

CueServer™ User's Manual

Software Version 4.1



Interactive
Technologies

Distributor:



Phone: 407-857-8770

Fax: 407-857-8771

Email: sales@techni-lux.com

www.techni-lux.com

Interactive Technologies, Inc.

5040 Magnolia Creek Drive

Cumming, GA 30028 USA

Phone: 678-455-9019

Fax: 678-455-9071

Email: info@interactive-online.com

support@interactive-online.com

Web: <http://www.interactive-online.com/>

CueServer, CueScript and the Interactive Technologies logo are trademarks of Interactive Technologies, Inc. Apple, iPod and iPhone are registered trademarks of Apple, Inc. All other trademarks referenced in this document are the property of their respective owners.

The CueServer firmware, design and documentation are copyrighted by Interactive Technologies, Inc. The firmware used in CueServer embodies valuable trade secrets proprietary to Interactive Technologies, Inc. and is licensed, not sold, and may not be duplicated in any way.

Specifications subject to change without notice.

Copyright © 2003-2010 Interactive Technologies, Inc. All rights reserved worldwide.

Printed in the United States of America.

Revised 2/18/2010

Table of Contents

Introduction	1
A Tour of CueServer's Main Components	2
Web Server	2
Cues	3
CueScript	3
Hardware Interfaces	4
LCD Display	4
Function Buttons	5
Ethernet	5
DMX-512	5
Serial	6
MIDI	6
Contact Closure Inputs	6
Digital Outputs	6
CueServer on Mobile Web Browsers (iPhone® and iPod® Touch)	7
Cues	9
An Overview of Cues	10
Recording Basic Cues	10
Playing Back Basic Cues	13
Where To Go From Here	14
Cue Types	15
Normal Cues	15
Streaming Cues	16
Working with Cues	17
Viewing Cues in the Cue List	17
Recording Cues	18
Adding Cues Using the Cue List Page	18
Adding Cues Using CueScript	20
Recording Only Selected Channels	21
Recording An Empty Cue	22
Recording a Streaming Cue	23
Playing Cues	27
Manual Cue Playback	27
Automated Cue Playback	28
Viewing Cue Playback	29
Editing Cues	31
Creating Loops and Chases	32

Playback Faders	35
Playback Fader Basics	35
Cue Playback	35
Manual Channels	36
Independent Timing	38
Split Fade Times	38
Disabling Timing	39
Advanced Playback Features	40
Combine Modes	41
Merge Mode	41
Override Mode	42
Scale Mode	43
Playback Submasters	44
Channel Enable Filters	44
An Example	44
Uses for Channel Disabling	46
Channel Parking	46
Uses for Channel Parking	49
Input Disabling	49
Triggers	51
Button Triggers	52
CueScript Function	55
CueScript (State-Driven) Function	56
Toggle Channel Function	57
Toggle Preset Function	59
Contact Closure Triggers	61
Timer Event Triggers	62
Timecode Triggers	64
Timecode Playback	65
Internal Timecode Generation	65
DMX Input Triggers	66
Event Ranges	67
Submaster Control	68
System Event Triggers	69
CueScript Command Language	71
Using CueScript	71
Executing Commands	71
Automation with Events and Actions	71
CueScript Basics	72
Command Syntax	72

Command Context	72
Levels	73
Percentage	73
Decimal	73
Hexadecimal	74
Binary (On/Off)	74
Variables	75
Assigning and Using Variables	75
System Variables	76
System Functions	76
CueScript Commands	77
At	77
At (with array value)	78
At (with relative value)	79
At Cue	80
At Input	81
At Playback	82
Break	83
Button	84
Channel	86
Clear	87
Contact	87
Cue	88
Delete	89
Device	90
Disable	91
Enable	92
Fade	93
Fixture	94
Follow	95
Follow Clear	95
Go	96
Group	97
HTP (deprecated)	97
IF ... THEN ... [ELSE ...] ENDIF	98
Input [Enable Disable]	99
Input Update	99
Join	100
Link	100
Log	101
LTP (deprecated)	101

Macro	102
Merge	103
Next	104
Off	105
On	105
Output	106
Override	107
Park	108
Playback	109
Previous	110
Record Cue	111
Record Group	112
Record Stop	112
Record Stream	113
Release	114
Reset	115
Scale	116
Self	117
SMPTE	118
Start	119
Station	119
Stop	120
Time	121
Toggle	122
Unjoin	123
Unpark	123
Update Cue	124
Wait	125
+ (and)	126
- (except)	126
> (through)	127
* (wildcard)	127
" " * (command broadcast)	128
" "~ (string store)	129
" " = (variable assignment)	131
{{ }} (variable substitution)	132
; (semicolon)	133

DMX-512 135

DMX Output	135
DMX Input	135

Contact Closure Inputs	139
Contact Closure Connections	139
Contact Closure Events	140
Binary Cue Select Feature	141
About the Binary Cue Select Feature	141
Enabling the Binary Cue Select Feature	142
Digital Outputs	145
Controlling the Low-Voltage Outputs	145
Low-Voltage Output Connections	146
Serial Port	149
Serial Port Settings	149
Receiving Commands via the Serial Port	151
Sending Serial Strings	152
Ethernet	153
Receiving Messages via UDP Packets	153
Sending Messages via UDP Packets	154
MIDI	155
MIDI Input	155
MIDI Timecode	155
Programming Timecode Events	156
MIDI Reset Messages	158
CueServer System Exclusive Messages	158
Note On Messages	159
Note Off Messages	159
MIDI Output	160
Sending MIDI Commands	160
Appendix A: CueScript Command Summary	163
Appendix B: System Variables	171
Appendix C: System Functions	173
Appendix D: Warranty Information	175

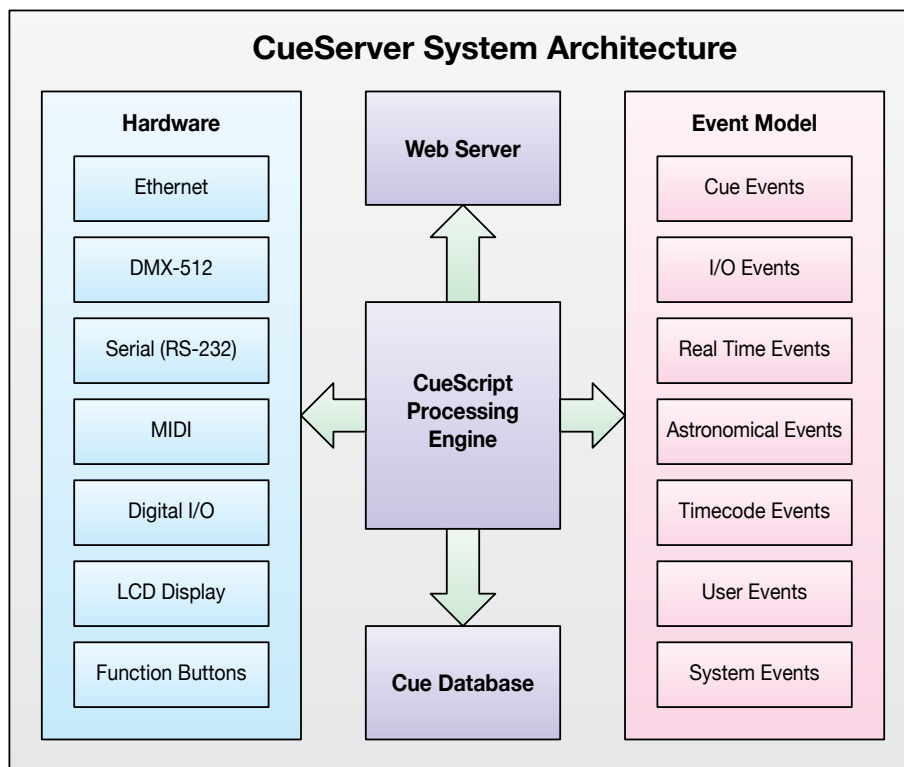
Introduction

CueServer is an advanced lighting control interface and playback device that leverages the power and ease-of-use of web-based operation and setup with robust show control features and a full complement of connections to external devices.

Use CueServer to play back a simple programmed show to DMX controlled lighting fixtures or set up CueServer to orchestrate a complex system of lighting playback, button inputs, wireless tablet controls, timers, astronomical events, MIDI controls, timecode triggering, serial devices, relay outputs and much more. CueServer easily scales from simple tasks to complex requirements without making CueServer difficult to understand, program or operate.

At the core of CueServer is the extremely flexible CueScript scripting language. This is the glue that ties together all of the functional units of the CueServer hardware, web server, cue database and event model.

Using the highly flexible model of Events and Actions, CueServer may be configured in nearly limitless ways, allowing the programmer to fully customize the function of CueServer and it's various I/O modules. For example, make buttons recall lighting sequences, have the LCD display the system's status, integrate with RS-232 controlled devices, synchronize timecode and other sequencers via MIDI, allow external PCs to trigger events via web pages, provide emergency backup lighting logic, have shows start at sunset, perform different tasks on different days -- do as much or as little as a project requires.



A Tour of CueServer's Main Components

Web Server

CueServer incorporates an “embedded” web server. This means that nearly any Windows PC or Mac OS computer with a web browser (and even iPhones, touchscreen iPods, some PDAs and cell phones) can connect to CueServer and view it's internally generated web pages.

CueServer uses it's web pages to perform a variety of tasks:

- **System Programming**
Use dynamic web pages to set up lighting cues, I/O logic, timing, events and much more.
- **Playback Monitoring**
See what's going on. View the live stage outputs and console controls.
- **User Input**
Create a user-interface for the end-user via web pages.

The screenshot displays the CueServer web interface. The top navigation bar includes tabs for Main, Console, Cue List, Groups, Stage, Keypad, and Triggers. The main content area is divided into two sections: Playback Controls and Stage View.

Playback Controls: This section shows four playback channels. Each channel has buttons for Go, Select, and Release, along with fields for Cue, Next Cue, Fade Time, Follow Time, Link Cue, Mode, and Status.

Playback	Cue	Next Cue	Fade Time	Follow Time	Link Cue	Mode	Status
Playback 1*	213	214	4.0 sec	10.0 sec	-	HTP	Running
Playback 2	-	-	Immediate	None	-	HTP	Running
Playback 3	-	400	Immediate	None	400	-	Running
Playback 4	-	203	10.0 sec	10.0 sec	-	-	-

Stage View: This section shows a grid of stage outputs. The grid is organized into columns representing different stage areas (1-100, 101-200, 201-300, 301-400, 401-500, 501-512). Each cell in the grid contains a numerical value or a status indicator (e.g., FL, 91, 33, 36, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60).

Command Line: A text input field at the bottom of the Stage View section contains the command `c30>39+50>59` and a Submit button. Below the input field, the text `Channel 30 > 39 + 50 > 59` is displayed.

CueServer's internal web pages are highly dynamic, using the latest in web technologies, to provide “live” information updating without requiring the pages to reload.

Cues

CueServer runs cues. A cue is an object that may contain lighting levels, timing information, and automation tasks. When a cue is executed, CueServer outputs the lighting levels and performs any actions associated with the cue.

Cues are further divided into two types. CueServer handles **Normal** cues and **Streaming** cues. Both cue types contain timing information and automation tasks, but they differ in the type of DMX lighting levels they contain.

Normal cues contain a snapshot of a lighting look that is to be applied to the lighting output of CueServer, combined with fade and follow times. When the cue is executed, the lighting output transitions to the recorded look. Fade times, follow times and links can also be specified for normal cues to build transitions, chases and loops.

Streaming cues contain a streamed recording of the lighting data from another console or controller. For this type of cue, CueServer stores every piece of incoming data from a console for playback later, allowing an external device to be used to compose a complex light show and then capture it into CueServer. When a streaming cue is executed, CueServer will replay the captured light show segment. This segment can be as simple as a slow rainbow effect for multiple fixtures or as complex as a moving-light “rock-and-roll” show. Streaming cues and normal cues may be combined in the same Cue List to produce a complete show.

CueServer has four separate playback faders that coordinate the execution of cues from a central Cue List database. This allows for as many as four separate time lines playing at once, either affecting the same lights and I/O devices or affecting separate groups of lights and devices.

CueScript

The most central part of CueServer is its CueScript command language. All of CueServer’s automation capabilities come from the application of CueScript commands to a wide range of available events in the CueServer system.

Simple light show setups need very little CueScript coding (if at all) to make CueServer perform its tasks. As the need for automation and/or integration into other systems expands, CueScript is the language that ties together all of CueServer’s hardware modules and events.

In brief, CueScript commands are very similar to those found on many lighting consoles. For example, “Channel 1 At 50%”, “Cue 7 Go”, “Button 3 Enable”, “Group 2 Release”, etc. CueServer makes many extensions to this familiar lighting control language to be able to handle some more advanced system automation tasks.

CueScript is used in two distinct ways while programming and operating CueServer, either as live commands, or as event actions.

When CueScript is being used live, the operator is typing commands into the CueServer command line and hitting Enter. For instance, if the operator wants to run a cue, they might type “Cue 34 Go” on the command line. These commands are typically entered on the command line field of many of


CueServer's web pages, but they may also be sent to CueServer through CueServer's serial port, Ethernet port or MIDI port. Since CueServer is always listening for CueScript commands on all of its external I/O ports, it's easy to allow other external devices to remotely control CueServer.

When CueScript is being used as an event action, this means that the system programmer has provided a series of CueScript commands that CueServer should execute when a specific event occurs in the system. Many user-definable events exist in CueServer, such as when a cue executes or when a button is pressed, or a contact closed, or the unit is powered on, or a specific time of the day occurs, etc. For instance, the programmer might choose to instruct CueServer to start running a show, starting with Cue 100, when CueServer turns on. The programmer would enter "Cue 100 Go" in the field for the System Startup Event.

Hardware Interfaces

CueServer has a wide complement of interfaces that connect CueServer to the outside world. These include an LCD Display, Function Buttons, Ethernet, DMX, Serial, MIDI, Contact Closure Inputs and Digital Outputs. Depending on the model of CueServer you have, some of the following features may not be present.

LCD Display



```
CueServer           May 1, 2006   1:42:53 PM
TC: 00:01:23.45    IP:192.168.123.4
```

The rack mounted CueServer Pro (CS-800) has a large 80-character LCD display that serves two primary functions. This display provides status information about the current operation of CueServer and it is part of a Menu system that allows the user to make adjustments to several of CueServer's parameters.

In its normal mode, the LCD Display is showing the current running status of CueServer. It typically shows the unit's name, the current date and time and the unit's IP Address. Which status information is displayed and where is completely configurable.

Also, through the use of CueScript commands, CueServer can write custom messages and labels to the LCD display, allowing the system programmer to customize the display for specific applications and make CueServer easier to use for the end-user:



```
System Ready       May 1, 2006   1:42:53 PM
Backup Mode Enabled - Press 1 to Disable
```

In the Menu mode (accessible from CueServer's front-panel menu buttons), the LCD Display is used to allow the end-user to modify several of CueServer's parameters, including the Date, Time, Network settings, and more.

