Duet Module Interface Specification

for an

Clearone Converge Pro 2

Audio Processor

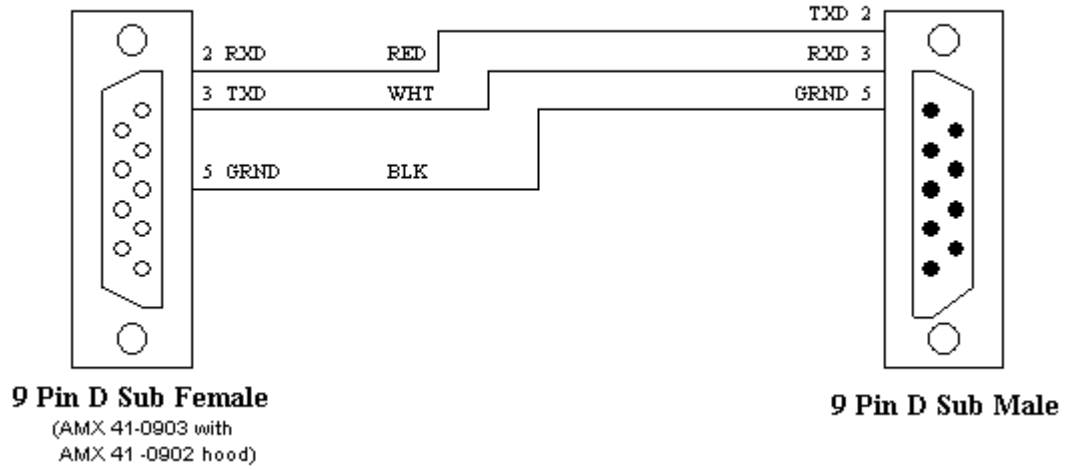
TABLE OF CONTENTS

Revision History

Date	Initials	Comments
4-29-2017	NJ	v1.0.0 Initial Release (Control Concepts, Inc.)
8-07-2018	JJM	v1.2.0 Skype for Business Dialer.

Introduction

This is a reference manual to describe the interface provided between an AMX NetLinx system and a Clearone Converge Pro 2. The Clearone Converge Pro 2 supports an RS-232 serial and an Ethernet protocol. The interface was tested using version 3.4.30.0 of the Converge Pro 2 firmware. The default baud rate is 57600. However, **when using the NI Series, we recommend communication settings to be set to a baud rate of 38400**, 8 data bits, 1 stop bit, no parity, and handshaking off. The cable for this device is FG#10-752-04. The wiring diagram for this cable is as follows: *(Note: The new NX series seem to have no issues at 115200 baud rate)*



This module was written using Café Duet firmware version v3.21.343. NetLinx Studio version 4.4 build 4.4.1626, Café Duet application platform and runtime version 2.0.5 and Café Duet application version 1.8.85.

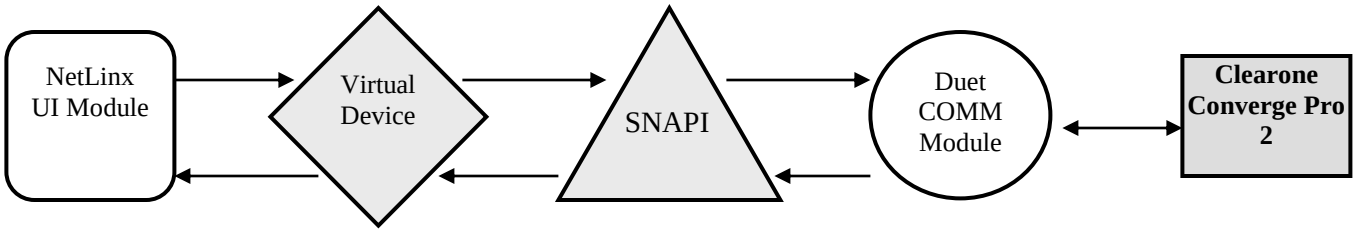
In the case of IP control, the Comm. module must be informed of the IP address of the device it is to connect to. This must be done manually via a telnet session, or can be automated from the UI module. The IP address must be set using the 'PROPERTY-' command. The 'REINIT' command is used to notify the Comm. module that new properties are now available and to start using them. When these two commands are used in the appropriate order, the Comm. module will attempt to connect to the specified IP address periodically until a connection is made or a new IP address is submitted using the 'PROPERTY-' and 'REINIT' commands. Please see the Programming Notes section for additional information.

Overview

The COMM module translates between the standard interface described below and audio processor serial protocol. It parses the buffer for responses from the audio processor, sends strings to control the audio processor, and receives commands from the UI module or telnet sessions.

A User Interface (UI) module is also provided. This module uses the standard interface described below and parses the command responses for feedback.

The following diagram gives a graphical view of the interface between the interface code and the Duet module.



Some functionality in the device interface may not be implemented in the API interface. In cases where device functions are desired but not API-supported, the PASSTHRU command may be used to send any and all device-protocol commands to the device. See the PASSTHRU command and the Adding Functions to Modules section for more information.

A sample UI module and a touch panel file are provided in the module package. These are not intended to cover every possible application, but can be expanded as needed by a dealer to meet the requirements of a particular installation.

Implementation

To interface to the AMX Clearone Converge Pro 2 module, the programmer must perform the following steps:

1. Define the device ID for the audio processor that will be controlled.
2. Define the virtual device ID that the audio processor's COMM module will use to communicate with the main program and User Interface. Duet virtual devices use device numbers 41000 - 42000.
3. If a touch panel interface is desired, a touch panel file Clearone Converge Pro 2 Demo**.**TP4 and module (Clearone Converge Pro 2 Example.axs) have been created for testing.
4. The Duet Clearone_ConvergePro2_Comm_dr1_0_0 module must be included in the program with a DEFINE_MODULE command. This command starts execution of the module and passes in the following key information: the device ID of the audio processor to be controlled, and the virtual device ID for communicating to the main program.

