STcontroller API UDP Examples

Updated: 20-Oct-2021

All commands and messages used between STcontroller and Studio Technologies' products can be recreated and transmitted with User Datagram Protocol (UDP) transmitter software. Studio Technologies has taken the time to test and outline the usage with a number of their products listed below. The construction of the UDP message will also be discussed.

Model 23X Announcer's Consoles

Table 1 - Model 23X Remote Settings

Setting ID	Setting Name	Setting Values
0x01	Microphone Preamplifier Gain	0x14-0x41 (20-65dB)
0x02	Microphone P48 Phantom Power	0x00 - Off 0x30 - On (48V)

These are settings within the Mic Pre Bus command sent to the M23X from STcontroller.

The data structure for these settings within the overall UDP packet is:

```
0x5A 0x12 0x00 0x02 <setting ID> <setting value> <crc8>
```

To set the gain to 27dB, the data structure would be:

```
0x5A 0x12 0x00 0x02 0x01 0x1B 0xD1
```

To turn on P48 Phantom Power, the data structure would be:

```
0x5A 0x12 0x00 0x02 0x02 0x30 0xEB
```

Usage

STcontroller communicates with our Dante products using Audinate's Packet Bridge protocol, which allows an OEM's CPU to receive UDP datagrams via the corresponding Dante interface. A reliable implementation of Packet Bridge requires the use and licensure of Dante API, however UDP datagrams sent to the appropriate address will suffice in this case. In order to construct a UDP message, a 24 byte header must be concatenated with data specific to the device being transmitted to. If a packet sniffing tool is used to analyze messages sent to a device from STcontroller, the header will be similar to the example below, but the example header can also be used in your own application. The example header is as follows:

`0xFF 0xFF 0x00 <msg_len> 0x07 0xE1 0x00 0x00 0x90 0xB1 0x1C 0x5B 0xD2 0x85 0x00 0x00 0x53 0x74 0x75 0x64 0x69 0x6F 0x2D 0x54 [data]`

`msg_len` is the combined length of the header and data, and is the only modifiable value in the example header.

Following the header is the unique device data. It is indicated with the Studio Technologies start byte `0x5A`. It is typically followed by the specific command ID (`cmd_id`), its data length (`cmd_data_len`), setting ID (`setting_id`) and value (`setting_val`), and finally a crc (`crc8`). Here is the typical structure:

`0x5A <cmd_id> <cmd_data_len> [<setting_id>, <setting_val>, …] <crc8>`

Note that multiple settings can be set at the same time if desired. `crc8` is calculated as CRC-8/DVB-S2 and uses the Studio Technologies start byte through the command data in its calculation.

The example command below is for toggling the Talkback button on the Model 209 Talent Console when the Talkback button is set for Latching. The setting ID and value can be found in Table 2.

`0x5A 0x0D 0x02 0x0A 0x01 0x7C`

If combined with the necessary header, the complete message to be sent to the Model 209 is:

`0xFF 0xFF 0x00` **`0x1E`** `0x07 0xE1 0x00 0x00 0x90 0xB1 0x1C 0x5B 0xD2 0x85 0x00 0x00 0x53 0x74 0x75 0x64 0x69 0x6F 0x2D 0x54 0x5A 0x0D 0x02 0x0A 0x01 0x7C`

Note that `msg_len` is `0x1E`, or 30 decimal, which is the entire length of the message.

The message must be sent to the device's Dante IP address on port 8700. This can be found using Dante Controller. It is suggested that only one device should be transmitted to at one time, and that there should be at least 200ms between each transmitted message to allow for ample processing time.

This approach is slightly different from STcontroller which creates a subscription to the device in order to transmit the message more reliably. The device will always acknowledge a received message, however this is to a multicast address. One can also confirm that setting messages have been received by periodically polling a device for its settings. In the case of the Model 209, the following command can be used:

```
0x5A 0x0C 0x96
```

The entire message will be:

```
0xFF 0xFF 0x00 0x1B 0x07 0xE1 0x00 0x00 0x90 0xB1 0x1C 0x5B 0xD2 0x85
0x00 0x00 0x53 0x74 0x75 0x64 0x69 0x6F 0x2D 0x54 0x5A 0x0C 0x96
```

The response to this message will also be sent to a multicast address, but may be easier to identify using a packet sniffing software.

Testing

Using a packet sniffing software to identify and parse commands from STcontroller will help to test UDP transmitter solutions with Studio Technologies' STcontroller compatible products. Contact support if you have questions on how to proceed, or for example messages for your products.

Using Elgato Stream Deck and Bitfocus Companion

The Elgato Stream Deck should be used in conjunction with Bitfocus Companion. This will give the device access to a number of modules, specifically the Generic TCP and UDP module. This will allow a Stream Deck button push to transmit UDP packets to the Studio Technologies device.

When configuring the module, the following settings are required:

Target IP: the Studio Technologies device's Dante IP address
Target Port: 8700
Connect with TCP/UDP: UDP

When configuring a button, choose a KEY UP/OFF ACTION. This will ensure the command is only sent once and not overloading the device. The selected action should be [UDP module name]: Send Command. The configuration should be:

Delay: 0
Command: Hexadecimal values for the desired UDP packet (see above, must include header and data sections) without "0x", and each preceded by '%'; for example, if the packet is to be `0x00 0x01 0x02`, the field should have `%00%01%02`
Command End Character: None