Function Buttons

On the front-panel of the rack mounted CueServer Pro are 8 user-definable function buttons, each with its own LED indicator. Each of these buttons may be configured with its own CueScript action or other pre-defined behavior, allowing the buttons to perform nearly any action possible on CueServer. The buttons may start shows, run cues, send commands to external devices, modify the LCD Display, toggle loads or presets and more. The indicator LEDs can be configured as well, allowing very flexible feedback options for each button.

Additional buttons (up to 512 in total) can be added to CueServer via the CueStation Network Hub, which can be attached to CueServer by either it's Ethernet or Serial port.

Ethernet

The Ethernet port on all CueServers are used for several important tasks. Most importantly, the primary way of communicating with CueServer is via its built-in web server. Use a computer with a web browser to connect to CueServer and view the dynamic web pages used to control, program and monitor all of CueServer's operations.

The Ethernet port can also be used in a network environment to allow external devices to send CueScript commands to CueServer, and CueServer can send arbitrary messages to other external network devices, allowing for complete Ethernet-based integration of CueServer with other systems.

CueServer responds to HTTP, TCP, UDP, XML, Telnet and several other protocols for a wide variety of system integration options.

DMX-512

DMX-512 is a lighting control protocol used for professional stage, studio and entertainment lighting fixtures. CueServer has extensive roots in it's ability to send and receive DMX signals. CueServer supports static, fading and streaming cues that can control nearly any DMX-based lighting fixture available. CueServer's DMX input port can pass data through to it's output, capture looks or streaming data and automatically provide backup scenarios -- all completely controlled by CueScript events and actions.

Serial

CueServer provides a standard RS-232 port for communicating with external devices, such as video playback systems, home automation systems, security panels, HVAC controls, building automation packages and much more. Some CueServer models have two serial ports. External devices may send CueScript commands to CueServer, and CueServer can send any arbitrary data out to control external devices.

Using the serial port, CueServer can be used as an intelligent bridge between a home automation system and DMX compatible lighting fixtures (such as color-changing LED lamps), or CueServer may take it's commands from a show control system or security panel. CueServer's simple to understand commands make it easy to integrate with a wide variety of other devices with serial ports.

MIDI

The rack mounted CueServer Pro (CS-800) has three MIDI ports (input, output and through) for interfacing with MIDI equipment and/or for receiving Timecode.

CueServer responds to many of the standard MIDI messages, allowing external devices to easily control many aspects of CueServer. Also, external devices may send special System Exclusive messages to CueServer that contain CueScript commands - allowing external devices to exert full remote control of CueServer.

CueServer can send nearly any MIDI command out to external MIDI devices in response to any internal event action, allowing very flexible control of external devices.

Timecode can be received into CueServer by way of MIDI-Timecode (MTC) to allow CueServer playback to be perfectly synchronized with an external video or audio program. SMPTE Timecode may be connected to CueServer by using a third-party SMPTE-to-MTC converter (such as the J.L. Cooper PPS-2).

Contact Closure Inputs

All CueServer models provide 8 separate contact closure inputs, which can trigger CueScript events. These inputs allow a wide array of external switch-type devices to be connected to CueServer and which can trigger all types of events.

Button stations, motion detectors, keyswitches, floor-sensors, photocells and much more can be attached to CueServer, allowing these devices to control or modify the behavior of CueServer.

Digital Outputs

All CueServer models provide 8 separate low-voltage digital outputs, which can be completely controlled by CueScript commands. Relays, LEDs, sounders and more can be directly connected to CueServer's digital outputs to allow CueServer to coordinate the operation of these devices with a show running in CueServer.

CueServer on Mobile Web Browsers (iPhone® and iPod® Touch)

An exciting feature of CueServer is that it is fully operational from advanced wireless web browsing devices (such as the Apple® iPhone® and the Apple® iPod Touch®).

Because of CueServer's strong roots with open-standards and web-based architecture, most any mobile device that contains a "real" web browser can effortlessly connect to CueServer and view its user-interface as if it were a world-class application running natively on that platform.

Use CueServer as the engine that can provide advanced lighting control to your iPhone or touch-screen iPod. These handheld devices can be used for wireless remote focus, macro triggering, DMX troubleshooting, and virtually any other lighting control task. CueServer acts as a complete web-based lighting console -- and now you can bring all of these features to your favorite handheld device.



Cues

Central to CueServer's design is the **Cue**. A cue is an object that may contain lighting levels, timing information, and/or automation tasks. When a cue is executed (or run), CueServer outputs the lighting levels stored in the cue and performs any additional automation actions that the cue may specify.

When programming CueServer, one can think of running a cue when something needs to change. For instance, a cue might turn on the lights on a stage or studio, or a cue might begin to slowly change the color of the front of a building to a combination of Red and Orange, or a cue might play back a sequence of complex moving light operations, or a cue might close the curtains in a boardroom and turn on a projector. If a cue contains DMX lighting levels, those levels will be sent to the lighting fixtures and if a cue contains CueScript actions for other automation tasks, those actions will be taken.

Cues are assigned **Cue Numbers** and are always referenced by their cue number when playing cues, looping to other cues, recording cues or editing cues. Cues may be assigned any number from 0.1 to 6499.9.

Cues are divided into two main types. CueServer handles **Normal** cues and **Streaming** cues. Both cue types contain timing information and automation tasks, but they differ in the type of DMX lighting levels they contain.

CueServer has up to four separate **Playback Faders** (depending on model) that coordinate the execution of cues from a central Cue List database. This allows for as many as four separate time lines playing at once, either affecting the same lights and I/O devices or affecting separate groups of lights and devices. The database of cues in a show can be viewed and edited via the **Cue List** web page. The state and operation of the playback faders can be viewed using the **Console** web page. For additional features and description of Playback Faders, see the Playback Faders chapter.

Cues can be executed manually by entering a CueScript command on many of CueServer's web pages such as "Cue 1 Go". CueServer can also execute cues when any of CueServer's event-based objects (buttons, timers, contact closures, other cues, etc.) are activated after they have been programmed with a CueScript action (such as "Cue 1 Go"). Also, cues may be executed automatically by the auto-follow and linking mechanisms in the Cue List.

Cues may be recorded in many ways, including the built-in web interface or by CueScript commands, by manually entering DMX levels, by taking a snapshot of the DMX input, or by recording an entire stream of changing DMX input levels.

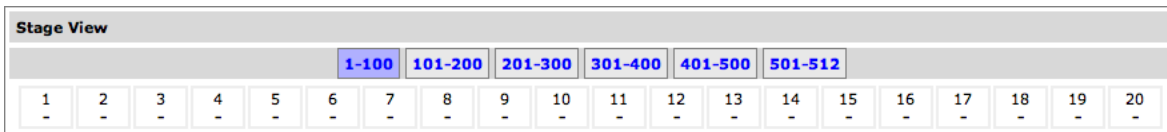
All of these features and concepts are described in greater detail in the following sections.

An Overview of Cues

This section will provide a quick introduction to using cues in CueServer. We will build a couple of cues and play them back. The sections that follow this brief introduction will describe all of the details of using cues in greater detail.

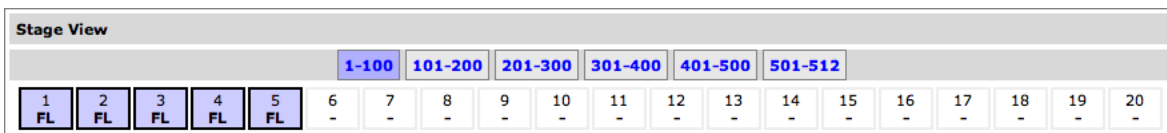
Recording Basic Cues

Start by viewing the Stage View web page (the view of CueServer’s lighting output channels) by clicking on the **Stage** button in the navigation bar:



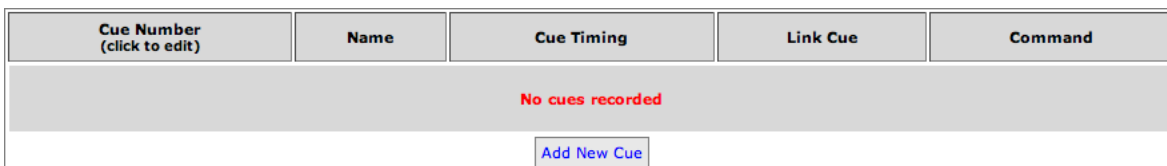
Turn on some lights by setting channel levels. Do this by entering the command:

```
Channel 1>5 At FL
```



Next, we will record this look as a cue.

Change to the Cue List web page by clicking on the **Cue List** button in the navigation bar. If you are starting with an empty CueServer, it should not contain any cues yet:



To record the current lighting look as a cue, click on the **Add New Cue** button. The New Cue web page will appear:

Record New Cue

Cue Type Standard Streaming

Cue Number

Cue Name

Fade Time

Follow Time

Link Cue
only needed to jump to non-sequential cue number

Automation Details

Command String

[Record Cue](#) [Cancel](#)

Use this web page to assign the new cue a cue number, name, fade and follow times, a link and a CueScript action.

For now, we will only assign a name and a fade time. Enter a name like “My First Cue” and a fade time of 5 seconds:

Record New Cue

Cue Type Standard Streaming

Cue Number

Cue Name

Fade Time

Follow Time

Link Cue
only needed to jump to non-sequential cue number

Automation Details

Command String

[Record Cue](#) [Cancel](#)

When finished entering the information, click on the **Record Cue** button.

CueServer will return to the **Cue List** page, displaying the newly recorded cue:

Cue Number <small>(click to edit)</small>	Name	Cue Timing	Link Cue	Command
1	My First Cue	Fade: 5.0 sec, Follow: ---	(none)	

[Add New Cue](#)

Next, return to the **Stage View** web page so we can change the levels to make a new scene:

Stage View																													
1-100					101-200					201-300					301-400					401-500					501-512				
1 FL	2 FL	3 FL	4 FL	5 FL	6 -	7 -	8 -	9 -	10 -	11 -	12 -	13 -	14 -	15 -	16 -	17 -	18 -	19 -	20 -										

Change the lighting levels by entering these commands:

```
Channel 1>8 At 50
Channel 3>6 At 75
```

Stage View																													
1-100					101-200					201-300					301-400					401-500					501-512				
1 50	2 50	3 75	4 75	5 75	6 75	7 50	8 50	9 -	10 -	11 -	12 -	13 -	14 -	15 -	16 -	17 -	18 -	19 -	20 -										

Then, return to the **Cue List** page and click on **Add New Cue** again.

Enter a name for the cue, and set the fade time to 10 seconds, like this:

Record New Cue

Cue Type Standard Streaming

Cue Number

Cue Name

Fade Time

Follow Time

Link Cue
only needed to jump to non-sequential cue number

Automation Details

Command String

When finished, click on the **Record Cue** button. CueServer will return to the Cue List page, displaying the two recorded cues:

Cue Number (click to edit)	Name	Cue Timing	Link Cue	Command
1	My First Cue	Fade: 5.0 sec, Follow: ---	(next)	
2	My Second Cue	Fade: 10.0 sec, Follow: ---	(none)	

Playing Back Basic Cues

Now that we have two cues recorded, we can play them back and watch the result.

First, change to the **Stage View**:

Stage View																			
1-100 101-200 201-300 301-400 401-500 501-512																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	50	75	75	75	75	50	50	-	-	-	-	-	-	-	-	-	-	-	-

Clear all current outputs by entering the following command:

```
Channel 1>8 Release
```

Stage View																			
1-100 101-200 201-300 301-400 401-500 501-512																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

To execute Cue 1, enter the following command:

```
Cue 1 Go
```

After entering this command, Cue 1 will begin playing back. Since it has a 5 second fade associated with it, the channel levels will gradually fade up to the scene recorded into Cue 1:

Stage View																			
1-100 101-200 201-300 301-400 401-500 501-512																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FL	FL	FL	FL	FL	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Next, to execute Cue 2, enter the following command:

```
Cue 2 Go
```

When this command is entered, the channel levels will gradually crossfade to the scene recorded in Cue 2 over 10 seconds:

Stage View																			
1-100 101-200 201-300 301-400 401-500 501-512																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	50	75	75	75	75	50	50	-	-	-	-	-	-	-	-	-	-	-	-

Please note that while the Stage View is displaying the channel levels changing, CueServer is also outputting the DMX lighting signal to the actual lighting fixtures. The update rate of the Stage View data in your web browser is not as fast as the DMX signal, and although the web page might appear to be skipping steps as channels fade up or down, CueServer's DMX signal is updating at 40 Hz and has very smooth crossfades.

Where To Go From Here

Although the example above is somewhat simple, CueServer's cue playback ability is quite flexible and can be extended to perform much more complex playback tasks.

You can use the built-in channel setting functions to set channel levels when recording cues, but you can also record cues by supplying a DMX signal from an external lighting console or controller.

CueServer supports not only the basic static cues demonstrated above (normal cues), but it also supports streaming cues, which are dynamic cues captured from an external lighting console or controller. A complex light show, including special animation, transitions or effects can be accurately captured and played back using CueServer's streaming cues.

Cues support crossfading, split fade times, auto-follow times, linking (for loops), and more, which are all described in the following sections.

Another important feature of cues is that each time a cue begins playing back, the cue can execute optional CueScript statements, which can be used to automate some other CueServer task. For example, a cue might be scripted with CueScript commands to display a prompt on the LCD Display, or to begin flashing an LED on one of the front-panel buttons, or to toggle a digital output, or send a serial string out the serial port, or to begin a different set of cues on another playback fader. The flexibility of this system provides nearly endless possibilities for customizing how CueServer controls its various resources during show playback.

It should also be noted that cues do not necessarily need to have DMX channel information recorded into them. Cues can be created that don't adjust the lighting output at all. Instead these, cues can be used solely for their ability to execute CueScript commands. In this case, a cue could be created that changes the LCD Display and sends a serial port string that causes a projection screen to lower, for example.

Finally, cues can be executed (or run, or played back) in a number of different ways. They can be executed manually by entering a CueScript command (such as `Cue 1 Go`), or you can assign the same CueScript commands to any of CueServer's event-based objects, which allows cues to be executed in response to a button press, timer, contact closure, system event, other cue, or many other triggers.

All of the features necessary to use cues to their full potential are described in the following sections.

Cue Types

CueServer works with two main types of cues, called Normal and Streaming cues. Although they are similar in many ways and can be mixed together in the same cue list, they have specific differences in the way that the DMX levels are stored in them and how they play back.

Normal Cues

Normal cues contain a static snapshot of a lighting look that is to be applied to the lighting output of CueServer, combined with optional fade and follow times. When a normal cue is executed, the lighting output transitions to the recorded look.

If no fade time is specified, the DMX levels in the cue appear immediately at the DMX output. If a fade time has been assigned to the cue, then when the cue is executed, the DMX output of CueServer begins a “crossfade” to the new look.

