RS232 Syntax from ZyPer Management Platform

ZeeVee, Inc.
295 Foster Street, Suite 200
Littleton, MA 01460 USA
February 28, 2024

Table of Contents

Introduction

It is possible to send an RS232 text string from the ZyPer Management Platform to any connected ZyPer4K, ZyPerUHD60 or ZyPerUHD endpoint on the network. (Encoder or Decoder) This function can be done from the ZMP command line interface or via a 3rd party control system. Details on syntax are shown below:

Note: ZyPer4K-XS and ZyPer4K-XR unit do not support RS232

RS232

The Management Platform (ZMP) must first be linked to the specific endpoint to send RS232 information. This can be done with either the dataConnect or switch command.

Examples:
dataConnect Dec1 server rs232 tunnelPort 1234

switch Dec1 server rs232

Note: The feature of dataConnect was added to allow a third party to connect to the ZMP server with a specific port and pass raw or telnet API commands (depending on the mode) to the server and port which is designated for a particular encoder or decoder.

Important Note: Issuing the dataConnect or switch command can cause the ZyPer endpoint to reboot to enable the link. Disconnecting the link can also cause the endpoint to reboot. The link should only be established once and then left alone to prevent undesired endpoint reboots.

When using any control system; that system is talking to our ZMP and not to any specific endpoint.

When sending RS232 commands to an encoder or decoder via the ZMP you must follow very specific syntax.

The ZeeVee command is: send <decoder_name> rs232 text

Here are examples on this. (Assume decoder name is *Dec1*)

Input command: send Dec1 rs232 Hello
Received at Dec1: Hello (Note, no line feed or carriage return)

Input command: send Dec1 rs232 Hello\r\n
Received at Dec1: Hello (with carriage return and line feed)

Input command: send Dec1 rs232 Hello World
Received at Dec1: Nothing. You get an error. Bad syntax. You cannot have a space between hello and world.

Input command: send Dec1 rs232 Hello_World
Received at Dec1: Hello_World (Note, no line feed or carriage return)

Input command: send Dec1 rs232 “Hello World”
Received at Dec1: Hello World (Note, no line feed or carriage return)

Input command: send Dec1 rs232 “Hello World”\r\n
Received at Dec1: Nothing. You get an error. Bad syntax. Token \r\n is invalid.

You need to contain the line feed and carriage return symbols inside the quotes in this case.

Input command: send Dec1 rs232 “Hello World\r\n”

Received at Dec1: Hello World (with carriage return and line feed)
Text can also be Hexadecimal Code as shown below:

Input command:
send Dec1 rs232 \x48\x65\x6c\x6c\x6f\x20\x57\x6f\x72\x6c\x64\x0A\x0D
Received at Dec1: Hello World (with carriage return and line feed)

Receiving RS232 feedback

The Management Platform also can receive RS232 communications that were input into a ZyPer endpoint. To view any such RS232 string, you use the "show responses" command.

Example:
Zyper$ show responses DEC1 rs232 since 0
device(d8:80:39:59:bf:57);
  device.rs232Response.0; string="Have a great day!\x0D"
  device.rs232Response.1; string="\x0A"
lastChangeId(2);
Success
Note that the ZyPer Management Platform can only show the user the responses. It has no means to act on any RS232 string it receives. This would require a 3rd party control system.

Important Note

When using Crestron as the control system, you need to append an extra \ symbol before the Carriage return symbol. Otherwise, carriage return does not work.

Example using Crestron to turn on/off LG display.

LG TV
ON
send DecoderName rs232 \x6B\x61\x20\x30\x31\x20\x30\x31\\x0D
OFF
send DecoderName rs232 \x6B\x61\x20\x30\x30\x20\x30\x30\\x0D

Tunnel Ports

There is a very convenient way to get RS232 data: TUNNELS.

```
Zyper$ dataConnect Decoder_1 server rs232
Dynamically assigned tunnel TCP port = 4100; telnet handshake mode
Success
Zyper$
Zyper$ show dataTunnels
data-sessions(d8:80:39:9b:9:a2);
  device: name=Decoder_1
  rs232Tunnel: port=4100
  rs232Tunnel-connections: none
Success
Zyper$
```

You can then connect to that tunnel port using TCP. Whatever is sent is forwarded to the device. Whatever the device returns is received on that TCP connection.

In the easiest case, you can just use telnet to connect to the tunnel.

You can specify the port number as well:

```
Zyper$ dataConnect Decoder_1 server rs232 tunnelPort 4101
tunnel TCP port = 4101; telnet handshake mode
Success
Zyper$
```

You can set the default TCP connection mode: raw|telnet (defaults to telnet).

```
Zyper$ set server dataTunnelMode raw|telnet
```

When in telnet mode the IAC commands are sent/received. Although most telnet clients will also work fine in raw mode.

