

DIRECTOR

COMMAND GUIDE

RAIN





DIRECTOR RAIN

COMMAND GUIDE V1.3.6

DIRECTOR





DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents

1 Introduction	15
1.1 Licence Requirements	15
1.2 Telnet Connection and API Commands	15
1.3 HTTP Requests	15
2 Command Overview	16
3 System Commands	17
3.1 Command reboot	17
3.1.1 Command usage	17
3.1.2 Description	17
3.1.3 Arguments	17
3.1.4 Notes	17
3.1.5 Return Value	17
3.1.6 Command Examples	17
3.1.7 Return Examples	17
4 Command join	18
4.1 Command join video	19
4.1.1 Command usage	19
4.1.2 Description	19
4.1.3 Arguments	19
4.1.4 Notes	19
4.1.5 Return Value	19
4.1.6 Command Examples	20
4.1.7 Return Examples	20
4.2 Command join audio	21
4.2.1 Command usage	21
4.2.2 Description	21
4.2.3 Arguments	21
4.2.4 Notes	21
4.2.5 Return Value	21
4.2.6 Command Examples	21
4.2.7 Return Examples	21
4.3 Command join serial	22
4.3.1 Command usage	22
4.3.2 Description	22
4.3.3 Arguments	22
4.3.4 Notes	22
4.3.5 Return Value	22
4.3.6 Command Examples	22
4.3.7 Return Examples	22
4.4 Command join all	23
4.4.1 Command usage	23
4.4.2 Description	23
4.4.3 Arguments	23
4.4.4 Notes	23
4.4.5 Return Value	23
4.4.6 Command Examples	24
4.4.7 Return Examples	24



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

4.5 Command join av	25
4.5.1 Command usage	25
4.5.2 Description	25
4.5.3 Arguments	25
4.5.4 Notes	25
4.5.5 Return Value	25
4.5.6 Command Examples	26
4.5.7 Return Examples	26
4.6 Command join wall	27
4.6.1 Command usage	27
4.6.2 Description	27
4.6.3 Arguments	27
4.6.4 Notes	27
4.6.5 Return Value	27
4.6.6 Command Examples	27
4.6.7 Return Examples	27
4.7 Command join ir	28
4.7.1 Command usage	28
4.7.2 Description	28
4.7.3 Arguments	28
4.7.4 Notes	28
4.7.5 Return Value	28
4.7.6 Command Examples	28
4.7.7 Return Examples	28



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

5 Command leave	28
5.1 Command leave video	28
5.1.1 Command usage	28
5.1.2 Description	28
5.1.3 Arguments	28
5.1.4 Notes	28
5.1.5 Return Value	28
5.1.6 Command Examples	28
5.1.7 Return Examples	28
5.2 Command leave audio	30
5.2.1 Command usage	30
5.2.2 Description	30
5.2.3 Arguments	30
5.2.4 Notes	30
5.2.5 Return Value	30
5.2.6 Command Examples	30
5.2.7 Return Examples	30
5.3 Command leave serial	31
5.3.1 Command usage	31
5.3.2 Description	31
5.3.3 Arguments	31
5.3.4 Notes	31
5.3.5 Return Value	31
5.3.6 Command Examples	31
5.3.7 Return Examples	31
5.4 Command leave all	32
5.4.1 Command usage	32
5.4.2 Description	32
5.4.3 Arguments	32
5.4.4 Notes	32
5.4.5 Return Value	32
5.4.6 Command Example	32
5.4.7 Return Examples	32
5.5 Command leave av	33
5.5.1 Command usage	33
5.5.2 Description	33
5.5.3 Arguments	33
5.5.4 Notes	33
5.5.5 Return Value	33
5.5.6 Command Example	33
5.5.7 Return Examples	33
5.6 Command leave ir	34
5.6.1 Command usage	34
5.6.2 Description	34
5.6.3 Arguments	34
5.6.4 Notes	34
5.6.5 Return Value	34
5.6.6 Command Example	34
5.6.7 Return Examples	34



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

6 Command stop	35
6.1 Command stop	35
6.1.1 Command usage	35
6.1.2 Description	35
6.1.3 Arguments	35
6.1.4 Notes	35
6.1.5 Return Value	35
6.1.6 Command Example	35
6.1.7 Return Examples	35
 7 Command start	 36
7.1 Command start	36
7.1.1 Command usage	36
7.1.2 Description	36
7.1.3 Arguments	36
7.1.4 Notes	36
7.1.5 Return Value	36
7.1.6 Command Examples	36
7.1.7 Return Examples	36
 8 Command set	 37
8.1 Command set audio_a_mute	37
8.1.1 Command usage	37
8.1.2 Description	37
8.1.3 Arguments	37
8.1.4 Notes	37
8.1.5 Return Value	37
8.1.6 Command Examples	37
8.1.7 Return Examples	37
8.2 Command set audio_d_mute	38
8.2.1 Command usage	38
8.2.2 Description	38
8.2.3 Arguments	38
8.2.4 Notes	38
8.2.5 Return Value	38
8.2.6 Command Examples	38
8.2.7 Return Examples	38
8.3 Command set edid	39
8.3.1 Command usage	39
8.3.2 Description	39
8.3.3 Arguments	39
8.3.4 Notes	39
8.3.5 Return Value	39
8.3.6 Command Examples	39
8.3.7 Return Examples	39



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

8.4 Command set listener (Licensed feature)	40
8.4.1 Command usage	40
8.4.2 Description	40
8.4.3 Arguments	40
8.4.4 Notes	40
8.4.5 Return Value	40
8.4.6 Command Examples	40
8.4.7 Return Examples	40
8.4.8 GC-100 Series IR/Sensor Connector Wiring	40
8.4.9 iTach and GlobalConnect Series IR/Sensor Connector Wiring	41
8.4.10 Flex Link Relay & Sensor Cable Wiring	41
8.4.11 Technical Notes	42
8.5 Command set overlay	43
8.5.1 Command usage	43
8.5.2 Description	43
8.5.3 Arguments	43
8.5.4 Notes	43
8.5.5 Return Value	43
8.5.6 Command Examples	43
8.5.7 Return Examples	43
8.6 Command set rotation	44
8.6.1 Command usage	44
8.6.2 Description	44
8.6.3 Arguments	44
8.6.4 Notes	44
8.6.5 Return Value	44
8.6.6 Command Examples	44
8.6.7 Return Examples	44
8.7 Command set scaler	45
8.7.1 Command usage	45
8.7.2 Description	45
8.7.3 Arguments	45
8.7.4 Notes	45
8.7.5 Return Value	45
8.7.6 Command Examples	45
8.7.7 Return Examples	45
8.8 Command set var	46
8.8.1 Command usage	46
8.8.2 Description	46
8.8.3 Arguments	46
8.8.4 Notes	46
8.8.5 Return Value	46
8.8.6 Command Examples	46
8.8.7 Return Examples	46
8.9 Command set video_mode	47
8.9.1 Command usage	47
8.9.2 Description	47
8.9.3 Arguments	47
8.9.4 Notes	47
8.9.5 Return Value	47
8.9.6 Command Examples	47
8.9.7 Return Examples	47
8.10 Command set volume	48
8.10.1 Command usage	48
8.10.2 Description	48
8.10.3 Arguments	48
8.10.4 Notes	48
8.10.5 Return Value	48
8.10.6 Command Examples	48
8.10.7 Return Examples	48



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

8.11 Command set ui_button (Licensed feature)	49
8.11.1 Command usage	49
8.11.2 Description	49
8.11.3 Arguments	49
8.11.4 Notes	49
8.11.5 Return Value	49
8.11.6 Command Examples	49
8.11.7 Return Examples	50
8.12 Command set ui_label (Licensed feature)	51
8.12.1 Command usage	51
8.12.2 Description	51
8.12.3 Arguments	51
8.12.4 Notes	51
8.12.5 Return Value	51
8.12.6 Command Example	51
8.12.7 Return Example	51
8.13 Command set ui_image (Licensed feature)	52
8.13.1 Command usage	52
8.13.2 Description	52
8.13.3 Arguments	52
8.13.4 Notes	52
8.13.5 Return Value	52
8.13.6 Command Example	52
8.13.7 Return Example	52
8.14 Command set ui_page (Licensed feature)	53
8.14.1 Command usage	53
8.14.2 Description	53
8.14.3 Arguments	53
8.14.4 Notes	53
8.14.5 Return Value	53
8.14.6 Command Example	53
8.14.7 Return Example	53
8.15 Command set ui (Licensed feature)	54
8.15.1 Command usage	54
8.15.2 Description	54
8.15.3 Arguments	54
8.15.4 Notes	54
8.15.5 Return Value	54
8.15.6 Command Example	54
8.15.7 Return Example	54
8.16 Command set sync	55
8.16.1 Command usage	55
8.16.2 Description	55
8.16.3 Arguments	55
8.16.4 Notes	55
8.16.5 Return Value	55
8.16.6 Command Example	55
8.16.7 Return Example	55
8.17 Command set events	56
8.17.1 Command usage	56
8.17.2 Description	56
8.17.3 Arguments	56
8.17.4 Notes	56
8.17.5 Return Value	56
8.17.6 Command Example	56
8.17.7 Return Example	56



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

9 Command get	57
9.1 Command get audio_a_mute	57
9.1.1 Command usage	57
9.1.2 Description	57
9.1.3 Arguments	57
9.1.4 Notes	57
9.1.5 Return Value	57
9.1.6 Command Examples	57
9.1.7 Return Examples	57
9.2 Command get audio_d_mute	58
9.2.1 Command usage	58
9.2.2 Description	58
9.2.3 Arguments	58
9.2.4 Notes	58
9.2.5 Return Value	58
9.2.6 Command Examples	58
9.2.7 Return Examples	58
9.3 Command get devices	59
9.3.1 Command usage	59
9.3.2 Description	59
9.3.3 Arguments	59
9.3.4 Notes	59
9.3.5 Return Value	59
9.3.6 Command Examples	59
9.3.7 Return Examples	59
9.4 Command get display_status	60
9.4.1 Command usage	60
9.4.2 Description	60
9.4.3 Arguments	60
9.4.4 Notes	60
9.4.5 Return Value	60
9.4.6 Command Examples	60
9.4.7 Return Examples	60
9.5 Command get edid	61
9.5.1 Command usage	61
9.5.2 Description	61
9.5.3 Arguments	61
9.5.4 Notes	61
9.5.5 Return Value	61
9.5.6 Command Examples	61
9.5.7 Return Examples	61
9.6 Command get joins	62
9.6.1 Command usage	62
9.6.2 Description	62
9.6.3 Arguments	62
9.6.4 Notes	62
9.6.5 Return Value	62
9.6.6 Command Examples	62
9.6.7 Return Examples	62
9.7 Command get overlay	63
9.7.1 Command usage	63
9.7.2 Description	63
9.7.3 Arguments	63
9.7.4 Notes	63
9.7.5 Return Value	63
9.7.6 Command Examples	63
9.7.7 Return Examples	63



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

9.8 Command get preferred	64
9.8.1 Command usage	64
9.8.2 Description	64
9.8.3 Arguments	64
9.8.4 Notes	64
9.8.5 Return Value	64
9.8.6 Command Examples	64
9.8.7 Return Examples	64
9.9 Command get rotation	65
9.9.1 Command usage	65
9.9.2 Description	65
9.9.3 Arguments	65
9.9.4 Notes	65
9.9.5 Return Value	65
9.9.6 Command Examples	65
9.9.7 Return Examples	65
9.10 Command get scaler	66
9.10.1 Command usage	66
9.10.2 Description	66
9.10.3 Arguments	66
9.10.4 Notes	66
9.10.5 Return Value	66
9.10.6 Command Examples	66
9.10.7 Return Examples	66
9.11 Command get status	67
9.11.1 Command usage	67
9.11.2 Description	67
9.11.3 Arguments	67
9.11.4 Notes	67
9.11.5 Return Value	67
9.11.6 Command Examples	67
9.11.7 Return Examples	67
9.12 Command get var	68
9.12.1 Command usage	68
9.12.2 Description	68
9.12.3 Arguments	68
9.12.4 Notes	68
9.12.5 Return Value	68
9.12.6 Command Examples	68
9.12.7 Return Examples	68
9.13 Command get ver	69
9.13.1 Command usage	69
9.13.2 Description	69
9.13.3 Arguments	69
9.13.4 Notes	69
9.13.5 Return Value	69
9.13.6 Command Examples	69
9.13.7 Return Examples	69
9.14 Command get video	70
9.14.1 Command usage	70
9.14.2 Description	70
9.14.3 Arguments	70
9.14.4 Notes	70
9.14.5 Return Value	70
9.14.6 Command Examples	70
9.14.7 Return Examples	70



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

9.15 Command get video_mode	71
9.15.1 Command usage	71
9.15.2 Description	71
9.15.3 Arguments	71
9.15.4 Notes	71
9.15.5 Return Value	71
9.15.6 Command Examples	71
9.15.7 Return Examples	71
9.16 Command get video_status	72
9.16.1 Command usage	72
9.16.2 Description	72
9.16.3 Arguments	72
9.16.4 Notes	72
9.16.5 Return Value	72
9.16.6 Command Examples	72
9.16.7 Return Examples	72
9.17 Command get volume	73
9.17.1 Command usage	73
9.17.2 Description	73
9.17.3 Arguments	73
9.17.4 Notes	73
9.17.5 Return Value	73
9.17.6 Command Examples	73
9.17.7 Return Examples	73
9.18 Command get ui_button (Licensed feature)	74
9.18.1 Command usage	74
9.18.2 Description	74
9.18.3 Arguments	74
9.18.4 Notes	74
9.18.5 Return Value	74
9.18.6 Command Examples	74
9.19 Command get ui (Licensed feature)	75
9.19.1 Command usage	75
9.19.2 Description	75
9.19.3 Arguments	75
9.19.4 Notes	75
9.19.5 Return Value	75
9.19.6 Command Examples	75
9.19.7 Return Examples	75
9.20 Command get events	76
9.20.1 Command usage	76
9.20.2 Description	76
9.20.3 Arguments	76
9.20.4 Notes	76
9.20.5 Return Value	76
9.20.6 Command Examples	76
9.20.7 Return Examples	76



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

10 Command send	77
10.1 Command send cec	77
10.1.1 Command usage	77
10.1.2 Description	77
10.1.3 Arguments	77
10.1.4 Notes	77
10.1.5 Return Value	77
10.1.6 Command Examples	77
10.1.7 Return Examples	77
10.2 Command send gc (Licensed feature)	78
10.2.1 Command usage	78
10.2.2 Description	78
10.2.3 Arguments	78
10.2.4 Notes	78
10.2.5 Return Value	78
10.2.6 Command Examples	78
10.2.7 Return Examples	78
10.3 Command send serial	79
10.3.1 Command usage	79
10.3.2 Description	79
10.3.3 Arguments	79
10.3.4 Notes	79
10.3.5 Return Value	80
10.3.6 Command Examples	80
10.3.7 Return Examples	80
10.4 Command send tcp	81
10.4.1 Command usage	81
10.4.2 Description	81
10.4.3 Arguments	81
10.4.4 Notes	81
10.4.5 Return Value	81
10.4.6 Command Examples	81
10.4.7 Return Examples	81
10.5 Command send ir	82
10.5.1 Command usage	82
10.5.2 Description	82
10.5.3 Arguments	82
10.5.4 Notes	82
10.5.5 Return Value	82
10.5.6 Command Examples	82
10.5.7 Return Examples	82



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

11 Command preset	83
11.1 Command preset add	83
11.1.1 Command usage	83
11.1.2 Description	83
11.1.3 Arguments	83
11.1.4 Notes	83
11.1.5 Return Value	83
11.1.6 Command Examples	83
11.1.7 Return Examples	83
11.2 Command preset delay	84
11.2.1 Command usage	84
11.2.2 Description	84
11.2.3 Arguments	84
11.2.4 Notes	84
11.2.5 Return Value	84
11.2.6 Command Examples	84
11.2.7 Return Examples	84
11.3 Command preset delete	85
11.3.1 Command usage	85
11.3.2 Description	85
11.3.3 Arguments	85
11.3.4 Notes	85
11.3.5 Return Value	85
11.3.6 Command Examples	85
11.3.7 Return Examples	85
11.4 Command preset load	86
11.4.1 Command usage	86
11.4.2 Description	86
11.4.3 Arguments	86
11.4.4 Notes	86
11.4.5 Return Value	86
11.4.6 Command Examples	86
11.4.7 Return Examples	86



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Table of Contents continued...