An example of how to do this is shown below.

```
DEFINE_DEVICE

dvDSP                                                   = 0:3:0                 //
Clearone Converge Pro 2 IP
//dvDSP                                                 = 5001:1:0              // Clearone Converge Pro 2
RS 232

dvPanelFader1                                   = 10002:01:0        // Any G4 Panel
dvPanelFader2                                   = 10002:02:0
dvPanelMic1                                             = 10002:03:0
```

```
dvPanelMic2                                                 = 10002:04:0
dvPanelMic3                                                 = 10002:05:0
dvPanelSpeakers                                     = 10002:06:0
dvPanelMic1ResponseTime   = 10002:07:0
dvPanelMic1TargetLevel    = 10002:08:0
dvPanelMic1HoldTime                         = 10002:09:0
dvPanelMic1Chairman                         = 10002:10:0
dvPanelMatrix                                       = 10002:12:0
dvPanelRoom                                                 = 10002:13:0
dvPanelAnalogDialer                         = 10002:14:0
dvPanelATCRX                                        = 10002:15:0
dvPanelVOIPDialer                           = 10002:16:0
dvPanelVOIPRX                                       = 10002:17:0
dvPanelPresets                                      = 10002:18:0
dvPanelMacros                                       = 10002:19:0

vdvDSP                                                                  = 41001:01:0        // Clearone Converge Pro 2
Duet Main Communication Device
vdvFader1                                                               = 41001:01:0
vdvFader2                                                               = 41001:02:0
vdvMic1                                                                 = 41001:03:0
vdvMic2                                                                 = 41001:04:0
vdvMic3                                                                 = 41001:05:0
vdvSpeakers                                                 = 41001:06:0
vdvMic1ResponseTime                         = 41001:07:0
vdvMic1TargetLevel                          = 41001:08:0
vdvMic1HoldTime                                     = 41001:09:0
vdvMic1Chairman                                     = 41001:10:0
vdvMic1ToRecord                                     = 41001:11:0
vdvMic1ToSpare1                                     = 41001:12:0
vdvMic1ToSpare2                                     = 41001:13:0
vdvMic1ToSpeakers                                   = 41001:14:0
vdvMic2ToRecord                                     = 41001:15:0
vdvMic2ToSpare1                                     = 41001:16:0
vdvMic2ToSpare2                                     = 41001:17:0
vdvMic2ToSpeakers                                   = 41001:18:0
vdvFader1ToRecord                                   = 41001:19:0
vdvFader1ToSpare1                                   = 41001:20:0
vdvFader1ToSpare2                                   = 41001:21:0
vdvFader1ToSpeakers                         = 41001:22:0
vdvFader2ToRecord                                   = 41001:23:0
vdvFader2ToSpare1                                   = 41001:24:0
vdvFader2ToSpare2                                   = 41001:25:0
vdvFader2ToSpeakers                         = 41001:26:0
vdvRoomWall1                                        = 41001:27:0
vdvRoomWall2                                        = 41001:28:0
vdvRoomWall3                                        = 41001:29:0
vdvAnalogDialer                                     = 41001:30:0
vdvATCRX                                                                = 41001:31:0
vdvVOIPDialer                                       = 41001:32:0
vdvVOIPRX                                                               = 41001:33:0

(***********************************************************)
(*            CONSTANT DEFINITIONS GO BELOW            *)
(***********************************************************)
DEFINE_CONSTANT

(***********************************************************)
(*            DATA TYPE DEFINITIONS GO BELOW           *)
(***********************************************************)
DEFINE_TYPE
```

```
(***********************************************************)
(*            VARIABLE DEFINITIONS GO BELOW            *)
(***********************************************************)
DEFINE_VARIABLE

(***********************************************************)
(*            LATCHING DEFINITIONS GO BELOW            *)
(***********************************************************)
DEFINE_LATCHING

(***********************************************************)
(*      MUTUALLY EXCLUSIVE DEFINITIONS GO BELOW          *)
(***********************************************************)
DEFINE_MUTUALLY_EXCLUSIVE

(***********************************************************)
(*       SUBROUTINE/FUNCTION DEFINITIONS GO BELOW        *)
(***********************************************************)

(***********************************************************)
(*             STARTUP CODE GOES BELOW                 *)
(***********************************************************)
DEFINE_START

DEFINE_MODULE 'Clearone_ConvergePro2_Comm_dr1_0_0' modClearoneConvergePro2(vdvDSP, dvDSP)

(***********************************************************)
(*             THE EVENTS GO BELOW                     *)
(***********************************************************)
DEFINE_EVENT

DATA_EVENT[vdvDSP]
{
        ONLINE:
        {
                WAIT 50
                {
          SEND_COMMAND vdvDSP,'DEBUG-4'

          // IP Device?
          IF (dvDSP.NUMBER == 0)
          {
                                SEND_COMMAND vdvDSP,'PROPERTY-IP_Address,192.168.187.53'
                                SEND_COMMAND vdvDSP,'PROPERTY-User_Name,clearone'
                                SEND_COMMAND vdvDSP,'PROPERTY-Password,converge'
                                SEND_COMMAND vdvDSP,'PROPERTY-Poll_Time,0'
          }
          // Serial Device
          ELSE
          {
                                SEND_COMMAND vdvDSP,'PROPERTY-Baud_Rate,38400'
          }

          ////////////////////////////////////////////////////
          // Configure Clearone Virtual Devices
          ////////////////////////////////////////////////////

          // Faders //
          // AUDIOPROCADD-<PORT>,GAINMUTE[<EndPointName>|<GAIN/GAIN_FINE>|<STEP>]
                        SEND_COMMAND vdvDSP,'"AUDIOPROCADD-
',ITOA(vdvFader1.PORT),',GAINMUTE[Fader_1|GAIN|2]'"
          SEND_COMMAND vdvDSP,'"AUDIOPROCADD-',ITOA(vdvFader2.PORT),',GAINMUTE[Fader_2|GAIN|2]'"
```

```
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic1.PORT),',GAINMUTE[Mic-
01|GAIN_FINE|2]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic2.PORT),',GAINMUTE[Mic-
02|GAIN_FINE|2]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic3.PORT),',GAINMUTE[Mic-
03|GAIN_FINE|2]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvSpeakers.PORT),',GAINMUTE[Speakers|GAIN|1]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvATCRX.PORT),',GAINMUTE[Dialer_1_Rx|GAIN|2]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvVOIPRX.PORT),',GAINMUTE[Voip_1_Rx|GAIN|2]"'

                                // EP_LEVEL //
                                // AUDIOPROCADD-
<PORT>,EP_LEVEL[<EndPointName>|<BlockName>|<ParameterName>|<MAX>|<MIN>|<STEP>]
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvMic1ResponseTime.PORT),',EP_LEVEL[Mic-01|AGC|RESPONSE_TIME|10000|100|1]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic1TargetLevel.PORT),',EP_LEVEL[Mic-
01|AGC|TARGET_LEVEL|20|-30|.5]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic1HoldTime.PORT),',EP_LEVEL[Mic-
01|GATING|HOLD_TIME|8|.1|.1]"'

                                // EP_States
                                // AUDIOPROCADD-<PORT>,EP_STATE[<EndPointName>|<BlockName>|<ParameterName>]
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic1Chairman.PORT),',EP_STATE[Mic-
01|GATING|CHAIRMAN]"'

                                // Crosspoints
                                // XPOINT[<InputEndPointName>|<OutputEndPointName>|<MAX>|<MIN>|<Type>]
                                // Valid values for Type: CROSSPOINT, GATED, NONGATED, PREAEC
                                // Types GATED, NONGATED and PREAEC apply only to MIC end points.
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic1ToRecord.PORT),',XPOINT[Mic-
01|Record|12|-20|GATED]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic1ToSpare1.PORT),',XPOINT[Mic-
01|Spare_1|12|-20|GATED]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic1ToSpare2.PORT),',XPOINT[Mic-
01|Spare_2|12|-20|GATED]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic1ToSpeakers.PORT),',XPOINT[Mic-
01|Speakers|12|-20|GATED]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic2ToRecord.PORT),',XPOINT[Mic-
02|Record|12|-20|GATED]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic2ToSpare1.PORT),',XPOINT[Mic-
02|Spare_1|12|-20|GATED]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic2ToSpare2.PORT),',XPOINT[Mic-
02|Spare_2|12|-20|GATED]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-',ITOA(vdvMic2ToSpeakers.PORT),',XPOINT[Mic-
02|Speakers|12|-20|GATED]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvFader1ToRecord.PORT),',XPOINT[Fader_1|Record|12|-20|CROSSPOINT]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvFader1ToSpare1.PORT),',XPOINT[Fader_1|Spare_1|12|-20|CROSSPOINT]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvFader1ToSpare2.PORT),',XPOINT[Fader_1|Spare_2|12|-20|CROSSPOINT]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvFader1ToSpeakers.PORT),',XPOINT[Fader_1|Speakers|12|-20|CROSSPOINT]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvFader2ToRecord.PORT),',XPOINT[Fader_2|Record|12|-20|CROSSPOINT]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvFader2ToSpare1.PORT),',XPOINT[Fader_2|Spare_1|12|-20|CROSSPOINT]"'
                                SEND_COMMAND vdvDSP,"'AUDIOPROCADD-
',ITOA(vdvFader2ToSpare2.PORT),',XPOINT[Fader_2|Spare_2|12|-20|CROSSPOINT]"'
```

```
                                SEND_COMMAND vdvDSP,'"AUDIOPROCADD-
',ITOA(vdvFader2ToSpeakers.PORT),',XPOINT[Fader_2|Speakers|12|-20|CROSSPOINT]"'

                                // RM_STATE //
                                // AUDIOPROCADD-<PORT>,RM_STATE[<RoomNumber>|<RoomOption>|<DividerNumber>]
                                SEND_COMMAND vdvDSP,'"AUDIOPROCADD-
',ITOA(vdvRoomWall1.PORT),',RM_STATE[1|IndividualDividerState|1]"'
                                SEND_COMMAND vdvDSP,'"AUDIOPROCADD-
',ITOA(vdvRoomWall2.PORT),',RM_STATE[1|IndividualDividerState|2]"'
                                SEND_COMMAND vdvDSP,'"AUDIOPROCADD-
',ITOA(vdvRoomWall3.PORT),',RM_STATE[1|IndividualDividerState|3]"'

                                // AUDIOPROCADD-<PORT>,DIALER[<EndPointName]
                                SEND_COMMAND vdvDSP,'"AUDIOPROCADD-
',ITOA(vdvAnalogDialer.PORT),',DIALER[Dialer_1_Rx]"'

                                // AUDIOPROCADD-<PORT>,VOIP[<EndPointName]
                                SEND_COMMAND vdvDSP,'"AUDIOPROCADD-',ITOA(vdvVOIPDialer.PORT),',VOIP[Voip_1]"'

            SEND_COMMAND vdvDSP,'REINIT'

                    }
        }
}
```

Upon initialization, the AMX Comm module will communicate with the audio processor and information will be exchanged.