The existing Tunnel Port can be closed/cancelled by issuing a new dataConnect command to “none”

Example:

```
Zyper$ dataConnect Decoder_1 none rs232
 Success
```

Example of connecting to a Tunnel Port and viewing RS232 data on that port.
In this example I have configured Tunnel Ports for two ZyPer endpoints. We can see this with the show dataTunnels command:

```
Zyper$ show dataTunnels
data-sessions(00:1c:d5:01:13:bd);
  device: name=UHD60-2EA
  rs232Tunnel: port=4100
  rs232Tunnel-connections: none
data-sessions(6c:df:fb:00:03:a0);
  device: name=XSE-Dec-Combo
  rs232Tunnel: port=4101
  rs232Tunnel-connections: none
Success
```

Now I open a connection to view the tunnel port. In this example I use telnet and will connect to tunnel port 4101.

```
artweeks$ telnet 192.168.0.21 4101
Trying 192.168.0.21...
Connected to 192.168.0.21.
Escape character is '^]'.
```

An update of the show dataTunnels command will show that a connection has been made and that that port is now being watched.

```
Zyper$ show dataTunnels
data-sessions(00:1c:d5:01:13:bd);
  device: name=UHD60-2EA
  rs232Tunnel: port=4100
  rs232Tunnel-connections: none
data-sessions(6c:df:fb:00:03:a0);
  device: name=XSE-Dec-Combo
  rs232Tunnel: port=4101
  rs232Tunnel-connections: remoteIp=192.168.0.99
```

So now that port is being watched. Let's send something to that port to see. I have a serial RS232 cable connected between the two ZyPer endpoints. I'll send RS232 data from the server out one ZyPer endpoint and into the other ZyPer endpoint.

Here is the command to send data out of the first ZyPer endpoint:

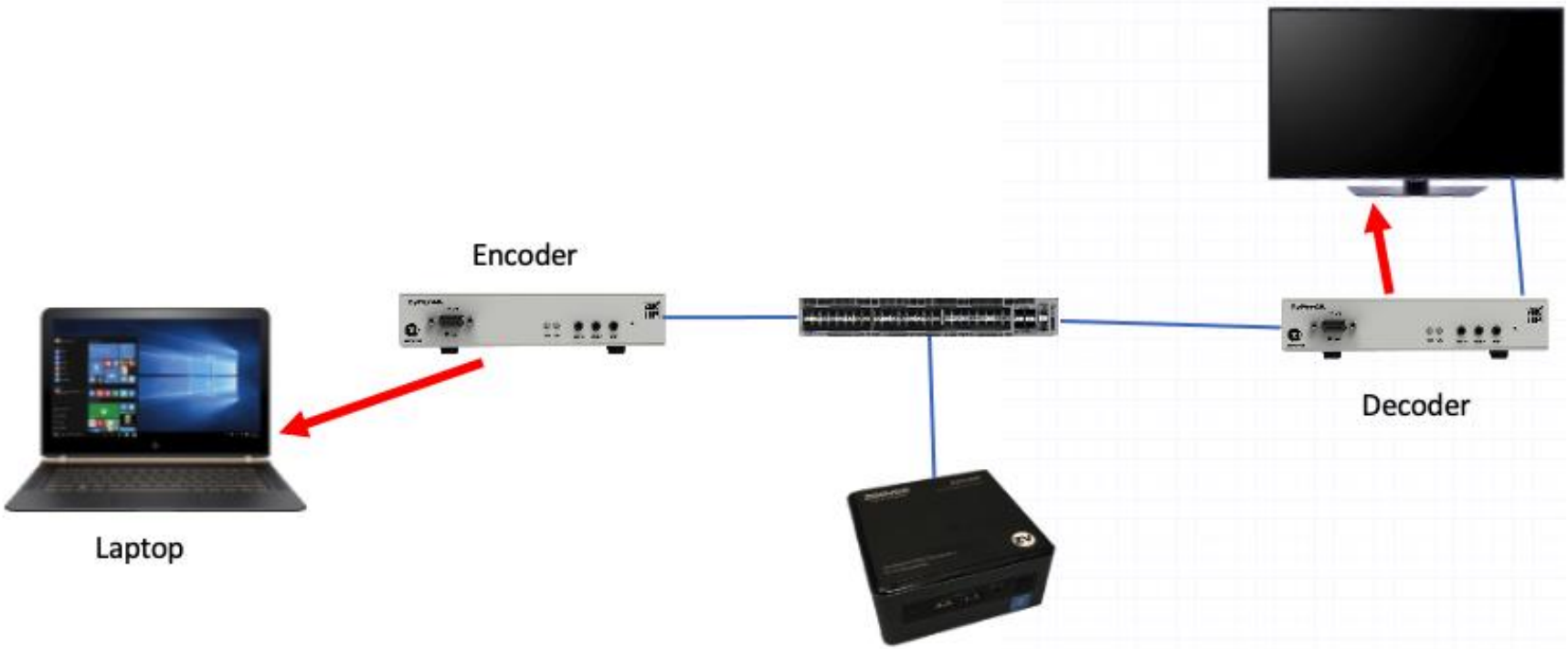
```
Zyper$ send UHD60-2EA rs232 "RS232 DataTunnel Demo\r\n"
Success
```