Fade times are specified in seconds (in 0.1 second increments). Normally, all DMX channels in the cue fade to their new output levels at the same time, but a “split fade time” may be specified that has different fade times for channels that will be rising and channels that will be falling. For example, a split fade time specified as “1 . 5 / 3” indicates that rising channels will fade in 1.5 seconds and falling channels will fade in 3 seconds.

In a normal cue, any combination of the 512 DMX output channels may be recorded into the cue. This allows all channels, or only some channels, or even no channels to be affected by the cue’s playback. Using this feature, a normal cue may affect all lighting output simultaneously, or it may only affect certain channels (such as a particular fixture or a particular fixture’s color, for instance). Normal cues may not have any lighting channels recorded in it at all, allowing the cue to be used solely as a cue for executing other automation tasks.

Each normal cue has an auto-follow time which allows the execution of the cue to automatically trigger the execution of the next cue in the cue list after a specified amount of time. For example, if a cue has a follow time of 5 seconds, then CueServer will automatically advance to the next cue in the cue list 5 seconds after the cue is executed.

Both normal and streaming cues have an optional link cue field, which allows the user to specify the cue number of the cue that should be executed after the current cue. By default, cues always execute in numerical order. This order can be changed by linking a cue to another cue. For example, to create a four-step chase, cue 4 can be linked back to cue 1.

Both normal and streaming cues have a CueScript action field, which allows the user to specify additional automation actions to take when the cue executes. The action field can contain any valid CueScript commands. Use the action field to turn on digital outputs, or flash one of the button LEDs, or to display a prompt on the LCD display or write a string out the serial port, or to begin executing other cues on a different playback fader, and more.

Streaming Cues

Streaming cues differ from normal cues in the way that the DMX lighting data is stored and played back. Streaming cues contain a streamed recording of the lighting data from another console or controller.

For this type of cue, CueServer stores every piece of incoming data from a console for playback later, allowing an external device to be used to compose a complex light show and then capture it into CueServer.

When a streaming cue is executed, CueServer will replay the captured light show segment. This segment can be as simple as a slow rainbow effect for multiple fixtures or as complex as a moving-light “rock-and-roll” show. Streaming cues and normal cues may be combined in the same Cue List to create a sophisticated playback of a combination of cue types.

Streaming cues do not have a user-defined auto-follow time. Streaming cues automatically perform an action (such as following to the next cue, or looping, or stopping) when the end of the data stream has been reached.

Streaming cues also do not have a fade time, as their purpose is to exactly play back a captured sequence of events recorded directly from another controller.

However, streaming cues do have links and CueScript actions just like normal cues in order to specify loops and/or provide additional automation details to the cue.

Working with Cues

Viewing Cues in the Cue List

Use the built-in **Cue List** web page to view a list of cues currently stored in CueServer.

The following example shows CueServer's Cue List page after several cues have been recorded into CueServer's memory. Each cue in the list shows its cue number, name, timing parameters, link and CueScript action.

Cue Number (click to edit)	Name	Cue Timing	Link Cue	Command
1	Start Show	Fade: ---, Follow: 10.0 sec	(next)	"Starting in 10 seconds"~0
10	Step 1	Fade: 1.0 sec, Follow: 2.0 sec	(next)	Macro 1
11	Step 2	Fade: 1.0 sec, Follow: 2.0 sec	(next)	
12	Step 3	Fade: 1.0 sec, Follow: 2.0 sec	10	
20	Gala Event	Stream Time: 00:00:15.0	(next)	Macro 2
99	All Off	Fade: 5.0 sec, Follow: ---	(none)	Button 1+2 Off; "System Off"~0

[Add New Cue](#)

Command Line

[Submit](#)

Last Command:

In this example, Cue 1 is named “Start Show”, has no fade time and a follow time of 10 seconds, no link (which defaults to the next cue) and a CueScript action that writes “Starting in 10 seconds” to the LCD Display.

Cues 10, 11 and 12 make up a 3-step loop (or chase). Each cue in the loop has a fade time of 1 second and a follow time of 2 seconds. This causes each cue to have a 1 second crossfade from the previous cue and a 2 second delay between the beginning of each cue. Cue 12 is linked back to Cue 10, creating the loop. Finally, Cue 10 has a CueScript action to execute Macro 1.

Cue 20 is the only streaming cue in the Cue List. It contains 15 seconds of DMX data that was captured from another lighting console or controller. When this cue is executed, it will execute Macro 2 and the lighting show recorded in it will last for 15 seconds.

Cue 99 has a 5 second fade and does not automatically follow to the next cue. It contains several CueScript actions to take when it executes, which is to turn off the LEDs on buttons 1 and 2, and to write “System Off” to the LCD Display.

Recording Cues

There are several ways to record cues into CueServer, which include using CueServer’s built-in web pages, or by entering CueScript commands. Cues can be recorded from CueServer’s own output or by capturing lighting data from an external lighting console or controller. Cues can be of the normal (static scene) type or they can be a real-time stream capture of a show playing from an external source.

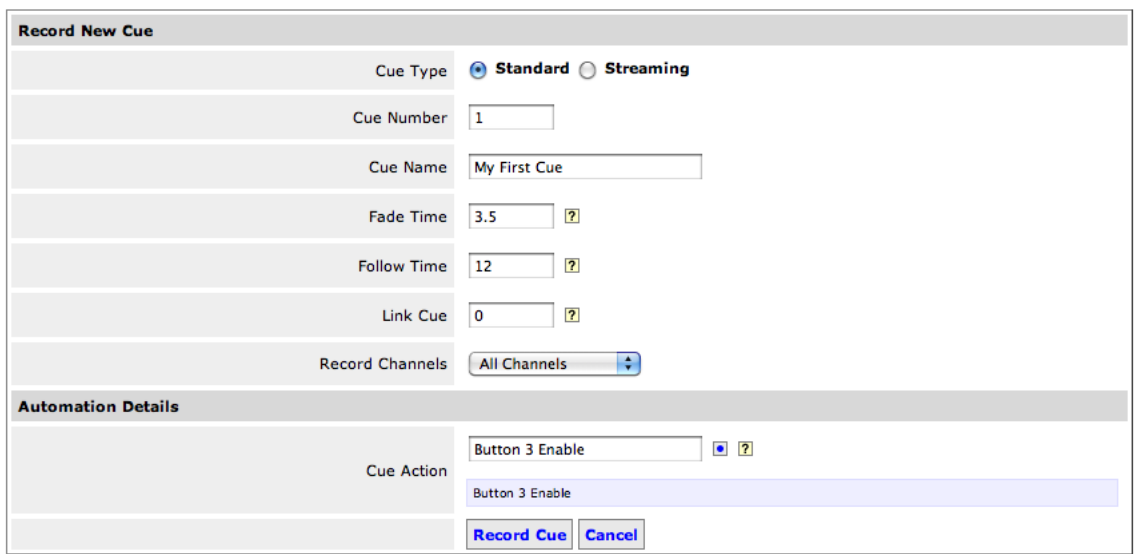
The following sections detail how each of these cue recording features work.

Adding Cues Using the Cue List Page

To add a new cue to the Cue List, click on the **Add New Cue** button:



The new cue web page will appear:

A screenshot of the "Record New Cue" web form. The form is divided into two main sections: "Record New Cue" and "Automation Details".
The "Record New Cue" section contains the following fields:

- Cue Type: Radio buttons for "Standard" (selected) and "Streaming".
- Cue Number: Text input field containing "1".
- Cue Name: Text input field containing "My First Cue".
- Fade Time: Text input field containing "3.5" with a help icon (?).
- Follow Time: Text input field containing "12" with a help icon (?).
- Link Cue: Text input field containing "0" with a help icon (?).
- Record Channels: A dropdown menu showing "All Channels".

The "Automation Details" section contains:

- Cue Action: A text input field containing "Button 3 Enable" with a help icon (?).
- A list of automation actions, with "Button 3 Enable" highlighted in blue.
- Buttons for "Record Cue" and "Cancel".

Use this web page to set up the new cue.

CueType

Choose the desired type of cue. Standard (or Normal) cues are regular crossfading cues with fade times. Streaming cues contain a continuous data stream captured from an external lighting console or controller. If Streaming is chosen, the page will change to the New Streaming Cue page, described in a later section of this chapter.

Cue Number

Specify a number for this cue. All cues are referenced by number. Each cue must have a unique cue number. Any number from 0.1 through 6499.9 may be used.

Cue Name

Each cue may have a user-assigned name. This is for your convenience only.

Fade Time

This is the crossfade time for the cue, in seconds. A time of 0 (zero) will cause the cue's levels to appear at the DMX output immediately when the cue is executed. Any time above zero will cause the cue's levels to gradually "fade" to the recorded look in the specified time.

Any time may be given from 0 to 6500 (about 1.8 hours) in 0.1 second increments.

Split fade times may be specified as well. Split fade times are two times separated by a slash (/), for example, 1.5/3. The time before the slash specifies the fade time for channels that are fading up and the time after the slash specifies the fade time for channels that are fading down.

Follow Time

If a cue is given a Follow Time, it will automatically execute the next cue in sequence after the specified time. A follow time of 0 (zero) means no auto-follow. Use this function to create sequences of cues that automatically step from one to another.

Link Cue

By default, cues always execute in numerical order (1 followed by 2, followed by 3, etc.). If a cue has a Link Cue, then the playback order of which cue comes next is overridden by this field. Use this feature to create loops or other jumps in cue playback order.

Record Channels

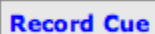
This pop-up menu specifies which channels will be recorded in the newly created cue. The following options are available:

- **All Channels** - All 512 channels are recorded into this cue.
- **Selected Channels** - Only the currently selected channels are recorded into this cue.
- **None** - No channels are recorded into this cue. The cue will be "empty".

Cue Action

This is the optional CueScript action that CueServer will execute when this cue executes. Any string of valid CueScript commands can be placed in this field. Use this field to have the cue run commands that can perform nearly any automation task, such as illuminating button LEDs, toggle digital outputs, output serial strings, display prompts on the LCD, run cues in the other playback faders and more. Unless otherwise specified, the same playback fader that was used to run the cue will be the default playback for CueScript commands.

After the parameters for the new cue have been entered, click the Record Cue button:

A rectangular button with a blue border and the text "Record Cue" in blue font.

When the cue is recorded, the current DMX levels being output by CueServer will be recorded into the cue. This can be the channel levels being output directly by CueServer, or may be the DMX data being input into CueServer (from an external console) and being passed through to CueServer's DMX output, or it may be a combination of both.

Adding Cues Using CueScript

Another way to record a cue is to enter a command using the CueScript language.

From any active command line within CueServer, enter a command such as:

```
Record Cue 1
```

The screenshot shows the CueServer interface. At the top is the 'Stage View' section, which contains a grid of 100 DMX channels (numbered 1 to 100) arranged in 5 rows and 20 columns. The grid is divided into six sections: 1-100, 101-200, 201-300, 301-400, 401-500, and 501-512. The first row (channels 1-20) is highlighted in blue and contains 'FL' for channels 1-10 and '-' for channels 11-20. The second row (channels 21-40) contains '-' for all channels. The third row (channels 41-60) contains '-' for channels 41-44, and '33' for channels 45-55. The fourth row (channels 61-80) contains '-' for all channels. The fifth row (channels 81-100) contains '-' for all channels. Below the Stage View is the 'Command Line' section, which has a text input field containing 'Record Cue 1' and a 'Submit' button. Below the input field, the text 'Record Cue 1' is displayed in a light blue box.

The command “Record Cue 1” will create a new cue with cue number 1 by capturing the current DMX Output of the CueServer.

When a cue is recorded from a CueScript command, the cue also captures the current fade time, follow time, and link.

For example, to record a cue with a fade time of 3 seconds, the following command may be entered:

```
Time 5; Record Cue 2
```

This records a new Cue 2 that has a fade time of 5 seconds.

Or, additional parameters may be specified:

```
Time 10; Follow 15; Link 1; Record Cue 3
```

These commands record a new Cue 3 with a 10 second fade time, a 15 second follow, linked back to Cue 1.

Recording Only Selected Channels

By default, cues store information for all 512 of the available channels and when played back they affect all of the channel outputs.

However, cues may be recorded to only contain a subset of channels, so that when they play back, they only affect a certain group of lights, fixtures or parameters. This is useful when one wants to record a cue that only operates on a particular fixture or that only changes the color of a fixture, etc.

When recording a cue using a CueScript command, the “\$” character is added before the cue number to specify that **only the selected channels** should be recorded into the cue.

For example:

Starting with the following levels being output:

Stage View																													
1-100					101-200					201-300					301-400					401-500					501-512				
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20										
33	33	33	33	33	33	33	33	33	33	33	33	-	-	-	-	-	-	-	-										
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40										
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-										

Note that channels 1 through 12 are being out at 33%, but only channels 4 through 9 are selected.

Then the following command is executed:

```
Record Cue $7
```

This command causes the levels of only channels 4 through 9 to be stored in Cue 7.

Later, if CueServer is outputting the following channels (1 through 20 at 75%):

Stage View																													
1-100					101-200					201-300					301-400					401-500					501-512				
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20										
75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75	75										
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40										
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-										

And Cue 7 is executed (by a command like “Cue 7 Go”), only channels 4 through 9 will be affected:

Stage View																													
1-100					101-200					201-300					301-400					401-500					501-512				
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20										
75	75	75	33	33	33	33	33	33	75	75	75	75	75	75	75	75	75	75	75										
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40										
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-										

Recording An Empty Cue

By default, cues store information for all 512 of the available channels and when played back they affect all of the channel outputs.

However, cues may be recorded without any channel level information at all. This type of cue does not affect the DMX output when played, it is more like a placeholder.

Empty cues still have a follow time and can link to other cues, which preserves its ability to be mixed in with other cues in the Cue List.

Empty cues also can execute CueScript actions, which allows an empty cue (one that does not directly affect the DMX output) to automate other tasks via CueScript commands. This would be useful to display prompts on the LCD Display, or illuminate LEDs on the front-panel buttons, or output serial strings, begin running cues on other playback faders and much more. A cue of this type might send the appropriate commands to lower a projection screen, for example.

When recording a cue using a CueScript command, the “#” character is added before the cue number to specify that **no DMX channels** should be recorded into the cue.

For example:

```
Record Cue #8
```

This command creates a new Cue 8 that does not contain any DMX channel information.

Recording a Streaming Cue

CueServer can record a special type of cue called a Streaming cue. This type of cue captures the lighting show playing at CueServer's DMX input port from an external lighting console or controller.