12 Message notify	87
12.1 Message notify serial	87
12.1.1 Message received	87
12.1.2 Description	87
12.1.3 Arguments	87
12.1.4 Notes	87
12.1.5 Return Value	87
12.1.6 Examples	87
12.2 Message notify network	88
12.2.1 Message received	88
12.2.2 Description	88
12.2.3 Arguments	88
12.2.4 Notes	88
12.2.5 Return Value	88
12.2.6 Examples	88
12.3 Message notify display	89
12.3.1 Message received	89
12.3.2 Description	89
12.3.3 Arguments	89
12.3.4 Notes	89
12.3.5 Return Value	89
12.3.6 Examples	89
12.4 Message notify source	90
12.4.1 Message received	90
12.4.2 Description	90
12.4.3 Arguments	90
12.4.4 Notes	90
12.4.5 Return Value	90
12.4.6 Examples	90
 Appendix A - How to HTTP request	 91
 Appendix B – Preset Logic	 94



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

1 Introduction

This document describes everything that a developer needs to be aware of to use the DIRECTOR H264 command guide and develop client control applications for H264 devices.

1.1 Licence Requirements

The Director Controller must have a valid licence key entered before use or trying to connect to the Telnet TCP control port 6980.

If no valid licence is active the Director Controller will return 'Invalid License' and terminate the TCP connection. Contact iMAGsystems or your local distributor for licencing information.

1.2 Telnet Connection

Third party controllers connect to the Director Controller and issue commands using ascii strings terminated with a carriage return <cr> 0x0D. This allows any Telnet client to be used with the system.

The Director Controller listens on TCP port 6980. Once a successful TCP connection is established you will receive a welcome message 'Connection Successful'.

A constant TCP connection to the Director Controller is recommended to maintain status changes of the system from notification events.

An optional security key can be used with all TCP API commands made to port 6980.

The keyword 'key:' along with the security key are added to the API command before any parameters.

1.3 HTTP requests

It is also possible to control the system with HTTP GET and POST requests.

A security key must be sent with any request. Security keys are generated by 'admin' level UI users on the Global Settings / Security Key tab.

GET = `http://<controllerURL>/api/command/<DIRECTOR_API_COMMAND>/<KEY>`

POST = `http://<controllerURL>/api/command/{'cmd': '<DIRECTOR_API_COMMAND>', 'key': '<KEY>'}`

Refer Appendix A - How to HTTP request

2 Command Overview

Commands are in a simple ascii text format. For each command, the Director Controller responds with a response which contains the return status (i.e. whether the command succeeded or not) and, if successful, the return value of the command if required. The API is to be used synchronous communication.

- All commands and returns are terminated with a carriage return <cr> 0x0D
- Commands are not case sensitive
- Invalid commands will return **error [Unknown]<cr>**
- Missing security key will return **error [security key missing]<cr>**
- Invalid security key will return **error [security key invalid]<cr>**

- join video encoder1 decoder1<cr>
- join video encoder1 all<cr>
- join video encoder1 MyGroup<cr>

- stop encoder1<cr>
- start encoder1<cr>

- send serial decoder1 "my data string\x0D"<cr>



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

3 System Commands

3.1 Command reboot

The **reboot** command is used to restart any or all Encoders and Decoders.

3.1.1 Command usage

```
reboot [key:<security_key>] <device_name><cr>
```

3.1.2 Description

Causes the target device to restart and become unavailable for a short period while restarting.

3.1.3 Arguments

<i>device_name</i>	Name of the Encoder, Decoder, Group or 'all' 'all_rx' 'all_tx'
--------------------	--------------------------------------------------------------------

3.1.4 Notes

- **all** is used as a destination when all devices are required to reboot.
- **all_rx** is used as a destination when all Decoders are required to reboot.
- **all_tx** is used as a destination when all Encoders are required to reboot.
- **group_name** is used as a destination when all Encoders and Decoders in a group are required to reboot.

3.1.5 Return Value

command	<space>	status	terminator
reboot	<space>	<i>success / error [message]</i>	<cr>

3.1.6 Command Examples

```
reboot Encoder1<cr>
reboot all<cr>
reboot all_rx<cr>
reboot all_tx<cr>
reboot MyGroup<cr>
reboot key:abc123 Encoder1<cr>
```

3.1.7 Return Examples

```
reboot success<cr>

reboot error [incomplete]<cr>
reboot error [device 'Encoder1' not found]<cr>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

4 Command join

The **join** commands are used for routing all the signals to their required destinations. Video, audio and serial can all be independently routed to their destinations.

join video command provides independent routing of the video.

join audio command provides independent routing of the audio.

join serial command provides independent routing of serial (RS-232).

join ir command provides independent routing of Infrared.

join all command provides combined routing of all signals.

join av command provides combined routing of the video and audio together.

join wall command is used to send a cropped portion of the video source to a display in a video wall configuration.

4.1 Command join video

4.1.1 Command usage

```
join video [key:<security_key>] <encoder_device_name> <decoder_device_name> [<exclusive>]
[<original> | <auto> | [size <mode>]]<cr>
```

4.1.2 Description

The command **join video** is used for independent routing of video signals.

4.1.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>decoder_device_name</i>	Device name of the <i>Decoder, Group</i> or ' all '
exclusive	Keyword ' exclusive ' allows for only the specified Decoder to be joined with the Encoder. All other Decoders joined with the Encoder will be removed. (optional)
original	Keyword ' original ' will set the Decoder resolution as pass-through to match the Encoder video resolution. (optional)
auto	Keyword ' auto ' is only available when the Decoder has a monitor connected with valid EDID. The Decoders resolution will be set to the connected displays preferred resolution. (optional)
size	Keyword ' size ' is followed by the video mode (optional)

4.1.4 Notes

- **all** can replace *decoder_device_name* when the video is required to be connected to all Decoders.
- Only use '**original**' or '**auto**' or '**size**'.
- The name of a group can replace *decoder_device_name* when the video is required to be connected to all Decoders in a group.
- '**auto**' = Decoder's connected displays preferred resolution
- '**original**' = Encoder source resolution
- Valid modes:
 - 'size 1080p60'
 - 'size 1080p50'
 - 'size 720p60'
 - 'size 720p50'

4.1.5 Return Value

command	<space>	mode	<space>	status	terminator
join	<space>	video	<space>	success / error [message]	<cr>



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

4.1.6 Command Examples

```
join video Encoder1 Decoder1<cr>  
join video Encoder1 all<cr>  
join video Encoder1 MyGroup<cr>  
join video Encoder1 Decoder1 exclusive<cr>  
join video Encoder1 MyGroup exclusive<cr>  
join video Encoder1 Decoder1 original<cr>  
join video Encoder1 Decoder1 auto<cr>  
join video Encoder1 Decoder1 size 1080p60<cr>  
join video key:abc123 Encoder1 Decoder1<cr>
```

4.1.7 Return Examples

```
join video success<cr>  
  
join video error [incomplete]<cr>  
join video error [join not permitted]<cr>  
join video error [invalid parameter]<cr>  
join video error [encoder 'Encoder1' not found]<cr>  
join video error [decoder 'Decoder1' not found]<cr>  
join video error [monitor not detected]<cr>
```

4.2 Command join audio

4.2.1 Command usage

join audio [key:<security_key>] <encoder_device_name> <decoder_device_name> [<exclusive>]<cr>

4.2.2 Description

The command **join audio** is used for independent routing of audio signals.

4.2.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>decoder_device_name</i>	Device name of the Decoder, Group or 'all'
exclusive	Keyword ' exclusive ' allows for only the specified Decoder to be joined with the Encoder. All other Decoders joined with the Encoder will be removed. (optional)

4.2.4 Notes

- **all** can replace *decoder_device_name* when the audio is required to be connected to all Decoders.
- The name of a group can replace *decoder_device_name* when the audio is required to be connected to all Decoders in a group.

4.2.5 Return Value

command	<space>	mode	<space>	status	terminator
join	<space>	audio	<space>	success / error [message]	<cr>

4.2.6 Command Examples

```
join audio Encoder1 Decoder1<cr>
join audio Encoder1 all<cr>
join audio Encoder1 MyGroup<cr>
join audio Encoder1 Decoder1 exclusive<cr>
join audio Encoder1 MyGroup exclusive<cr>
join audio key:abc123 Encoder1 Decoder1<cr>
```

4.2.7 Return Examples

```
join audio success<cr>

join audio error [incomplete]<cr>
join audio error [join not permitted]<cr>
join audio error [invalid parameter]<cr>
join audio error [encoder 'Encoder1' not found]<cr>
join audio error [decoder 'Decoder1' not found]<cr>
```

4.3 Command join serial

4.3.1 Command usage

join serial [key:<security_key>] <encoder_device_name> <decoder_device_name> [<exclusive>]<cr>

4.3.2 Description

The command **join serial** is used for independent routing of serial signals.

4.3.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>decoder_device_name</i>	Device name of the Decoder, Group or 'all'
exclusive	Keyword ' exclusive ' allows for only the specified Decoder to be joined with the Encoder. All other Decoders joined with the Encoder will be removed. (optional)

4.3.4 Notes

- **all** can replace *decoder_device_name* when the serial RS232 is required to be connected to all Decoders.
- The name of a group can replace *decoder_device_name* when the serial RS232 is required to be connected to all Decoders in a group.
- Device must be in Matrix mode to allow joins.

4.3.5 Return Value

command	<space>	mode	<space>	status	terminator
join	<space>	serial	<space>	success / error [message]	<cr>

4.3.6 Command Examples

```
join serial Encoder1 Decoder1<cr>
join serial Encoder1 all<cr>
join serial Encoder1 MyGroup<cr>
join serial Encoder1 Decoder1 exclusive<cr>
join serial Encoder1 MyGroup exclusive<cr>
join serial key:abc123 Encoder1 Decoder1<cr>
```

4.3.7 Return Examples

```
join serial success<cr>

join serial error [incomplete]<cr>
join serial error [join not permitted]<cr>
join serial error [invalid parameter]<cr>
join serial error [encoder 'Encoder1' not found]<cr>
join serial error [decoder 'Decoder1' not found]<cr>
```

4.4 Command join all

4.4.1 Command usage

```
join all [key:<security_key>] <encoder_device_name> <decoder_device_name> [<exclusive>]
[<original> | <auto> | [size <mode>]]<cr>
```

4.4.2 Description

The command **join all** is used for combined routing of all signals.

4.4.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>decoder_device_name</i>	Device name of the Decoder, Group or 'all'
exclusive	Keyword 'exclusive' allows for only the specified Decoder to be joined with the Encoder. All other Decoders joined with the Encoder will be removed. (optional)
original	Keyword 'original' will set the Decoder resolution as pass-through to match the Encoder video resolution. (optional)
auto	Keyword 'auto' is only available when the Decoder has a monitor connected with valid EDID. The Decoders resolution will be set to the connected display's preferred resolution. (optional)
size	Keyword 'size' is followed by the video mode (optional)

4.4.4 Notes

- **all** can replace *decoder_device_name* when all signal are required to be connected to all Decoders.
- Only use **'original'** or **'auto'** or **'mode'**.
- The name of a group can replace *decoder_device_name* when all signals are required to be connected to all Decoders in a group.
- **'auto'** = Decoder's connected displays preferred resolution
- **'original'** = Encoder source resolution
- Valid modes:
 - 'size 1080p60'
 - 'size 1080p50'
 - 'size 720p60'
 - 'size 720p50'

4.4.5 Return Value

command	<space>	mode	<space>	status	terminator
join	<space>	all	<space>	success / error [message]	<cr>



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

4.4.6 Examples

```
join all Encoder1 Decoder1<cr>
join all Encoder1 all<cr>
join all Encoder1 MyGroup<cr>
join all Encoder1 MyGroup exclusive<cr>
join all Encoder1 Decoder1 original<cr>
join all Encoder1 Decoder1 auto<cr>
join all Encoder1 Decoder1 size 1080p60<cr>
join all key:abc123 Encoder1 Decoder1<cr>
```

4.4.7 Return Examples

```
join all success<cr>

join all error [incomplete]<cr>
join all error [join not permitted]<cr>
join all error [invalid parameter]<cr>
join all error [encoder 'Encoder1' not found]<cr>
join all error [decoder 'Decoder1' not found]<cr>
join all error [monitor not detected]<cr>
```


4.5 Command join av

4.5.1 Command usage