Then here is what I see on the window watching the tunnel port 4101. (Highlighted in red is the change)

```
artweeks$ telnet 192.168.0.21 4101
Trying 192.168.0.21...
Connected to 192.168.0.21.
Escape character is '^]'.
RS232 DataTunnel Demo
```

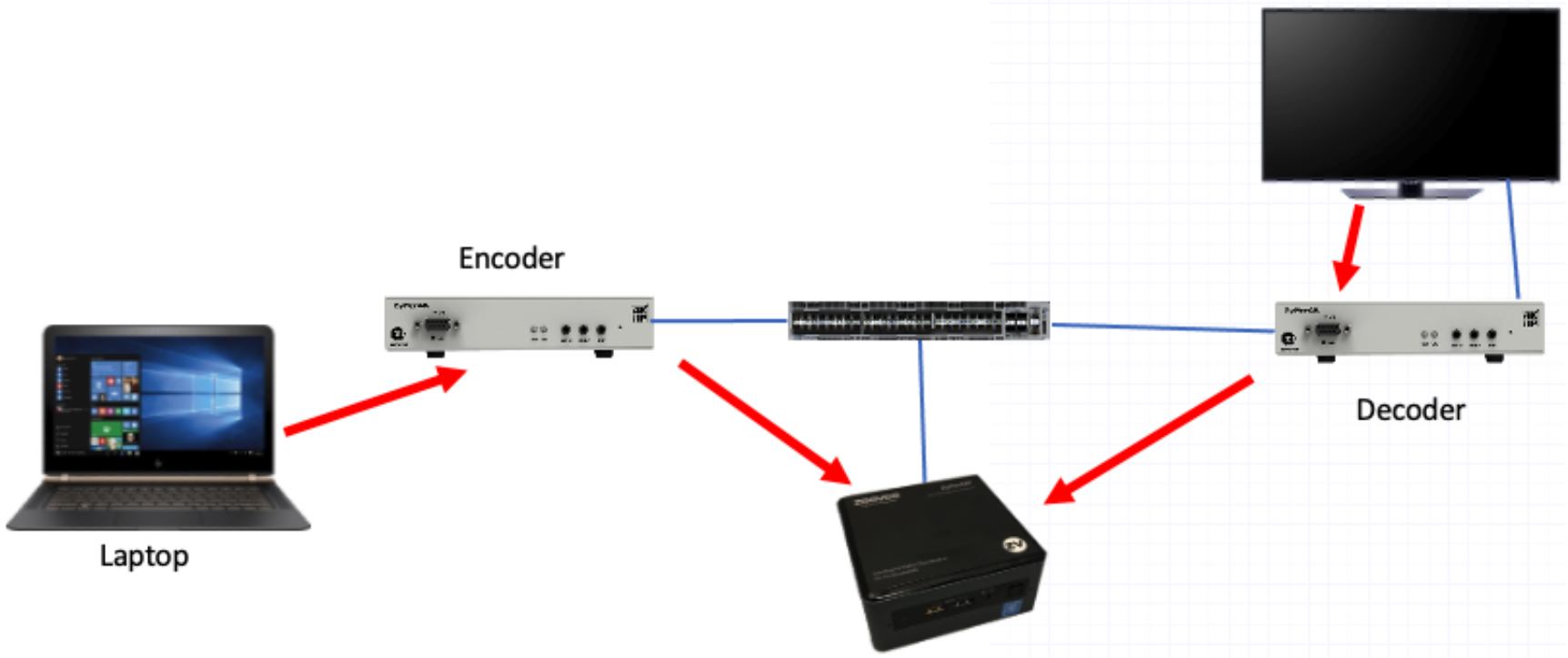
RS-232 Control via ZyPer Management Platform

- The ZyPer Management Platform (ZMP) can be used to manually send RS-232 commands to any endpoint. This allows someone in central location to control devices anyplace on the network.
- Command: send <deviceMac|deviceName> rs232 <text:string>