Port Mapping

This module uses multiple virtual devices in order distinguish events for one zone from another. The functionality of each zone will vary based on what Audio Processor components you map to what virtual port. You will map these components using the "AUDIOPROCADD-" command. This will be discussed later in this document. This module has been configured to support 250 components (250 Virtual Ports). You can only map a single component to each port on the virtual device using the AUDIOPROCADD- command. These components are as follows, we will discuss more about these later in this manual.

1) GAINMUTE[]
2) EP_LEVEL[]
3) EP_STATE[]
4) XPOINT[]
5) RM_STATE[]
6) DIALER[]
7) VOIP[]
8) SKYPE[]

Virtual Device	Channels	Levels	Control	Feedback
41001:1:0 – **Presets & System Init use Only.**	**251,252**	**N/A**	See Command Control List	See Command Feedback Lists
41001:x:0 – **GAINMUTE Type Mapping**	**24, 25, 26, 199**	**1**	See Command Control List	See Command Feedback Lists

41001:x:0 – **EP_Level Type Mapping**	**24,25**	**1**	See Command Control List	See Command Feedback Lists
41001:x:0 – **EP_STATE Type Mapping**	**26, 199**	**N/A**		
41001:x:0 – **XPOINT Type Mapping**	**24, 25, 26**	**N/A**	See Command Control List	See Command Feedback Lists
41001:x:0 – **RM_STATE Type Mapping**	**26, 199**	**N/A**		
41001:x:0 – **Dialer/VOIP/SKYPE type Mapping**	**201, 202, 204, 208, 238, 239**	**N/A**	See Command Control List	See Command Feedback Lists

Table 1 - Port Mapping

Clearone Audio Components

Because the Clearone Converge Pro 2 product line is highly configurable and flexible it is important to teach the duet module what it needs to control. So in order to teach the duet module, you must use the "AUDIOPROCADD-" command. This section will go into details and examples on how this works.

Seven duet software components have been created to help teach the duet module. Each component has several properties that must be filled in accurately for the duet module to control the Clearone Converge pro 2 components correctly. We will touch on each one.

1) GAINMUTE[]
2) EP_LEVEL[]
3) EP_STATE[]
4) XPOINT[]
5) RM_STATE[]
6) DIALER[]
7) VOIP[]
8) SKYPE[]

GAINMUTE

The GAINMUTE software component will more than likely be the most used component, because most of the controls that tend to be used on a touch panel are volume controls. The GAINMUTE component has a number of properties to enable it to be used in a flexible manner. To successfully understand the properties of the GAINMUTE component, you must understand the basics of the Clearone Converge Pro 2 end point message.

The Clearone Converge Pro 2 protocol parameters are as follows.

EP <EndPointName> <BlockName> <ParameterName> <Value>

For the GAINMUTE software component we are interested in the ones highlighted in BLUE. The GAINMUTE component simplifies the programming by only caring about the EndPointName and the ParameterName. The BlockName is handled by the Duet module automatically.

The EndPoint name is the name of any block in the Clearone Converge Pro 2 configuration that has a LEVEL BlockName. The available types of endpoints for a GAINMUTE component are MIC, TELCO_RX, TELCO_TX, VOIP_RX, VOIP_TX, OUTPUT, SPEAKER, PROC, FADER, BFM, USB_RX, USB_TX, SGEN, SRMIC, DANTE_RX, and DANTE_TX.

The parameter is important because based on the type of EndPoint the EndPointName points to, either GAIN or GAIN_FINE have to be specified. This is because the Duet module has no way to know if the EndPoint is a mic or not. In short if the EndPointName is pointing to a microphone use GAIN_FINE and if not use GAIN.

I will show you some examples to help.

It is important to understand the Clearone Converge Pro 2 protocol is highly case sensitive and these must match exactly!

GAINMUTE[<EndPointName>|<GAIN/GAIN_FINE>|<STEP>]

Example 1:

I want to control the GAIN and MUTE of a Matrix input named Mic-01. Since I know that this is configured to be a microphone I know that I must use GAIN_FINE. The step can be in any number that falls within a .5 increment. For example, I am going to use 2 for the step value. I would like to assign the control of this GAIN Mute component to the 10th port on my AMX duet virtual device.

So, I would send the following command to the first virtual port of the duet module on the online event of the virtual device.

```
//GAINMUE[EndPointName|GAIN/GAIN_FINE|STEP]
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-10,GAINMUTE[Mic-01|GAIN_FINE|2]'

// You must send this last after all configurations have been sent.
SEND-COMMAND 41001:1:0,'REINIT'
```

Once you have sent this command and the Duet module has successfully initialized, then you can control the gain and mute controls by using the "GAINMUTE" software component API on virtual port 10. Here are three examples of the GAINMUTE API. A list of all available API commands, channels and levels is available later in this manual.

```
PULSE[41001:10:0, 24] // Raises the Gain level
PULSE[41001:10:0, 25] // Lowers the Gain level
PULSE[41001:10:0, 26] // Cycle the Mute state
```

Example 2:

I want to control the GAIN and MUTE of the Matrix input representing the ATC RX named Dialer_1_RX. Since this End Point is not a microphone this component definition will use GAIN. Just like the previous example the step value supports an increment of .5. I am going to set it to 1. I am going to control this component on the 31st port on my AMX duet virtual device.

So I send the following command to the first virtual port of the duet module on the online event of the virtual device.

```
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-31,GAINMUTE[Dialer_1_RX|GAIN|1]'

// You must send this last after all configurations have been sent.
```

SEND_COMMAND 41001:1:0,'REINIT'
Once you have sent this command and the Duet module has successfully initialized, than you can control the cross point control by using the "GAINMUTE" software component API on virtual port 31. Here are three examples of the GAINMUTE API. A list of all available API commands, channels and levels is available later in this manual.

PULSE[41001:31:0, 24] // Raises the Gain level
PULSE[41001:31:0, 25] // Lowers the Gain level
PULSE[41001:31:0, 26] // Cycle the Mute state

EP_LEVEL

The EP_LEVEL software component works like the GAIN portion of the GAINMUTE software component. However unlike the GAINMUTE software component the EP_LEVEL component is not constrained to only controlling the level value of End Points with a Block Name of LEVEL and a Parameter Name of GAIN or GAIN_FINE.

The EP_Level software component is designed to control any End Point that has a Parameter Name which contains a value that supports a range that is greater than a min of 0 to a max of 1.

The following are the valid " Block Name" and "Parameter Name" combination of values that the EP_LEVEL software component will allow. **It is important to understand the Clearone Converge Pro 2 protocol is highly case sensitive and these must match exactly!**

Block Name, Parameter Name

- AEC, NLP
- NC, DEPTH
- AGC, TARGET_LEVEL
- AGC, RESPONSE_TIME
- AGC, THRESHOLD
- AGC_ALC, MODE
- GATING, GROUP
- GATING, MODE
- GATING, AMB_LEVEL
- GATING, OFF_ATTEN
- GATING, GATE_RATIO
- GATING, HOLD_TIME
- GATING, DECAY_RATE
- FILTER_1, TYPE
- FILTER_1, FCY
- FILTER_1, GAIN
- FILTER_1, BW
- FILTER_2, TYPE
- FILTER_2, FCY

- FILTER_2, GAIN
- FILTER_2, BW
- FILTER_3, TYPE
- FILTER_3, FCY
- FILTER_3, GAIN
- FILTER_3, BW
- FILTER_4, TYPE
- FILTER_4, FCY
- FILTER_4, GAIN
- FILTER_4, BW
- FILTER_5, TYPE
- FILTER_5, FCY
- FILTER_5, GAIN
- FILTER_5, BW
- FILTER_6, TYPE
- FILTER_6, FCY
- FILTER_6, GAIN
- FILTER_6, BW
- FILTER_7, TYPE
- FILTER_7, FCY
- FILTER_7, GAIN
- FILTER_7, BW
- FILTER_8, TYPE
- FILTER_8, FCY
- FILTER_8, GAIN
- FILTER_8, BW
- FILTER_9, TYPE
- FILTER_9, FCY
- FILTER_9, GAIN
- FILTER_9, BW
- FILTER_10, TYPE
- FILTER_10, FCY
- FILTER_10, GAIN
- FILTER_10, BW
- FILTER_11, TYPE
- FILTER_11, FCY
- FILTER_11, GAIN
- FILTER_11, BW
- FILTER_12, TYPE
- FILTER_12, FCY
- FILTER_12, GAIN
- FILTER_12, BW
- FILTER_13, TYPE
- FILTER_13, FCY