```
join av [key:<security_key>] <encoder_device_name> <decoder_device_name> [<exclusive>]
[<original> | <auto> | [size <mode>]]<cr>
```

4.5.2 Description

The command **join av** is used for combined routing of audio and video signals.

4.5.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>decoder_device_name</i>	Device name of the Decoder, Group or 'all'
exclusive	Keyword ' exclusive ' allows for only the specified Decoder to be joined with the Encoder. All other Decoders joined with the Encoder will be removed. (optional)
original	Keyword ' original ' will set the Decoder resolution as pass-through to match the Encoder video resolution. (optional)
auto	Keyword ' auto ' is only available when the Decoder has a monitor connected with valid EDID. The Decoders resolution will be set to the connected display's preferred resolution. (optional)
size	Keyword ' size ' is followed by the video mode (optional)

4.5.4 Notes

- **all** can replace *decoder_device_name* when the audio and video are required to be connected to all Decoders.
- Only use '**original**' or '**auto**' or '**mode**'.
- The name of a group can replace *decoder_device_name* when the audio and video are required to be connected to all Decoders in a group.
- '**auto**' = Decoder's connected displays preferred resolution
- '**original**' = Encoder source resolution
- Valid modes:
 - 'size 1080p60'
 - 'size 1080p50'
 - 'size 720p60'
 - 'size 720p50'

4.5.5 Return Value

command	<space>	mode	<space>	status	terminator
join	<space>	av	<space>	success / error [message]	<cr>



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

4.5.6 Command Examples

```
join av Encoder1 Decoder1<cr>
join av Encoder1 all<cr>
join av Decoder1 MyGroup<cr>
join av Encoder1 MyGroup exclusive<cr>
join av Encoder1 Decoder1 original<cr>
join av Encoder1 Decoder1 auto<cr>
join av Encoder1 Decoder1 size 1080p60<cr>
join av key:abc123 Encoder1 Decoder1<cr>
```

4.5.7 Return Examples

```
join av success<cr>

join av error [incomplete]<cr>
join av error [join not permitted]<cr>
join av error [invalid parameter]<cr>
join av error [encoder 'Encoder1' not found]<cr>
join av error [decoder 'Decoder1' not found]<cr>
join av error [monitor not detected]<cr>
```

4.6 Command join wall

4.6.1 Command usage

```
join wall [key:<security_key>] <encoder_device_name> <decoder_device_name> <wall_type>
<display_position> [size <width> <height> <fps>] [bezel <display_width> <viewable_width> <display_height>
<viewable_height>]<cr>
```

4.6.2 Description

The command **join wall** is used to join an Encoder to a Decoder within a video wall layout.

4.6.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>decoder_device_name</i>	Device name of the Decoder
<i>wall_type</i>	Define the video wall configuration as columns x rows
<i>display_position</i>	Define the display position from the top left
<i>size (optional)</i>	Define the display resolution
<i>width</i>	Display resolution horizontal in px (used with optional size)
<i>height</i>	Display resolution vertical in px (used with optional size)
<i>fps</i>	Display frame rate (used with optional size)
<i>bezel (optional)</i>	Define the size of the display bezel
<i>display_width</i>	Define the overall width of the display in mm (used with optional bezel)
<i>viewable_width</i>	Define the viewable width of the display in mm (used with optional bezel)
<i>display_height</i>	Define the overall height of the display in mm (used with optional bezel)
<i>viewable_height</i>	Define the viewable height of the display in mm (used with optional bezel)

4.6.4 Notes

4.6.5 Return Value

command	<space>	mode	<space>	status	terminator
join	<space>	wall	<space>	success / error [message]	<cr>

4.6.6 Command Examples

```
join wall Encoder1 Decoder1 3x3 1<cr>
join wall Encoder1 Decoder1 3x3 1 size 1920 1080 60<cr>
join wall Encoder1 Decoder1 3x3 1 bezel 1000 980 800 780<cr>
join wall Encoder1 Decoder1 3x3 1 size 1920 1080 60 bezel 1000 980 800 780<cr>
```

4.6.7 Return Examples

```
join wall success<cr>

join wall error [incomplete]<cr>
join wall error [join not permitted]<cr>
join wall error [invalid display_position]<cr>
join wall error [invalid parameter]<cr>
join wall error [encoder 'Encoder1' not found]<cr>
join wall error [decoder 'Decoder1' not found]<cr>
```

4.7 Command join ir

4.7.1 Command usage

```
join ir [key:<security_key>] <encoder_device_name> <decoder_device_name> / <group_name> / <all> [<exclusive>]<cr>
```

4.7.2 Description

The command **join ir** is used for independent routing of infrared signals.

4.7.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>decoder_device_name</i>	Device name of the Decoder, Group or ' all '
<i>exclusive</i>	Keyword ' exclusive ' will remove all other joins (optional)

4.7.4 Notes

- **all** can replace *decoder_device_name* when the infrared is required to be connected to all Decoders.
- The name of a group can replace *decoder_device_name* when the infrared is required to be connected to all Decoders in a group.

4.7.5 Return Value

command	<space>	mode	<space>	status	terminator
join	<space>	ir	<space>	success / error [message]	<cr>

4.7.6 Command Examples

```
join ir Encoder1 Decoder1<cr>
join ir Encoder1 all<cr>
join ir Encoder1 MyGroup<cr>
join ir Encoder1 Decoder1 exclusive<cr>
join ir Encoder1 MyGroup exclusive<cr>
join ir key:abc123 Encoder1 Decoder1<cr>
```

4.7.7 Return Examples

```
join ir success<cr>

join ir error [incomplete]<cr>
join ir error [join not permitted]<cr>
join ir error [source 'Encoder1' not found]<cr>
join ir error [destination 'Decoder1' not found]<cr>
```

5 Command leave

The **leave** commands are used with a Decoder to disconnect from an Encoder's stream.

5.1 Command leave video

5.1.1 Command usage

```
leave video [key:<security_key>] <decoder_device_name><cr>
```

5.1.2 Description

The command **leave video** is used to disconnect a Decoder from the video stream it is receiving.

5.1.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder, Group or 'all'
----------------------------	--------------------------------------------

5.1.4 Notes

- **all** can replace *decoder_device_name* when all the Decoders are required to leave video subscriptions.
- The name of a group can replace *decoder_device_name* when all Decoders in a group are required to leave video subscriptions.

5.1.5 Return Value

command	<space>	mode	<space>	status	terminator
leave	<space>	video	<space>	<i>success / error [message]</i>	<cr>

5.1.6 Command Examples

```
leave video Decoder1<cr>
leave video all<cr>
leave video MyGroup<cr>
leave video key:abc123 Decoder1<cr>
```

5.1.7 Return Examples

```
leave video success<cr>
leave video error [incomplete]<cr>
leave video error [decoder 'Decoder1' not found]<cr>
```

5.2 Command leave audio

5.2.1 Command usage

leave audio [key:<security_key>] <decoder_device_name><cr>

5.2.2 Description

The command **leave audio** is used to disconnect a Decoder from the audio stream it is receiving.

5.2.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder, Group or 'all'
----------------------------	--------------------------------------------

5.2.4 Notes

- **all** can replace *decoder_device_name* when all the Decoders are required to leave audio subscriptions.
- The name of a group can replace *decoder_device_name* when all Decoders in a group are required to leave audio subscriptions.

5.2.5 Return Value

command	<space>	mode	<space>	status	terminator
leave	<space>	audio	<space>	success / error [message]	<cr>

5.2.6 Command Examples

```
leave audio Decoder1<cr>
leave audio all<cr>
leave audio MyGroup<cr>
leave audio key:abc123 Decoder1<cr>
```

5.2.7 Return Examples

```
leave audio success<cr>
leave audio error [incomplete]<cr>
leave audio error [decoder 'Decoder1' not found]<cr>
```

5.3 Command leave serial

5.3.1 Command usage

leave serial[key:<security_key>] <decoder_device_name><cr>

5.3.2 Description

The command **leave serial** is used to disconnect a Decoder from the serial RS232 stream it is receiving.

5.3.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder, Group or 'all'
----------------------------	--------------------------------------------

5.3.4 Notes

- **all** can replace *decoder_device_name* when all the Decoders are required to leave serial subscriptions.
- The name of a group can replace *decoder_device_name* when all Decoders in a group are required to leave serial subscriptions.

5.3.5 Return Value

command	<space>	mode	<space>	status	terminator
leave	<space>	serial	<space>	success / error [message]	<cr>

5.3.6 Command Examples

```
leave serial Decoder1<cr>
leave serial all<cr>
leave serial MyGroup<cr>
leave serial key:abc123 MyGroup<cr>
```

5.3.7 Return Examples

```
leave serial success<cr>

leave serial error [incomplete]<cr>
leave serial error [decoder 'Decoder1' not found]<cr>
```

5.4 Command leave all

5.4.1 Command usage

leave all [key:<security_key>] <decoder_device_name><cr>

5.4.2 Description

The command **leave all** is used to disconnect a Decoder from all the streams it is receiving.

5.4.3 Arguments

<i>decoder_device_name</i>	Name of the Decoder, Group or 'all'
----------------------------	-------------------------------------

5.4.4 Notes

- **all** can replace *decoder_device_name* when all the Decoders are required to leave all subscriptions.
- The name of a group can replace *decoder_device_name* when all Decoders in a group are required to leave all subscriptions.

5.4.5 Return Value

command	<space>	mode	<space>	status	terminator
leave	<space>	all	<space>	success / error [message]	<cr>

5.4.6 Command Example

```
leave all Decoder1<cr>
leave all all<cr>
leave all MyGroup<cr>
leave all key:abc123 Decoder1<cr>
```

5.4.7 Return Examples

```
leave all success<cr>
leave all error [incomplete]<cr>
leave all error [decoder 'Decoder1' not found]<cr>
```


5.5 Command leave av

5.5.1 Command usage

leave av [key:<security_key>] <decoder_device_name><cr>

5.5.2 Description

The command **leave av** is used to disconnect a Decoder from both the audio and video streams it is receiving.

5.5.3 Arguments

<i>decoder_device_name</i>	Name of the Decoder, Group or 'all'
----------------------------	-------------------------------------

5.5.4 Notes

- **all** can replace *decoder_device_name* when all the Decoders are required to leave audio and video subscriptions.
- The name of a group can replace *decoder_device_name* when all Decoders in a group are required to leave audio and video subscriptions.

5.5.5 Return Value

command	<space>	mode	<space>	status	terminator
leave	<space>	av	<space>	success / error [message]	<cr>

5.5.6 Command Example

```
leave av Decoder1<cr>
leave av all<cr>
leave av MyGroup<cr>
leave av key:abc123 Decoder1<cr>
```

5.5.7 Return Examples

```
leave av success<cr>
leave av error [incomplete]<cr>
leave av error [decoder 'Decoder1' not found]<cr>
```

5.6 Command leave ir

5.6.1 Command usage

leave ir [key:<security_key>] <decoder_device_name><cr>

5.6.2 Description

The command **leave ir** is used to disconnect a Decoder from the Infrared stream it is receiving.

5.5.3 Arguments

<i>decoder_device_name</i>	Name of the Decoder, Group or 'all'
----------------------------	-------------------------------------

5.5.4 Notes

- **all** can replace *decoder_device_name* when all the Decoders are required to leave audio and video subscriptions.
- The name of a group can replace *decoder_device_name* when all Decoders in a group are required to leave audio and video subscriptions.

5.5.5 Return Value

command	<space>	mode	<space>	status	terminator
leave	<space>	ir	<space>	success / error [message]	<cr>

5.5.6 Command Example

```
leave ir Decoder1<cr>
leave ir all<cr>
leave ir MyGroup<cr>
leave av key:abc123 Decoder1<cr>
```

5.5.7 Return Examples

```
leave ir success<cr>
leave ir error [incomplete]<cr>
leave ir error [decoder 'Decoder1' not found]<cr>
```

6 Command stop

The **stop** command is used to stop AV Encoder streams from being sent on the network. Joins are maintained between Encoder and Decoders but no data is sent from the Encoder.

6.1 Command stop

6.1.1 Command usage

```
stop [key:<security_key>] <mode> <encoder_device_name>]<cr>
```

6.1.2 Description

The **stop** command is used to stop all Encoder streams from being sent on the network. Joins are maintained between Encoders and Decoders but no data is sent from the Encoder.

6.1.3 Arguments

<i>mode</i>	Always 'all'
<i>encoder_device_name</i>	Device name of the Encoder or Group

6.1.4 Notes

- The name of a group can replace *encoder_device_name* when all Encoders in a group are required to stop.
- All stream types are started together.
- A streaming connection can be made again with the command **join** or **start**.

6.1.5 Return Value

command	<space>	mode	<space>	status	terminator
stop	<space>	all	<space>	success / error [message]	<cr>

6.1.6 Command Examples

```
stop all Encoder1<cr>
stop all MyGroup<cr>
stop key:abc123 all Encoder1<cr>
```

6.1.7 Return Examples

```
stop all success<cr>
stop error [incomplete]<cr>
stop error [invalid mode]<cr>
stop all error [incomplete]<cr>
stop all error [encoder 'Encoder1' not found]<cr>
```

7 Command start

The **start** command is used to start AV Encoder streams.

7.1 Command start

7.1.1 Command usage

`start [key:<security_key>] <mode> <encoder_device_name><cr>`

7.1.2 Description

The command **start** is used to start all Encoder streams.

7.1.3 Arguments

<i>mode</i>	Always 'all'
<i>encoder_device_name</i>	Device name of the Encoder or Group

7.1.4 Notes

- A stream cannot be joined unless it has started, so a **join** command will automatically send the **start** command if the Encoder streams are stopped.
- All stream types are started together.
- The name of a group can replace *encoder_device_name* when all Encoders in a group are required to start all streams.

7.1.5 Return Value

command	<space>	mode	<space>	status	terminator
start	<space>	all	<space>	success / error [message]	<cr>

7.1.6 Command Examples

```
start all Encoder1<cr>
start all MyGroup<cr>
start key:abc123 all Encoder1<cr>
```

7.1.7 Return Examples

```
start all success<cr>

start error [incomplete]<cr>
start error [invalid mode]<cr>
start all error [incomplete]<cr>
start all error [encoder 'Encoder1' not found]<cr>
```

8 Command set

The **set** commands are used to change the working conditions of an Encoder or Decoder.

8.1 Command set audio_a_mute

8.1.1 Command usage

```
set audio_a_mute [key:<security_key>] <device_name> <state><cr>
```

8.1.2 Description

The command **set audio_a_mute** is used to mute analog audio.

8.1.3 Arguments

<i>device_name</i>	Device name of either the Encoder or Decoder
<i>state</i>	true false

8.1.4 Notes

- Use the command **get audio_a_mute** to retrieve this setting.

8.1.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	audio_a_mute	<space>	success / error [message]	<cr>

8.1.6 Command Examples