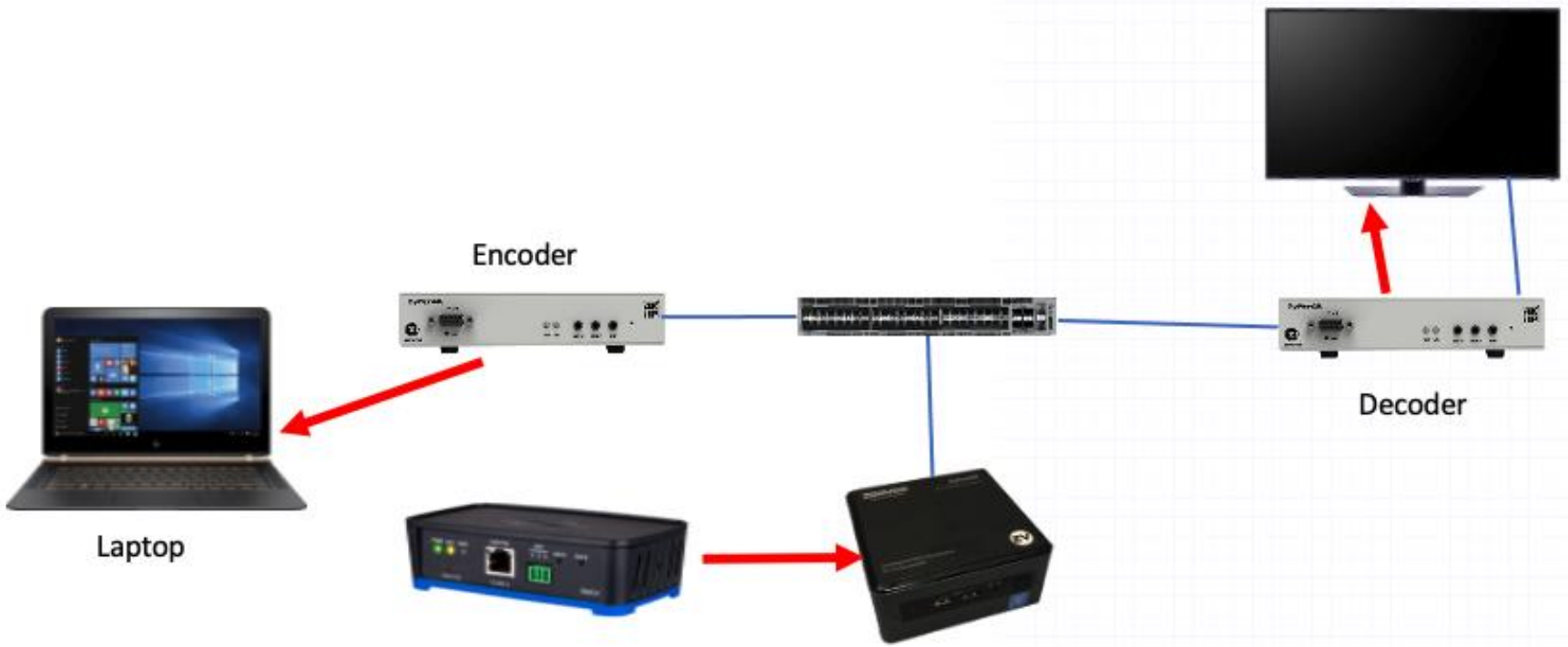
RS-232 Control via ZyPer Management Platform

- The ZMP can receive RS-232 from any endpoint. These text strings are stored on the server and can be retrieved any time.
- Command: show responses <deviceMac|deviceName> rs232 last
- Command: show responses <deviceMac|deviceName> rs232 since 0



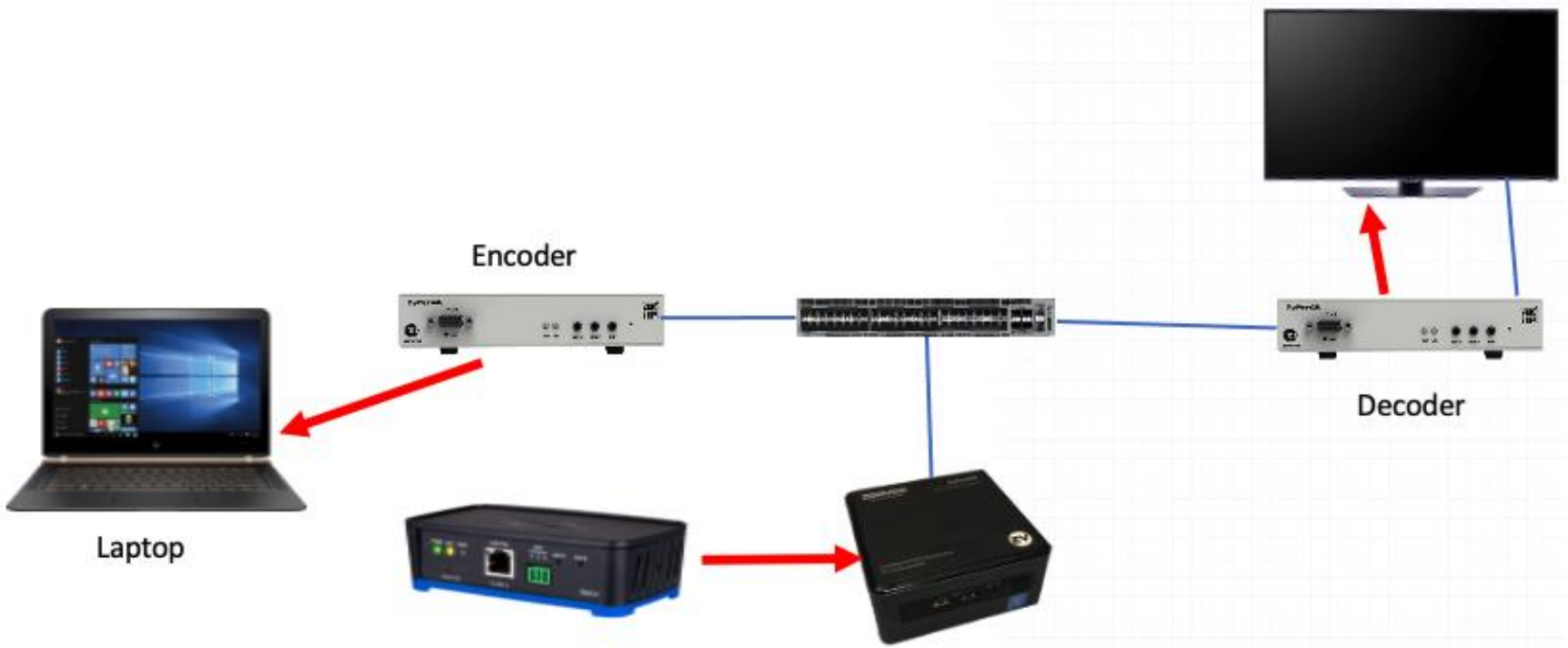
RS-232 Control via 3rd Party Systems (Crestron etc..)

