property should be used when registering. Selecting this option will add the listed proxy server(s) to the route headers of the SIP REGISTER request.
voip.user	TEXT	SIP username.		SIP username for the account used to authenticate with the SIP registrar and proxies. Required for VoIP support.
voip.password	TEXT	SIP user password.		SIP password for the account used to authenticate with the SIP registrar and proxies. Required for VoIP call support.
voip.id	TEXT	SIP ID.		SIP ID used for SIP registration. If this field is left blank, the voip.user (User name) field will be used as the ID.
voip.name	TEXT	VoIP Display Name.		VoIP Display Name. The Display Name is shown when an outbound call is made. If no Display Name is provided, the User name will be used. Please note that the IP PBX might override the Display name sent by the device and replace it with names configured in the PBX.

voip.reg-timeout	NUMBER	1..604800 : Registration timeout in seconds	60	Registration Timeout is the optional timeout for SIP account registration, in seconds.
voip.rereg-delay	NUMBER	1..604800 : Auto re-registration retry interval in seconds	300	Configure auto re-registration retry interval in seconds. If SIP registration is unsuccessful, this property specified the time duration between retry attempts in seconds.
voip.use-timer	INDEXED_ OPTION	0 : Inactive 1 : Optional 2 : Mandatory 3 : Always	1	Specify the preference for using SIP session keep-alive timers.
voip.timer-se	NUMBER	90..604800 : Session timer in seconds	1800 (30 minutes)	The expiration period (seconds) is the interval at which the phone will consider the SIP session timed out if it does not receive a refresh message from the remote phone.
voip.timer-min-se	NUMBER	1..604800 : Minimum session timer value in seconds	90	SIP session timer minimal expiration period (seconds). This is the minimum period that the device will accept when negotiating the expiration period with the remote phone. If the session timer expiration duration is less than this value, this value is used instead. It is measured in seconds.

voip.use-100rel	BOOLEAN	0 : Disable reliable provisional responses 1 : Enable reliable provisional responses	0	Implements reliable SIP provisional responses. SIP is a request-response type of protocol with two types of responses: provisional and final. Final responses are sent reliably, using an ACK to ensure receipt. Provisional responses by default are not sent reliably and do not require an ACK; however, in some cases, such as for PSTN interoperability support, reliability of provisional types of responses is needed. Choose this option to add the PRACK (provisional ACK) message support for reliability.
voip.auto-update-nat	BOOLEAN	0 : Disable NAT 1 : Enable NAT	1	Configure whether SIP traversal behind NAT is disabled. Used for device behind a symmetric NAT (Network Address Translation). When enabled, the device will keep track of the public IP address from the response of the REGISTER request. If it detects that the address has changed, it will unregister the current Contact, update the Contact with the transport address obtained from the Via header, and register a new Contact to the SIP registrar. This option will also update the public name of the UDP transport if STUN is configured.
voip.dtmf-method	TEXT	0 : RTP -- RFC2833 1 : SIP INFO 2 : In-band	0	DTMF signaling method.
voip.dtmf-rtp-payload-type	TEXT	96..127	96	Specify the dynamic RTP payload type for DTMF transport using RTP.

voip.media-onhold-method	BOOLEAN	0 : M line only -- RFC3264 1 : M and C line -- RFC2543	0	The Media on Hold method setting allows switching the Media on Hold behavior between the different RFC definitions.
voip.use-srtp	INDEXED_ OPTION	0 : Disabled -- Do not use SRTP; always use RTP 1 : Optional -- Use the optional disposition for SRTP in SDP; if the remote end supports SRTP then use SRTP; otherwise use RTP 2 : Mandatory -- Force use of SRTP; if the remote end does not support SRTP the call does not connect	0	Controls Secure Real-time Transport Protocol (SRTP) usage.
voip.rtp-port	NUMBER	1024..65535 : Port number	4000	Base port number for VoIP RTP. RTP is originated and received on even port numbers, and the associated RTCP uses the next higher odd port number.