- FILTER_13, GAIN
- FILTER_13, BW
- FILTER_14, TYPE
- FILTER_14, FCY
- FILTER_14, GAIN
- FILTER_14, BW
- FILTER_15, TYPE
- FILTER_15, FCY
- FILTER_15, GAIN
- FILTER_15, BW
- SETTINGS, AUTO_ANSWER_RINGS
- SETTINGS, AUTO_DISCONNECT_MODE
- SETTINGS, RING_TYPE
- SETTINGS, HOOK_FLASH_DURATION
- SETTINGS, RING_LEVEL
- SETTINGS, HOOK_LEVEL
- SETTINGS, RING_ON_TIME
- SETTINGS, RING_OFF_TIME
- SETTINGS, COUNTRY_CODE
- GRAPHICEQ, GAIN_1
- GRAPHICEQ, GAIN_2
- GRAPHICEQ, GAIN_3
- GRAPHICEQ, GAIN_4
- GRAPHICEQ, GAIN_5
- GRAPHICEQ, GAIN_6
- GRAPHICEQ, GAIN_7
- GRAPHICEQ, GAIN_8
- GRAPHICEQ, GAIN_9
- GRAPHICEQ, GAIN_10
- LIMITER, THRESHOLD
- DELAY, VALUE
- COMPRESSOR, GROUP
- COMPRESSOR, POST_GAIN
- COMPRESSOR, THRESHOLD
- COMPRESSOR, ATTACK
- COMPRESSOR, RATIO
- COMPRESSOR, RELEASE
- COMPRESSOR, DELAY

The EP_LEVEL software component, as you saw, is very flexible. To get this flexibility the command to add the component to the list of components that the duet module will know to control the properties have to be a little more complicated To successfully understand the properties of the EP_LEVEL component, you must understand the basics of the Clearone Converge Pro 2 end point message.

The Clearone Converge Pro 2 protocol parameters are as follows.

EP <EndPointName> <BlockName> <ParameterName> <Value>

For the EP_LEVEL software component we are interested in the ones highlighted in BLUE.

The End Point name works just like the End Point Name property in the GAINMUTE software component. The Block Name and Parameter Name properties can be set based off of any of the available combinations listed above.

In addition the range being used by the EP_LEVEL software component must be specified. The range used by the component doesn't have to match the level ranges of the Block Name and Parameter Name but the range must fall within it.

EP_LEVEL[<ENDPOINTNAME>|<BLOCKNAME>|<PARAMETERNAME>|<MAX>|<MIN>|<STEP>]

Example 1:

I want to control the AGC Response Time of Mic-01 on the 12th port of my AMX duet virtual device. I want to use the full range available to the parameter and an increment of 1.

So I would send the following command to the first virtual port to the duet module on the online event of the virtual device..

```
// EP_LEVEL[ENDPOINTNAME|BLOCKNAME|PARAMETERNAME|MAX|MIN|STEP]
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-12,EP_LEVEL[Mic-
01|AGC|RESPONSE_TIME|10000|100|1]'

// You must send this last after all configurations have been sent.
SEND-COMMAND 41001:1:0,'REINIT'
```

Once you have sent this command and the Duet module has successfully initialized, then you can control the level control by using the "EP_LEVEL" software component API on virtual port 12. Here are two examples of the EP_LEVEL API. A list of all available API commands, channels and levels is available later in this manual.

```
PULSE[41001:12:0, 24] // Raises the level
PULSE[41001:12:0, 25] // Lowers the level
```

Example 2:

I want to control the AGC Target Level of Mic-01 on the 13th port of my AMX duet virtual device. Again I want to use the full range that the command allows me with a .5 increment. So, I would send the following command to the first virtual port.

```
// EP_LEVEL[ENDPOINTNAME|BLOCKNAME|PARAMETERNAME|MAX|MIN|STEP]
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-13,EP_LEVEL[Mic-
01|AGC|TARGET_LEVEL|20|-30|.5]'

// You must send this last after all configurations have been sent.
SEND_COMMAND 41001:1:0,'REINIT'
```

Once you have sent this command and the Duet module has successfully initialized, then you can control the state control by using the "EP_LEVEL" software component API on virtual port 13. Here are two examples of the EP_LEVEL API. A list of all available API commands, channels and levels is available later in this manual.

```
PULSE[41001:13:0, 24] // Raises the level
PULSE[41001:13:0, 25] // Lowers the level
```

EP_STATE

The EP_State software component is like the EP_Level but for controlling properties with a value range of 0 to 1 with a granularity of 1 which is in essence a Boolean on and off state.

The following are the valid " Block Name" and "Parameter Name" combination of values that the EP_STATE software component will allow. **It is important to understand the Clearone Converge Pro 2 protocol is highly case sensitive and these must match exactly!**

Block Name, Parameter Name

- AEC, ENABLE
- NC, ENABLE
- GATING, NONE
- GATING, CHAIRMAN
- GATING, PA_ADAPT
- GATING, ADAPT_AMB
- FILTER_1, ENABLE
- FILTER_2, ENABLE
- FILTER_3, ENABLE
- FILTER_4, ENABLE
- FILTER_5, ENABLE
- FILTER_6, ENABLE
- FILTER_7, ENABLE
- FILTER_8, ENABLE
- FILTER_9, ENABLE
- FILTER_10, ENABLE
- FILTER_11, ENABLE
- FILTER_12, ENABLE
- FILTER_13, ENABLE

- FILTER_14, ENABLE
- FILTER_15, ENABLE
- SETTINGS, ADAPT
- SETTINGS, HOOK_ENABLE
- EC, ENABLE
- CE, ENABLE
- ALC, ENABLE
- GRAPHICEQ, ENABLE
- LIMITER, ENABLE
- DELAY, ENABLE
- COMPRESSOR, ENABLE
- COMPRESSOR, DELAY_ENABLE
- BF, ZONE_1
- BF, ZONE_2
- BF, ZONE_3
- BF, ZONE_4
- BF, ZONE_5
- BF, ZONE_6

EP_STATE[<EndPointName>|<BlockName>|<ParameterName>]

Example 1:

I want to be able to toggle the Chairman status of Mic-01. By looking at the list above of the valid states I see that the chairman property is under the block name GATING. I want to control this state using the 14th port of the AMX duet virtual device. So I would send the following command to the first virtual port to the duet module on the online event of the virtual device.

```
// EP_STATE[EndPointName|BlockName|Parametername]
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-14,EP_STATE[Mic-01|GATING|CHAIRMAN]'

// You must send this last after all configurations have been sent.
SEND_COMMAND 41001:1:0,'REINIT'
```

Once you have sent this command and the Duet module has successfully initialized, then you can control the state control by using the "EP_STATE" software component API on virtual port 14. Here is one example of the EP_STATE API. A list of all available API commands, channels and levels is available later in this manual.

```
PULSE[41001:14:0, 26] // Toggles the State
```

XPOINT

The XPOINT software component is for controlling the cross-point attenuation state and level. The type of cross point that will be used is defined when the XPOINT is defined.

The max and min values can be defined but the increment is always 1. The max and min values do not have to be the full range of the cross-point but they must fall within the range.

Unlike the other components, the XPOINT software component uses two endpoints to define itself. The first end point is the input and the second is the output.

Valid values for Type: CROSSPOINT, GATED, NONGATED, PREAEC
ypes GATED, NONGATED and PREAEC apply only to MIC end points.

XPOINT[InputEndPointName|OutputEndPointName|Max|Min|TYPE]

Example 1:

I would like to control the cross-point route of Mic-01 to the speakers output and have it be gated. I'm going to use the full range of the cross-point. I would like to assign the software component to virtual port 2 on the duet module. So, on the online event of the first virtual port, I will send theses commands.

```
//XPOINT[InputEndPointName|OutputEndPointName||Max|Min|Type]
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-2,XPOINT[Mic-01|Speakers|12|-20|GATED]'

// You must send this last after all configurations have been sent.
SEND_COMMAND 41001:1:0,'REINIT'
```

Once you have sent this command and the Duet module has successfully initialized, then you can control the cross-point control by using the "XPOINT" software component API on virtual port 2. Here is one example of the XPOINT API. A list of all available API commands, channels and levels is available later in this manual.