```
set audio_a_mute Encoder1 false<cr>
set audio_a_mute Encoder1 true<cr>
set audio_a_mute key:abc123 Decoder1 false<cr>
```

8.1.7 Return Examples

```
set audio_a_mute success<cr>
set audio_a_mute error [incomplete]<cr>
set audio_a_mute error [invalid value '<value>']<cr>
set audio_a_mute error [device 'Encoder1' not found]<cr>
set audio_a_mute error [device 'Decoder1' disconnected]<cr>
```

8.2 Command set audio_d_mute

8.2.1 Command usage

set audio_d_mute [key:<security_key>] <decoder_device_name> <state><cr>

8.2.2 Description

The command **set audio_d_mute** is used to mute HDMI audio.

8.2.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
<i>state</i>	true false

8.2.4 Notes

- Use the command **get audio_d_mute** to retrieve this setting.

8.2.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	audio_d_mute	<space>	success / error [message]	<cr>

8.2.6 Command Examples

```
set audio_d_mute Decoder1 false<cr>
set audio_d_mute Decoder1 true<cr>
set audio_d_mute key:abc123 Decoder1 false<cr>
```

8.2.7 Return Examples

```
set audio_d_mute success<cr>

set audio_d_mute error [incomplete]<cr>
set audio_d_mute error [invalid value '<value>']<cr>
set audio_d_mute error [device 'Decoder1' not found]<cr>
set audio_d_mute error [device 'Decoder1' disconnected]<cr>
```



8.3.1 Command usage

8.3.2 Description

8.3.3 Arguments

8.3.4 Notes

- ### 8.3.5 Return Value

8.3.6 Command Examples

8.3.7 Return Examples

```
set edid success<cr>
set edid error [incomplete]<cr>
set edid error [invalid edid_data]<cr>
set edid error [encoder 'Encoder1' not found]<cr>
set edid error [encoder 'Encoder1' disconnected]<cr>
```

8.4 Command set listener

8.4.1 Command usage

```
set listener [key:<security_key>] <notify_ip> <notify_port> <protocol> [<device_ip>] <condition> <state> <device_io> <preset_name> [delay]<cr>
```

8.4.2 Description

The command **set listener** utilises Global Caché device sensor notify functionality. This enables a preset to be triggered from a sensor notify beacon sent when a devices input status changes from a switch or PIR.

8.4.3 Arguments

<i>notify_ip</i>	239.255.250.250
<i>notify_port</i>	9160
<i>protocol</i>	UDP
<i>condition</i>	'on' 'off' 'all'
<i>state</i>	'true' 'false'
<i>device_io</i>	Physical Global Caché device input sensor port 1 to 6
<i>preset_name</i>	Name of the preset to be executed
<i>delay (optional)</i>	Optional delay time of preset execution in minutes

8.4.4 Notes

- Supported Global Caché devices: GC-100-06, GC100-12, GC-100-18, IP2IR, WF2IR, GCIR3, Flex series with Flex Link Relay & Sensor Cable.
- "**condition**" is the the input as '**on**' (closed contact or voltage) or '**off**' (open contact or no voltage)
- "**state**" is the running state of the set listener service as '**true**' or '**false**'.
- Optional "**delay**" is used to delay to preset for the specified amount of time in minutes. The delay is reset each time the command is used.

8.4.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	listener	<space>	<i>success / error [message]</i>	<cr>

8.4.6 Command Examples

```
set listener 239.255.250.250 9160 udp 172.30.10.222 on enabled 1 preset2 30<cr>
set listener 239.255.250.250 9160 udp 172.30.10.222 on disabled 1<cr>
set listener key:abc123 239.255.250.250 9160 udp 172.30.10.222 on enabled 1 preset1<cr>
```

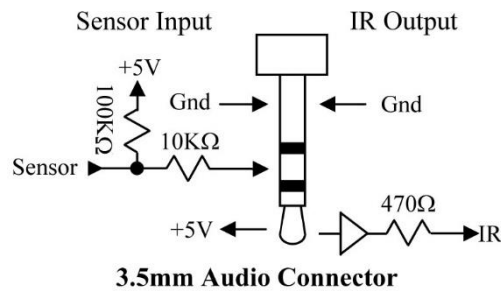
8.4.7 Return Examples

```
set listener success<cr>

set listener error [incomplete]<cr>
set listener error [reserved notify port]<cr>
set listener error [invalid parameter]<cr>
set listener error [preset not found]<cr>

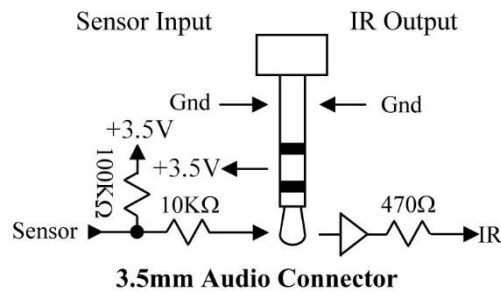
set listener error [no active listener found]<cr>
set listener error [invalid device port]<cr>
```


8.4.8 GC-100 Series IR/Sensor Connector Wiring



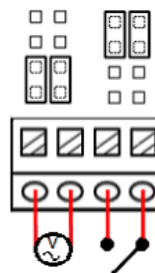
Note: Use connector RING and SLEEVE for closed contact detection

8.4.9 iTach and GlobalConnect Series IR/Sensor Connector Wiring



Note: Use connector TIP and SLEEVE for closed contact detection

8.4.10 Flex Link Relay & Sensor Cable Wiring



Note: Check jumper positions for either voltage or closed contact detection

8.4.1.1 Technical Notes

Global Cache sensor cables

GC-100 series

- GC-SV1 Video Out Sensor
- GC-SP1 AC/DC Voltage Sensor
- GC-SC1 Contact Closure Sensor *

IP2IR, WF2IR and GCIR3

- IT-SV1 Video Out Sensor
- IT-SP1 AC/DC Voltage Sensor
- IT-SC1 Contact Closure Sensor *

Flex series

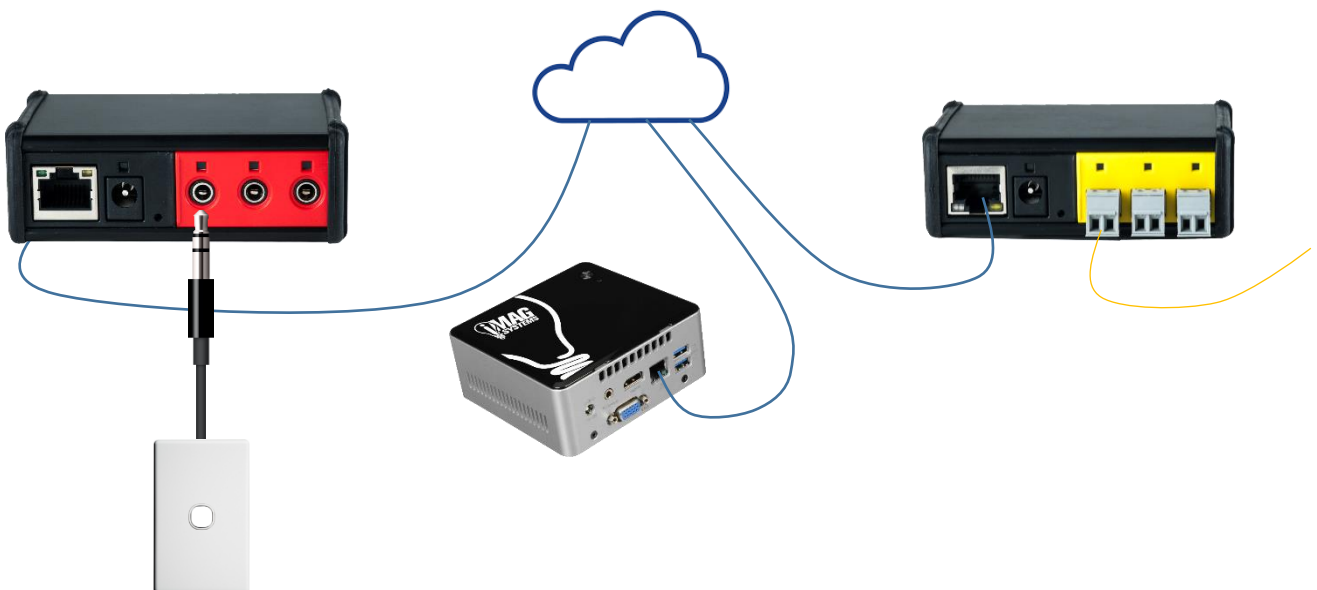
Not required

* For closed contact detection a GC-SC1 or IT-SC1 Contact Closure Sensor cable is not really required if you can make your own. With a 3.5mm stereo phono connector wired with tip and sleeve or ring and sleeve depending on the model as described above in 3.1.8 and 3.1.9.

IR outputs and sensor inputs share a common connector and LED indicator. Each 3.5mm audio connector is independently configured using the assistant. Each connector has three contacts configured as either an infrared (IR) output or sensor input.

When configured as an output, the indicator will blink as an IR command is transmitted. When functioning as a sensor, the indicator is "on" when a positive input or no connection is present. The maximum sensor input voltage is $\pm 24V$, with an "on" indication for voltages greater than 2.5V and "off" when less than 0.8V with an input impedance of $\sim 100K\Omega$.

Here is a simple example of a system using an iTach IP2IR connected to a push button to activate an iTach IP2CC relay.



ToC

8.5 Command set overlay

8.5.1 Command usage

```
set overlay [key:<security_key>] <decoder_device_name> <state> [<horizontal> <vertical>]<cr>
```

8.5.2 Description

The command **set overlay** is used to display a preloaded image on the Decoders display.

8.5.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
<i>state</i>	true false
<i>horizontal</i>	Horizontal position 1..1920 (optional)
<i>vertical</i>	Vertical position 1..1080 (optional)

8.5.4 Notes

- Images are preloaded on the Decoder settings overlay image tab.

8.5.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	overlay	<space>	success / error [message]	<cr>

8.5.6 Command Examples

```
set overlay Decoder1 false<cr>
set overlay key:abc123 Decoder1 true 1 1<cr>
```

8.5.7 Return Examples

```
set overlay success<cr>
set overlay error [incomplete]<cr>
set overlay error [invalid state '<state>']<cr>
set overlay error [invalid horizontal '<horizontal>']<cr>
set overlay error [invalid vertical '<vertical>']<cr>
set overlay error [decoder 'Decoder1' not found]<cr>
set overlay error [decoder 'Decoder1' disconnected]<cr>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

8.6 Command set rotation

8.6.1 Command usage

set rotation [key:<security_key>] <decoder_device_name> <value><cr>

8.6.2 Description

The command **set rotation** is used to rotate the video for displays that have been rotated.

8.6.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
<i>value</i>	0 90 180 270

8.6.4 Notes

8.6.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	rotation	<space>	success / error [message]	<cr>

8.6.6 Command Examples

```
set rotation Decoder1 0<cr>
set rotation key:abc123 Decoder1 5<cr>
```

8.6.7 Return Examples

```
set rotation success<cr>

set rotation error [incomplete]<cr>
set rotation error [invalid value '<value>']<cr>
set rotation error [decoder 'Decoder1' not found]<cr>
set rotation error [decoder 'Decoder1' disconnected]<cr>
```

8.7 Command set scaler

8.7.1 Command usage

set scaler [key:<security_key>] <decoder_device_name> <mode><cr>

8.7.2 Description

The command **set scaler** is used to change the Decoders video output resolution.

8.7.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
<i>mode</i>	auto original 1080p60 .. 720p50

8.7.4 Notes

- Valid mode values:
 - auto = Decoder's connected displays preferred resolution
 - original = Encoder source video resolution
 - 1080p60
 - 1080p50
 - 720p60
 - 720p50

8.7.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	scaler	<space>	success / error [message]	<cr>

8.7.6 Command Examples

```
set scaler Decoder1 auto<cr>
set scaler Decoder1 original<cr>
set scaler Decoder1 1080p60<cr>
set scaler key:abc123 Decoder1 12<cr>
```

8.7.7 Return Examples

```
set scaler success<cr>

set scaler error [incomplete]<cr>
set scaler error [invalid mode '<mode>']<cr>
set scaler error [decoder 'Decoder1' not found]<cr>
set scaler error [decoder 'Decoder1' disconnected]<cr>
set scaler error [monitor not detected]<cr>
```

8.8 Command set var

8.8.1 Command usage

```
set var [key:<security_key>] <var_name> <value> [delete]<cr>
```

8.8.2 Description

The command **set var** is used to store any user defined variable values.

8.8.3 Arguments

<i>var_name</i>	Name of the variable
<i>value</i>	String or boolean true false
delete	Keyword used to delete the variable from the system

8.8.4 Notes

- var_name and value can be any string up to 256 characters.
- Use the command **get var** to retrieve this setting.
- String values with spaces are to be wrapped in "quotation marks".

8.8.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	var	<space>	success / error [message]	<cr>

8.8.6 Command Examples

```
set var MyVar true<cr>
set var myVar "any string upto 256"<cr>
set var MyVar [delete]<cr>
set var key:abc123 MyVar false<cr>
```

8.8.7 Return Examples

```
set var success<cr>

set var error [incomplete]<cr>
set var error [var_name max 256 characters]<cr>
set var error [value max 256 characters]<cr>
```

8.9 Command set video_mode

8.9.1 Command usage

```
set video_mode [key:<security_key>] <decoder_device_name> <mode><cr>
```

8.9.2 Description

The command **set video_mode** is used to set the Decoders video output delay mode.

8.9.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
<i>mode</i>	low high

8.9.4 Notes

- low will provide low latency with low video quality
- high will provide high latency with high video quality

8.9.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	video_mode	<space>	success / error [message]	<cr>

8.9.6 Command Examples

```
set video_mode Decoder1 low<cr>
set video_mode key:abc123 Decoder1 high<cr>
```

8.9.7 Return Examples

```
set video_mode success<cr>
set video_mode error [incomplete]<cr>
set video_mode error [invalid mode '<mode>']<cr>
set video_mode error [decoder 'Decoder1' not found]<cr>
set video_mode error [decoder 'Decoder1' disconnected]<cr>
```

8.10 Command set volume

8.10.1 Command usage

set volume [key:<security_key>] <decoder_device_name> <value> <source><cr>

8.10.2 Description

The command **set volume** is used to set the volume level of either the HDMI or analog audio.