- 3rd Party control systems such as Crestron Series 4 can send RS-232 strings to any ZyPer endpoint via the ZMP. Basically, by sending the exact needed API command to the ZyPer Management Platform.
- 3rd Party control such as Creston connect to the ZMP over telnet
- Command: send <deviceMac|deviceName> rs232 <text:string>



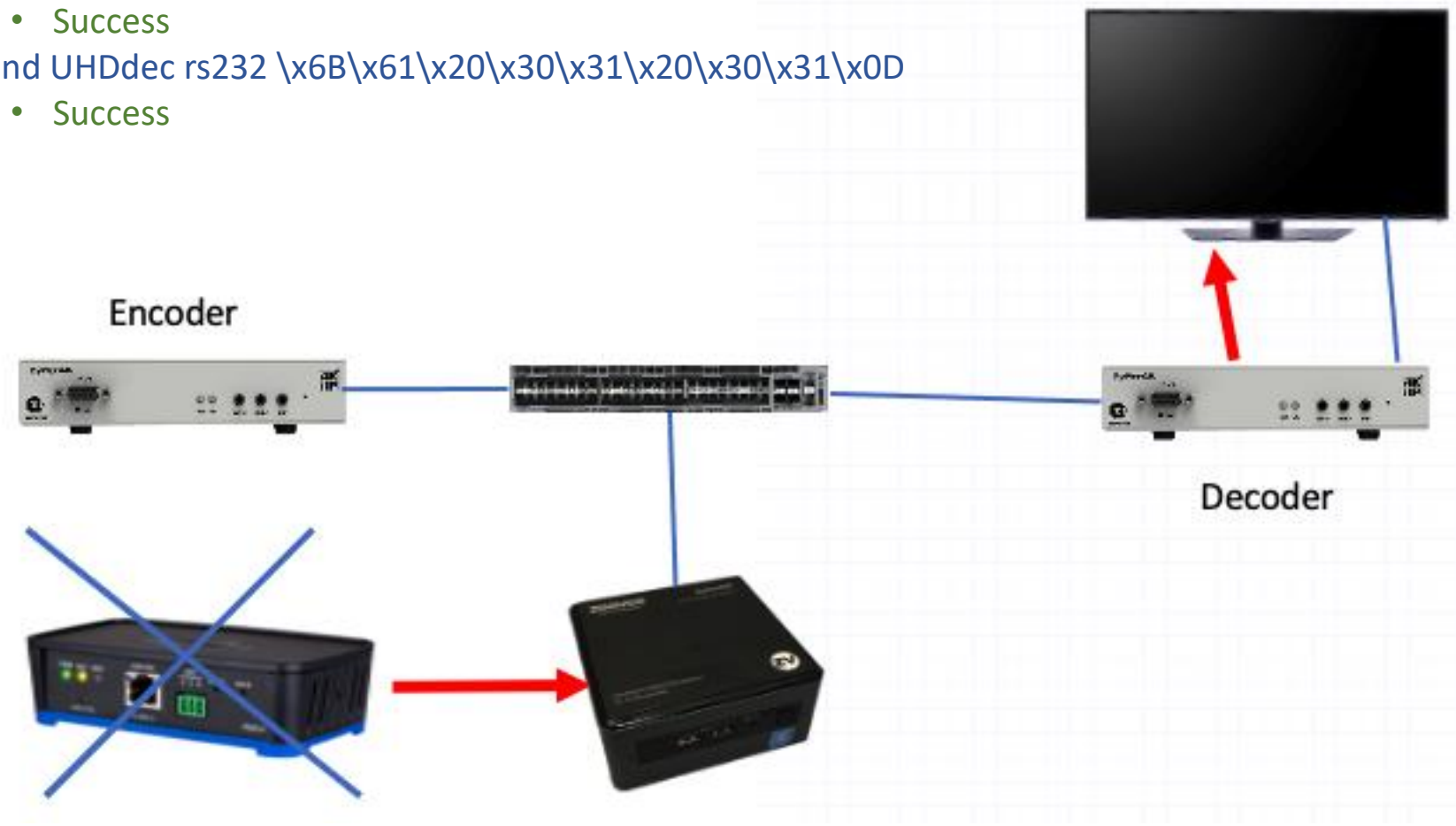
Troubleshooting RS-232 Control via 3rd Party Systems