RM_STATE

The RM_STATE software component has been designed to handle the individual divider state and the divider polarity for the room command type.

Unlike all the other component types the RM_State software component doesn't use an end point at all.

RM_STATE[<RoomNumber>|<RoomOption>|<DividerNumber>]

Example 1:

I want to control the states of three walls in a three-way combinable space. These three rooms are the first set of rooms in the DSP configuration. To accomplish this I have to create a RM_STATE software component for each wall. I'll use ports 27 through 29 on the AMX duet virtual device. So, I would send the following commands to the first virtual port to the duet module on the online event of the virtual device.

```
//RM_STATE[RoomNumber|RoomOption|DividerNumber]
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-27,RM_STATE[1|IndividualDividerState|1]
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-28,RM_STATE[1|IndividualDividerState|2]
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-29,RM_STATE[1|IndividualDividerState|3]

// You must send this last after all configurations have been sent.
SEND_COMMAND 41001:1:0,'REINIT'
```

Once you have sent this command and the Duet module has successfully initialized, then you can control the state of the wall by using the "RM_STATE" software component API on their respective virtual ports, 27, 28 and 29. Here is one example of the RM_STATE API. A list of all available API commands, channels and levels is available later in this manual.

```
PULSE[41001:14:0, 26] // Toggles the State
```

DIALER

The dialer software component has been designed to control the Clearone Converge Pro 2 Dialer. You will use the DIALER software component API to control the Clearone Converge Pro 2 Dialer.

DIALER[EndPointName]

Example 1:

I would like to control the Dialer in the configuration it is set to the end point name Dialer_1_Rx. I'll use ports 33 on the AMX duet virtual device. So, I would send the following command to the first virtual port to the duet module on the online event of the virtual device.

```
// DIALER[EndPointName]
SEND_COMMAND 41001:1:0,'AUDIOPROCDD-33,DIALER[Dialer_1_Rx]'

// You must send this last after all configurations have been sent.
SEND-COMMAND 41001:1:0,'REINIT'
```

Once you have sent this command and the Duet module has successfully initialized, then you can control the dialer control by using the "DIALER" software component API on virtual port 33. Here is an example of the API. A list of all available API commands, channels and levels is available later in this manual.

// DIALER (dials a phone number)
SEND_COMMAND 41001:33:0,'DIALNUMBER-8002220193'

VOIP

The VOIP software component has been designed to control the Clearone Converge Pro 2 VOIP Dialer. You will use the DIALER software component API to control the Clearone Converge Pro 2 Dialer. Some custom API methods have been added to complete control the VOIP Dialer.Like the Dialer software component for analog calls the VOIP software component only requires the end point name. However the end point name is not the endpoint name of the VOIP RX input it is simply the VOIP name.

VoIP[EndPointName]

Example 1:

I would like to control the VOIP Dialer that was named "Voip_1" on the assigned virtual port 34 to the duet module. So, on the online event of the first virtual port, I will send theses commands.

// VoIP[EndPointName]
SEND_COMMAND 41001:1:0,'AUDIOPROCDD-34,VOIP[Voip_1]'

// You must send this last after all configurations have been sent.
SEND-COMMAND 41001:1:0,'REINIT'

Once you have sent this command and the Duet module has successfully initialized, then you can control the VOIP dialer control by using the "DIALER" software component API on virtual port 34. Here is an example of the DIALER API. A list of all available API commands, channels and levels is available later in this manual.

// DIALER (dials a phone number)
SEND_COMMAND 41001:33:0,'DIALNUMBER-8002220193'

SKYPE

The SKYPE software component has been designed to control the Clearone Converge Pro 2 SKYPE Dialer. You will use the SKYPE software component API to control the Clearone Converge Pro 2 Dialer. Some custom API methods have been added to complete control the SKYPE Dialer.Like the Dialer software component for analog calls the SKYPE software component only requires the end point name.

SKYPE[EndPointName]

Example 1:

```
SEND_COMMAND 41001:1:0,'AUDIOPROCADD-10,SKYPE[Skype_Name_2_01]'

// You must send this last after all configurations have been sent.
SEND_COMMAND 41001:1:0,'REINIT'
```

Once you have sent this command and the Duet module has successfully initialized, then you can control the SKYPE dialer control by using the "DIALER" software component API on virtual port 34. Here is an example of the DIALER API. A list of all available API commands, channels and levels is available later in this manual.

```
// DIALER (dials a phone number)
SEND_COMMAND 41001:10:0,'DIALNUMBER-8002220193'
```

Channels

The UI module controls the audio processor via channel events (NetLinx commands *pulse, on, and off*) sent to the COMM module. The channels supported by the COMM module are listed below. These channels are associated with the virtual device(s) and are independent of the channels associated with the touch panel device. Not all channels will be available on all virtual ports; it will depend on what type of component is mapped to which virtual port as discussed above.

Note: An '*' indicates an extension to the standard API

Channel	Description
24	PULSE: Increment audio processor level (GAINMUTE EP_LEVEL and XPOINT software components)
25	PULSE: Decrement audio processor level (GAINMUTE EP_LEVEL and XPOINT software components)
26	PULSE: Cycle audio processor state (GAINMUTE EP_STATE and XPOINT software components)
199	ON: Audio processor state ON – provides feedback also OFF: Audio processor state OFF – provides feedback also (GAINMUTE EP_STATE and XPOINT software components)
201	PULSE: Redials the last call *(DIALER and VOIP software components)*
204	PULSE: Cycles Off Hook *(DIALER software components)*
208	PULSE: Performs a Hook Flash *(DIALER and VOIP software components)*
238	ON: Place a DIALER component off hook. OFF: Place a DIALER component on hook. *(DIALER software components)*
239	On: Places DIALER component in auto-answer mode. OFF: Places DIALER component out of auto-answer mode. *(DIALER software components)*
251	ON: Device is Online – feedback only OFF: Device is not Online – feedback only *(First Virtual Port Only)*
252	ON: Data is Initialized – feedback only OFF: Data is not Initialized – feedback only *(First Virtual Port Only)*
*301	PULSE: Performs Hold functionality. *(VoIP software Component)*
*302	ON: Outgoing calls ringing – feedback only *(VoIP software Component)*

*303	ON: Incoming calls ringing – feedback only *(VoIP software Component)*
*304	ON: Indicates busy – feedback only *(VoIP software Component)*
*305	ON: Indicates the DSP encountered a warning or error – feedback only *(VoIP software Component)*
*306	ON: Call Waiting – feedback only *(VoIP software Component)*
*307	PULSE: Transfer *(VoIP software component)*
*308	PULSE: Blind Transfer *(VoIP software component)*
*311	PULSE: Conference Call 1 *(VoIP software component)*
*312	PULSE: Conference Call 2 *(VoIP software component)*
*313	PULSE: Conference Call 3 *(VoIP software component)*
*314	PULSE: Conference Call 4 *(VoIP software component)*
*315	PULSE: Conference Call 5 *(VoIP software component)*

Table 2 - Virtual Device Channel Events

Levels

The UI module controls the audio processor via level events (NetLinx command *send_level*) sent to the COMM module. The levels supported by the COMM module are listed below. These levels are associated with the virtual device(s) and are independent of the levels associated with the touch panel device.

Note: An '*' indicates an extension to the standard API.

Level	Description
1	Audio Processor Level(0..255) *(Level and LEVELSTATE software Component)*

Table 3 - Virtual Device Level Events

Command Control

The UI module controls the audio processor via command events (NetLinx command *send_command*) sent to the COMM module. The commands supported by the COMM module are listed below.

Note: An '*' indicates an extension to the standard API.

