

TCP/IP Extension to the Ashly Ethernet Control Protocol.

Extension Goal:

With the Goal of simplicity to the end user in mind it is necessary that we provide products are easy to configure. The Ashly Ethernet Control Protocol is built upon a UDP/IP Protocol that has allowed the idea of plug it in and it will just Work. However for the contracting market it is important that we also provide way to control the product which Third party programmers are comfortable working with. Both AMX & Crestron recommend TCP/IP instead of UDP/IP. TCP/IP will require IP parameters to be setup properly however in an installed situation this can be assumed. Therefore the Ashly Protocol will be extended to support TCP communications.

Introduction:

This document defines the communications protocol used by Ashly Audio NE Devices such as the ne24.24M to communicate over TCP/IP Network. This protocol will sit on top of TCP/IP (as opposed to UDP/IP used by the standard Ashly Protocol). This will allow PCs and third party controllers to establish a connection and both send changes and receive updates on a single connection. This will also provide some guarantee of data arrival without the need for acknowledgements to be implemented by the installer.

Establishing a Connection:

The Ashly devices will listen for TCP connections on TCP port 3100 (this is the same port used for UDP). A maximum of 12 concurrent TCP connections will be allowed to a single Ashly device. Using Crestron or AMX a connection may be established and will be maintained until closed. The Ashly devices will periodically send updates even if there are no changes to ensure the connection is still valid.

Message Structures:

There are two different structures used in this protocol, The Data Request/ Response & Updates are used to communicate between the PC and Device about changes in the data set that have occurred. The Set Data Messages are used to change the data set on a device.

Data Requests/ Response & Updates

The Data Request Response/ Update protocol is typically used by the PC/ Control Platform to request the current settings inside an Ashly Device. Also this is the same protocol used by the device to send "Unsolicited" Updates to the control platform when something is changed on the device. For example if the user presses Mute on another control platform, the device will send a Mute update to all other PCs/ Control Platforms connected. These messages cannot change settings and therefore there is no security in the message. These messages should only be used with the "Working Settings" (0x00 for byte 12). After the header the data payload consists of options as defined in the Ashly Protocol Appendix A. After the last option the End Opt MUST be present (0xFF). This

signifies the end of the usable message. For requests of data it is only necessary to specify enough data to uniquely identify the parameter. For example for a Mute you may set the bytes to follow to 2 and only specify the channel type and channel and skip dummyming in the data typically occupying byte 3. This was done in the example message.

<u>Byte#</u>	<u>Value</u>	<u>Description</u>
1-4	0x8F	Header Identifier
5-10	0x00	Reserved Leave All 6 locations blank (0x00)
11	0y	Request/reply (0 = request, 1 = reply)
12	xx	Source 00 – Default (working Settings)
13	0x00	Reserved, Interbox Communications (Ignore data in response)
14	0x00	Reserved, Value ignored should be set to 0x00
15-??	--	Option Payload must be terminated with END_OPT (0xFF)

Sample Message (Mute Request Input 1) sent to device

0x8F 0x8F 0x8F 0x8F 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02 0x02 0x01 0x00 0xFF

Sample Response/ Update, Sent from Device (Muted)

0x8F 0x8F 0x8F 0x8F 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0x00 0x00 0x02 0x03 0x01 0x00 0x01 0xFF

Set Data Message:

The Set Data message is used to change data on an Ashly Device. This protocol is similar to the Request message however this message basically contains security information, which may or may-not be used. A user name/ password may be specified (using ASCII encoding) or left as null (0x00 for all locations) if this is left null then the last obtained security level is used. If security was never obtained then the default users security level is used. For those who do not wish to implement security it is sufficient to just use the default users level (always leave the user name and password as null). The Message Number may also be dummied in as the same value for all messages. This is only important if multiple set messages are sent before acknowledgments are received and the PC/Control system cares about the acknowledgment response. The Acknowledgement Status may also be ignored on the reply if the end user does not care since under normal conditions it should always be successful.

<u>Byte#</u>	<u>Value</u>	<u>Description</u>
1-4	0xAA	Header Identifier
5-10	0x00	Reserved Leave All 6 locations blank (0x00)
11-18	yy	User Name (May be left blank, 0x00)
19-26	yy	Password (May be left black, 0x00)
27	zz	Message Number (byte 1)
28	zz	Message Number (byte 2)
29	aa	Acknowledgment status
30	0x00	Reserved
31-??	--	Option Payload must be terminated with END_OPT (0xFF)

Message Number:

A number associated with a message. This should be implemented as a circular counter between 0 and 65535. For correct messages received the acknowledgment may contain only the header information sent. **If only one message is sent at a time this may be dummied in using the same value for all set messages.**

Acknowledgment status

The status of the acknowledgment may be either

0 – indicates that it is the originals message not the acknowledgment.

1 – message received ok.

2 – indicating the message was not passed due to insufficient security.

3 – indicates that at least one parameter in the previous message was not accepted due to insufficient resources (namely DSP horsepower).

4 – indicates a DSP Error has occurred (reserved byte 31 indicates the Error Code

5 – indicates that it is already in bulk update mode (returned only for opt OPT_BULK_UPDATE

16 - Save to Temp Buffer (original message) (Processed)

Sample Set Message (Mute Set on Input 1) sent to device

0xAA 0xAA 0xAA 0xAA 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02 0x03 0x01 0x00 0x01 0xFF

Sample Set Response, Sent from Device

0xAA 0xAA 0xAA 0xAA 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x01 0x00 0xFF