8.10.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
<i>value</i>	-100..12
<i>source</i>	analog hdmi

8.10.4 Notes

8.10.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	volume	<space>	success / error [message]	<cr>

8.10.6 Command Examples

```
set volume Decoder1 0 analog<cr>
set volume key:abc123 Decoder1 -100 hdmi<cr>
```

8.10.7 Return Examples

```
set volume success<cr>

set volume error [incomplete]<cr>
set volume error [invalid value '<value>']<cr>
set volume error [invalid source '<source>']<cr>
set volume error [device 'Encoder1' not found]<cr>
set volume error [device 'Encoder1' disconnected]<cr>
```


8.11 Command set ui_button

8.11.1 Command usage

```
set ui_button [key:<security_key>] <ui_name> <button_name / button_group> <function> [<button_position>] [<value>]<cr>
```

8.11.2 Description

The command **set ui_button** is used to control a button within an active User Interface.

8.11.3 Arguments

<i>ui_name</i>	Name of the User Interface
<i>button_name / button_group</i>	Name of the button or preset logic <<button_name>> Or name of a button group
<i>function</i>	position state text press
<i>button_position</i>	up down both
<i>value</i>	up down, enabled disabled or text string depending on the function

8.11.4 Notes

- Control UI must be enabled as a feature for command to be used.
 - The UI service must be enabled.
 - Used with split buttons, <<encoder>> and <<decoder>> can be used for button text <value>.
 - Used within a preset, <<button_name>> can be used in place of <button_name>.
 - For button groups, only **position** and **state** functions are available. Radio Toggle buttons can only use **position up**.
 - function > position** and **text** uses **up | down**
 - function > position** is only valid for Toggle, Radio Toggle and Split button types.
 - function > state** uses **enabled | disabled**
 - button_position** is only required for function text for Toggle, Radio Toggle and Split button types otherwise 'up' can only be used.
- | | | | | |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> Momentary <ul style="list-style-type: none"> state text press | <ul style="list-style-type: none"> Toggle <ul style="list-style-type: none"> position state text press | <ul style="list-style-type: none"> Radio Toggle <ul style="list-style-type: none"> position state text press | <ul style="list-style-type: none"> Split <ul style="list-style-type: none"> state text press | <ul style="list-style-type: none"> Repeat <ul style="list-style-type: none"> state text press |
|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

8.11.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	ui_button	<space>	success / error [message]	<cr>

8.11.6 Command Examples

```
set ui_button myUI myButton position up<cr>
set ui_button myUI myButton position down<cr>
set ui_button myUI myButton state enabled<cr>
set ui_button myUI myButton state disabled<cr>
set ui_button myUI myButton text up "MyText"<cr>
set ui_button myUI MyButtonGroup position up<cr>
set ui_button myUI MyButtonGroup state disabled<cr>
set ui_button myUI myButton text down <<encoder>><cr>
set ui_button myUI <<button_name>> text both <<encoder>><cr>
set ui_button key:abc123 myUI myButton press<cr>
```

8.11.7 Return Examples

```
set ui_button success<cr>

set ui_button error [incomplete]<cr>
set ui_button error [ui 'myUI' not found]<cr>
set ui_button error [button 'myButton' not found]<cr>
set ui_button error [invalid parameter]<cr>
set ui_button error [invalid button_position]<cr>
set ui_button error [service disabled]<cr>
set ui_button error [invalid button type]<cr>
```

8.12 Command set ui_label

8.12.1 Command usage

set ui_label [key:<security_key>] <ui_name> <label_name> <function> <value><cr>

8.12.2 Description

The command **set ui_label** is used to control a label within a User Interface.

8.12.3 Arguments

<i>ui_name</i>	Name of the User Interface
<i>label_name</i>	Name of the label
<i>function</i>	color visibility text
<i>value</i>	depending on the function

8.12.4 Notes

- Control UI must be enabled as a feature for command to be used.
- The UI service must be enabled.
- When used with split buttons, <<encoder>> and <<decoder>> can be used for label text.
- function color uses HEX RGB color code from 000000 to FFFFFFFF
- function visibility uses true | false

8.12.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	ui_label	<space>	success / error [message]	<cr>

8.12.6 Command Examples

```
set ui_label myUI myLabel color 000000<cr>
set ui_label myUI myLabel visibility false<cr>
set ui_label myUI myLabel visibility true<cr>
set ui_label myUI myLabel text "MyText"<cr>
set ui_label key:abc123 myUI myLabel text <<encoder>><cr>
```

8.12.7 Return Examples

```
set ui_label success<cr>

set ui_label error [incomplete]<cr>
set ui_label error [ui 'myUI' not found]<cr>
set ui_button error [invalid parameter]<cr>
set ui_label error [label 'myLabel' not found]<cr>
```

8.13 Command set ui_image

8.13.1 Command usage

```
set ui_image [key:<security_key>] <ui_name> <image_name> <function> <value><cr>
```

8.13.2 Description

The command **set ui_image** is used to control an image within a User Interface.

8.13.3 Arguments

<i>ui_name</i>	Name of the User Interface
<i>image_name</i>	Name of the image
<i>function</i>	visibility
<i>value</i>	true false

8.13.4 Notes

- Control UI must be enabled as a feature for command to be used.
- The UI service must be enabled.

8.13.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	ui_image	<space>	success / error [message]	<cr>

8.13.6 Command Examples

```
set ui_image myUI myImage visibility false<cr>
set ui_image key:abc123 myUI myImage visibility true<cr>
```

8.13.7 Return Examples

```
set ui_image success<cr>

set ui_image error [incomplete]<cr>
set ui_image error [ui 'myUI' not found]<cr>
set ui_image error [invalid parameter]<cr>
set ui_image error [image 'myImage' not found]<cr>
```

8.14 Command set ui_page

8.14.1 Command usage

```
set ui_page [key:<security_key>] <ui_name> <page_name><cr>
```

8.14.2 Description

The command **set ui_page** is used to change the displayed page in active User Interface.

8.14.3 Arguments

<i>ui_name</i>	Name of the User Interface
<i>page_name</i>	Name of the page

8.14.4 Notes

- Control UI must be enabled as a feature for command to be used.
- The UI service must be enabled.

8.14.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	ui_page	<space>	<i>success / error [message]</i>	<cr>

8.14.6 Command Examples

```
set ui_page myUI myPage<cr>
set ui_page key:abc123 myUI Page2<cr>
```

8.14.7 Return Examples

```
set ui_page success<cr>

set ui_page error [incomplete]<cr>
set ui_page error [ui 'myUI' not found]<cr>
set ui_page error [page 'myPage' not found]<cr>
set ui_page error [invalid parameter]<cr>
set ui_page error [service disabled]<cr>
```

8.15 Command set ui

8.15.1 Command usage

```
set ui [key:<security_key>] <ui_name> <service> [timeout <timeout>] [clients <clients>] [login <pin>]<cr>
```

8.15.2 Description

The command **set ui** is used to enable and disable a User Interface service.

8.15.3 Arguments

<i>ui_name</i>	name of the User Interface
<i>service</i>	'enabled' 'disabled' 'logout'
<i>clients</i>	Optional with enabled service to set the maximum client limit from 1 to 100
<i>pin</i>	Optional with enabled service to set a fixed or random 4 digit login pin code

8.15.4 Notes

- Control UI must be enabled as a feature for command to be used.
- Service logout is the same as disabled then enabled.

8.15.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	ui	<space>	success / error [message]	<cr>

8.15.6 Command Examples

```
set ui myUI disabled<cr>
set ui myUI logout<cr>
set ui myUI enabled<cr>
set ui myUI enabled clients 10 login 1234<cr>
set ui key:abc123 myUI enabled clients 10 login random<cr>
```

8.15.7 Return Examples

```
set ui success<cr>

set ui error [incomplete]<cr>
set ui error [ui 'myUI' not found]<cr>
set ui error [invalid clients '<value>']<cr>
set ui error [invalid pin '<value>']<cr>
set ui error [invalid parameter]<cr>
```

8.16 Command set sync

8.16.1 Command usage

set sync [key:<security_key>] <encoder_device_name> <decoders><cr>

8.16.2 Description

The command **set sync** is used to synchronise the Encoder video stream to all specified Decoders. Used for devices in a video wall configuration or where a single Encoder is joined to multiple Decoders and synchronisation is required between displays.

8.16.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>decoders</i>	Device names of all the Decoders

8.16.4 Notes

- Decoder names are separated with a space.

8.16.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	sync	<space>	success / error [message]	<cr>

8.16.6 Command Examples

```
set sync Encoder1 Decoder1 Decoder2 Decoder3 Decoder4<cr>
set sync key:abc123 Encoder1 Decoder1 Decoder2 Decoder3 Decoder4 Decoder5 Decoder6<cr>
```

8.16.7 Return Examples

```
set sync success<cr>

set sync error [incomplete]<cr>
set sync error [encoder 'Encoder1' not found]<cr>
set sync error [decoder 'Decoder1' not found]<cr>
```

8.17 Command set events

8.17.1 Command usage

```
set events [key:<security_key>] <event_name> <function> <value><cr>
```

8.17.2 Description

The command **set events** is used to enable or disable events created on the UI's Scheduler and Events tabs.

8.17.3 Arguments

<i>event_name</i>	Name of the event
<i>function</i>	Current only " state " supported
<i>value</i>	" enabled " or " disabled "

8.17.4 Notes

- Event must be created on UI's Scheduler or Events tab before using command.

8.17.5 Return Value

command	<space>	mode	<space>	status	terminator
set	<space>	events	<space>	<i>success / error [message]</i>	<cr>

8.17.6 Command Examples

```
set events MyEvent enabled<cr>
set events MyEvent disabled<cr>
```

8.17.7 Return Examples

```
set events success<cr>

set events error [incomplete]<cr>
set events error [event '<event_name>' not found]<cr>
set events error [invalid parameter]<cr>
```


9 Command get

The **get** commands are used to retrieve information from the system or Encoders and Decoders.

9.1 Command get audio_a_mute

9.1.1 Command usage

```
get audio_a_mute [key:<security_key>] <device_name><cr>
```

9.1.2 Description

The command **get audio_a_mute** is used to retrieve the analog audio mute status.

9.1.3 Arguments

<i>device_name</i>	Device name of the Encoder or Decoder
--------------------	---------------------------------------

9.1.4 Notes

- Returned value is either **true** or **false**.
- Use the command **set audio_a_mute** to change this setting.

9.1.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	audio_a_mute	<space>	<i>success data / error [message]</i>	<space>	<string>	<cr>

9.1.6 Command Example

```
get audio_a_mute Encoder1<cr>  
get audio_a_mute key:abc123 Decoder1<cr>
```

9.1.7 Return Examples

```
get audio_a_mute success true<cr>  
get audio_a_mute success false<cr>  
  
get audio_a_mute error [incomplete]<cr>  
get audio_a_mute error [device 'Encoder1' not found]<cr>  
get audio_a_mute error [device 'Decoder1' disconnected]<cr>
```

9.2 Command get audio_d_mute

9.2.1 Command usage

get audio_d_mute [key:<security_key>] <decoder_device_name><cr>

9.2.2 Description

The command **get audio_d_mute** is used to retrieve the HDMI audio mute status.

9.2.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
----------------------------	----------------------------

9.2.4 Notes

- Returned value is either **true** or **false**.
- Use the command **set audio_d_mute** to change this setting.

9.2.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	audio_d_mute	<space>	success data / error [message]	<space>	<string>	<cr>

9.2.6 Command Example

```
get audio_d_mute Decoder1<cr>
get audio_d_mute key:abc123 Decoder1<cr>
```

9.2.7 Return Examples

```
get audio_d_mute success true<cr>
get audio_d_mute success false<cr>

get audio_d_mute error [incomplete]<cr>
get audio_d_mute error [device 'Decoder1' not found]<cr>
get audio_d_mute error [device 'Decoder1' disconnected]<cr>
```

9.3 Command get devices

9.3.1 Command usage

get devices [key:<security_key>] <target><cr>

9.3.2 Description

The command **get devices** is used to retrieve the name and MAC Address of available devices.

9.3.3 Arguments

<i>target</i>	all all_rx all_tx
---------------	-----------------------

9.3.4 Notes

- Return value <device_name> = name of device
- Return value <device_id> = device MAC Address

9.3.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	devices	<space>	success data / error [message]	<space>	<string>	<cr>

9.3.6 Command Example

```
get devices all<cr>
get devices all_tx<cr>
get devices all_rx<cr>
get devices key:abc123 all<cr>
```

9.3.7 Return Examples

```
get devices success '<device_name>--<device_id>','<device_name>--<device_id>'<cr>
get devices success '<device_name>--<device_id>'<cr>

get devices error [incomplete]<cr>
get devices error [invalid target '<target>']<cr>
```

9.4 Command get display_status

9.4.1 Command usage

```
get display_status [key:<security_key>] <decoder_device_name><cr>
```

9.4.2 Description

The command **get display_status** is used to find if a Decoder has a display connected.

9.4.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
----------------------------	----------------------------

9.4.4 Notes

- Returned mode is either **true** or **false**.
- Some non-compliant displays will need to be powered on before detection is possible.

9.4.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	display_status	<space>	<i>success data / error [message]</i>	<space>	<string>	<cr>

9.4.6 Command Example

```
get display_status Decoder1<cr>
get display_status key:abc123 Decoder1<cr>
```

9.4.7 Return Examples

```
get display_status success true<cr>
get display_status success false<cr>

get display_status error [incomplete]<cr>
get display_status error [decoder 'Decoder1' not found]<cr>
get display_status error [decoder 'Decoder1' disconnected]<cr>
```



9.5.1 Command usage

9.5.2 Description

9.5.3 Arguments

9.5.4 Notes

- ## 9.5.5 Return Value

9.5.6 Command Example

9.5.7 Return Examples

ToC

9.6 Command get joins

9.6.1 Command usage

```
get joins [key:<security_key>] <decoder_device_name> <subscription><cr>
```

9.6.2 Description

The command **get joins** is used to retrieve the Encoder name subscribed to a Decoder's subscription.