voip.set-qos	BOOLEAN	0 : Disable QoS 1 : Enable QoS	0	Option to enable QoS (Quality of Service) tagging for SIP and media. For layer 3, at the Internet layer, the DiffServ (Differentiated Services) precedence level is Class 3. The Differentiated Services Code Point (DSCP) in the IP header is set to 24 (0x18). For layer 2, IEEE 802.1p tagging is supported.
voip.udp-tcp-selection	BOOLEAN	0 : UDP 1 : TCP	0	Transport that will be used for SIP messages.
voip.local-port	NUMBER	1024..65535 : Port number	5060	Specifies the local port for SIP transport.
voip.ip-addr	TEXT	ip-address : IP address		Configure optional address to advertise as the address of this transport. Can specify any address or hostname for this field. For example, it can point to the public address of a NAT router where port mappings have been configured for SIP..
voip.bound-addr	TEXT	ip-address : IP address		Configure bound IP address we intend to use.
voip.no-refer-sub	BOOLEAN	0 : Disable suppressing subscription 1 : Enable suppressing subscription	0	Enable or disable suppress subscription during transfer. When transferring a SIP call, the SIP REFER process automatically establishes a temporary event subscription to notify the party initiating the transfer about the receiver's status in handling the REFER. In some cases these event subscriptions and notifications are not needed, such as when forking is not used. Enable this option to suppress the automatic event subscriptions when transferring calls. (See IETF RFC 4488.)

voip.min-size	BOOLEAN	0 : Do not use compact format 1 : Use compact format.	0	Use compact SIP message format. The SIP protocol specifies that header field names can be in the full name form or in the abbreviated form. Abbreviated form is useful when messages might be too large to be carried on the available transport, for example when exceeding UDP's Maximum Transmission Unit (MTU). Enable this option to encode SIP headers in their short forms to reduce size. By default, the option is not enabled and SIP headers in outgoing messages will be encoded in their full names. (See SIP protocol standard, IETF RFC 3261.)
voip.allow-strict	BOOLEAN	0 : Enable strict routing 1 : Disable strict routing	0	Allow strict routing for SIP registration proxies. By default, proxies specified for SIP registration will be configured as loose-routing proxies. The loose-routing designation will be automatically appended to each proxy address when the proxy is added to the SIP Route header. Older proxies may be strict-routing (see IETF RFC 2543), not supporting loose routing (see IETF RFC 3261). Enable this option if you are using strict-routing proxies. If this option is enabled and you are specifying one or more loose-routing proxies in the Proxy field, then you must manually add the suffix to each loose-routing proxy address. For example, “10.134.123.101;lr”.

voip.stun-srv	TEXT	stun-address : String. IP address, domain name, or host name, and optional port number. Separate multiple addresses by comma.		Specifies the STUN (Session Traversal Utilities for NAT) server IP address or name to use to determine if the phone is behind a NAT, the type of NAT, and the public address of the phone. The field can contain a comma separated list of servers. Each server can be a domain name, host name, or IP address, and it may contain an optional port number. (For STUN see IETF RFC 5389.)
voip.use-ice	BOOLEAN	0 : Disable ICE 1 : Enable ICE	0	Enables the use of the ICE (Interactive Connectivity Establishment) protocol for NAT traversal. ICE takes advantage of STUN and TURN to identify candidates (IP addresses and ports) for communication, evaluating and prioritizing the candidate pairs to select the best route. Expensive candidates, such as using a media relay, are selected only as a last resort. (For ICE see IETF RFC 5245.)
voip.ice-regular	BOOLEAN	0 : Regular nomination method 1 : Aggressive nomination method	1	ICE nomination method. When using ICE, select the preferred ICE Nomination Method. To validate candidate pairs (IP addresses and ports for the local and remote nodes), CS-700 sends STUN binding requests as part of the media connectivity tests. When a candidate is nominated for use, a STUN binding request is sent with a flag indicating that the candidate pair is nominated.

voip.ice-max-hosts	NUMBER	0 : No maximum 1..10 : Maximum number of host candidates	5	Maximum number of ICE host candidate. An ICE host candidate is an actual local transport address in the host. Host transport addresses are obtained by binding to attached network interfaces. These interfaces include both physical interfaces and virtual interfaces such as VPN. This option specifies the maximum number of local ICE host candidates that may be used in evaluating candidate pairs when determining the best route.
voip.ice-no-rtcp	BOOLEAN	0 : Do not disable RTCP 1 : Keep RTCP disabled	0	Option to not disable the RTCP component in ICE.
voip.use-turn	BOOLEAN	0 : Disable TURN 1 : Enable TURN	0	Enables the use of a TURN (Traversal Using Relay NAT) relay when using ICE. A TURN relay is a media relay server residing on the public internet which can relay media data packet between clients. TURN relays are used when other preferred mechanisms are not available, such as STUN or direct connectivity. If TURN is enabled, the other TURN settings (server, username and password) must also be specified.
voip.turn-srv	TEXT	address : String. The format is either 'DOMAIN:PORT' or 'HOST:PORT'.		TURN server domain name or hostname, and port. The format is either 'DOMAIN:PORT' or 'HOST:PORT'
voip.turn-user	TEXT	User name.		Username to authenticate against the TURN server.
voip.turn-passwd	TEXT	Password.		Password to authenticate against the TURN server.
voip.turn-tcp	BOOLEAN	0 : Do not use TCP; use UDP 1 : Use TCP	0	Configure whether to use TCP on TURN relay; otherwise use UDP.