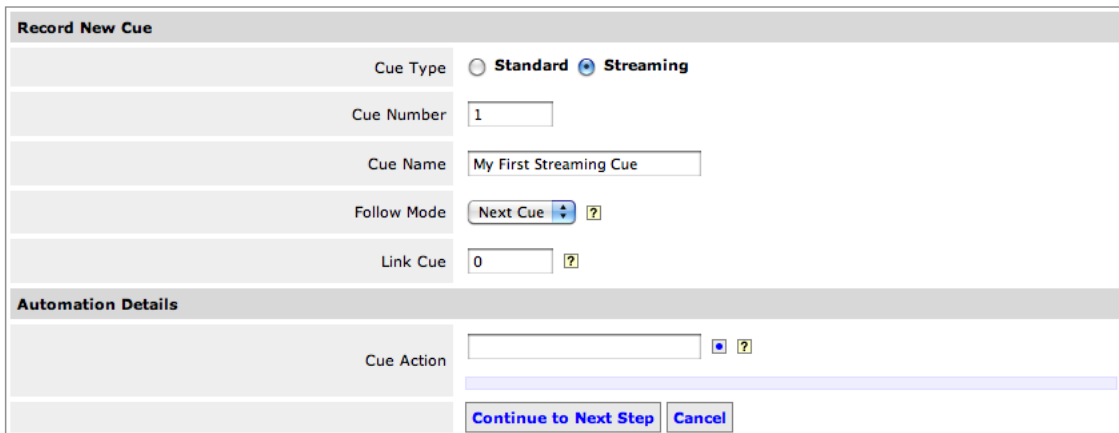
While a Streaming cue is being recorded, CueServer stores every DMX update being sent from the external device, similarly to how a tape recorder might capture the show. Then, when the cue is played back, the entire sequence exactly as it was recorded is output from the CueServer.

It should also be noted here that streaming cues can be recorded and placed in the same Cue List as normal cues. A static cue might play first, followed by streaming playback, followed by another static cue, etc. Also, because CueServer has four playback faders, multiple streaming cues and/or static cues can be playing back at the same time.

To record a streaming cue, click on the **Add New Cue** button from the **Cue List** page:

Add New Cue

Next, click on the Streaming radio-button to choose a streaming-type cue:



The screenshot shows a web form titled "Record New Cue". At the top, there are two radio buttons for "Cue Type": "Standard" (unselected) and "Streaming" (selected). Below this are several input fields: "Cue Number" with the value "1", "Cue Name" with the value "My First Streaming Cue", "Follow Mode" with a dropdown menu showing "Next Cue" and a help icon, and "Link Cue" with the value "0" and a help icon. A section titled "Automation Details" contains a "Cue Action" field with a dropdown menu and a help icon. At the bottom of the form are two buttons: "Continue to Next Step" and "Cancel".

The page will change to only show properties that are appropriate for a streaming cue. These properties are described below:

Cue Number

Specify a number for this cue. All cues are referenced by number. Each cue must have a unique cue number. Any number from 0.1 through 6499.9 may be used.

Cue Name

Each cue may have a user-assigned name. This is for your convenience only.

Follow Mode

Streaming cues have a follow mode that specifies what CueServer should do as soon as the end of the stream has been reached. The following options are available:

- **Next Cue**
When the stream completes, the playback fader will advance to the next cue in the list (or follow to the linked cue, if specified).
- **Loop**
When the stream completes, the stream will immediately loop back to its beginning. This cue will loop indefinitely.
- **Hold**
When the stream completes, the playback fader will hold the last frame without advancing automatically to the next cue.
- **Release**
When the stream completes, the playback fader will be released, causing all of its channel levels to be released.

Link Cue

By default, cues always execute in numerical order (1 followed by 2, followed by 3, etc.). If a cue has a Link Cue, then the playback order of which cue comes next is overridden by this field. Use this feature to create loops or other jumps in cue playback order.

Cue Action

This is the optional CueScript action that CueServer will execute when this cue executes. Any string of valid CueScript commands can be placed in this field. Use this field to have the cue run commands that can perform nearly any automation task, such as illuminating button LEDs, toggle digital outputs, output serial strings, display prompts on the LCD, run cues in the other playback faders and more. Unless otherwise specified, the same playback fader that was used to run the cue will be the default playback for CueScript commands.

After the cue number is chosen and the cue's other properties are set, click on the **Continue to Next Step** button to proceed to record the stream:

Continue to Next Step

Next, CueServer will display the Record Streaming Cue page:

CueServer	Main	Console	Cue List	Groups	Stage	Keypad	Triggers
Record DMX Input Stream							
This page will perform a real-time recording of the DMX Input supplied to CueServer. During the recording process, only the DMX Input will be recorded (CueServer's playback faders will be disabled).							
Cue Number	102						
Trigger Channel <small>(optional input channel that will automatically start/stop the recording process)</small>	<input type="text" value="512"/>						
Stream Duration <small>(optional recording duration, in seconds)</small>	<input type="text"/>						
Recording Controls							
DMX Recorded	00:00:00.0						
Memory Available	99.7%						
Trigger Channel Status	No DMX Input						
Recording Status	Ready to Record (will wait for DMX Input)						
	<input type="button" value="Start Recording"/>						
	<input type="button" value="Erase Stream"/>						
	<input type="button" value="Return to Cue"/>						

From this page, you can Start and Stop recording manually, specify an optional Trigger Channel for automatic recording, specify an optional recording duration, or re-record a stream of DMX data.

The function of the fields and buttons on this page are described below:

Cue Number

This is the number of the cue being recorded.

Trigger Channel

This field activates a function to automatically trigger the starting and stopping of the recording process by monitoring one of the DMX input channels being supplied by the external console or controller. Specifying a channel number here will cause CueServer to wait for that channel to rise above 0 (zero) to automatically start recording, and then automatically stop recording as soon as the channel returns to 0 (zero). This feature is very useful for programming the external show to automatically mark its own start and stop times (or in-point and out-point). To disable this feature and record a stream manually, enter a trigger channel of 0 (zero).

Stream Duration

This field specifies the duration of a stream to be recorded. Supply a number of seconds (accurate to 0.1s) of the desired recording length. CueServer will stop recording when this time limit is reached. If not specified, stream recording continues until manually stopped by clicking the Stop button or automatically stopped when the trigger channel returns to zero or when available memory is exhausted.

DMX Recorded

This field displays the duration of the current recording in Hours, Minutes, Seconds and Tenths. While recording is in process, this number updates live.

Memory Available

This is the approximate amount of memory available to record the streaming cue into. Depending on the update rate of the source and the complexity of the show being captured, this number will decrease at varying rates while recording is in process. If the source signal is being recorded, but none of the DMX input levels are changing, this number will decrease very slowly, for example.

Trigger Channel Status

This field displays the live status of the chosen trigger channel.

Recording Status

This field shows the current status of recording. It may indicate that CueServer is ready to record, or currently recording, or waiting for DMX input, or waiting for the trigger channel to become active, etc.

Start Recording

The Start Recording button is used to begin the recording process. If a trigger channel is specified, CueServer will wait for that channel to rise above 0 (zero) before beginning to capture data. If no DMX Input is present, CueServer will wait for DMX input to start before beginning to capture data. Finally, if DMX input is available and there is no trigger channel specified, CueServer will begin capturing data as soon as the Start Recording button is pressed.

Note that when CueServer is recording a stream or waiting to record a stream, the Start Recording button changes to Stop Recording.

Erase Stream

The Erase Stream button becomes enabled after a stream has been recorded. This button lets you erase the current stream and record the stream again. You must erase an existing stream before CueServer will allow you to re-record a new stream into the cue.

When satisfied with the stream recorded in the cue, exit back to the cue details page by pressing the **Return to Cue** button:

Return to Cue

Playing Cues

There are several ways to play back cues. Cues can be executed manually via CueServer's built-in web pages, or they can be remotely triggered, or they can be run as the result of an event-based object such as a front-panel button, timer, contact closure or other CueServer trigger.

The following sections describe in detail the various ways that cues can be played back.

Manual Cue Playback

From any of CueServer's web pages with a live command line, cues can be played back using CueScript commands.

Use the command string "`Cue (cue number) Go`" to run a cue. For example:

```
Cue 30 Go
```

When this command is entered, the current playback fader (by default Playback 1) will begin running Cue 30.

Each time a cue runs, the playback fader keeps track of which cue is the next cue. To play the next cue in sequence, simply use the command:

```
Go
```

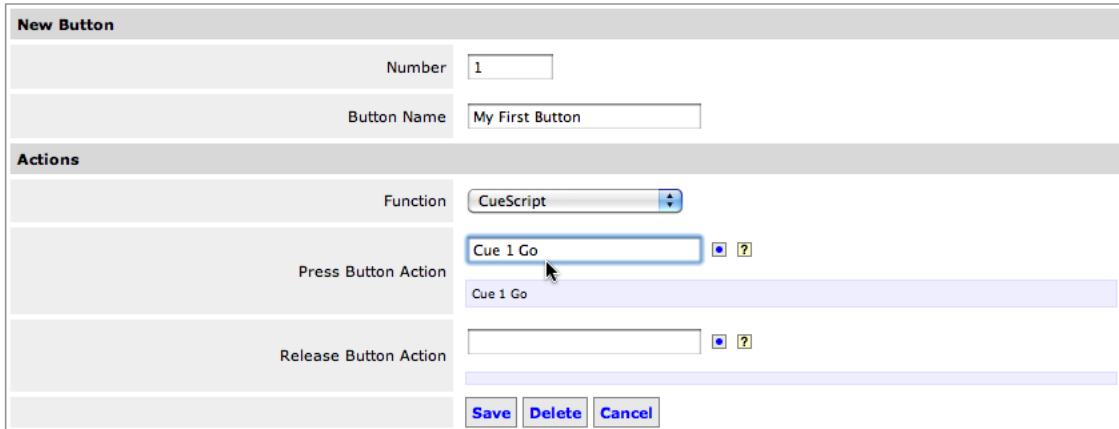
Using the `Go` command without first specifying a cue causes the next cue in sequence to run. In this example, Cue 31 would play next. On the next `Go`, Cue 32 would play, etc.

Automated Cue Playback

Buttons, timers, contact closures, DMX input, timecode events, system events and more can be programmed to run cues when they are activated. Each of these objects generate events in CueServer, and each of these events can execute CueScript commands.

For example, to program a button (or any of the other event-based objects) to run a cue when activated, use CueServer's **Triggers** web pages to program the object to run the desired cue.

For example, enter the **Function Buttons** page of the **Triggers** area and add a button to the list (or edit an existing button). After choosing a button, the Button Setup page appears:



New Button	
Number	<input type="text" value="1"/>
Button Name	<input type="text" value="My First Button"/>
Actions	
Function	<input type="text" value="CueScript"/>
Press Button Action	<input type="text" value="Cue 1 Go"/> <input type="checkbox"/> <input type="button" value="?"/>
Release Button Action	<input type="text" value=""/> <input type="checkbox"/> <input type="button" value="?"/>
<input type="button" value="Save"/> <input type="button" value="Delete"/> <input type="button" value="Cancel"/>	

In this example, we named Button 1 “My First Button”, made it a CueScript-type button and assigned it the command “Cue 1 Go” when the button is pressed.

For additional information about all of CueServer’s Trigger features, see the chapter on Triggers.

Viewing Cue Playback

CueServer's **Console** web page displays a live view of the operation of each of CueServer's four playback faders.

Playback 1*					Go	Select	Clear	View
Current:	Cue 20	Fade Time:	Immediate	Link Cue:	20	Output:	100%	
Next Cue:	30	Follow Time:	60.0 sec	Follow Timer:	-	Mode:	Merge	
Playback 2					Go	Select	Clear	View
Current:	Empty	Fade Time:	Immediate	Link Cue:	-	Output:	100%	
Next Cue:	-	Follow Time:	None	Follow Timer:	-	Mode:	Merge	
Playback 3					Go	Select	Clear	View
Current:	Empty	Fade Time:	Immediate	Link Cue:	-	Output:	100%	
Next Cue:	-	Follow Time:	None	Follow Timer:	-	Mode:	Merge	
Playback 4					Go	Select	Clear	View
Current:	Empty	Fade Time:	Immediate	Link Cue:	-	Output:	100%	
Next Cue:	-	Follow Time:	None	Follow Timer:	-	Mode:	Merge	

While cues are playing back in any of the four playback faders, the Console page shows which cue is active, which cue is next, the current fade time, follow time, link, combine mode and active status.

Each of the fields that appear for each playback fader are described below:

Current

This field shows the current activity of the playback fader. It will indicate that the playback fader is Empty, or it contains a Cue, Active Channels, is playing back a streaming cue, or other descriptive information about the fader.

Next Cue

This is the next cue that will be executed when the playback fader receives a Go command or if it performs an auto-follow.

Fade Time

The fade time that will be used for the next Go.

Follow Time

The auto-follow time that will be used for the next Go.

Link Cue

The link cue that will be used for the next Go.

Follow Timer

If an auto-follow was assigned to the last cue that was executed, this field shows the count-down timer that will perform the auto-follow when it reaches zero.

Output

The submaster output level of this playback fader. Typically 100%, but can be changed by using the `Playback` command.

Mode

The combine mode for the playback fader. A playback fader may be in Merge, Override or Scale mode. Merge causes each channel of the fader to be output if it is a higher level than those of lower-numbered faders or the DMX input. Override causes each channel of the fader to be output instead of those of lower-numbered faders or the DMX input. Scale causes each channel of the fader to proportionally scale the output of the faders or DMX input that precede it. See the descriptions of the `Merge`, `Override` and `Scale` commands.

For additional information about the Console web page, see Console section of the Web Pages chapter.

Editing Cues

To edit an existing cue, click on the cue's number link from the Cue List web page:

Cue Number (click to edit)	Name
1	Start Show
10	Step 1
11	Step 2
12	Step 3

The edit cue page will appear for the chosen cue (assuming the cue is a normal-type cue):

Cue Details	
Cue Type	Normal
Cue Number	<input type="text" value="10"/>
Cue Name	<input type="text" value="Step 1"/>
Fade Time	<input type="text" value="1.0"/>
Follow Time	<input type="text" value="2.0"/>
Link Cue <small>only needed to jump to non-sequential cue number</small>	<input type="text" value="0"/>
Cue Levels	
Channel Levels	View Levels
Automation Details	
Command String	<input type="text" value="Macro 1"/> <input type="text" value="Macro 1"/>
	Save Delete Cancel

From this page, most of the cue's parameters can be modified.

Cue Type

This is the type of the cue. Once a cue has been recorded, its type cannot be changed. If a cue of a different type is needed, first delete the existing cue and re-record a new cue of a different type.

Cue Number

This is the cue's number. This value can be changed to assign a different number to the cue, which can change the cue's position in the Cue List. The cue number of a streaming cue cannot be changed.

Cue Name, Fade Time, Follow Time, Link Cue

The cue's name, fade time, follow time and link may all be changed.

Channel Levels

Click on the **View Levels** buttons to view the channel levels recorded in this cue.

Command String

The cue's CueScript action may be changed.

After a cue's details have been modified, click on the Save button to record the changes:

Save

Or, to delete the cue from the Cue List, click on the Delete button:

Delete

Creating Loops and Chases

To create a sequence of cues that loop or repeat indefinitely (until another event causes a different cue to play), use the **Link Cue** feature to set the last cue of the sequence to then link back to the first cue of the sequence. Links can be used anywhere you want to override the normal numerical-order playback of cues.