- Sometimes 3rd party control systems have issues sending the correct RS-232 commands to devices such as displays when going thru the ZMP.
- The following pages detail a procedure to troubleshoot such issues.



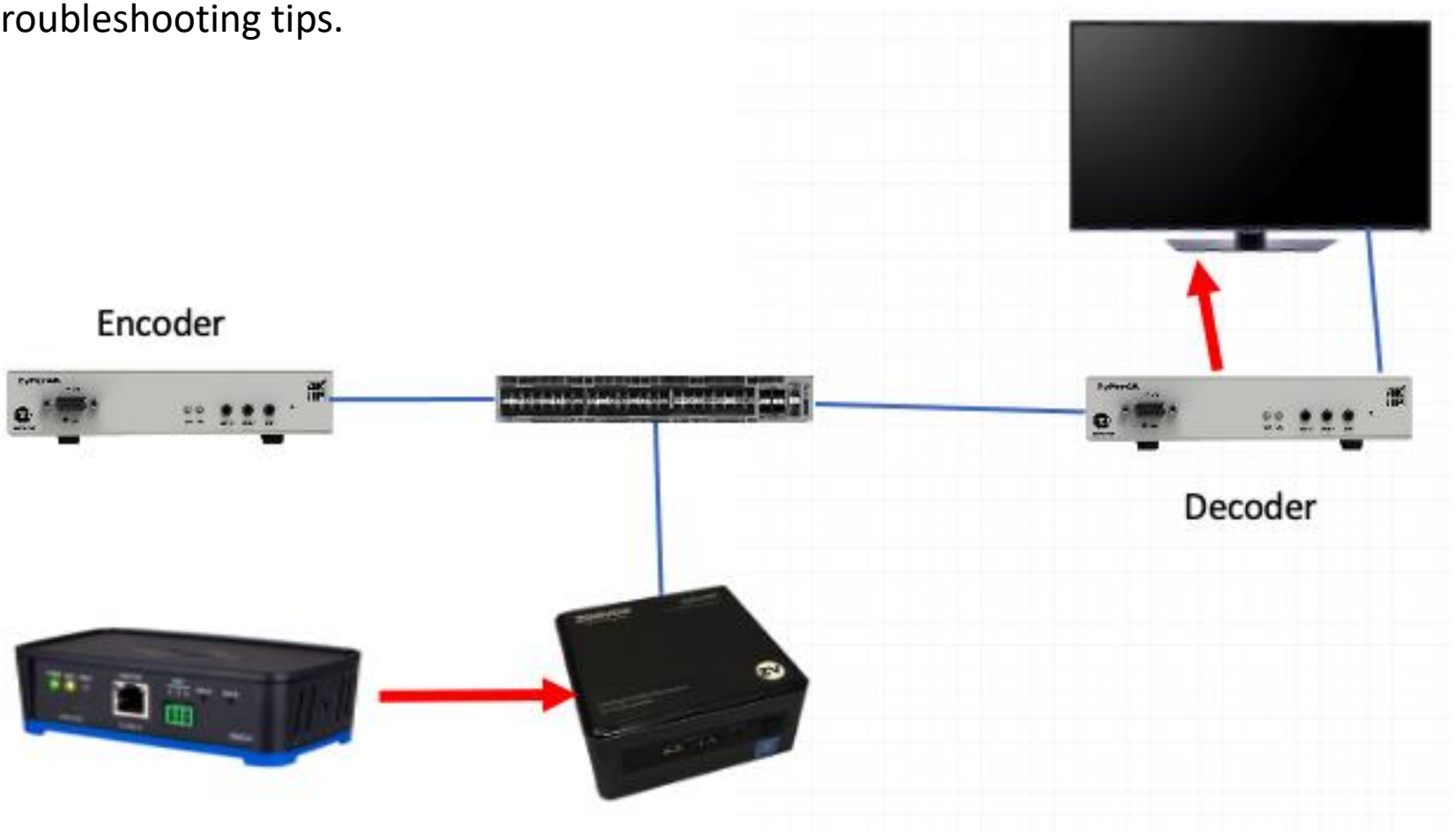
Troubleshooting RS-232 Control via 3rd Party Systems