voip.codec1	INDEXED_ OPTION	0 : None 1 : G.722 2 : G.711 u-law (PCMU) 3 : G.711 A-law (PCMA) 4 : G.726 5 : G.729	1	Highest prioritized codec. At least one codec different from “None” has to be selected in voip.codec1-voip.codec5.
voip.codec2	INDEXED_ OPTION	0 : None 1 : G.722 2 : G.711 u-law (PCMU) 3 : G.711 A-law (PCMA) 4 : G.726 5 : G.729	2	Second highest prioritized codec. At least one codec different from “None” has to be selected in voip.codec1-voip.codec5.
voip.codec3	INDEXED_ OPTION	0 : None 1 : G.722 2 : G.711 u-law (PCMU) 3 : G.711 A-law (PCMA)) 4 : G.726 5 : G.729	3	Third highest prioritized codec. At least one codec different from “None” has to be selected in voip.codec1-voip.codec5.
voip.codec4	INDEXED_ OPTION	0 : None 1 : G.722 2 : G.711 u-law (PCMU) 3 : G.711 A-law (PCMA) 4 : G.726 5 : G.729	4	Fourth highest prioritized codec. At least one codec different from “None” has to be selected in voip.codec1-voip.codec5.
voip.codec5	INDEXED_ OPTION	0 : None 1 : G.722 2 : G.711 u-law (PCMU) 3 : G.711 A-law (PCMA) 4 : G.726 5 : G.729	5	Lowest prioritized codec. At least one codec different from “None” has to be selected in voip.codec1-voip.codec5.

voip.ptime	NUMBER	10..60 : ptime interval in ms	20	The ptime (packetization interval) value for a codec determines the length of time in milliseconds represented by the media in an RTP packet which is used to transmit audio traffic.
voip.mwi	BOOLEAN	0 : Disable MWI signaling 1 : Enable MWI signaling	0	Enable displaying the message waiting indicator (MWI) on the device and enable receiving message waiting notifications from the PBX. The PBX must be configured to support voice mail for the registered user in order for this feature to work properly.
voip.vm-number	TEXT	VoIP voicemail number.		The number that is dialed when voicemail is called from the UI.
voip.do-not-disturb	BOOLEAN	0 : Disable DND 1 : Enable DND	0	Configure do-not-disturb (DND) setting.
voip.auto-answer	NUMBER	0 : Disable auto-answer 100..699 : RESPONSE to send when answering	0	Auto answer incoming VoIP calls. We recommend enabling this feature only for test purposes. If the phone is set to Do Not Disturb or if there are no available lines, the Forward rules will apply. If there are no Forward rules specified, the incoming call will be sent to voice mail. If voice mail is not supported, the call will be rejected.
voip.duration	NUMBER	0 : No maximum 1..10080 : Maximum VoIP call duration in minutes	0	Specifies the maximum VoIP call duration in minutes. When the call duration reaches the maximum duration, the call will be automatically terminated.
voip.dial-plan	TEXT	dialplan : Dial plan string.		Specifies the VoIP dial plan string. See the User's Guide for a detailed description of dial plan settings.