9.6.3 Arguments

<i>decoder_device_name</i>	Device name of a Decoder
<i>subscription</i>	video audio serial ir

9.6.4 Notes

9.6.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	joins	<space>	success data / error [message]	<space>	<string>	<cr>

9.6.6 Command Examples

```
get joins Decoder1 video<cr>
get joins Decoder1 audio<cr>
get joins Decoder1 serial<cr>
get joins Decoder1 ir<cr>
get joins key:abc123 Decoder1 video<cr>
```

9.6.7 Return Examples

```
get joins success Encoder1 <cr>

get joins error [incomplete]<cr>
get joins error [invalid subscription '<subscription>']<cr>
get joins error [no encoder connected]<cr>
get joins error [decoder 'Decoder1' not found]<cr>
get joins error [decoder 'Decoder1' disconnected]<cr>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

9.7 Command get overlay

9.7.1 Command usage

```
get overlay [key:<security_key>] <decoder_device_name><cr>
```

9.7.2 Description

The command **get overlay** is used to retrieve the Decoder overlay display status.

9.7.3 Arguments

<i>decoder_device_name</i>	Device name of a Decoder
----------------------------	--------------------------

9.7.4 Notes

- Use the command **set overlay** to change this setting.
- Returned value will be either **true** or **false**.

9.7.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	overlay	<space>	<i>success data / error [message]</i>	<space>	<string>	<cr>

9.7.6 Command Examples

```
get overlay Decoder1<cr>
get overlay key:abc123 Decoder1<cr>
```

9.7.7 Return Examples

```
get overlay success true<cr>
get overlay success false<cr>

get overlay error [incomplete]<cr>
get overlay error [decoder 'Decoder1' not found]<cr>
get overlay error [decoder 'Decoder1' disconnected]<cr>
```

9.8 Command get preferred

9.8.1 Command usage

get preferred [key:<security_key>] <decoder_device_name> <option><cr>

9.8.2 Description

The command **get preferred** is used to retrieve the preferred resolution of a display connected to a Decoder.

9.8.3 Arguments

<i>decoder_device_name</i>	Device name of Decoder
<i>option</i>	width height fps

9.8.4 Notes

- A display must be connected to the Decoder for the EDID to be retrieved. Some non-compliant displays may need to be switched on before the EDID can be accessed.

9.8.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	preferred	<space>	success data / error [message]	<space>	<string>	<cr>

9.8.6 Command Example

```
get preferred Decoder1 width<cr>
get preferred Decoder1 height<cr>
get preferred Decoder1 fps<cr>
get preferred key:abc123 Decoder1 width<cr>
```

9.8.7 Return Example

```
get preferred success 1920<cr>
get preferred success 1080<cr>
get preferred success 60<cr>

get preferred error [incomplete]<cr>
get preferred error [invalid option '<option>']<cr>
get preferred error [no EDID available]<cr>
get preferred error [decoder 'Decoder1' not found]<cr>
get preferred error [decoder 'Decoder1' disconnected]<cr>
```


9.9 Command get rotation

9.9.1 Command usage

get rotation [key:<security_key>] <decoder_device_name><cr>

9.9.2 Description

The command **get rotation** is used to retrieve the video output rotation status of the specified Decoder.

9.9.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
----------------------------	----------------------------

9.9.4 Notes

- Return value will be between 0 and 7.
- Use the command **set rotation** to change this setting.

9.9.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	rotation	<space>	success data / error [message]	<space>	<string>	<cr>

9.9.6 Command Example

```
get rotation Decoder1<cr>
get rotation key:abc123 Decoder1<cr>
```

9.9.7 Return Examples

```
get rotation success 0<cr>
get rotation success 7<cr>

get rotation error [incomplete]<cr>
get rotation error [decoder 'Decoder1' not found]<cr>
get rotation error [decoder 'Decoder1' disconnected]<cr>
```

9.10 Command get scaler

9.10.1 Command usage

get scaler [key:<security_key>] <decoder_device_name> <option><cr>

9.10.2 Description

The command **get scaler** is used to retrieve the scaled video resolution of the Decoder's HDMI video.

9.10.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
<i>option</i>	all width height fps

9.10.4 Notes

9.10.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	scaler	<space>	success data / error [message]	<space>	<string>	<cr>

9.10.6 Command Examples

```
get scaler Decoder1 all<cr>
get scaler Decoder1 width<cr>
get scaler Decoder1 height<cr>
get scaler Decoder1 fps<cr>
get scaler key:abc123 Decoder1 all<cr>
```

9.10.7 Return Examples

```
get scaler success 1920 1080 60<cr>
get scaler success 1920<cr>
get scaler success 1080<cr>
get scaler success 60<cr>

get scaler error [incomplete]<cr>
get scaler error [invalid option '<option>']<cr>
get scaler error [decoder 'Decoder1' not found]<cr>
get scaler error [decoder 'Decoder1' disconnected]<cr>
```

9.11 Command get status

9.11.1 Command usage

get status [key:<security_key>] <device_name> [<stream>]<cr>

9.11.2 Description

The command **get status** is used to retrieve the status of the specified device or individual Encoder device stream or Decoder subscription.

9.11.3 Arguments

<i>device_name</i>	Name of the Encoder or Decoder
<i>stream</i>	video audio serial ir

9.11.4 Notes

- When no stream is specified the return will be as seen on the status of the Encoder / Decoder UI Status tab, this is a general status of the device.

9.11.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	status	<space>	success data / error [message]	<space>	<string>	<cr>

9.11.6 Command Example

```
get status Encoder1<cr>
get status Encoder1 audio<cr>
get status Encoder1 video<cr>
get status Encoder1 serial<cr>
get status Encoder1 ir<cr>
get status key:abc123 Encoder1 ir<cr>
```

```
get status Decoder1<cr>
get status Decoder1 audio<cr>
get status Decoder1 video<cr>
get status Decoder1 serial<cr>
get status Decoder1 ir<cr>
```

9.11.7 Return Examples

```
get status success streaming<cr>
get status success stopped<cr>
get status success timeout<cr>
get status success disconnected<cr>
get status success out of range<cr>

get status error [incomplete]<cr>
get status error [invalid stream '<stream>']<cr>
get status error [device 'Encoder1' not found]<cr>
```

9.12 Command get var

9.12.1 Command usage

```
get var [key:<security_key>] <var_name><cr>
```

9.12.2 Description

The command **get var** is used to retrieve the value of the specified user defined variable.

9.12.3 Arguments

<i>var_name</i>	Name of the variable
-----------------	----------------------

9.12.4 Notes

9.12.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	var	<space>	<i>success data / error [message]</i>	<space>	<string>	<cr>

9.12.6 Command Example

```
get var MyVar<cr>
```

9.12.7 Return Examples

```
get var success <value><cr>
```

```
get var error [incomplete]<cr>
```

```
get var error [var '<var_name>' not found]<cr>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

9.13 Command get ver

9.13.1 Command usage

get ver [key:<security_key>] <device_name><cr>

9.13.2 Description

The command **get ver** is used to retrieve the current firmware version of a Decoder or Encoder.

9.13.3 Arguments

<i>device_name</i>	Device name of either a Decoder or Encoder
--------------------	--------------------------------------------

9.13.4 Notes

9.13.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	ver	<space>	success data / error [message]	<space>	<string>	<cr>

9.13.6 Command Examples

```
get ver Encoder1<cr>
get ver key:abc123 Decoder1<cr>
```

9.13.7 Return Examples

```
get ver success 1.1.2<cr>
get ver error [incomplete]<cr>
get ver error [device 'Encoder1' not found]<cr>
get ver error [device 'Encoder1' disconnected]<cr>
```



9.14 Command get video

9.14.1 Command usage

get video [key:<security_key>] <encoder_device_name> <option><cr>

9.14.2 Description

The command **get video** is used to retrieve the connected video information from an Encoder.

9.14.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
<i>option</i>	all width height fps sm

9.14.4 Notes

- **all** returns <width> <height> <frames_per_second> <scan_mode>

9.14.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	video	<space>	<i>success data / error [message]</i>	<space>	<string>	<cr>

9.14.6 Command Example

```
get video Encoder1 all<cr>
get video Encoder1 width<cr>
get video Encoder1 height<cr>
get video Encoder1 fps<cr>
get video Encoder1 sm<cr>
get video key:abc123 Encoder1 all<cr>
```

9.14.7 Return Examples

```
get video success 1920 1080 60 PROGRESSIVE<cr>
get video success 1920<cr>
get video success 1080<cr>
get video success 60<cr>
get video success PROGRESSIVE<cr>
get video success INTERLACED <cr>

get video error [incomplete]<cr>
get video error [invalid option '<option>']<cr>
get video error [encoder 'Encoder1' not found]<cr>
get video error [encoder 'Encoder1' disconnected]<cr>
```

9.15 Command get video_mode

9.15.1 Command usage

```
get video_mode [key:<security_key>] <decoder_device_name><cr>
```

9.15.2 Description

The command **get video_mode** is used to retrieve the video mute status of the specified Decoder.

9.15.3 Arguments

<i>decoder_device_name</i>	Device name of the Decoder
----------------------------	----------------------------

9.15.4 Notes

- Return value will be between either **low** or **high**.
- Use the command **set video_mode** to change this setting.

9.15.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	video_mode	<space>	<i>success data / error [message]</i>	<space>	<string>	<cr>

9.15.6 Command Example

```
get video_mode Decoder1<cr>  
get video_mode key:abc123 Decoder1<cr>
```

9.15.7 Return Examples

```
get video_mode success low<cr>  
get video_mode success high<cr>  
  
get video_mode error [incomplete]<cr>  
get video_mode error [decoder 'Decoder1' not found]<cr>  
get video_mode error [decoder 'Decoder1' disconnected]<cr>
```

9.16 Command get video_status

9.16.1 Command usage

```
get video_status [key:<security_key>] <encoder_device_name><cr>
```

9.16.2 Description

The command **get video_status** is used to find if an Encoder has a stable video source connected.

9.16.3 Arguments

<i>encoder_device_name</i>	Device name of the Encoder
----------------------------	----------------------------

9.16.4 Notes

- Returned mode is either **true** or **false**.

9.16.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	video_status	<space>	success data / error [message]	<space>	<string>	<cr>

9.16.6 Command Example

```
get video_status Encoder1<cr>
get video_status key:abc123 Encoder1<cr>
```

9.16.7 Return Examples

```
get video_status success true<cr>
get video_status success false<cr>

get video_status error [incomplete]<cr>
get video_status error [encoder 'Encoder1' not found]<cr>
get video_status error [encoder 'Encoder1' disconnected]<cr>
```




DIRECTOR RAIN

COMMAND GUIDE V1.3.6

9.17 Command get volume

9.17.1 Command usage

get volume [key:<security_key>] <device_name><cr>

9.17.2 Description

The command **get volume** is used to retrieve the current analog audio volume level.

9.17.3 Arguments

<i>device_name</i>	Device name of either a Decoder or Encoder
--------------------	--------------------------------------------

9.17.4 Notes

- Use the command **set volume** to change this setting.

9.17.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	volume	<space>	success data / error [message]	<space>	<string>	<cr>

9.17.6 Command Examples

```
get volume Encoder1<cr>
get volume key:abc123 Decoder1<cr>
```

9.17.7 Return Examples

```
get volume success 0<cr>
get volume success 100<cr>

get volume error [incomplete]<cr>
get volume error [device 'Encoder1' not found]<cr>
get volume error [device 'Encoder1' disconnected]<cr>
```

9.18 Command get ui_button

9.18.1 Command usage

```
get ui_button [key:<security_key>] <ui_name> <button_name> <function><cr>
```

9.18.2 Description

The command **get ui_button** is used to retrieve the current state of a User Interface button.

9.18.3 Arguments

<i>ui_name</i>	Name of the User Interface
<i>button_name</i>	Name of the button in the User Interface or preset logic <<button_name>>
<i>function</i>	position state

9.18.4 Notes

- Control UI must be enabled as a feature for command to be used.
- The UI service must be enabled.
- Use the command **set ui_button** to change this setting.
- function > position** uses **up** | **down**
- function > state** uses **enabled** | **disabled**
- Momentary**
 - state
 - Toggle**
 - position
 - state
 - Radio Toggle**
 - position
 - state
 - Split**
 - position
 - state
 - Repeat**
 - state

9.18.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	ui_button	<space>	success data / error [message]	<space>	<string>	<cr>

9.18.6 Command Examples

```
get ui_button MyUI MyButton position<cr>
get ui_button MyUI <<button_name>> position<cr>
get ui_button key:abc123 MyUI MyButton state<cr>
```

9.18.7 Return Examples

```
get ui_button position success up<cr>
get ui_button position success down<cr>
get ui_button state success enabled<cr>
get ui_button state success disabled<cr>

get ui_button error [incomplete]<cr>
get ui_button error [invalid parameter]
get ui_button error [User Interface '<ui_name>' not found]<cr>
get ui_button error [button '<button_name>' not found]<cr>
```

9.19 Command get ui

9.19.1 Command usage

```
get ui [key:<security_key>] <ui_name><cr>
```

9.19.2 Description

The command **get ui** is used to retrieve the User Interface status.

9.19.3 Arguments

<i>ui_name</i>	Name of the User Interface
----------------	----------------------------

9.19.4 Notes

- Control UI must be enabled as a feature for command to be used.

9.25.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	ui	<space>	<i>success data / error [message]</i>	<space>	<string>	<cr>

9.25.6 Command Example

```
get ui MyUI<cr>
```

9.25.7 Return Examples

```
get ui disabled<cr>
get ui enabled<cr>
get ui enabled timeout 240<cr>
get ui enabled clients 1<cr> *number of users 1 to 100
get ui enabled login 1234<cr> *0000 to 9999
get ui enabled timeout 240 clients 1 login 1234<cr>

get ui error [user interface 'MyUI' not found]<cr>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

9.20 Command get events

9.20.1 Command usage

```
get events [key:<security_key>] <event_name> <function><cr>
```

9.20.2 Description

The command **get events** is used to retrieve the state of events created on the UI's Scheduler and Events tabs.

9.20.3 Arguments

<i>event_name</i>	Name of the event
<i>function</i>	Current only " state " supported

9.20.4 Notes

- Event must be created on UI's Scheduler or Events tab before using command.

9.20.5 Return Value

command	<space>	mode	<space>	status	<space>	value	terminator
get	<space>	events	<space>	<i>success data / error [message]</i>	<space>	<string>	<cr>

9.20.6 Command Example

```
get events state MyEvent<cr>
```

9.20.7 Return Examples

```
get events success enabled<cr>
get events success disabled<cr>

get events error [incomplete]<cr>
get events error [event '<event_name>' not found]<cr>
get events error [invalid parameter]<cr>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

10 Command send

The **send** commands are used to send serial RS-232 data to any or all Encoders or Decoders from a 3rd party control system. Any TCP controllable device or Global Caché device can be controlled.

10.1 Command send cec

10.1.1 Command usage

```
send cec [key:<security_key>] <device_name> / <group_name> / <all> / <all_tx> / <all_rx> <data_hex><cr>
```

10.1.2 Description

The command **send cec** is used to send cec data from a control system to a Decoder's connected display.