- First, ensure that the desired command can be transmitted to the display or other device when the 3rd party control system is removed from the equation.
- Telnet directly into the ZMP and issue the needed commands.
- Example to turn LG display on: (Blue are API commands, Green is response from API)
- dataConnect UHDdec server rs232
  - Dynamically assigned tunnel TCP port = 4101; telnet handshake mode
  - Success
- send UHDdec rs232 \x6B\x61\x20\x30\x31\x20\x30\x31\x0D
  - Success



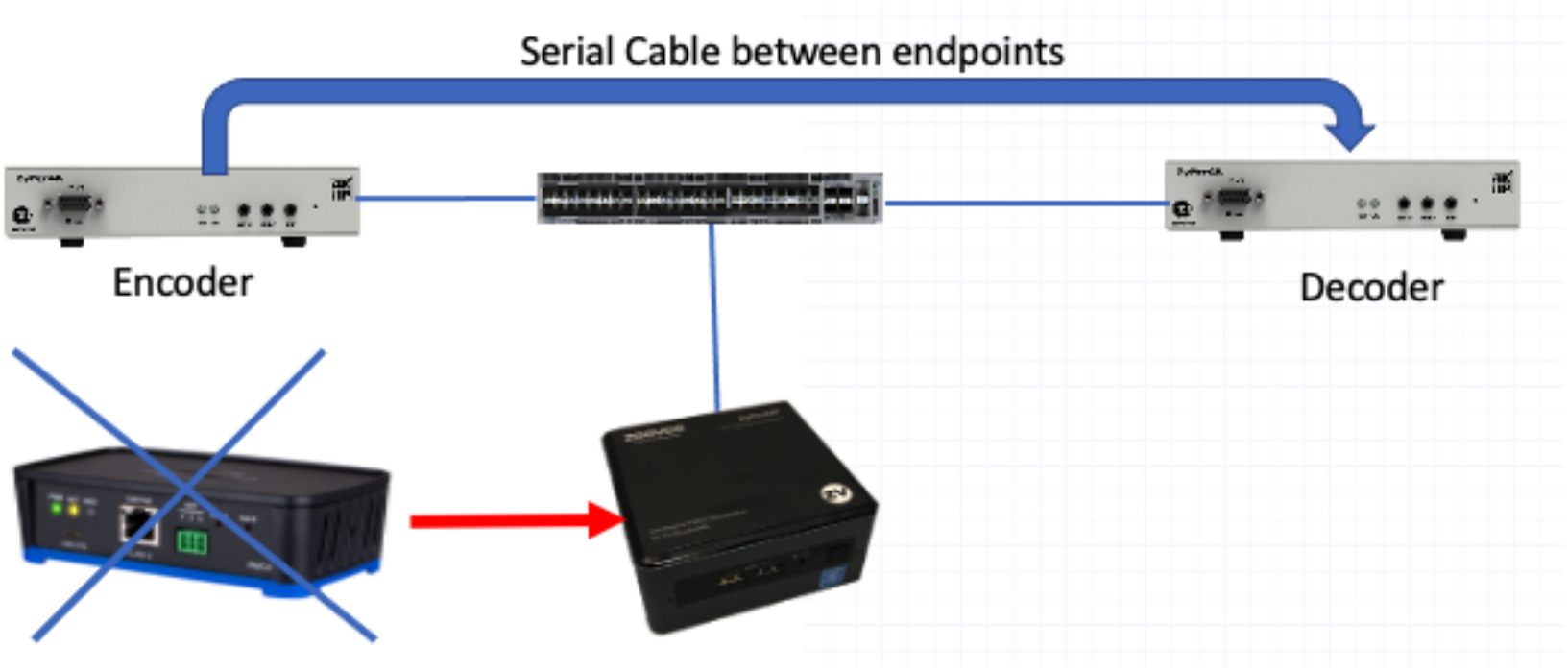
Troubleshooting RS-232 Control via 3rd Party Systems

- Assuming the command works directly from the API, copy/paste the exact syntax into the 3rd party control system for transmission to the ZMP. In most cases this will resolve the issue.
- If the issue is not resolved, then something is getting changed, added, corrupted between the 3rd party control system and the ZMP. See following slides for troubleshooting tips.