Command	Description
AUDIOPROCADD-<index>, <address>	Adds an audio processor component at the specified index. Use the first virtual port when sending this command. <index> : 1-250 <address>: GAINMUTE[<EndPointName>|<GAIN/GAIN_FINE>|<STEP>] EP_LEVEL[<EndPointName>|<BlockName>|<ParameterName>|<MAX>|<MIN>|<STEP>] EP_STATE[<EndPointName>|<BlockName>|<ParameterName>] XPOINT[<InputEndPointName>|<OutputEndPointName>|<MAX>|<MIN>|<Type>] RM_STATE[<RoomNumber>|<RoomOption>|<DividerNumber>] DIALER[<EndPointName>] VOIP[<EndPointName>] AUDIOPROCADD-2,GAINMUTE[Fader1|GAIN|2] AUDIOPROCADD-3,EP_LEVEL[Mic-01|AGC|RESPONSE_TIME|10000|100|1] AUDIOPROCADD-4,EP_STATE[Mic-01|GAITING|CHAIRMAN] AUDIOPROCADD-5,XPOINT[Mic-01|Speakers|12|-20|GATED] AUDIOPROCADD-6,RM_STATE[Router1|IndividualDividerState|1] AUDIOPROCADD-7,DIALER[Dialer_1_Rx] AUDIOPROCADD-8,VOIP[Voip_1]
DEBUG-<value>	Set the state of debugging messages in the UI module and the Comm. module. Use the first virtual port when sending this command. **Note:** See Programming Notes section. <value> : 1 = set only error messages on 2 = set error and warning messages on 3 = set error, warning & debug messages on 4 = set all messages on DEBUG-1

?DEBUG	Request the state of the debug feature. Use the first virtual port when sending this command. ?DEBUG
PASSBACK-<state>	Enable or disable response reporting from the device. When enabled device responses will be sent as strings to the virtual device. Use the first virtual port when sending this command. **Note:** By default, this is set to off at startup. <state> : 0 = Off (default) 1 = On PASSBACK-1 PASSBACK-0
PASSTHRU-<string>	Allows user the capability of sending commands directly to whatever unit is attached with minimal processing by the Duet module. User must be aware of the protocol implemented by the unit to use this command. This gives the user access to features that may not be directly supported by the module. For more information, see the "Adding Functions to Modules" section below. Use the first virtual port when sending this command. <string> : string to send to unit PASSTHRU-@PPS:0 (pause)

PROPERTY-<key>,<value>	Set the value of property <key> to <value>. This must be followed by the REINIT command to take effect. These values are not initialized by default. Use the first virtual port when sending this command. <key> : IP_Address <value> : String = representing an IP address <key> : User_Name <value> : String = representing a user name <key> : Password <value> : String = representing a password <key> : Poll_Time <value> : 0 = off,1000..300000 – Time in milliseconds <key> : Baud_Rate <value> : 9600 19200 38400 57600 115200 = default PROPERTY-IP_Address,10.0.0.5 PROPERTY-Poll_Time,30000 (Time in milliseconds) PROPERTY-Baud_Rate,38400 (Any valid baud rate)
?PROPERTY-<key>	Get the value of a property <key>. If the value is not initialized, an empty string is returned. Use the first virtual port when sending this command. <key> : IP_Address Poll_Time Baud_Rate User_Name Password ?PROPERTY-IP_Address ?PROPERTY-Poll_Time ?PROPERTY-Baud_Rate
REINIT	Re-initializes the communication link and data. **Note:** This command deletes any messages waiting to go out to the device. Use the first virtual port when sending this command. REINIT
?VERSION	Query for the current version number of the Duet module. Use the first virtual port when sending this command. ?VERSION

ACTUAL_LEVEL_VALUE-<level>	* Allows you to set the actual value the hardware requires, bypassing the 0-255 AMX level range. <level> : Actual Valid Hardware Level Value // sets value to -50 ACTUAL_LEVEL_VALUE—50 (GAINMUTE EP_LEVEL and XPOINT software components)
?ACTUAL_LEVEL_VALUE	* This will allow you to query the actual hardware level, bypassing the 0-255 AMX level range. ?ACTUAL_LEVEL_VALUE (GAINMUTE EP_LEVEL and XPOINT software components)
?DIALERSTATUS	Get the dialer's current status. Use the mapped virtual port when sending this command. ?DIALERSTATUS (DIALER AND VOIP Software components)
DIALNUMBER-<number>	Dial the specified number. Use the mapped virtual port when sending this command. <number> : string = represents the number to be dialed DIALNUMBER-4696248000 (DIALER AND VOIP Software components)
DTMF-<digit>	Send a DTMF tone for the specified character without regard for hook status. Use the mapped virtual port when sending this command. <digit> : 0..9,*,# = digit DTMF-9 (DIALER AND VOIP Software components)
VOIP_CALL_SELECT-<id>	* The Clearone Converge Pro 2 VoIP dialer can have up to 5 calls at the same time in various stages. This command picks which call your API is controlling. <id> : 1..5 = call id VOIP_CALL_SELECT-2 (VoIP Software Component)
?VOIP_CALL_SELECT	*Queries for the current active call id. ?VOIP_CALL_SELECT (VOIP Software Component)

MACRO-<macro name>	*The Clearone Converge Pro 2 will run a macro (a series of commands) on a box. Macros are defined using the Converge Pro 2 Console application. MACRO-Macro_1
AUTO_ANSWER_RING_COUNT-<Number Of Rings>	*This sets the number of rings until the Dialer answers the call automatically. <Number of Rings> : 0 ..4 0 = disabled 1 – 4 = Number of rings before the call is answered automatically. AUTO_ANSWER_RING_COUNT-2 (DIALER Software Component)
?OFF_HOOK_DURATION	*Poll the Dialer for the length of time the dialer has been off the hook. (DIALER Software Component)
*?SKYPE_ACTIVE_SESSION	Requests the Info of the Current Active Session. Returns: SKYPE_ACTIVE_SESSION_MEETING_ID-"<id>" SKYPE_ACTIVE_SESSION_STATUS-<status> SKYPE_ACTIVE_SESSION_IS_CONFERENCE-<TRUE|FALSE> SEND_COMMAND 41001:10:0,'?SKYPE_ACTIVE_SESSION'
*SKYPE_ADD_CONTACT_TO_ACTIVE_SESSION	Adds the currently selected contact from the list to the active session. The selected active contact is displayed with: SKYPE_CONTACT_LIST_SELECTION_ID-<id> SKYPE_CONTACT_LIST_SELECTION_NAME-<name> SKYPE_CONTACT_LIST_SELECTION_TYPE-<type> SKYPE_CONTACT_LIST_SELECTION_GROUP-<group> SEND_COMMAND 41001:10:0,'SKYPE_ADD_CONTACT_TO_ACTIVE_SESSION'
*SKYPE_ADD_CONTACT_TO_GROUP	Adds the currently selected contact from the list to a group. Group selection will follow this command. The selected active contact is displayed with: SKYPE_CONTACT_LIST_SELECTION_ID-<id> SKYPE_CONTACT_LIST_SELECTION_NAME-<name> SKYPE_CONTACT_LIST_SELECTION_TYPE-<type> SKYPE_CONTACT_LIST_SELECTION_GROUP-<group> SEND_COMMAND 41001:10:0,'SKYPE_ADD_CONTACT_TO_GROUP'
*SKYPE_CALL_ACCEPT	Accepts the INCOMINGCALL. SEND_COMMAND 41001:10:0,'SKYPE_CALL_ACCEPT'

*SKYPE_CALL_HOLD	Puts the ACTIVE Session on hold. Active Session is displayed using these. SKYPE_ACTIVE_SESSION_MEETING_ID-"<id>" SKYPE_ACTIVE_SESSION_STATUS-<status> SKYPE_ACTIVE_SESSION_IS_CONFERENCE-<TRUE|FALSE> SEND_COMMAND 41001:10:0,'SKYPE_CALL_HOLD'
*SKYPE_CALL_LEAVE	Leaves the ACTIVE Session. Active Session is displayed using these. SKYPE_ACTIVE_SESSION_MEETING_ID-"<id>" SKYPE_ACTIVE_SESSION_STATUS-<status> SKYPE_ACTIVE_SESSION_IS_CONFERENCE-<TRUE|FALSE> SEND_COMMAND 41001:10:0,'SKYPE_CALL_LEAVE'
*SKYPE_CALL_RESUME	Resumes the ACTIVE Session. Active Session is displayed using these. SKYPE_ACTIVE_SESSION_MEETING_ID-"<id>" SKYPE_ACTIVE_SESSION_STATUS-<status> SKYPE_ACTIVE_SESSION_IS_CONFERENCE-<TRUE|FALSE> SEND_COMMAND 41001:10:0,'SKYPE_CALL_RESUME'
*?SKYPE_CONTACT_LIST	Refreshes the current contact list selection. Contact list is displayed using these. SKYPE_CONTACT_LIST-<item>,<count>,"<name>" SEND_COMMAND 41001:10:0,'?SKYPE_CONTACT_LIST'

*SKYPE_CONTACT_LIST_SELECTION-<item>	Selects an item from the contact list.
	