voip.always-forwarding	BOOLEAN	0 : Disable 1 : Enable	0	Enable or disable forwarding all incoming VoIP calls to the specified number.
voip.always-forwarding-num	TEXT	VoIP dialing number.		Forward all incoming VoIP calls to the specified number.
voip.busy-forwarding	BOOLEAN	0 : Disable 1 : Enable	0	Enable or disable forwarding incoming calls to the specified number if the local phone is in 'Do Not Disturb' mode or if both lines are busy.
voip.busy-forwarding-num	TEXT	VoIP dialing number.		Forward incoming calls to the specified number if the local phone is in 'Do Not Disturb' mode or if both lines are busy.
voip.noanswer-forwarding	BOOLEAN	0 : Disable 1 : Enable	0	Enable or disable forwarding incoming VoIP calls to the specified number if the call is not answered within the duration specified in the 'noanswer-delay' attribute.
voip.noanswer-forwarding-num	TEXT	VoIP dialing number.		Forward incoming VoIP calls to the specified number if the call is not answered within the duration specified in the 'noanswer-delay' attribute.
voip.noanswer-delay	NUMBER	2..30 : Seconds to wait before forwarding a call	10	Number of seconds to wait before forwarding an unanswered incoming call to the 'noanswer-forwarding-num' number.

7 SNMP

The CS-700 includes an SNMP agent that can be configured to provide SNMP support.

The table below describes the SNMP configuration settings that the CS-700 administrator must configure to enable SNMP.

snmp-enable	Enable or disable SNMP support. Disabled (0) means that SNMP is not supported. Enabled (1) means that SNMP is available.
snmp-address	Specifies SNMP server address to which traps will be sent. Leave blank to disable traps.

snmp-community	Specifies the SNMP read-only community string used for queries from the server and transmitted traps. Read-only indicates the authorization level. The device does not support write operations initiated through SNMP.
snmp-contact-name	Specifies the contact name, typically the system administrator. This string is informational and can include an email address. It is not associated with traps.
snmp-device-location	Specifies the location of the device for informational purposes.

The following read-only properties and traps are supported.

Read-only Properties

- Serial number
- MAC address
- Base FW version
- USB connection active (0,1)
- USB microphone audio stream active (0,1)
- USB speaker audio stream active (0,1)
- USB video stream active (0,1)
- Bluetooth connection active (0,1)
- VoIP call active (0,1)

Traps

- USB connection state changes

Below is the SNMP MIB for the CS-700.

```
CS700-MIB DEFINITIONS ::= BEGIN

    IMPORTS
        OBJECT-TYPE, NOTIFICATION-TYPE, MODULE-IDENTITY, enterprises FROM
SNMPv2-SMI
        OBJECT-GROUP FROM SNMPv2-CONF
        DisplayString FROM SNMPv2-TC
    ;

yamahaAgentMIB MODULE-IDENTITY
    LAST-UPDATED "201710010000Z"
    ORGANIZATION "www.uc.yamaha.com"
    CONTACT-INFO
        "postal: Yamaha Unified Communications
                 144 North Rd
                 Sudbury, MA 01776
         email:  uc-customersupport@music.yamaha.com"
    DESCRIPTION
        "Defines monitoring structures for the Yamaha SNMP agent for CS-700."
    REVISION "201710010000Z"
    DESCRIPTION "Initial revision"

    ::= { enterprises 1182 }
    cs700             OBJECT IDENTIFIER ::= { yamahaAgentMIB 7386 }
    cs700Traps        OBJECT IDENTIFIER ::= { cs700 1 }
    cs700TrapsObjects OBJECT IDENTIFIER ::= { cs700Traps 1 }


--
-- CS-700 objects
--

serial OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    "Serial number of type String."
    ::= {cs700 11}

version OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    "Base version of type String."
    ::= {cs700 12}

usbConnection OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    "USB connection active (0,1) of type String."
    ::= {cs700 13}

usbMicStream OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    "USB microphone audio stream active (0,1) of type String."
    ::= {cs700 14}

usbSpkStream OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    "USB speaker audio stream active (0,1) of type String."
    ::= {cs700 15}

usbVidStream OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    "USB video stream active (0,1) of type String."
    ::= {cs700 16}

btConnected OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    "Bluetooth connection active (0,1) of type String."
    ::= {cs700 17}

voipCallStatus OBJECT-TYPE
    SYNTAX DisplayString
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
    "Current VoIP call lines 1-3 call state of type String."
    ::= {cs700 18}

--
-- CS-700 traps
--

usbConnTrap OBJECT-TYPE
    SYNTAX      DisplayString
    MAX-ACCESS  accessible-for-notify
    STATUS      current
    DESCRIPTION "USB connection trap data"
    ::= { cs700TrapsObjects 1 }

usbConnTrapHit NOTIFICATION-TYPE
    OBJECTS     { usbconnTrap }
    STATUS      current
    DESCRIPTION "Generated when USB connection state changes"
    ::= { cs700Traps 1 }
END
```

Appendix A – Volume Mappings

The table below identifies the volume mapping between the CS-700 device and the host operating systems. Windows 7 and 8 are shown in column W7. Windows 10 is column W10. Mac and Chrome are shown in column MAC.