10.1.3 Arguments

<i>device_name</i>	Device name of the Decoder, Encoder, Group or 'all' 'all_tx' 'all_rx'
<i>data_hex</i>	String of ascii characters representing the hexadecimal cec code

10.1.4 Notes

- The **data_hex** argument is a hexadecimal string which represent the cec code to be sent.
- The command will return with an error when using a single device if no source is connected on an Encoder or no display connected on a Decoder.
- **group_name** is used as a destination when all Encoders and Decoders in a group are required to send CEC.

10.1.5 Return Value

command	<space>	mode	<space>	status	terminator
send	<space>	cec	<space>	success / error	<cr>

10.1.6 Command Examples

```
send cec Decoder1 F004<cr>
send cec MyGroup F004<cr>
send cec key:abc123 Decoder1 F004<cr>
```

10.1.7 Return Examples

```
send cec success<cr>

send cec error [incomplete]<cr>
send cec error [This device does not support CEC]<cr>
send cec error [HDMI disconnected]<cr>
send cec error [decoder 'Decoder1' not found]<cr>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

10.2 Command send gc

The command **send gc** provides a seamless integration with Global Caché products. For more information on Global Caché visit www.globalcache.com.

10.2.1 Command usage

```
send gc [key:<security_key>] <address> <port> <gc_api><cr>
```

10.2.2 Description

The **send gc** command allows control over numerous Global Caché products via the same single TCP connection used for the Controller.

10.2.3 Arguments

<i>address</i>	Global Caché IP address
<i>port</i>	Global Caché TCP port. Usually 4998 and for serial com1: 4999 com2: 5000
<i>gc_api</i>	Global Caché API string

10.2.4 Notes

- The **gc_api** string uses the same standard Global Caché control string format as found in the Global Caché API manuals downloaded from: www.globalcache.com/downloads/
- Use keyword "[disconnect]" in place of **gc_api** string to terminate the connection.
- String sent to port 4999 or 5000 must be wrapped in quotations "my string".

10.2.5 Return Value

Return values will be in the standard Global Caché format under normal conditions. An exception to this would be if a TCP connection was not possible to a device, in which case an error such as `send gc error [device '172.30.1.111' not found]<cr>` would be sent.

10.2.6 Examples

```
send gc 172.30.1.111 4999 "a string to send"<cr>
send gc 172.30.1.111 4998 setstate,1:1,1<cr>
send gc 172.30.1.111 4998 sendir,1:2,4444,34500,1,1,34,48,24,12,24,960,24,12,24...<cr>
send gc 172.30.1.111 4998 [disconnect]<cr>
send gc key:abc123 172.30.1.111 4999 "\x00\x01"<cr>
```

10.2.7 Return Examples

```
a serial string<cr>
state,1:1,1<cr>
completeir,1:1,1<cr>

send gc error [incomplete]<cr>
send gc error [device '172.30.1.111' not found]<cr>
send gc error [device '172.30.1.111' timeout]<cr>
send gc error [invalid]<cr>
ERR_<XX><cr>
```

10.3 Command send serial

10.3.1 Command usage

send serial [key:<security_key>] <device_name> / <group_name> / <all> / <all_tx> / <all_rx> "<data_string>" [<feedback> ["<feedback_string>"]]<cr>

10.3.2 Description

The command **send serial** is used to send serial RS-232 data from a control system to Encoders and Decoders.

10.3.3 Arguments

<i>device_name</i>	Device name of Encoder, Decoder, Group or 'all' 'all_tx' 'all_rx'
<i>data_string</i>	String of ASCII characters
<i>feedback</i>	Keywords reply , equals or contains (optional)
<i>feedback_string</i>	String used with equals or contains to compare with the feedback string

10.3.4 Notes

- When either 2-way communication is required and the control system is expecting a reply from the serial equipment or unsolicited serial data is expected, then the device must be set for **control** mode in the RS232 serial port settings. When unsolicited serial data is received the Controller will raise a 'notify serial' event.
- The **data_string** and **feedback_string** arguments are ASCII text strings when set to ASCII mode. When the device is in ASCII mode it will only be possible to send escaped \x0D carriage return and / or \x0A line feed.
 - abcdefghijklmnopqrstuvwxyz0123456789\x0D
 - \x0D\x0A
- The **data_string** and **feedback_string** arguments are ASCII text strings of bytes when set to HEX mode.
 - 00FF
 - \x00\xff
- The **feedback** option uses keywords **reply**, **equals** or **contains** to set the type of feedback. To receive a string only, use **reply**. For comparison with the specified **feedback_string** use **equals** for an exact match, or **contains** for a match within the string. Can only be used when **device_name** is a single Encoder or Decoder.
- feedback_string** contains the expected device's feedback string result.
- group_name** is used as a destination when all Encoders and Decoders in a group are required to send.



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

10.3.5 Return Value

command	<space>	mode	<space>	status	terminator
send	<space>	serial	<space>	<i>success [data] / error [message]</i>	<cr>

10.3.6 Command Examples

```
send serial Decoder1 "my data string\x0D"<cr>
send serial Encoder1 "\x00\x01\x02\x03\x04"<cr>
send serial Decoder1 "my data string" reply<cr>
send serial Decoder1 "my data string\x0D" contains "\x0D"<cr>
send serial Decoder1 "my data string\x0D\x0A" equals "OK"<cr>
send serial MyGroup "my data string\x0D"<cr>
send serial key:abc123 Decoder1 "000102FF"<cr>
```

10.3.7 Return Examples

```
send serial success [my return string]<cr>
send serial success [00FF0D]<cr> * Received HEX in HEX mode
send serial success [\x00\xff\x0D]<cr> * Received HEX in ASCII mode
send serial success []<cr>
send serial success<cr>

send serial error [incomplete]<cr>
send serial error [invalid HEX data]<cr>
send serial error [invalid HEX feedback]<cr>
send serial error [invalid parameter]<cr>
send serial error [device 'Encoder1' not found]<cr>
```


10.4 Command send tcp

10.4.1 Command usage

```
send tcp [key:<security_key>] <address> <port> "<command>" [<feedback> ["<feedback_string>"]]<cr>
```

10.4.2 Description

The command **send tcp** provides a seamless integration with any TCP controllable device.

10.4.3 Arguments

<i>address</i>	TCP IP address
<i>port</i>	TCP port
<i>command</i>	Command string to send to TCP device
<i>feedback</i>	Keyword reply , equals or contains (optional)
<i>feedback_string</i>	Expected feedback string used with equals or contains

10.4.4 Notes

- The TCP device must be in the same range as the Controller.
- To send HEX add \x before the HEX byte. \x0D for carriage return
- The feedback option uses keywords reply, equals or contains to set the type of feedback.
To receive a string only, use reply.
For comparison with the specified feedback_string use **equals** for an exact match, or **contains** for a match within the string. Can only be used when device_name is a single Encoder or Decoder.
- feedback_string** contains the expected device's feedback string result.
- Use keyword "[**disconnect**]" in place of **command** string to terminate the connection.

10.4.5 Return Value

A success return value will contain what is returned from the TCP device.

10.4.6 Examples

```
send tcp 172.30.1.111 1000 "ascii string"<cr>      send tcp 172.30.1.111 1000 "an mixed string\x0D"<cr>
send tcp 172.30.1.111 1000 "\x00\x01\x02\x03"<cr>
send tcp 172.30.1.111 1000 "ascii string" contains "feedback string"<cr>
send tcp 172.30.1.111 1000 "ascii string" equals "feedback string"<cr>
send tcp 172.30.1.111 1000 "ascii string" reply<cr> send tcp 172.30.1.111 1000 [disconnect]<cr>
send tcp key:abc123 172.30.1.111 1000 "ascii string"<cr>
```

10.4.7 Return Examples

```
send tcp success<cr>
send tcp success [<feedback>]<cr>
send tcp error [incomplete]<cr>
send tcp error [device '172.30.1.111' not found]<cr>
send tcp error [device '172.30.1.111:1000' not connected]<cr>

send tcp success [disconnected]<cr>
send tcp error [invalid parameter]<cr>
send tcp error [<feedback>]<cr>
```



10.5.1 Command usage

10.5.2 Description

10.5.3 Arguments

10.5.4 Notes

- ### 10.5.5 Return Value

10.5.6 Command Examples

10.5.7 Return Examples

```
send ir error [incomplete]<cr>
send ir error [max length exceeded]<cr>
send ir error [Length of hex data should be in multiples of 4 bytes]<cr>
send ir error [device 'Encoder1' not found]<cr>
```

11 Command preset

The preset commands are used to store and apply a series of commands available from this manual. A sequence of commands can be used to create routing tables or video wall. Refer Appendix B - Preset logic for logic that can be applied within a preset.

11.1 Command preset add

11.1.1 Command usage

```
preset add [key:<security_key>] <preset_name> <preset_data><cr>
```

11.1.2 Description

The command **preset add** is used to create and append commands to a specified preset.

11.1.3 Arguments

<i>preset_name</i>	The name defined as the preset
<i>preset_data</i>	A valid Control Command string

11.1.4 Notes

- The preset is executed with the **preset load** command.
- Preset commands are not allowed in a preset itself, only used to create, delete and execute presets.

11.1.5 Return Value

command	<space>	function	<space>	name	<space>	status	terminator
preset	<space>	add	<space>	<i>preset1</i>	<space>	<i>success / error</i> <i>[message]</i>	<cr>

11.1.6 Command Examples

```
preset add preset1 join all encoder1 decoder1<cr>
preset add key:abc123 preset1 join all Encoder1 Decoder1<cr>
```

11.1.7 Return Examples

```
preset add preset1 success<cr>
preset add error [incomplete]<cr>
```

11.2 Command preset delay

11.2.1 Command usage

preset delay [key:<security_key>] <sec>

11.2.2 Description

The command **preset delay** is used within a preset to add a delay between commands.

11.2.3 Arguments

sec	Delay time in milliseconds up to 9999
-----	---------------------------------------

11.2.4 Notes

- This command can only be used within a preset.

11.2.5 Return Value

None

11.2.6 Command Examples

```
preset delay 1000  
preset delay key:abc123 500
```

11.2.7 Return Example

None

11.3 Command preset delete

11.3.1 Command usage

preset delete [key:<security_key>] <preset_name><cr>

11.3.2 Description

The command **preset delete** is used to delete the specified preset from the Thunder Director Controller or directly from the UI.

11.3.3 Arguments

<i>preset_name</i>	The name defined as the preset
--------------------	--------------------------------

11.3.4 Notes

11.3.5 Return Value

command	<space>	function	<space>	name	<space>	status	terminator
preset	<space>	delete	<space>	<i>preset1</i>	<space>	<i>success / error [message]</i>	<cr>

11.3.6 Command Examples

```
preset delete preset1<cr>
preset delete key:abc123 preset1<cr>
```

11.3.7 Return Examples

```
preset delete preset1 success<cr>
preset delete error [incomplete]<cr>
preset delete error [preset 'preset1' not found]<cr>
preset error [invalid mode]<cr>
```

11.4 Command preset load

11.4.1 Command usage

preset load [key:<security_key>] <preset_name> [timeout]<cr>

11.4.2 Description

The command **preset load** is used to apply stored commands within the specified preset.

11.4.3 Arguments

<i>preset_name</i>	The name defined as the preset
<i>delay</i>	Delay in minutes before preset is applied (optional)

11.4.4 Notes

- The preset is created with the **preset add** command or directly via the UI.
- Optional "**delay**" is used to delay a preset for the specified amount of time in minutes. The delay is reset each time the command is used and -1 will terminate the command.

11.4.5 Return Value

command	<space>	function	<space>	name	<space>	status	terminator
preset	<space>	load	<space>	<i>preset1</i>	<space>	<i>success / error</i> <i>[message]</i>	<cr>

11.4.6 Command Examples

```
preset load preset1<cr>
preset load preset1 30<cr>
preset load preset1 -1<cr>
preset load key:abc123 preset1<cr>
```

11.4.7 Return Examples

```
preset load preset1 success<cr>

preset load error [incomplete]<cr>
preset load error [preset 'preset1' not found]<cr>
preset load preset1 error [...]<cr>
```

12 Message notify

The **notify** messages are sent from the Controller to a third party control system connected on Telnet port 6980 with updated event notifications. A **notify** message will be sent on the following events:

- Serial RS-232 data received from Encoder or Decoder
- Encoder or Decoder network connectivity
- Decoder display connectivity
- Encoder source connectivity

12.1 Message notify serial

12.1.1 Message received

notify serial <device_name> <data_string><cr>

12.1.2 Description

A **notify serial** message is sent when serial RS-232 data from an Encoder or Decoder is received.

12.1.3 Arguments

<i>device_name</i>	Device name of Encoder or Decoder
<i>data_string</i>	String of ascii characters

12.1.4 Notes

- Before **notify serial** message can be received, the device must be set to **control** mode.

12.1.5 Return Value

message	<space>	mode	<space>	data_string	terminator
notify	<space>	serial	<space>	<i>ascii_data</i>	<cr>

12.1.6 Examples

```
notify serial 'Decoder1' my data string\x0D<cr>
notify serial 'Decoder1' \x00\x01\x02\x03\xFF<cr>
notify serial 'Encoder1' 0001020304<cr>
```



12.2 Message notify network

12.2.1 Message received

notify network <device_name> <state><cr>

12.2.2 Description

A **notify network** message is sent whenever an Encoder or Decoder is connected or disconnected from the network.

12.2.3 Arguments

<i>device_name</i>	Device name of Encoder or Decoder
<i>state</i>	'true' 'false'

12.2.4 Notes

- A **notify network** message will be sent when the Controller is unable to connect or connects with an Encoder or Decoder. For example, a false then true message will be sent during a device power cycle or reboot.

12.2.5 Return Value

message	<space>	mode	<space>	status	terminator
notify	<space>	network	<space>	true false	<cr>

12.2.6 Examples

```
notify network 'Decoder1' false<cr>
notify network 'Decoder1' true<cr>
```


12.3 Message notify display

12.3.1 Message received

notify display <decoder_device_name> <state><cr>

12.3.2 Description

A **notify display** message is sent whenever a display is connected or disconnected from a Decoder.

12.3.3 Arguments

<i>decoder_device_name</i>	Device name of Decoder
<i>state</i>	'true' 'false'

12.3.4 Notes

- Some non-compliant displays will require power for this event to be raised.

12.3.5 Return Value

message	<space>	mode	<space>	status	terminator
notify	<space>	display	<space>	true false	<cr>

12.3.6 Examples