We'll create a four step loop in the following example. The first step is to create four cues (see the section on recording cues):

Cue Number (click to edit)	Name	Cue Timing	Link Cue	Command
10	First	Fade: 1.0 sec, Follow: ---	(next)	
11	Second	Fade: 1.0 sec, Follow: ---	(next)	
12	Third	Fade: 1.0 sec, Follow: ---	(next)	
13	Fourth	Fade: 1.0 sec, Follow: ---	(none)	

[Add New Cue](#)

Each of these cues was assigned a name and a 1 second fade time.

If you start playing back these cues with a "Cue 10 Go" command, Cue 10 will run. With each subsequent Go, Cue 11, then Cue 12, then Cue 13 will run. Issuing additional Go commands after Cue 13 runs will have no effect, as the end of the Cue List was reached.

To create the loop, edit Cue 13 and set its Link Cue to point back to Cue 10. The Cue List will now look like this:

Cue Number (click to edit)	Name	Cue Timing	Link Cue	Command
10	First	Fade: 1.0 sec, Follow: ---	(next)	
11	Second	Fade: 1.0 sec, Follow: ---	(next)	
12	Third	Fade: 1.0 sec, Follow: ---	(next)	
13	Fourth	Fade: 1.0 sec, Follow: ---	10	

[Add New Cue](#)

Now, when starting with Cue 10, each subsequent Go will cause the cues to play back in the following order: 10, 11, 12, 13, 10, 11, 12, 13, 10, 11, 12, 13...

Cues may also have an **Auto Follow Time** set in them. In this case, when a cue is run with a Go command, the playback fader starts its auto-follow timer. When the timer expires, the playback fader issues its own Go so that the next cue in sequence will run automatically. Use auto-follow times to create a sequence of cues that automatically step from one to another, when the first cue is run, it begins a process where subsequent cues automatically follow after a specified amount of time.

In the following example, we'll add a follow time of 2 seconds to each of the cues of a loop by editing each cue and entering 2 seconds in the cue's Follow Time field:

Cue Number (click to edit)	Name	Cue Timing	Link Cue	Command
10	First	Fade: 1.0 sec, Follow: 2.0 sec	(next)	
11	Second	Fade: 1.0 sec, Follow: 2.0 sec	(next)	
12	Third	Fade: 1.0 sec, Follow: 2.0 sec	(next)	
13	Fourth	Fade: 1.0 sec, Follow: 2.0 sec	10	

[Add New Cue](#)

Now, when Cue 10 is played (by a command like "Cue 10 Go"), the channels in Cue 10 will fade up in 1 second as expected. But, 2 seconds after the cue started running (1 second after the fade completes), the playback fader will issue its own Go command, which runs Cue 11 automatically. Then, 2 seconds later, Cue 12 is run. Then Cue 13. Then Cue 10 again (because of the link), etc.

Using cues with follow times allows chases and other automated cue stepping to occur.

Playback Faders

CueServer coordinates all DMX lighting playback through four independent Playback Faders. Each playback fader contains a combination of 512 DMX output channels, cue stack information, timing parameters and mode settings.

Each playback fader can play back either normal or streaming cues. They can also hold static or crossfading channel levels. Each playback fader can be placed in any of three combine modes, Merge, Override or Scale. Individual channels in each playback fader may have separate fade times, may be parked and may be enabled or disabled.

Using the various powerful features of CueServer's playback faders, CueServer can be used to play back simple cues, or be extended to play shows in multiple zones, provide partitioning features, coordinate automatic backup, enable emergency lighting and more.

Playback Fader Basics

In the simple case, when cues are played back, this occurs in one of the four playback faders. Each fader keeps track of which cue was most recently played and which cue will be next. Each time a `Go` command is issued for a playback fader, it advances to the next cue. This is very similar to how an entertainment-based lighting console operates.

Because CueServer has four playback faders, this allows the user to have as many as four independent cues running at the same time.

Cue Playback

To run multiple cues at once, you can issue commands to each playback fader:

```
Playback 1 Cue 101 Go
Playback 2 Cue 201 Go
```

These commands would tell Playback 1 to load Cue 101 and execute it, then tell Playback 2 to load Cue 201 and execute it.

Later, if you execute a `Go` command for either of the playbacks, they will advance to the next cue in the cue list:

```
Playback 1 Go
Playback 2 Go
```

These commands would advance Playback 1 to Cue 102 and Playback 2 to Cue 202.

Each time a cue is executed with the `Go` command, the associated playback fader loads the cue's channel levels and begins crossfading to the new scene over the specified fade time.

CueServer's Console web page displays the current status of each of the four playback faders:

Playback 1*					Go	Select	Clear	View
Current:	Cue 101	Fade Time:	5.0 sec	Link Cue:	-	Output:	100%	
Next Cue:	102	Follow Time:	60.0 sec	Follow Timer:	-	Mode:	Merge	
Playback 2					Go	Select	Clear	View
Current:	Cue 201	Fade Time:	7.5 sec	Link Cue:	-	Output:	100%	
Next Cue:	202	Follow Time:	60.0 sec	Follow Timer:	-	Mode:	Override	
Playback 3					Go	Select	Clear	View
Current:	Empty	Fade Time:	Immediate	Link Cue:	-	Output:	100%	
Next Cue:	-	Follow Time:	None	Follow Timer:	-	Mode:	Merge	
Playback 4					Go	Select	Clear	View
Current:	Empty	Fade Time:	Immediate	Link Cue:	-	Output:	100%	
Next Cue:	-	Follow Time:	None	Follow Timer:	-	Mode:	Merge	

This page shows that following the example above, Cues 101 and 201 have been played in playback faders 1 and 2 and that Cues 102 and 202 are waiting for the next `Go` command before executing.

If the cues playing in a playback fader have auto-follow times or links, these parameters appear in the fader's display on the Console page. As cues auto-advance for chases or loops, all of the timing and link information appear as well.

Manual Channels

Not only may each playback fader be used to play cues back, but they can also be used to set and fade manual channel levels.

For example, start with CueServer reset to its default state:

```
Reset (RESET)
```

Entering the following commands would set channel levels in multiple playback faders:

```
Playback 1 (P1)
Channel 1>10 @ 50 (C1>10A50)
Playback 2 (P2)
Channel 5>15 @ 75 (C5>15A75)
Playback 3 (P3)
Channel 9+10+14>20 @ 66 (C9+10+14>20A66)
```

Go to the Stage view page to see the result of setting these levels:

DMX Output Levels																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100		101-200		201-300		301-400		401-500		501-512							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	50	50	50	75	75	75	75	75	75	75	75	75	75	75	66	66	66	66	66
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Note that by default, each of the playback faders merges its output by using a “highest takes precedence” (HTP) mode. This means that for each output channel, the playback fader with the highest level will contribute to the output.

The Stage view color-codes each channel with the matching color for the playback fader that is contributing to the output.

To see which channels are active in each playback, click on the tab buttons for each playback fader:

Playback 1 Levels (Merge Mode)																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100		101-200		201-300		301-400		401-500		501-512							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	50	50	50	50	50	50	50	50	50	-	-	-	-	-	-	-	-	-	-
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Playback 2 Levels (Merge Mode)																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100		101-200		201-300		301-400		401-500		501-512							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	-	-	-	75	75	75	75	75	75	75	75	75	75	75	-	-	-	-	-
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Playback 3 Levels (Merge Mode)																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100		101-200		201-300		301-400		401-500		501-512							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	-	-	-	-	-	-	-	66	66	-	-	-	66	66	66	66	66	66	66
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

When the result of each playback is combined via the Merge mode, the channels in Playback 2 set to 75% take precedence over the 66% levels in Playback 3, which take precedence over the 50% levels in Playback 1.

Independent Timing

Each channel in a playback fader may be given independent timing when fading to a new channel level.

First, clear Playback 2 (which also selects Playback 2):

```
Playback 2 Clear (P2CL)
```

Then, set several channels to different levels with different fade times:

```
Time 90 Channel 10 At FL (T90C10AFL)
Time 70 Channel 11 At 90 (T70C11A90)
Time 50 Channel 12 At 80 (T50C12A80)
Time 30 Channel 13 At 70 (T30C13A70)
Time 10 Channel 14 At 60 (T10C14A60)
```

As each command is entered, each channel begins fading toward its destination with a different fade time -- each channel will arrive at its desired level at a different time (which is unlike a regular cue, when all channels arrive at their destination at the same time).

When all of the fades are complete, the Playback 2 stage view should look like this (make sure to click on the P2 button to view only Playback 2 channels):

Playback 2 Levels (Merge Mode)																			
View: Input P1 P2 P3 P4 Output								Channels: 1-100 101-200 201-300 301-400 401-500 501-512											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	-	-	-	-	-	-	-	-	FL	90	80	70	60	-	-	-	-	-	-
21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

The Group and Fixture commands (or anything else that can set output levels) may be used in the same way:

```
Time 15 Group 5 At 75 (T15U5A75)
Time 30 Group 6 At 66 (T30U6A66)
```

For additional flexibility, independent timing commands may be placed in Macros or used as command strings placed in cues to produce special timing effects when running a cue.

Split Fade Times

CueServer supports the concept of Split Fade Times. Instead of simply providing a single time during which the fader fades from one scene to another, two times can be provided -- the first time applies to channel values which are fading up and the second time applies to channel values that are fading down. This can provide an advanced scene transition (popular in traditional stage lighting) that allows the new scene to appear quickly while the old scene fades out more slowly.

To specify a split fade time, use the slash character ‘/’ to separate two different times (the first for channels fading up and the second for channels fading down):

```
Time 1.5/3 Channel 1>10 At 50
```

In the above example, channels 1 through 10 will fade to 50%. Depending on the previous values of each of these ten channels, if a channel was previously lower than 50% (meaning that it will have to fade up to reach 50%), it will do so in 1.5 seconds. If a channel was previously higher than 50% (meaning that it will have to fade down to reach 50%), it will do so in 3 seconds.

Cues may be recorded with split fade times, or they can be used manually with CueScript commands (like the example above).

Disabling Timing

Timing may be temporarily disabled for a playback fader. When timing is disabled, fades no longer occur, auto-follow timers do not run and stream playback is frozen. This function can be useful during programming when one wants to recall a cue that has a long fade time or an auto-follow timer that would otherwise automatically advance to the next cue.

To disable the timing of a playback fader, use the `stop` command. To re-enable the timing, use the `start` command.

For example:

```
Playback 1 Stop (P1STO)
```

or

```
Playback 2 Start (P2STA)
```

When a playback fader's timing is disabled (stopped), it displays this state in the Console page:

Playback 1* (Timing Stopped)				Go	Select	Clear	View
Current:	Cue 101	Fade Time:	5.0 sec	Link Cue:	-	Output:	100%
Next Cue:	102	Follow Time:	60.0 sec	Follow Timer:	-	Mode:	Merge

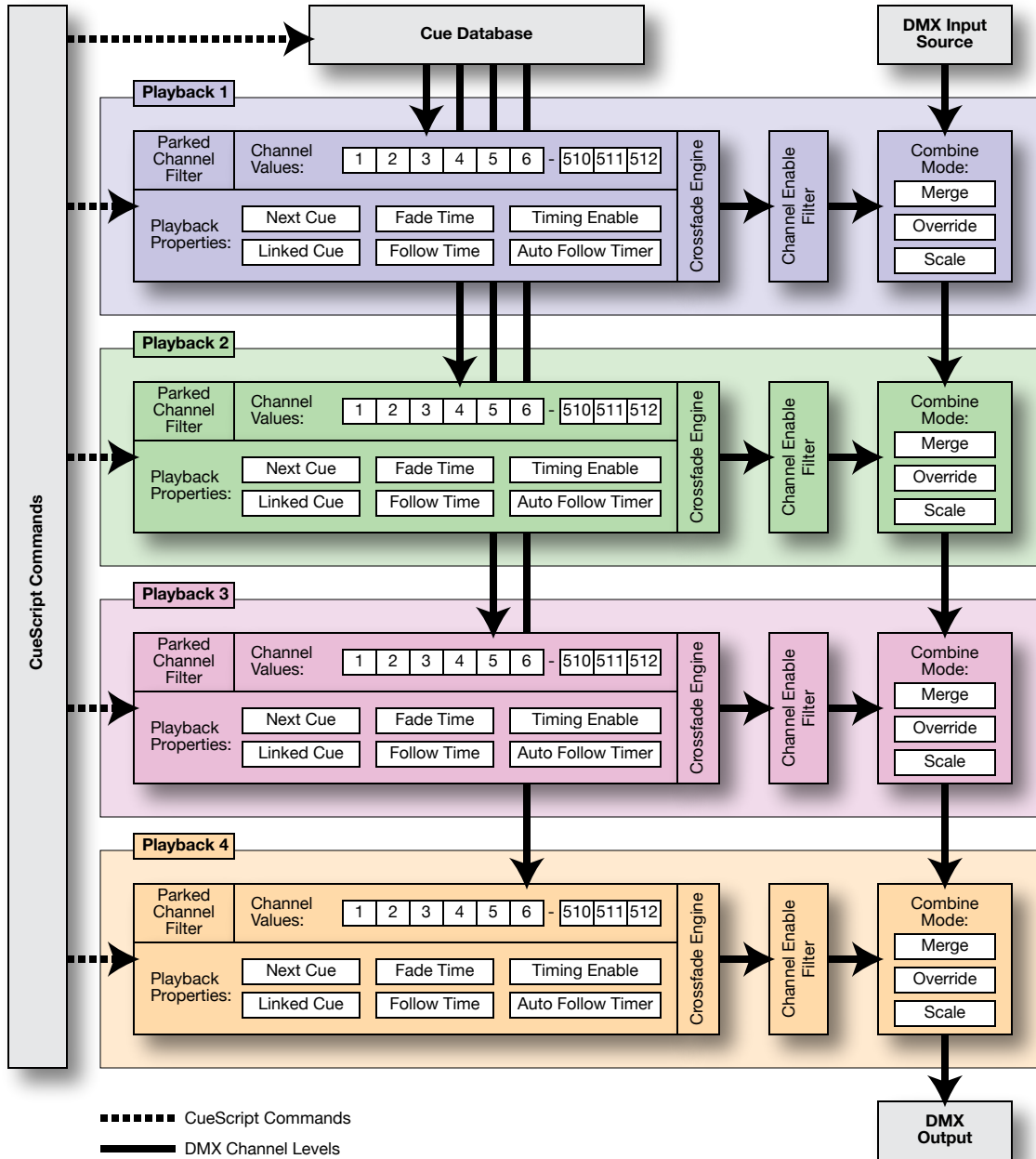
If a playback fader's timing is stopped and a cue is executed, the cue will ignore its programmed fade time. The cue's channels appear immediately. If the cue has an auto-follow time, that timer is stopped also, which keeps loops and/or chases from running. If the cue is a streaming cue, it will be frozen at the first frame of playback, waiting for a `start` command to resume its timeline.