	Contact list is displayed using these.
	SKYPE_CONTACT_LIST-<item>,<count>,"<name>"
	
	// Selects the first item in the list.
	SEND_COMMAND 41001:10:0,'SKYPE_CONTACT_LIST_SELECTION-1'
	
	// Goes to the TOP most level of the list. So resets the list to the TOP.
	SEND_COMMAND 41001:10:0,'SKYPE_CONTACT_LIST_SELECTION-0'

*SKYPE_CREATE_SESSION_USING_SELECTION	Creates a call session using the selection from the contact list. If a group is selected, then everyone in the group is added to the session. If a contact is selected only that contact is added to the session. These are used to display the current selection from the contact list. SKYPE_CONTACT_LIST_SELECTION_NAME-<name> SKYPE_CONTACT_LIST_SELECTION_TYPE-<type> SKYPE_CONTACT_LIST_SELECTION_GROUP-<group> SEND_COMMAND 41001:10:0,'SKYPE_CREATE_SESSION_USING_SELECTION'
*SKYPE_DELETE_GROUP	Deletes the currently selected GROUP from the contact list. If selection must be a GROUP and not an individual. This will remove all contacts from this group,but will not delete the contacts. These are used to display the current selection from the contact list. SKYPE_CONTACT_LIST_SELECTION_NAME-<name> SKYPE_CONTACT_LIST_SELECTION_TYPE-<type> SKYPE_CONTACT_LIST_SELECTION_GROUP-<group> SEND_COMMAND 41001:10:0,'SKYPE_DELETE_GROUP'
*SKYPE_DIAL_CONTACT	Dials the currently selected contact from the contact list. The selection must be a contact and not a group. These are used to display the current selection from the contact list. SKYPE_CONTACT_LIST_SELECTION_NAME-<name> SKYPE_CONTACT_LIST_SELECTION_TYPE-<type> SKYPE_CONTACT_LIST_SELECTION_GROUP-<group> SEND_COMMAND 41001:10:0,'SKYPE_DIAL_CONTACT'
*SKYPE_FILTER_SEARCH_TEXT-<text>	This can be used to filter contacts from the current known contacts from the contact list and recent searches. To do an exhaustive search, use 'SKYPE_PERFORM_SEARCH' in conjunction with entering the text. The text is not case sensitive. SEND_COMMAND 41001:10:0,'SKYPE_FILTER_SEARCH_TEXT-A'
*?SKYPE_FILTER_SEARCH_TEXT	This is used to query the current filtered/search text.
*SKYPE_JOIN_MEETING-<id>	Join an already in progress meeting using it's ID. SEND_COMMAND 41001:10:0,'SKYPE_JOIN_MEETING-2231231@microsoft.com'
*SKYPE_JOIN_CONFERENCE-<id>	Join an already in progress conference using it's ID. SEND_COMMAND 41001:10:0,'SKYPE_JOIN_ CONFERENCE-2231231@microsoft.com'

*SKYPE_MEET_NOW	Start a meeting now. SEND_COMMAND 41001:10:0,'SKYPE_MEET_NOW'
*SKYPE_PERFORM_SEARCH	To do an exhaustive search, use 'SKYPE_PERFORM_SEARCH' in conjunction with entering the text. The text is not case sensitive. SEND_COMMAND 41001:10:0,'SKYPE_PERFORM_SEARCH'
*?SKYPE_REG_STATUS	Query the current registration status of the SKYPE component. SEND_COMMAND 41001:10:0,'?SKYPE_REG_STATUS'
*SKYPE_REMOVE_CONTACT_FROM_GROUP	Removes the currently selected contact from selected group from the contact list. The selection must be a contact and not a group. The group to remove will be the group that is currently associated with the displayed contact. These are used to display the current selection from the contact list. SKYPE_CONTACT_LIST_SELECTION_NAME-<name> SKYPE_CONTACT_LIST_SELECTION_TYPE-<type> SKYPE_CONTACT_LIST_SELECTION_GROUP-<group> SEND_COMMAND 41001:10:0, 'SKYPE_REMOVE_CONTACT_FROM_GROUP'
*SKYPE_REMOVE_CONTACT_FROM_ALL_GROUPS	Removes the currently selected contact from the contact list. The selection must be a contact and not a group. The group to remove will be the group that has been selected. These are used to display the current selection from the contact list. SKYPE_CONTACT_LIST_SELECTION_NAME-<name> SKYPE_CONTACT_LIST_SELECTION_TYPE-<type> SKYPE_CONTACT_LIST_SELECTION_GROUP-<group> SEND_COMMAND 41001:10:0, 'SKYPE_REMOVE_CONTACT_FROM_GROUP'
*SKYPE_SELECT_SESSION-<item>	Select a session from the session list and makes it the active session. These are used to display the current sessions. SKYPE_SESSION_LIST-<item>,<count>,"<id>" SEND_COMMAND 41001:10:0,'SKYPE_SELECT_SESSION-2'
*?SKYPE_SESSION_LIST	Refreshes the current sessions. These are used to display the current sessions. SKYPE_SESSION_LIST-<item>,<count>,"<id>" SEND_COMMAND 41001:10:0,'?SKYPE_SESSION_LIST'
?LOCAL_NUMBER	*Poll the dialer for the phone line number the DSP is connected to. (DIALER Software Component)

VOIP_CALL_FORWARD_MODE-<mode>	*Set the call forwarding mode of the VOIP dialer. These modes effect when the call is forwarded. <Mode> : DISABLED UNCONDITIONAL BUSY NO_REPLY VOIP_CALL_FORWARD_MODE-NO_REPLY (VOIP Software Component)
?VOIP_CALL_FORWARD_MODE	*Queries for the current call forward mode. ?VOIP_CALL_FORWARD_MODE (VOIP Software Component)
VOIP_DO_NOT_DISTURB_STATE-<mode>	*Set the do not disturb mode of the VOIP dialer. <Mode< : DND_NOT_SET DND_CALL_MUTE DND_CALL_REJECT VOIP_DO_NOT_DISTURB_STATE-DND_CALL_MUTE (VOIP Software Component)
?VOIP_DO_NOT_DISTURB_STATE	*Queries for the current do not disturb mode of the VOIP dialer. ?VOIP_DO_NOT_DISTURB_STATE (VOIP Software Component)
?VOIP_CALL_LED_STATE-<id>	*Queries the VOIP software component for the state of a “LED” for the selected call. <id> : 1..5 = call id ?VOIP_CALL_LED_STATE-1 (VOIP Software Component)
?VOIP_CALL_STATE-<id>	*Queries the VOIP dialer for the for the state of the selected call. <id> : 1..5 = call id ?VOIP_CALL_STATE-1 (VOIP Software Component)

?VOIP_CALL_INFO_TEXT-<id>	*Queries the VOIP dialer for the info text of the selected call. <id> : 1..5 = call id ?VOIP-CALL_INFO_TEXT-1 (VOIP Software Component)

Table 4 – Send Command Definitions

Command Feedback

The COMM module provides feedback to the User Interface module for audio processors changes via command events. The commands supported are listed below.

PLEASE NOTE: Feedback is only provided when there is a state change. If no state change resulted from the command sent in, then no feedback will be returned.