```
notify display 'Decoder1' false<cr>
notify display 'Decoder1' true<cr>
```



12.4 Message notify source

12.4.1 Message received

notify source <encoder_device_name> <state><cr>

12.4.2 Description

A **notify source** message is sent whenever a source is connected or disconnected from an Encoder.

12.4.3 Arguments

<i>encoder_device_name</i>	Device name of Encoder
<i>state</i>	'true' 'false'

12.4.4 Notes

- The Controller is looking for a stable valid video signal.

12.4.5 Return Value

message	<space>	mode	<space>	status	terminator
notify	<space>	source	<space>	true false	<cr>

12.4.6 Examples

```
notify source 'Encoder1' false<cr>
notify source 'Encoder1' true<cr>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Appendix A - How to HTTP request

GET = `http://<controllerURL>/api/command/<DIRECTOR_API_COMMAND>/<KEY>`

POST = `http://<controllerURL>/api/command{"cmd": "<DIRECTOR_API_COMMAND>", "key": "<KEY>"}`

Example 1: POST - ajax

```
<script language='JavaScript' type='text/javascript'>
  var controllerIP = '169.254.1.1'; *change this to the same IP address as the SDVoE Director controller
  var BaseURL = 'http://' + controllerIP + '/api/command';
  var MAXIMUM_WAITING_TIME = 5000; *timeout in milliseconds
  var CheckStatusTimer;
  var key = '123xyz'; *replace this with the generated security key
  var command = '123xyz'; *replace with any Director API command

  $.ajax({
    type: 'POST',
    crossDomain: true,
    contentType: 'application/json; charset=utf-8',
    dataType: 'text',
    url: BaseURL,
    data: '{"cmd":"' + command + '", "key":"' + key + '"}',
    timeout: MAXIMUM_WAITING_TIME,
    success: function(data, textStatus, XMLHttpRequest){
      console.log(data);
    },
    error: function (XMLHttpRequest, textStatus, errorThrown) {
      console.log('ERROR = ' + errorThrown);
    }
  });
</script>
```

Example 2: POST - xhr

```
<script language='JavaScript' type='text/javascript'>
  var controllerIP = '169.254.1.1'; *change this to the same IP address as the SDVoE Director controller
  var BaseURL = 'http://' + controllerIP + '/api/command';
  var key = '123xyz'; *replace this with the generated security key
  var command = '123xyz'; *replace with any DIRECTOR API command
  var xmlRequest = new XMLHttpRequest();
  xmlRequest.open('POST', BaseURL, true);
  var params = '{"cmd":"' + command + '", "key":"' + key + '"}';
  var MAXIMUM_WAITING_TIME = 5000;
  xmlRequest.onreadystatechange = function () {
    if (this.readyState == 4) {
      clearTimeout(xmlTimer);
      if(this.status == 200){
        console.log(this.responseText);
      }else{
        console.log('ERROR = ' + this.status + ' ' + this.statusText);
      }
    }else{
      if(this.status != 200){
        console.log('ERROR = ' + this.status + ' ' + this.statusText);
      }
    }
  };
  xmlRequest.send(params);
  var xmlTimer = setTimeout(function() {
    xmlRequest.abort();
    console.log('ERROR = timeout');
  }, MAXIMUM_WAITING_TIME);
</script>
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Example 3: GET - xhr

```
<script language='JavaScript' type='text/javascript'>
  var controllerIP = '172.30.0.220'; *change this to the same IP address as the SDVoE Director controller
  var BaseURL = 'http://' + controllerIP + '/api/command/';
  var key = '123xyz'; //replace this with the generated security key
  var command = '123xyz'; //replace with an DIRECTOR API command
  var MAXIMUM_WAITING_TIME = 5000;
  var btn2xhr = new XMLHttpRequest();
  btn2xhr.open('GET', BaseURL + '<command>/' + key);
  btn2xhr.onreadystatechange = function () {
    if (this.readyState == 4) {
      if (this.status == 200) {
        document.getElementById('Text0').innerHTML = this.responseText;
      } else {
        document.getElementById('Text0').innerHTML = 'ERROR = ' + this.status + ' ' + this.statusText;
      }
    } else {
      if (this.status != 200) {
        document.getElementById('Text0').innerHTML = 'ERROR = ' + this.status + ' ' + this.statusText;
      }
    }
  };
  btn2xhr.send(null);
  var btn2xhrTimer = setTimeout(function() {
    btn2xhr.abort();
    document.getElementById('Text0').innerHTML = 'ERROR = timeout';
  }, MAXIMUM_WAITING_TIME);
</script>
```

Example 4: IFTTT – Webhooks – POST

URL

http://<controllerIP>/api/command

Method

POST

Content Type

application/json

Body

{"cmd":"<DIRECTOR_API_COMMAND>","key":"<KEY>"}

Example 5: IFTTT – Webhooks – GET

URL

http://<controllerIP>/api/command/<DIRECTOR_API_COMMAND>/<KEY>

Method

GET

Content Type

text/plain



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Example 6: bt.tn – Auxiliary HTTP request GET

URL=<controllerIP>/api/command/<DIRECTOR_API_COMMAND>/<KEY> port=80

**Note: All spaces in the <DIRECTOR_API_COMMAND> must be url-encoded as %20.*

See Percent-encoding chart below:

!	#	\$	&	'	()	*	+	,	/	:	;	=	?	[]
%21	%23	%24	%26	%27	%28	%29	%2A	%2B	%2C	%2F	%3A	%3B	%3D	%3F	%5B	%5D
space	'	%	-	.	<	>	\	^	_	`	{		}	~	@	
%20	%22	%25	%2D	%2E	%3C	%3E	%5C	%5E	%5F	%60	%7B	%7C	%7D	%7E	%40	

Example 7: bt.tn – HTTP POST

HTTP URL - Specify URL

http://<controllerIP>/api/command

HTTP METHOD

POST

ARGUMENTS - application/json

```
{"cmd":"< DIRECTOR_API_COMMAND>","key":"<KEY>"}
```

Appendix B – Preset Logic

Basic if else logic can be applied within a preset to allow you to build some *smarts* into your system. All **get** commands can be used as an expression.

The following syntax applies:

```
if (something) {  
    <do_this>  
    ...  
} elseif (something_else) {  
    <do_this>  
    ...  
} else {  
    <do_this_instead>  
    ...  
}
```

Within presets only, the following commands can be used to manipulate strings:

- substr(<string>,<start>,<optional_length>)
- trim (<string>)

Substr() will extract a string from a string.

Trim() will remove any HEX after the last ASCII character, like terminators <cr> and <lf> from the end of the string.

Strings can be extracted with command **substr()** as follows:

```
substr(<string>,<start>,<optional_length>)  
  
set var myVar "123456"  
set ui_label myUI lbl01 text substr(get var MyVar,1)  
    lbl01 text = "123456"  
set ui_label myUI lbl01 text substr(get var MyVar,3,2)  
    lbl01 text = "34"
```

Strings can be trimmed of trailing hexadecimal with a **trim()** command as follows:

```
if (trim(send tcp 172.30.10.141 4998 "setstate,1:1,1\x0d" reply) == "state,1:1,1"){  
    set ui_label AT01 lbl01 text "good"  
}else{  
    set ui_label AT01 lbl01 text "bad"  
}  
  
set var MyVar send tcp 172.30.10.141 4998 "setstate,1:1,1\x0d" reply  
if (trim(get var MyVar) == "state,1:1,1"){  
    set ui_label AT01 lbl01 text "good"  
}elseif (trim(get var MyVar) == "state,1:1,0" {  
    set ui_label AT01 lbl01 text "bad"  
} else {  
    set ui_label AT01 lbl01 text "error"  
}  
  
set var MyVar trim(send tcp 172.30.10.141 4998 "setstate,1:1,1\x0d" reply)  
if (get var MyVar == "state,1:1,1"){  
    set ui_label AT01 lbl01 text "good"  
}elseif (get var MyVar == "state,1:1,0" {  
    set ui_label AT01 lbl01 text "bad"  
} else {  
    set ui_label AT01 lbl01 text "error"  
}
```

Within presets only, the following variables can be used to identify buttons and devices:

- <<**button_name**>> which identifies the button pressed by name
Note: If the button does not have a name then "<<button_name>>" will be the result
- <<**encoder**>> which identifies the Encoder by name associated with selected Split Encoder button
- <<**decoder**>> which identifies the Decoder by name associated with selected Split Decoder button

Appendix B – Preset logic continued...

The following **get** commands will return a **string** value that can be used with:

- == (equal to)
- != (not equal to)

- get audio_source
- get devices
- get edid
- get scaler <encoder_device_name> all
- get status
- get var
- get ver
- get video <encoder_device_name> all
- get joins
- get ui_button

Example 1:

```
if (get joins Decoder1 == Encoder1) {  
    join all Encoder2 Decoder1  
} else {  
    join all Encoder1 Decoder2  
}
```

Example 2:

```
if (get audio_source Decoder1 != hdmi) {  
    set audio_source Decoder1 hdmi  
}
```

Example 3:

```
if (get var MyVar == "<string>") {  
    <do_this>  
} else {  
    <do_this_instead>  
}
```

Example 4:

```
if (get ui_button UIexample btn01 position == "down") {  
    send gc 172.30.20.129 4998 setstate,1:1,1  
}
```

Appendix B – Preset logic continued...

The following **send** commands will return a **string** value that can be used with:

- == (equal to)
- != (not equal to)
- send tcp > feedback = reply
- send serial > feedback = reply
- send gc > (port 4999 / 5000) feedback = reply
- send gc > (port 4998) command get

Example 1:

```
if (send tcp 169.254.1.70 4998 "setstate,1:1,1\x0D" reply == "state,1:1,1\x0D") {  
    <do_this>  
} else {  
    <do_this_instead>  
}  
  
if (trim(send tcp 169.254.1.70 4998 "setstate,1:1,1\x0D" reply) == "state,1:1,1") {  
    <do_this>  
} else {  
    <do_this_instead>  
}
```

Example 2:

```
if (send gc 169.254.1.70 4999 "My String" reply == "This String") {  
    set ui_button UIexample btn01 position down  
}
```

Example 3:

```
if (send gc 169.254.1.70 4998 getstate,1:1 == "state,1:1,1\x0D") {  
    set ui_button UIexample btn01 position down  
}  
  
if (trim(send gc 169.254.1.70 4998 getstate,1:1) == "state,1:1,1") {  
    set ui_button UIexample btn01 position down  
}
```

Example 4:

```
if (send serial Decoder1 "My String" reply == "This String") {  
    set ui_button UIexample btn01 position down  
}
```


Appendix B – Preset logic continued...

The following **get** commands will return an **integer** value that can be used with:

- == (equal to)
- != (not equal to)
- < (less than)
- > (greater than)

- get preferred
- get rotation
- get scaler <encoder_device_name> width
- get scaler <encoder_device_name> height
- get scaler <encoder_device_name> fps
- get video <encoder_device_name> width
- get video <encoder_device_name> height
- get video <encoder_device_name> fps
- get video <encoder_device_name> sm
- get video quality
- get volume

Example:

```
if (get rotation Encoder1 != 0) {  
    set rotation Encoder1 0  
} else {  
    join all Encoder1 Decoder1
```

The following **get** commands will return a **boolean** value that can be used with:

- not

- get display_status
- get video_mute
- get video_status
- get var

Example 1:

```
if (get video_status Encoder1) {  
    join all Encoder1 Decoder1  
} elseif (get video_status Encoder2) {  
    join all Encoder2 Decoder1  
}
```

Example 2:

```
if not (get video_status Encoder1) {  
    join all Encoder2 Decoder1  
}
```

Appendix B – Preset logic continued...

The following **send** commands will return a **boolean** value that can be used with:

- not
 - send tcp > feedback = none, equals or contains
 - send gc > feedback = none, equals or contains

Example 1:

```
if (send tcp 172.30.20.129 4998 "setstate,1:1,1\x0D" contains "setstate,1:1,1") {  
    set ui_button UIexample btn01 position down  
} else {  
    set ui_button UIexample btn01 position up  
}
```

Example 2:

```
if not (send gc 172.30.20.129 4998 setstate,1:1,1) {  
    <do something>  
} else {  
    <do something else>  
}
```



DIRECTOR RAIN

COMMAND GUIDE V1.3.6

Appendix B – Preset logic continued...

Multiple conditional statements can be included using "&&" (and) as well as "||" (or).

```
if (get video_status Encoder1 && get video_status Encoder2) {  
    join all Encoder1 Decoder1  
    join all Encoder2 Decoder2  
}  
  
if (get video_status Encoder1 || get video_status Encoder2) {  
    join all Encoder1 Decoder1  
    join all Encoder2 Decoder2  
}
```

Split button Decoder state 1 preset logic to display connected Encoder on Decoder buttons or labels.

```
set ui_button buttons <<button_name>> text both <<encoder>>  
  
if (get ui_button buttons split4 position == down) {  
    set ui_label buttons lb01 text <<encoder>>  
}  
  
if (get ui_button buttons split5 position == down) {  
    set ui_label buttons lb02 text <<encoder>>  
}  
  
if (get ui_button buttons split6 position == down) {  
    set ui_label buttons lb03 text <<encoder>>  
}  
  
if (get ui_button buttons split7 position == down) {  
    set ui_label buttons lb04 text <<encoder>>  
}
```

Split button Decoder state 2 preset logic to clear connected Encoder on Decoder buttons or labels.

```
set ui_button buttons <<button_name>> text both None  
  
if (<<button_name>> == split4) {  
    set ui_label buttons lb01 text None  
}  
  
if (<<button_name>> == split5) {  
    set ui_label buttons lb02 text None  
}  
  
if (<<button_name>> == split6) {  
    set ui_label buttons lb03 text None  
}  
  
if (<<button_name>> == split7) {  
    set ui_label buttons lb04 text None  
}
```

Using a variable as string to store a send tcp/serial command result string:

```
set var myVar send tcp 10.1.1.10 6970 "{status}\x0D" reply  
  
if (get var myVar == "option1") {  
    <do something>  
} elseif (get var myVar == "option2") {  
    <do something>  
} else if (get var myVar == "option3") {  
    <do something>  
} else {  
    <do something>  
}  
  
set ui_label myControl lbl01 text get var myVar
```

Using a variable as boolean to set/store a button state:

```
set var myVar false  
  
if (get var Test01) {  
    set ui_button myControl btn01 position up  
} else {  
    set ui_button myControl btn01 position down  
}
```