Alternatively, if timing is enabled and a crossfading cue or streaming cue is currently in the process of playing back and then the `stop` command is received during the fade or stream, the fade or stream playback is frozen at its current point in time. The channel levels will remain constant until a `start` command is issued, which will resume the fade or stream from its current point in time.

Advanced Playback Features

Each of the playback faders has a robust set of features for customizing how it operates. Each fader can contribute its output by merge, override and scale modes, independent channels may be parked or disabled and more. These features can allow CueServer to be further customized to satisfy a wide variety of complex lighting control projects.

The following diagram shows how each playback fader works internally and how they contribute their parts to the overall DMX output of the CueServer:



Note that in CueServer, DMX channel levels flow from CueServer’s DMX Input port through each of the four playback faders in succession and then finally to the DMX Output port. The specific combine mode of each playback fader changes the way that each playback contributes its channel levels to the next playback fader or the overall output of the device.

There are independent channel enabling filters on each channel that can restrict its level from appearing at the combine stage for each fader.

Each channel in each fader may be independently parked, which causes incoming commands or cues that would affect parked channels to be discarded, causing those channel levels to become locked at their current value until those channels are unparked.

All of these components of CueServer’s playback engine are described in further detail in the following sections.

Combine Modes

Each of CueServer’s four playback faders can be assigned to operate in one of three combine modes: Merge, Override or Scale. The combine mode affects how the channels in one playback fader are combined with the flow of DMX levels from one playback to the next and finally to the overall output of the device.

Merge Mode

When a playback fader is in Merge mode (the default mode), then as channel levels flow from one playback fader to the next, each fader’s channels are contributed to the output only if the new level is higher than the previous level. This is very similar to how the “Highest Takes Precedence” (HTP) mode works on many entertainment-class lighting controllers.

A playback fader is placed in Merge mode by executing the Merge command. For example:

```
Playback 1 Merge (P1ME)
```

For instance, if the following commands are issued to set levels in the first three playback faders, then each fader would contribute its levels to the output when its channels contain the highest level of the other playback faders:

```
Playback 1 Channel 3>6 At 75 (P1C3>6@75)
Playback 2 Channel 1>10 At 33 (P2C1>10@33)
Playback 3 Channel 5>8 At 50 (P3C5>8@50)
```

Each of these three playback faders would contain the following channels:

Playback 1 Levels (Merge Mode)																									
View:		Input	P1	P2	P3	P4	Output	Channels:												1-100	101-200	201-300	301-400	401-500	501-512
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20						
-	-	75	75	75	75	-	-	-	-	-	-	-	-	-	-	-	-	-	-						

Playback 2 Levels (Merge Mode)																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
33	33	33	33	33	33	33	33	33	33	-	-	-	-	-	-	-	-	-	-

Playback 3 Levels (Merge Mode)																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	-	-	-	50	50	50	50	-	-	-	-	-	-	-	-	-	-	-	-

And the combined output of CueServer would look like:

DMX Output Levels																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
33	33	75	75	75	75	50	50	33	33	-	-	-	-	-	-	-	-	-	-

The 75% levels in Playback 1 take precedence over the 50% levels in Playback 3, which take precedence over the 33% levels in Playback 2.

Override Mode

When a playback fader is in Override mode, then if the fader contains channels, they supersede any channels coming from previous (lower-numbered) playback faders. Putting a playback fader in Override mode makes that fader work in a priority-based strategy -- channels in higher-numbered playback faders take precedence over lower-numbered playback faders.

A playback fader is placed in Override mode by executing the `Override` command. For example:

```
Playback 3 Override (P3OV)
```

For example, assuming that the first three playback faders contained the same example levels set in the Merge example above, but Playback 3 is set to Override mode, then the channels that are in Playback 3 will take precedence over the channels in Playback 1 or 2 regardless of their levels.

The resulting output from CueServer would change to:

DMX Output Levels																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
33	33	75	75	50	50	50	50	33	33	-	-	-	-	-	-	-	-	-	-

Notice that the 50% levels for channels 5 and 6 in Playback 3 are overriding the higher levels of 75% in Playback 1.

Scale Mode

When a playback fader is in Scale mode, then each channel in the fader is used to proportionally scale the output levels given to it by the previous (lower-numbered) playback faders. A playback fader in Scale mode works to inhibit or reduce channel levels by a percentage -- a feature that can be used to create inhibitive submasters or a grandmaster for CueServer (concepts used by entertainment lighting controllers), or can be used to control the relative intensity of a preset group, zone or room (concepts used by architectural lighting).

A playback fader is placed in Scale mode by executing the `Scale` command. For example:

```
Playback 4 Scale (P4SC)
```

For example, if the output from CueServer initially looks like this (the result of following the last example):

DMX Output Levels																			
View: Input P1 P2 P3 P4 Output								Channels: 1-100 101-200 201-300 301-400 401-500 501-512											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
33	33	75	75	50	50	50	50	33	33	-	-	-	-	-	-	-	-	-	-

Then, Playback 4 is placed in Scale mode and Channels 1 through 6 are set to 50%.

```
Playback 4 Scale (P4SC)
Channel 1>6 At 50 (C1>6A50)
```

When viewing the levels in Playback 4, the stage page will reveal that the fader is in scale mode and each of its channels are set to 50%:

Playback 4 Levels (Scale Mode)																			
View: Input P1 P2 P3 P4 Output								Channels: 1-100 101-200 201-300 301-400 401-500 501-512											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	50	50	50	50	50	-	-	-	-	-	-	-	-	-	-	-	-	-	-

But, when looking at the combined output of CueServer, the result of scaling channels 1>6 by 50% shows that the levels being output by Playbacks 1 to 3 are being reduced by 50%:

DMX Output Levels																			
View: Input P1 P2 P3 P4 Output								Channels: 1-100 101-200 201-300 301-400 401-500 501-512											
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
16	16	37	37	25	25	50	50	33	33	-	-	-	-	-	-	-	-	-	-

To eliminate the effects of the scaling in Playback 4, either set the channel levels in Playback 4 back up to 100% (to not scale down by any percent), or release the channels in Playback 4 by executing the `Release` command.

Playback Submasters

Each playback fader includes a “submaster” that controls the overall intensity level of all channels being output by that playback.

To adjust the submaster level of a playback fader, use the “Playback *x* At *y*” commands.

For example, the command “Playback 3 at 75” would set the submaster for Playback 3 to 75%.

When setting the submaster level, the current fade time (as set by the Time command) is used to ramp the submaster level up or down over a period of time.

The Console page shows the current level of the submaster of each playback fader (labelled “Output”):

Playback 1*				Go	Select	Clear	View
Current:	Cue 101	Fade Time:	Immediate	Link Cue:	-	Output:	33%
Next Cue:	102	Follow Time:	60.0 sec	Follow Timer:	-	Mode:	Merge

In this example above, the submaster of Playback 1 is set to 33%.

Channel Enable Filters

The contribution of each channel in each playback fader may be independently enabled or disabled. When a channel is disabled, the playback fader still maintains its level and continues to perform crossfades and other operations on the channel, but the channel does not affect the output of the playback fader, and subsequently it doesn’t appear at CueServer’s output.

To enable or disable one or more channels in a playback fader, use the Enable and Disable commands:

```
Channel 1 Disable (C1DIS)
Channel 10>20 Enable (C10>20ENA)
```

An Example

The following example will help illustrate how channel disabling works. Start with all playback faders empty. Then, set a few levels:

```
Playback 1; Channel 1>10 At 50 (P1C1>10A50)
Playback 2; Channel 3>8 At 75 (P2C3>8A75)
```

The combined output of CueServer should look like:

DMX Output Levels																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	50	75	75	75	75	75	75	50	50	-	-	-	-	-	-	-	-	-	-

Next, disable channels 5 and 6 in Playback 2 by using the command:

Playback 2; Channel 5+6 Disable (P2C5+6DIS)

DMX Output Levels																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	50	75	75	50	50	75	75	50	50	-	-	-	-	-	-	-	-	-	-

After channels 5 and 6 of Playback 2 are disabled, then the 50% levels from Playback 1 are no longer suppressed by the higher levels in Playback 2. The output from CueServer on channels 5 and 6 becomes 50%.

To investigate further, view the contents of Playback 1 and 2:

Playback 1 Levels (Merge Mode)																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	50	50	50	50	50	50	50	50	50	-	-	-	-	-	-	-	-	-	-

Playback 2 Levels (Merge Mode)																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	-	75	75	75	75	75	75	-	-	-	-	-	-	-	-	-	-	-	-

Note that although Playback 2 contains 75% levels for channels 3 through 8, channels 5 and 6 are greyed-out, showing that they're currently disabled (not making a contribution to the output).

Set channel 5 to 66%:

Playback 2; Channel 5 At 66 (P2C5+6DIS)

Playback 2 Levels (Merge Mode)																			
View:		Input	P1	P2	P3	P4	Output	Channels:											
		1-100	101-200	201-300	301-400	401-500	501-512												
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-	-	75	75	66	75	75	75	-	-	-	-	-	-	-	-	-	-	-	-

Then, switch back to viewing the stage output:

DMX Output Levels																			
View: Input P1 P2 P3 P4 Output Channels: 1-100 101-200 201-300 301-400 401-500 501-512																			
1 50	2 50	3 75	4 75	5 50	6 50	7 75	8 75	9 50	10 50	11 -	12 -	13 -	14 -	15 -	16 -	17 -	18 -	19 -	20 -

Nothing has changed on the output, because channels 5 and 6 are still disabled. Then, re-enable channels 5 and 6 by using the `Enable` command:

```
Channel 5+6 Enable (C5+6ENA)
```

Now, see that the output from Playback 2 re-appears at the output with the newly set level:

DMX Output Levels																			
View: Input P1 P2 P3 P4 Output Channels: 1-100 101-200 201-300 301-400 401-500 501-512																			
1 50	2 50	3 75	4 75	5 66	6 75	7 75	8 75	9 50	10 50	11 -	12 -	13 -	14 -	15 -	16 -	17 -	18 -	19 -	20 -

Uses for Channel Disabling

The following is an example of how channel disabling can be used:

External Console Priority

A project requires that CueServer outputs a light show each evening, but on special nights an external lighting console will be attached to CueServer’s DMX Input jack and CueServer should automatically allow the console to control the lights.

To accomplish this, set the DMX Input Restore trigger (in the System Events page) to disable all channels in Playback 1 (the playback that the CueServer show is running in). Do this with the command “Playback 1 Channel * Disable”. Then, in the DMX Input Loss trigger, use the command “Playback 1 Channel * Enable”.

Now, when an external console is attached, CueServer will automatically disable its internal show’s outputs which will allow the console’s input to pass-through without being modified. Then, when the console’s input is removed, CueServer will automatically switch back to Playback 1 being enabled, which will begin outputting whatever show was silently running in the background.

Channel Parking

Each channel in each of the playback faders may be independently Parked. When a channel is parked, it becomes “frozen” at its current output level. Subsequent commands or cues that would normally effect parked channels will have no effect on those channels.

To park one or more channels in a playback fader, use the `Park` command. To reverse the effect of a parked channel, use the `Unpark` command.

Parking a channel is different from Disabling a channel, because when a channel is parked, it becomes “stuck” at its current level, while a channel that is disabled stops contributing to the fader’s output levels even though the channel still holds its last level (and can be changed to other levels).

For example, start with CueServer reset to its default state:

Reset (RESET)

Then, set a few channels to a level:

Channel 1>10 @ 10 (C1>10A10)

See the results in the Stage page, using the Playback 1 view:

Playback 1 Levels (Merge Mode)																			
View: <input type="button" value="Input"/> <input checked="" type="button" value="P1"/> <input type="button" value="P2"/> <input type="button" value="P3"/> <input type="button" value="P4"/> <input type="button" value="Output"/>																			
Channels: <input checked="" type="button" value="1-100"/> <input type="button" value="101-200"/> <input type="button" value="201-300"/> <input type="button" value="301-400"/> <input type="button" value="401-500"/> <input type="button" value="501-512"/>																			
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
10	10	10	10	10	10	10	10	10	10	-	-	-	-	-	-	-	-	-	-

Next, Park a few channels using the Park command:

Channel 4>7 Park (C4>7PARK)

Playback 1 Levels (Merge Mode)																			
View: Input P1 P2 P3 P4 Output										Channels: 1-100 101-200 201-300 301-400 401-500 501-512									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
10	10	10	10	10	10	10	10	10	10	-	-	-	-	-	-	-	-	-	-

Note that when viewing a playback fader's levels in the Stage window, parked channels display in Red text.

Finally, change channel values to test the parked channels:

Channel 1>10 @ FL (C1>10AFL)

Playback 1 Levels (Merge Mode)																			
View: Input P1 P2 P3 P4 Output										Channels: 1-100 101-200 201-300 301-400 401-500 501-512									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FL	FL	FL	10	10	10	10	FL	FL	FL	-	-	-	-	-	-	-	-	-	-

Notice that even though a command was executed that attempted to set all channels from 1 through 10 to Full, the parked channels remained at their parked levels. It should be noted that any attempt to change channels 4 through 7 including setting a group, running a static or streaming cue or releasing the channels will also have had no effect on their values.

Next, Unpark these channels using the Unpark command:

Channel 4>7 Unpark (C4>7UNPARK)

Playback 1 Levels (Merge Mode)																			
View: Input P1 P2 P3 P4 Output										Channels: 1-100 101-200 201-300 301-400 401-500 501-512									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FL	FL	FL	10	10	10	10	FL	FL	FL	-	-	-	-	-	-	-	-	-	-

Note that channels 4 through 7 remain at their previously parked 10% level. Unparking channels does not immediately effect any channel levels -- it just allows subsequent commands that would effect the channels to be able to do so.

Finally, set some new levels for all of the channels to demonstrate that they're unparked:

Channel 1>10 @ 66

(C1>10A66)

Playback 1 Levels (Merge Mode)																			
View: Input P1 P2 P3 P4 Output										Channels: 1-100 101-200 201-300 301-400 401-500 501-512									
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
66	66	66	66	66	66	66	66	66	66	-	-	-	-	-	-	-	-	-	-

Uses for Channel Parking

The following is an example of how channel parking can be used:

Locking Out Parts Of A Show

A project uses CueServer to control both the water jets and underwater lighting fixtures of a water fountain. In certain circumstances, the wind speed is too high to allow the water jets to function, but the light show should continue as normal.

To accomplish this, a contact closure is connected to CueServer that indicates from the wind speed meter that the wind speed is too high. In the contact closure's settings, the contact is programmed to execute the command "Group 10; Release; Park". These commands first select all channels stored in Group 10 (which are the channels that control the water jets), then those channels are released and then they are parked in their released state.

As shows are run throughout the day, all of the unparked channels continue to follow the cues that are executing without the water jets firing.

When the wind speed drops to an acceptable level, the contact opens. The contact is programmed to execute the command "Group 10 Unpark" when the contact opens. This command unparks the channels in Group 10, allowing those channels become part of the show again.