Command	Description
ACTUAL_LEVEL_VALUE-<level>	* Reports the current absolute hardware value for the mapped component, bypassing the 0-255 AMX level range. <level> : Actual Valid Hardware Level Value // -50 ACTUAL_LEVEL_VALUE--50 (GAINMUTE EP_LEVEL and XPOINT software components)
DEBUG-<value>	Reports the state of debugging messages in the UI module and the Comm. module. This is reported on the first virtual port. <value> : 1 = set only error messages on 2 = set error and warning messages on 3 = set error, warning and debug messages on 4 = set all messages on DEBUG-1

PROPERTY-<key>,<value>	Reports the value of property <key>. This is reported on the first virtual port. <key> : IP_Address <value> : String = representing an IP address <key> : User_Name <value> : String = representing a user name <key> : Password <value> : String = representing a password <key> : Poll_Time <value> : 0 = off,1000..300000 – Time in milliseconds <key> : Baud_Rate <value> : 9600 19200 38400 57600 115200 = default PROPERTY-IP_Address,10.0.0.5 PROPERTY-Poll_Time,30000 (Time in milliseconds) PROPERTY-Baud_Rate,38400 (Any valid baud rate)
VERSION-<value>	Reports the current version number of the Duet module. This is reported on the first virtual port. <value> : current version number in xx.yy.zz format VERSION-1.0.0
DIALERSTATUS-<state>	Reports the current state of the dialer. This is reported on the mapped virtual port. <state> : BUSY CONNECTED DIALING DISCONNECTED FAULT NEGOTIATING RINGING INVALID DIALERSTATUS-DIALING (DIALER and VOIP Software Components)
INCOMINGCALL-<number>	Reports when and incoming call is detected. This is reported on the mapped virtual port. <number> : Phone Number INCOMINGCALL-8002220193 (Dialer Software Component)

SKYPE_ACTIVE_SESSION_MEETING_ID-"<id>"	Reports when Active Session has changed or queried. SKYPE_ACTIVE_SESSION_MEETING_ID-"hdhdhdh@mcmcmcm.com"
SKYPE_ACTIVE_SESSION_STATUS-<status>	Reports when Active Session has changed or queried. Status: UNKNOWN IDLE CONNECTING RINGING BUSY ACTIVE HOLD INCOMING CONFERENCE_JOIN INVITE_JOIN_AUDIO SKYPE_ACTIVE_SESSION_STATUS-CONNECTING
SKYPE_ACTIVE_SESSION_IS_CONFERENCE-<TRUE|FALSE>	Reports when Active Session has changed or queried. TRUE equals active session is of type CONFERENCE. SKYPE_ACTIVE_SESSION_IS_CONFERENCE-TRUE
SKYPE_CONTACT_LIST-<item>,<count>,"<name>"	Reports when contact list has changed or queried. SKYPE_CONTACT_LIST-1,5,"Joe Smith" SKYPE_CONTACT_LIST-2,5,"Jill Smith" SKYPE_CONTACT_LIST-3,5,"Jack Smith" SKYPE_CONTACT_LIST-4,5,"JakeSmith" SKYPE_CONTACT_LIST-5,5,"Judy Smith"
SKYPE_CONTACT_LIST_TYPE-<type>	Reports when contact list has changed or queried. This represents that type of items displayed. Types: UNKNOWN GROUPS CONTACTS INFOONLY SKYPE_CONTACT_LIST_TYPE-GROUPS
SKYPE_CONTACT_LIST_HELP_TEXT-<text>	Reports when contact list has changed or queried. This represents that user help text and prompting info. SKYPE_CONTACT_LIST_HELP_TEXT-Select Contact from List
SKYPE_CONTACT_LIST_SELECTION_ID-<id>	Reports when contact list selection has changed or queried. SKYPE_CONTACT_LIST_SELECTION_ID-sip:name@domain.com
SKYPE_CONTACT_LIST_SELECTION_NAME-<name>	Reports when contact list selection has changed or queried. SKYPE_CONTACT_LIST_SELECTION_NAME-Jake Smith

SKYPE_CONTACT_LIST_SELECTION_TYPE-<type>	Reports when contact list selection has changed or queried. Types: UNKNOWN GROUPS CONTACTS SKYPE_CONTACT_LIST_SELECTION_TYPE-CONTACTS
SKYPE_CONTACT_LIST_SELECTION_GROUP-<group>	Reports when contact list selection has changed or queried. SKYPE_CONTACT_LIST_SELECTION_GROUP-Favorites
SKYPE_ERROR_MESSAGE-<error>	Reports when the Clearone S4B has caused and error. SKYPE_ERROR_MESSAGE-“Some Error Text!”
SKYPE_FILTER_SEARCH_TEXT-<text>	Repors when filter/search text has been changed or queried. SKYPE_FILTER_SEARCH_TEXT-A
SKYPE_REG_STATUS-<status>	Reports when contact list selection has changed or queried. Statuses: REGISTERED NOT_REGISTERED NO_PROXY_DEFINED SKYPE_REG_STATUS-REGISTERED
SKYPE_SESSION_LIST-<item>,<count>,"<id>"	Reports when session list has changed or queried. SKYPE_SESSION_LIST-1,2,"test@meeting1.com" SKYPE_SESSION_LIST-2,2,"test@meeting2.com"
VOIP_CALL_SELECT-<id>	*Reports the active call that the API is controlling. <id> : 1..5 = call id VOIP_ACTIVE_CALL-2 (VOIP Software Component)
VOIP_DO_NOT_DISTURB_STATE-<mode>	*Reports the current call prompt for the call id. You will receive the changes unsolicited. <mode> : DND_NOT_SET DND_CALL_MUTE DND_CALL_REJECT VOIP_DO_NOT_DISTURB_STATE-DND_NOT_SET (VOIP Software Component)

VOIP_CALL_STATE-<id>,<text>	*Reports the current call state for the call id. You will receive the changes unsolicited. <id> : 1..5 = call id <text> : UNKNOWN IDLE DIALING DIAL_TONE ACTIVE CONNECTED BUSY HOLD INCOMING INPROCESS INVALID BLIND_TRANSFER_HOLD BLIND_TRANSFERING_DIAL_TONE BLIND_TRANSFERING_DIALING BLIND_TRANSFERING_IN_PROCESS BLIND_TRANSFERING_RINGING CONFERENCE_ACTIVE CONFERENCE_HOLD VOIP_CALL_STATE-3,ACTIVE (VOIP Software Component)
VOIP_CALL_INFO_TEXT-<id>,<text>	*Reports the current call number for the call id. You will receive the changes unsolicited. <id> : 1..5 = call id <text> : phone number VOIP_CALL_INFO_TEXT-1,600 (VoIP Software Component)
VOIP_CALL_FORWARD_MODE-<mode>	*Reports the call forward mode the VOIP dialer is set to. <mode> : DISABLED UNCONDITIONAL BUSY NO_REPLY VOIP_CALL_FORWARD_MODE-NO_REPLY (VOIP Software Component)

VOIP_CALL_LED_STATE- <id><state>	*Reports the state of a "LED" for the selected call in the VOIP dialer software component. <id> : 1..5 = call id <state> : OFF ON BLINK VOIP_CALL_LED_STATE-2,BLINK (VOIP Software Component)

Table 5 - Command Feedback Definitions

Programming Notes

PLEASE NOTE: In order to establish a communication connection between the module and the device, the module must know what IP address to connect to. Use the PROPERTY- command to set the module's IP address and then use the REINIT command to force it to take affect. Here is an example of how these two commands are used:
Send_Command 41001:1:0, 'PROPERTY-IP_Address,192.168.103.30'
Send_Command 41001:1:0, 'REINIT'
Note that your virtual device (41001:1:0) and the IP address of your device (192.168.103.30) may differ from this example. Substitute the appropriate values where necessary.

Adding Functions to Modules

Commands to the device

This module supplies a mechanism to allow additional device features to be added to software using the module. This is the 'PASSTHRU-' command, which allows protocol strings to be passed through the module. The device-specific protocol must be known in order to use this feature.

As an example, suppose that a module for a projector has not implemented the 'white balance adjustment' feature. The command that the projector protocol requires is 03H, 10H, 05H, 14H, followed by a checksum. The documentation for the 'PASSTHRU-' command specifies that the module will automatically generate the checksum. In this case, the following string should be sent from the UI code to implement 'white balance adjustment'.

```
send_command vdvDevice, "'PASSTHRU-',$03,$10,$05,$14"
```

The reason to use 'PASSTHRU-' instead of sending a protocol string directly to the device port is that the device may require command queuing, calculation of checksums, or other internal processing, which would not be done if the string was sent directly. Because of this, it is best to filter all communication TO the device through the module. (The module documentation will indicate any processing that will be automatically done to the 'PASSTHRU-' command like checksum calculation.)

Responses from the device

The module will automatically interpret replies from the device and pass these on to the application code according to the documented API. Some device replies may not be passed on to the application code. To see all replies from the device unfiltered by the module, enable PASSBACK and use a DATA_EVENT with a string handler in the UI code. Again, the device-specific protocol must be known in order to interpret these responses. Even when PASSBACK is enabled, the module will still interpret device responses according to the standard API.