		DSP		Windows 7	Windows 10	MAC OS	ChromeOS	Thresholds
Intervals ->		18		50	50	16	*25 (really 20)	
0	0	0	-31.5	E080 (-31.5)	E080 (-31.5)	E080 (-31.5)	E080 (-31.5)	less or eq: -30.0 (E200) -> 0
1	2	1	-27.5	E3FD	E35C		E280	
2	4	1	-27.5	E6D3	E5C4		E480	less or eq: -20.4 (EBA0) -> 1
3	6	2	-25	E936	E7D8		E680	
4	8	2	-25	EB47	E9AB		E880	
5	10	2	-25	ED18	EB4C	EA80	EA80	less or eq: -16.4 (EF9B) -> 2
6	12	3	-22.5	EEB7	ECC5		EC80	
7	14	3	-22.5	F02E	EED1	EE80	EE80	
8	16	3	-22.5	F185	EF59		F080	less or eq: -13.1 (F2E9) -> 3
9	18	4	-20	F2BF	F07D	F180	F280	
10	20	4	-20	F3E2	F18C			
11	22	4	-20	F4F1	F28A	F480	F480	less or eq: -10.5 (F577) -> 4
12	24	5	-17.5	F5EE	F378			
13	26	5	-17.5	F6DC	F459			
14	28	5	-17.5	F7BC	F52D	F680	F680	less or eq: -7.6 (F851) -> 5
15	30	6	-15	F890	F5F7			
16	32	6	-15	F958	F6B6	F880	F880	
17	34	6	-15	FA17	F76C	FA80	FA80	less or eq: -5.2 (FAB7) -> 6
18	36	7	-12.5	FACD	F851			
19	38	7	-12.5	FB7B	F8C1			
20	40	7	-12.5	FC22	F961	FC80	FC80	less or eq: -3.2 (FCCA) -> 7
21	42	8	-10	FCC2	F9FB			
22	44	8	-10	FD5B	FA8F			
23	46	8	-10	FDEE	FB1D			less or eq: -1.8 (FE2D) -> 8
24	48	9	-7.5	FE7D	FBA7			
25	50	9	-7.5	FF06	FC2B			
26	52	9	-7.5	FF8A	FCAC	FE80	FE80	less or eq: -0.1 (FFD9) -> 9
27	54	10	-5	0009	FD28			
28	56	10	-5	0085	FDA0			
29	58	10	-5	00FE	FE15	0080	0080	less or eq: +1.1 (0111) -> 10
30	60	11	-2.5	0172	FE86			
31	62	11	-2.5	01E3	FEF4			
32	64	11	-2.5	0251	FF5F	0180	0280	less or eq: +2.4 (0261) -> 11
33	66	12	0	02BC	FFC8			
34	68	12	0	0324	002D			
35	70	12	0	038A	0095	0380		less or eq: +3.6 (0396) -> 12
36	72	13	1.7	03ED	00FF			
37	74	13	1.7	044D	016D			
38	76	13	1.7	04AB	01DE	0480	0480	less or eq: +4.7 (04B4) -> 13
39	78	14	3.4	0507	0253			
40	80	14	3.4	0561	02CA			
41	82	14	3.4	05B8	0346			less or eq: +5.8 (05BF) -> 14
42	84	15	5.1	060E	03C6			
43	86	15	5.1	0662	044A			
44	88	15	5.1	06B4	04D3	0580	0680	less or eq: +6.8 (06B8) -> 15
45	90	16	6.8	0705	0561			
46	92	16	6.8	0753	05F4			
47	94	16	6.8	07A1	068D	0780		less or eq: +7.9 (07D4) -> 16
48	96	17	8.5	07EC	072C			
49	98	17	8.5	0837	07D2			
50	100	17	8.5	0880 (+8.5)	0880 (+8.5)	0880 (+8.5)	0880 (+8.5)	higher than +7.9 (07D4) -> 17

Appendix B – Crestron Integration TCP/IP

Integration with Crestron over TCP/IP is through IP interface.