Input Disabling

By default, the DMX Input port receives any DMX levels coming into CueServer and passes them through to the stack of playback faders -- the DMX Input is merged with Playback 1, then this result is merged with Playback 2, then this result is merged with Playback 3, etc.

If the CueServer show wants to obscure any input coming from the DMX Input port, it has only two options. The first is to set Playback 1 into Override mode, which will cause all of the channels in Playback 1 to override the levels coming in from the DMX Input port. However, in cases where those channels need to be released, or Playback 1 needs to be in Scale more, this option may not accomplish the desired function.

The second solution is to disable the influence of the DMX Input on the playback faders. A special command is available to do this:

```
Input Disable (INDIS)
```

When the DMX Input port is disabled, it continues to receive DMX Input -- but none of these levels are passed along the chain of playback faders -- and they do not reach the output.

However, when DMX Input is disabled the levels can still be viewed from the Input tab in the Stage view, they will appear on the LCD Display when viewing DMX Input, they will still be recorded when Stream Recording is activated, they still activate DMX Input triggers, and the `At Input` command will still grab DMX Input levels.

To re-enable the DMX Input influence on the playback faders, execute the Input Enable command:

```
Input Enable (INENA)
```

Triggers

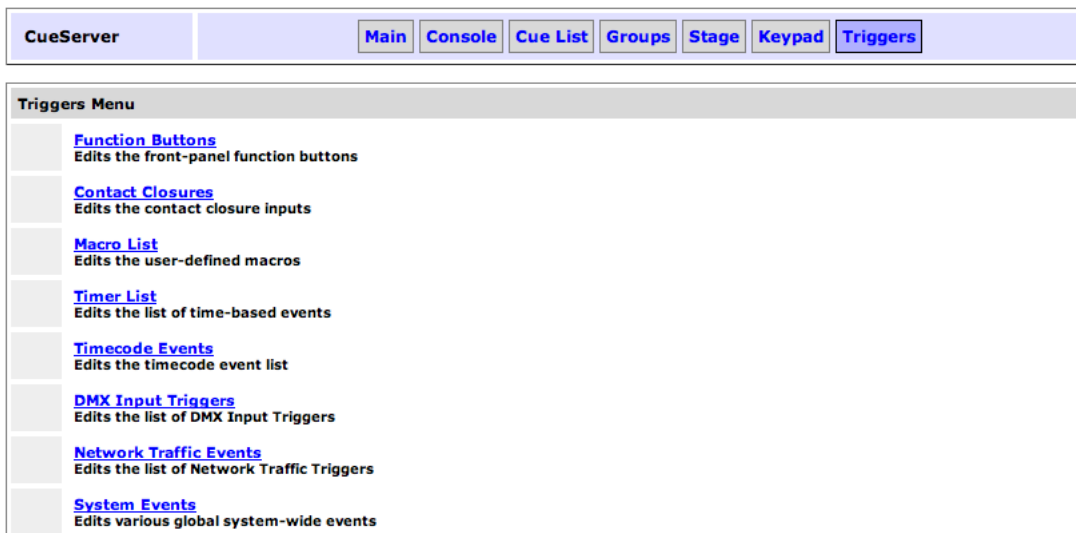
Central to CueServer's automation capabilities is the concept of a **Trigger**. A trigger is an object that is created in the CueServer show that responds to an **Event**. For example, you can create a Button Trigger that responds to events sent by one of the front-panel buttons on the CueServer, or you can create a Timer Trigger that responds to events that are sent by the CueServer's internal clock.

Each trigger in CueServer can be assigned one or more CueScript commands. This allows you to program the action that will be taken when an event occurs. For example, you might want to program a button to start a show or create a timer that automatically turns off the lights at a specific time.

There are several different types of triggers defined by CueServer. They are:

- Button Triggers
- Contact Closure Triggers
- Timer Triggers
- Timecode Triggers
- DMX Input Triggers
- Network Traffic Triggers
- System Event Triggers

To view or edit any of the triggers for a CueServer show, use the Triggers page:



The following sections describe each of CueServer's trigger mechanisms in detail.

Button Triggers

On CueServer models with programmable pushbuttons and/or when using external buttons (such as those connected to a CueStation Network), triggers may be assigned to each button to perform various actions when the button is pressed, released and/or held down.

Buttons can execute CueScript commands or be programmed to directly control a DMX channel or lighting preset, and other functions as well.

To set up button triggers, go to the **Triggers > Function Buttons** page. This example shows several buttons that have been configured with names, functions and actions:

Function Buttons				
Button (click to edit)	Name	Enabled	Function	Action
1	Start Show	Yes	CueScript	Press = [Cue 1 Go]
2	Stop Show	Yes	CueScript	Press = [Clear]
5	Timer Enable	Yes	CueScript	Off = ["TimerEnable" = "1" Button 5 On] On = ["TimerEnable" = "0" Button 5 Off]
6	Work Lights	Yes	Toggle Channel	Channel = 501, Fade = 1.0 sec, Level = 100%
7	Maintenance Mode	Yes	CueScript	Off = [Macro 201 Button 7 On] On = [Macro 202 Button 7 Off]
8	Emergency Override	Yes	CueScript	Press = [Macro 999]

[Add New Button](#)

The following columns are shown in the Function Button list:

Button

This is the button number of each button. Button numbers correspond to which physical button will activate the events for that button.

Name

This is the user-defined name of the button. This field is for your own descriptive use.

Enabled

This column shows if the button is currently enabled or disabled (using the `Enable` or `Disable` commands).

Function

This column shows the button's function mode.

Action

This column shows a summary of the button's properties.

To edit a button trigger, click on the number of the button you want to edit.

To add a new button trigger, click on the **Add New Button** button.

When adding or editing a button, a page similar to the following will appear:

New Button	
Number	<input type="text" value="1"/>
Button Name	<input type="text"/>
Actions	
Function	<input type="text" value="CueScript"/>
Press Button Action	<input type="text"/> <input type="checkbox"/> ?
Release Button Action	<input type="text"/> <input type="checkbox"/> ?
<input type="button" value="Save"/> <input type="button" value="Delete"/> <input type="button" value="Cancel"/>	

Each button's setup page contains fields for setting the button's number, name, function, and other parameters. Depending on the chosen function, the button parameters will include different choices.

All button definitions have a button number, name and function:

Button Number

Every button definition must have a unique number. There are two ways for button numbers to be specified. Buttons can be numbered from 1 through 512, or buttons can be numbered by a combination of a "station number" and "button number" (separated by a decimal point).

When using the "station.button" method, each "station" has 8 logical buttons assigned to it, so the first 8 buttons would be referred to as Button 1.1 through 1.8 and then Buttons 9 through 16 would be equivalent to 2.1 through 2.8, Buttons 17 through 24 equivalent to 3.1 through 3.8, and so on.

Any place that a button number is required (on the web page or in a CueScript command), either the simple button number (1 to 512) or the station/button number (1.1 to 64.8) can be used.

On CueServer models with front-panel buttons (such as the CS-800), those buttons activate button events 1 through 8. To create an action for Button 1 on the front-panel, create a button definition with the number 1 (or 1.1).

When using external buttons (such as CueStations), the dip switches on the back of each station assigns the station a "station number". To make the buttons on a station report to CueServer as Station #3, set the dip switches accordingly. Then, in CueServer, create button definition for buttons 17 through 24 (or, more simply, 3.1 through 3.8).

Button Name

This is the name of the button for your own descriptive use.

Button Function

This is the function of the button. The following choices are available for button functions:

- **CueScript**
When this option is selected, the button will execute CueScript commands when the button is pressed and/or released. Use this mode to specify commands (such as “Cue 1 Go”) to be executed when a button is pressed.
- **CueScript (State-Driven)**
When this option is selected, the button will execute one of two different CueScript commands, depending on the state of the button’s LED indicator at the moment the button is pressed by the user. One command is executed when the button’s indicator is off, and another command is executed when the indicator is on. It is up to the programmer to turn on and off the indicator by using the appropriate commands as part of the event actions. Use this mode to implement toggle buttons, radio-style buttons and other similar schemes.
- **Toggle Channel**
When this option is selected, the button will be set up to automatically toggle a channel on and off to a “memorized” preset level. Options are available for fade times, indicator behavior and a press-hold-cycle feature. Use this mode in a traditional architectural setting to control a “load”.
- **Toggle Preset**
When this option is selected, the button will be set up to automatically toggle a preset on and off. A preset is specified by both a group of channels and a cue number to hold the preset levels. Options are available for fade times, indicator behavior and a press-hold-record feature. Use this mode in a traditional architectural setting to control a “preset”.

Depending on the Button Function chosen, the properties of the button will be different. The following sections describe the available properties of each button function.

CueScript Function

When a button is in “CueScript” mode, it is capable of executing one CueScript statement when the button is pressed and another CueScript statement when the button is released.

This button function is best suited for general-purpose programming of buttons. A button can be instructed to perform any action (or combination of actions) that CueServer can perform by using CueScript statements.

The screen similar to the following appears while editing a CueScript button:

The screenshot shows a web interface titled "Edit Button 1 (Station 1, Button 1)". It contains the following elements:

- Button Name:** A text input field containing "Start Show".
- Actions:** A section header.
- Function:** A dropdown menu currently set to "CueScript".
- Press Button Action:** A text input field containing "Cue 1 Go", followed by a small blue square icon and a question mark icon. Below this is a light blue horizontal bar.
- Release Button Action:** An empty text input field, followed by a small blue square icon and a question mark icon. Below this is another light blue horizontal bar.
- Buttons:** Three buttons labeled "Save", "Delete", and "Cancel" at the bottom.

Press Button Action

This field allows you to enter the CueScript statement that should be executed when the user presses the button.

Release Button Action

This field allows you to enter the CueScript statement that should be executed when the user releases the button.

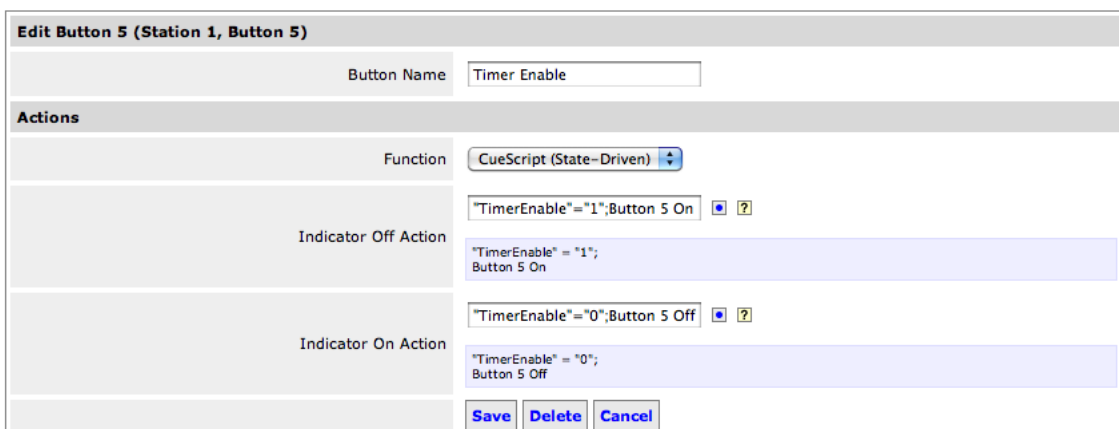
CueScript (State-Driven) Function

When a button is in “CueScript (State-Driven)” mode, it is capable of executing one CueScript statement when the button is pressed while the button’s LED indicator is off, and another CueScript statement when the button is pressed while the button’s LED indicator is on.

This button function is best suited for situations where the state of the button’s LED should dictate what action the button should take. This mode makes it easy to implement toggle buttons and radio-style groups of buttons.

It is important to note that when using this mode, you must manually maintain the state of the button’s LED. This means that when your user presses the button, you should turn “on” the button’s LED so that the next time the user presses the button, the other button action will be taken. Use the `Button` command to turn on or off the button’s LED.

The screen similar to the following appears while editing a State-Driven CueScript button:



Edit Button 5 (Station 1, Button 5)	
Button Name	Timer Enable
Actions	
Function	CueScript (State-Driven)
Indicator Off Action	*TimerEnable" = "1"; Button 5 On
Indicator On Action	*TimerEnable" = "0"; Button 5 Off
Save Delete Cancel	

Indicator Off Action

This field allows you to enter the CueScript statement that should be executed when the user presses the button while the button’s LED is off.

Indicator On Action

This field allows you to enter the CueScript statement that should be executed when the user releases the button while the button’s LED is on.

Toggle Channel Function

When a button is in “Toggle Channel” mode, it is programmed to toggle a single channel on and off to a memorized brightness level. Options are available for fade times, press-hold-cycle, saving the new level and indicator behavior.

This button function is designed for use in traditional architectural control applications where a button may be set up to turn on and off a “load”.

The screen similar to the following appears while editing a Toggle Channel button:

Edit Button 6 (Station 1, Button 6)	
Button Name	Work Lights
Actions	
Function	Toggle Channel
Channel	501
Level	100 %
Fade Time	1.0
Hold to Cycle	<input type="checkbox"/>
Save Level	<input type="checkbox"/>
Indicator	
Function	None
On Color	<input type="checkbox"/>
Off Color	<input type="checkbox"/>
Save Delete Cancel	

Channel

This is the channel number of the channel that should be controlled by this button.

Level

This is the brightness level that the channel should be set to when the button “turns on” the load.

Fade Time

This is the fade time (or split fade time) used to fade up or down the channel when it is being turned on or off.

Hold to Cycle

This check box enables the press-hold-cycle feature. If enabled, when the user presses and holds the button, the channel will slowly ramp up or down, allowing the user to release the button when the desired light level is reached.

Save Level

This check box enables the save level feature. If enabled, when the user adjusts the light level using the press-hold-cycle feature (above), the new light level will be saved (memorized) for toggling the channel on and off.

Indicator Function

This pop-up menu selects what automatic function the button's LED indicator will have. Available options include turning the indicator on when the channel (load) is non-zero, at the memorized level, at 100% and more.

Indicator On Color

This check box and field allows this button to have a different "on color" than the default on color. Enable the check box to override the default and enter the number of the color/flash-pattern for the indicator to use in it's on state.

Indicator Off Color

This check box and field allows this button to have a different "off color" than the default off color. Enable the check box to override the default and enter the number of the color/flash-pattern for the indicator to use in it's off state.

Toggle Preset Function

When a button is in “Toggle Preset” mode, it is programmed to toggle a group of channels on and off to a preset lighting look. Options are available for fade times, press-hold-record, and indicator behavior.

This button function is designed for use in traditional architectural control applications where a button may be set up to turn on and off a “preset”.