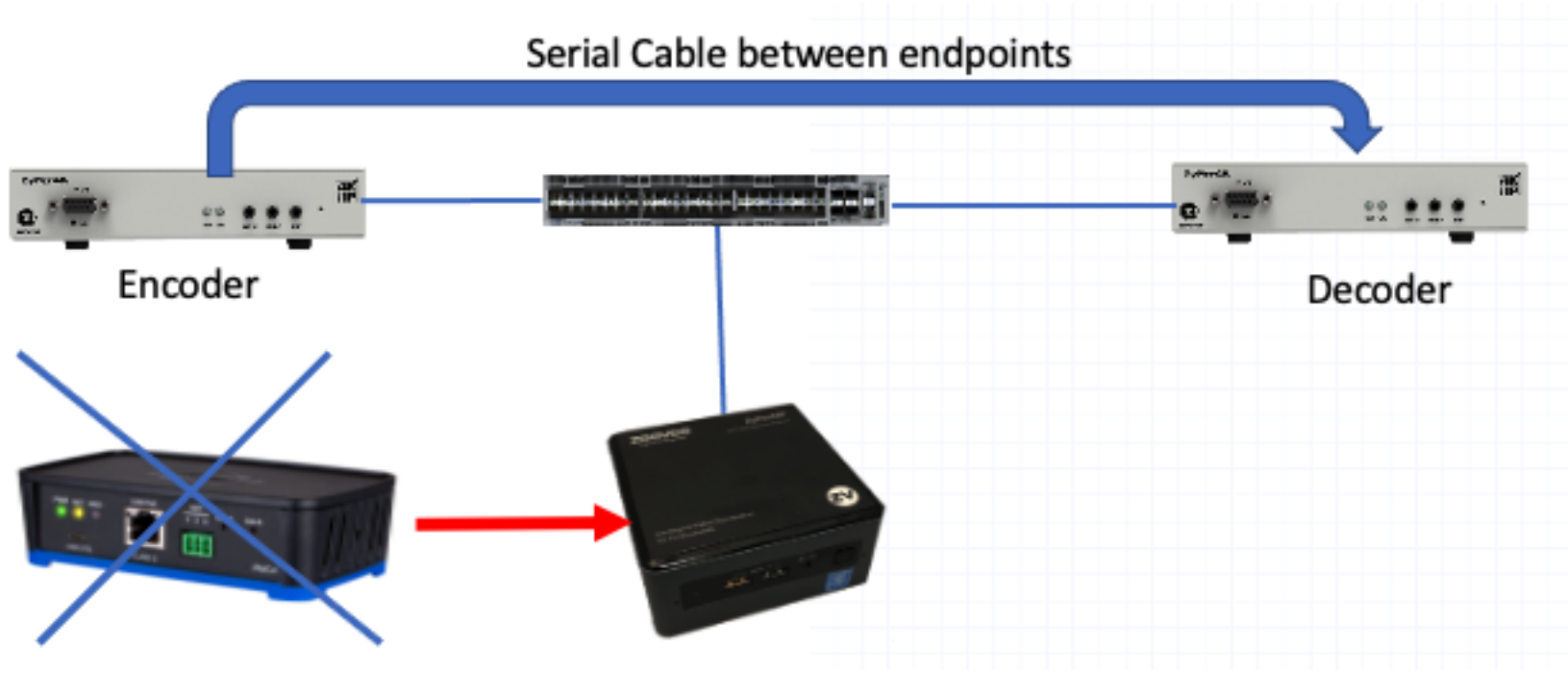
Troubleshooting RS-232 Control via 3rd Party Systems

- Directly connect the RS-232 ports using serial cable between 2 ZyPer endpoints. (Encoder and Decoder in this example.) Can be any 2 endpoints.
- Ensure both endpoints configured for same baud rate, parity etc..
- Ensure both endpoints dataConnected for RS232 to the ZMP Server
- dataConnect UHDdec server rs232
- dataConnect UHDenc server rs232



Troubleshooting RS-232 Control via 3rd Party Systems

- From the API, send the desired command to one of the endpoints. Example the decoder.
- send UHDdec rs232 \x6B\x61\x20\x30\x31\x20\x30\x31\x0D
  - Success
- Now use the show responses command to see exactly what was received on the RS-232 line at the encoder.
- show responses UHDEnc rs232 since 0
  - device(34:1b:22:81:2b:53);
  - device.rs232Response.0; string="**ka 01 01\x0D**"
  - lastChangeId(1);
  - Success
- In this example you can see that “ka 01 01\x0D” was transmitted to the decoder thus received at the encoder.



Troubleshooting RS-232 Control via 3rd Party Systems

- Now try sending the same exact command to the decoder via the 3rd party control system.
- Then via the API use the show responses command again to see what exactly was received at the encoder.
- **Here is where you look for discrepancies. Missing characters, added characters, missing or added carriage return or line feed commands.**
    - Note: We have seen cases where Crestron needs to add an extra \ to work.
        - Example: send UHDdec rs232 \x6B\x61\x20\x30\x31\x20\x30\x31\\x0D
- Adjust as needed in 3rd party control system until commands match as expected.
- Reconnect decoder to target device (display) and test.