Crestron is required to maintain a continuous IP session with CS-700 in order to control its behavior properly. When integrating with Crestron, following these steps will provide a good user experience:

1- Crestron device must maintain a session with CS-700 at all times. On a lost session, Crestron device must repeatedly attempt the telnet/SSH connection till the connection is re-established.
2- On a reconnection with CS-700, Crestron device must provide username and password as in IP Interface.
3- On a successful login, Crestron device must issue the following commands:
    a. regnotify
    b. set dialer-connection-mode rc
4- Crestron device must issue "get registration" and receive a reply "val registration 200" before attempting SIP calls.

Note that on a power-up, CS-700 will come up un-registered with SIP server. SIP registration procedure begins when the Crestron device issues "set dialer-connection-mode rc" command in the steps above.

The following examples show how a typical dialer state machine will control the CLI to make calls and to mix them in a conference.

Make two calls then conference them

The user makes a call to 1234 on line 1, then puts the call on hold. The user then makes a call to 7890 on line 2, then creates a conference between line 1 and line 2.

```
dial 1 1234

hold 1

dial 2 7890
```

Held calls cannot be conferenced, user must have both calls active before initiating a conference bridge.

```
resume 1

join 1 1 2 1
```

Since USB and Bluetooth are both optional in the join, join 1 1 2 1 is identical to join 1 1 2 1 usb 0 bt 0.

Hold calls in conference

The hold command automatically removes calls from conference, but if you just send the hold command for line 1, line 2 will remain in conference. It is best practice to send the join command to split the conference first.

```
join 1 0 2 0
hold 1
hold 2
```

Resume calls and re-establish the conference

Both lines need to be in active state before they can be joined. Note, leaving two lines in active state without joining them will produce undefined telephony behavior.

```
resume 1
resume 2
join 1 1 2 1
```

Remove calls from conference and leave one call active

A single line cannot be put into a conference with the join command. Similarly, a single line cannot be left in a conference with the join command. Commands join 1 1 2 0 and join 1 0 2 1 will also remove both calls from the conference.

Similarly, if no calls are in the conference, join 1 1 2 0 does not change the line 1 status to indicate that it's in conference. If no calls are in the conference, join 1 0 2 1 does not change the line 2 status to indicate that it's in conference.

Following is the correct way to remove calls from conference.

```
join 1 0 2 0
```

At this point both calls are connected to the speaker and microphone locally, but they do not send audio to each other. Note, leaving two lines in active state without holding either or both will produce undefined telephony behavior.

```
hold 2
```

At this point, line 1 remains active, and line 2 goes on hold.

Answer SIP calls with active USB line, then conferencing all of them together

The USB line is for PC based calls and audio, where CS-700 is the speaker/microphone device. The user starts with USB line active, an incoming SIP call on line 1, and an incoming SIP call on line 2.

```
hold usb
answer 1
```

At the point, SIP line 1 is in active state, and the USB line is in hold state.

```
hold 1
```

At the point, both SIP line 1, and USB line are in hold state.

```
answer 2
```

At the point, both SIP line 1, and USB line are in hold state, and SIP line 2 is in active state. The user may now initiate a conference. Note, held calls cannot be added in a conference.

```
resume 1
resume usb
join 1 1 2 1 usb 1
```

At this point, all calls (USB, line 1, and line 2) are in a conference.

Hold only one call

Starting from the previous example, the user puts line 1 on hold.

```
hold 1
```

Because a held call cannot be in conference, this automatically removes line 1 from the conference.

At this point, calls (USB, line 2) are in an active conference, and line 1 in on hold, outside the conference.

Hold all calls

The user starts with all lines in conference.

```
hold all
```

At this point, all lines are on hold, and not in conference.

Resume calls and conference them

The user starts with all lines on hold, and not in conference.

```
resume 1
resume usb
```

At this point, line 1 and USB are in active state, and line 2 is in held state. Note, leaving two lines in active state without holding either or both will produce undefined telephony behavior.

```
join 1 1 2 0 usb 1
```

At this point, line 1 and USB are in conference, and line 2 is on hold outside the conference. The user now puts line 2 in conference.

```
resume 2
```

At this point, line 2 and the conference are in active state.

```
join 1 1 2 1 usb 1
```

At this point, all lines are in active state in the conference

Hangup a call in conference

The user starts with line 1 and USB in an active conference.

```
hangup 1
```

At this point, line1 is disconnected and removed from conference, but USB is still in conference.

```
join 1 0 2 0 usb 0
```

At this point, USB is active and outside the conference.