The screen similar to the following appears while editing a Toggle Preset button:

Edit Button 9 (Station 2, Button 1)	
Button Name	Test
Actions	
Function	Toggle Preset
Preset (Cue)	1
Zone (Group)	1
Fade Time	3.0
Hold to Record	<input checked="" type="checkbox"/>
Indicator	
Function	Preset Active
On Color	<input type="checkbox"/>
Off Color	<input type="checkbox"/>
<input type="button" value="Save"/> <input type="button" value="Delete"/> <input type="button" value="Cancel"/>	

Preset (Cue)

This is the number of the Cue that will hold the preset’s channel levels. When this preset is activated, the channel levels in the specified cue will be recalled.

Zone (Group)

This is the number of the Group that specifies which channels are included in this preset. When this preset is activated, only the channels in this group will be affected. Specify 0 (zero) to have the preset recall all channels from the preset cue.

Fade Time

This is the fade time (or split fade time) used to fade up or down the preset when it is being turned on or off.

Hold to Record

This check box enables the press-hold-record feature. If enabled, when the user presses and holds the button for 5 seconds, the preset will be re-recorded, using the current levels of all of the channels in the preset.

Indicator Function

This pop-up menu selects what automatic function the button's LED indicator will have. Available options include turning the indicator on when the preset is active or not-active.

Indicator On Color

This check box and field allows this button to have a different "on color" than the default on color. Enable the check box to override the default and enter the number of the color/flash-pattern for the indicator to use in it's on state.

Indicator Off Color

This check box and field allows this button to have a different "off color" than the default off color. Enable the check box to override the default and enter the number of the color/flash-pattern for the indicator to use in it's off state.

Contact Closure Triggers

On CueServer models with contact-closure inputs, triggers may be assigned to each contact-closure input to perform various actions when the contact is closed, opened and/or maintained.

Contact closures can execute CueScript commands or be programmed to directly control a DMX channel or lighting preset, and other functions as well.

To set up contact-closure triggers, go to the **Triggers > Contact Closures** page.

Contact closures in CueServer have the same general functions and programming model as the Function Buttons. Please see the previous section on Function Buttons for detailed explanation of the various screens, functions and properties available to contact closures.

Timer Event Triggers

CueServer contains an extensive timer/calendar system. Timers can be programmed that react to the time of the day, day of the week, day of the year, relative to calculated sunrise and sunset times (astronomical time) or any combination of these. When a timer event occurs, CueServer can run any CueScript commands you assign to the trigger.

To view the list of Timer Events that are programmed into CueServer, view the **Timer List** page in the **Triggers** section. A page similar to the following will appear:

Current Date: 10/04/07 - TimeZone: -5.0 - Sunrise: 6:34 AM - Sunset: 6:19 PM				
Timer (click to edit)	Name	Days	Trigger Time	Command
1	Museum Open	-MTWT--	10:00 AM	Macro 1
2	Museum Open Weekends	----FS	9:00 AM	Macro 1
3	Exterior Attract Show	-MTWTFS	Sunset + 15 Min	Macro 2
4	Light Show 1	-MTWTFS	7:00 PM	<ShowEnable> Macro 3
5	Light Show 2	----FS	8:00 PM	<ShowEnable> Macro 3
6	Soft Close	-MTWTFS	10:00 PM	<AutoClose> Macro 8
7	Close	-MTWTFS	10:45 PM	<AutoClose> Macro 9
8	Sunrise/Reset	Every Day	Sunrise - 30 Min	Playback 1 Clear; Playback 2 Clear; Playback 3 Clear

[Add New Timer Event](#)

In this example, several timers have already been set up with names, days of the week, times and CueScript actions.

To add a new Timer Event, click on the **Add New Timer Event** button.

[Add New Timer Event](#)

To edit an existing Timer Event, click on the number of the timer event.

In both cases, a page similar to the following will appear:

Timer Details	
Timer Name	Museum Open
Days of the Week This timer will only operate on selected weekdays	<input type="checkbox"/> Sun <input checked="" type="checkbox"/> Mon <input checked="" type="checkbox"/> Tue <input checked="" type="checkbox"/> Wed <input checked="" type="checkbox"/> Thu <input type="checkbox"/> Fri <input type="checkbox"/> Sat
Specific Days This timer will only operate on the chosen day or range of days (if checked)	<input type="checkbox"/> January 1 through January 1
Specific Time	<input checked="" type="radio"/> 10 : 0 <input checked="" type="radio"/> AM <input type="radio"/> PM
Relative Time	<input type="radio"/> Sunrise + 0 Minutes
Automation Details	
Command String	Macro 1 Macro 1 <input type="button" value="Save"/> <input type="button" value="Delete"/> <input type="button" value="Cancel"/>

In this example, the timer has already been configured to operate on weekdays from Monday through Thursday at 10:00AM. When the timer is triggered, it will execute Macro 1.

Each of the Timer Trigger's fields are described below:

Timer Name

This is the name of the timer event for your own descriptive use.

Days of the Week

This field allows each of the seven days of the week to be included or excluded from this timer's schedule. To have a timer operate every day, check all of the check boxes. A timer will not operate at all if none of the check boxes are checked.

Specific Days

This field allows a specific day or range of days to be included in the requirements for this timer's schedule. If the check box is checked, then the timer will only trigger if the current day is within the range of days specified.

Specific Time

When this field's radio button is selected, the timer will trigger at a specific time of the day.

Relative Time

When this field's radio button is selected, the timer will trigger at a time of the day, relative to the calculated Sunrise or Sunset time plus or minus an offset. For example, you can specify that the timer operate at Sunset + 15 minutes, or Sunrise - 30 minutes, etc.

Command String

This field holds the CueScript commands that will be executed when the timer event is triggered.

Timecode Triggers

CueServer can trigger events based on specific timecode markers. The CueServer Pro can receive and respond to MIDI Timecode (MTC) or SMPTE Timecode (by using a simple SMPTE to MTC converter, such as the JL Cooper PPS-2).

To view the list of Timecode Events that are programmed into CueServer, view the **Timecode Event List** in the **Triggers** page. A page similar to the following will appear (this example includes already defined timecode events):

Timecode (click to edit)	Name	Command
00:01:00.00	Tape Start	
00:01:04.12	Opening Scene	Cue 101 Go
00:01:47.03	Sunburst	Cue 102 Go
00:01:53.23	Meteor Shower	Cue 102.1 Go; Output 1 On
00:01:56.13	Meteor 2	Go; Output 2 On
00:02:01.17	Meteor 3	Go; Output 1 + 2 Off; Output 3 On
00:02:13.09	Expanse of Space	Cue 103 Go; Output 3 Off
00:03:10.22	Closing	Cue 105 Go
00:03:44.02	Reset	Playback 1 Clear

[Add New Timecode Event](#)

To add a new Timecode Event, click on the **Add New Timer Event** button.

[Add New Timecode Event](#)

To edit an existing Timecode Event, click on the number of the timecode event.

In both cases, a page similar to the following will appear:

Timecode Event Details	
Timecode	<input type="text" value="00"/> : <input type="text" value="01"/> : <input type="text" value="53"/> . <input type="text" value="23"/>
Event Name	<input type="text" value="Meteor Shower"/>
Command String	<input type="text" value="Q102.1G;O1ON"/> <input type="text" value="Cue 102.1 Go;
Output 1 On"/>
	<input type="button" value="Save"/> <input type="button" value="Delete"/> <input type="button" value="Cancel"/>

In this example, the timecode event has already been configured to trigger at time 00:01:53:23, and it will run Cue 102.1 and also turn on digital output 1.

Each of the Timecode Event's fields are described below:

Timecode

This is the timecode at which the event will trigger. The time is expressed in Hours, Minutes, Seconds and Frames.

Event Name

This is the name of the timer event for your own descriptive use.

Command String

This field holds the CueScript commands that will be executed when the timecode event is triggered.

Timecode Playback

When CueServer begins receiving timecode from an external source (SMPTE or MTC), CueServer will automatically trigger events that appear in the Timecode Triggers list. No additional work or configuration is necessary.

Internal Timecode Generation

CueServer can generate its own timecode clock by using the SMPTE CueScript commands. Using this set of commands, the timecode clock can be reset, started, stopped, cleared and set to an arbitrary time position. See the SMPTE command in the CueScript Commands chapter for details.

DMX Input Triggers

CueServer can trigger events or perform actions based on DMX Input channels.

One of two types of DMX Input triggering can be chosen for each channel:

- **Event Range**
CueScript commands can be triggered when any DMX input channel either enters or exits a range of input values. Multiple event ranges can be set for each channel.
- **Submaster Control**
A DMX Input channel can be assigned to directly control the submaster level of a playback fader.

To view the list of DMX Input Trigger that are programmed into CueServer, view the **DMX Input Triggers** in the **Triggers** page. A page similar to the following will appear (this example includes already defined DMX Input triggers):

Trigger (click to edit)	Name	Channel	Function	Action
1	Neon Show	1	Event Range	Range = [128..255] Enter = [Playback 1 Cue 101 Go] Exit = [Playback 1 Clear]
2	Fiber Show	2	Event Range	Range = [128..255] Enter = [Playback 2 Cue 201 Go] Exit = [Playback 2 Clear]
3	iPlayer Stop	3	Event Range	Range = [0..63] Enter = ["X0100" ~4]
4	iPlayer Show 1	3	Event Range	Range = [64..127] Enter = ["X0101" ~4]
5	iPlayer Show 2	3	Event Range	Range = [128..191] Enter = ["X0102" ~4]
6	iPlayer Show 3	3	Event Range	Range = [192..255] Enter = ["X0103" ~4]
7	Submaster Patch	500	Submaster Control	Playback = [1]

7 of 256 Events Used

[Add New DMX Input Trigger](#)

To add a new DMX Input Trigger, click on the **Add New DMX Input Trigger** button.

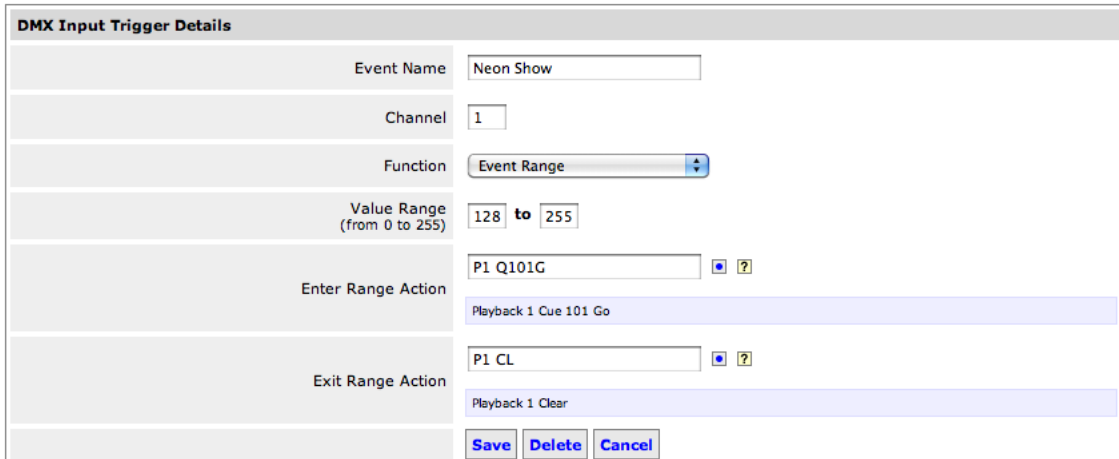
[Add New DMX Input Trigger](#)

To edit an existing DMX Input Trigger, click on the number of the trigger.

In both cases, pages similar to the following will appear.

Event Ranges

When a DMX Input trigger is set to execute CueScript commands based on entering or exiting a range of DMX channel values, the trigger's page appears like this:



The screenshot shows a web interface titled "DMX Input Trigger Details". It contains several input fields and buttons:

- Event Name:** A text input field containing "Neon Show".
- Channel:** A text input field containing "1".
- Function:** A dropdown menu with "Event Range" selected.
- Value Range (from 0 to 255):** Two text input fields containing "128" and "255", separated by the word "to".
- Enter Range Action:** A text input field containing "P1 Q101G" with a small blue square icon and a question mark icon to its right. Below it is a light blue bar containing the text "Playback 1 Cue 101 Go".
- Exit Range Action:** A text input field containing "P1 CL" with a small blue square icon and a question mark icon to its right. Below it is a light blue bar containing the text "Playback 1 Clear".
- Buttons:** Three buttons labeled "Save", "Delete", and "Cancel" are located at the bottom right of the form.

In this example, the DMX Input trigger has been configured to watch DMX Input channel 1 and when it rises to between 128 and 255 (in decimal), it will run Cue 101 on Playback 1. When channel 1 falls to below 128, it will clear Playback 1.

Each of the DMX Input Trigger's fields are described below:

Event Name

This is the name of the timer event for your own descriptive use.

Channel

This is the DMX Input channel that the trigger will watch.

Function

This pop-up menu chooses the trigger type. In this case it is "Event Range".

Value Range

This is the range of values that the trigger will be watching for. Values are expressed in decimal numbers, from 0 to 255.

Enter Range Command

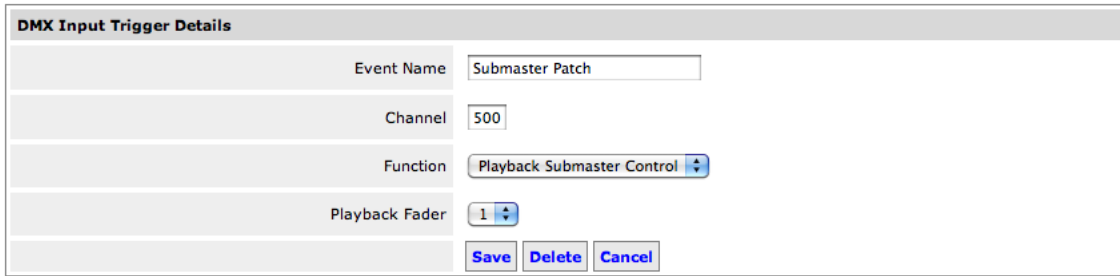
This field holds the CueScript commands that will be executed when the DMX Input on the specified channel enters the specified range of values.

Command String

This field holds the CueScript commands that will be executed when the DMX Input on the specified channel exits the specified range of values.

Submaster Control

When a DMX Input trigger is set to control the submaster level of a playback fader, the trigger's page appears like this:



The screenshot shows a form titled "DMX Input Trigger Details" with the following fields and values:

DMX Input Trigger Details	
Event Name	Submaster Patch
Channel	500
Function	Playback Submaster Control
Playback Fader	1
Save Delete Cancel	

In this example, the DMX Input trigger has been configured to take the input level of Channel 500 and use it to automatically adjust the submaster level of Playback 1.

Each of the DMX Input Trigger's fields are described below:

Event Name

This is the name of the timer event for your own descriptive use.

Channel

This is the DMX Input channel that the trigger will watch.

Function

This pop-up menu chooses the trigger type. In this case it is "Playback Submaster Control".

Playback Fader

This pop-up menu selects which playback fader is being controlled by this trigger event.

System Event Triggers

CueServer has several system events that can be used to trigger various actions. These events include:

- System Startup
- DMX Input Restore
- DMX Input Loss

To display the list of System Event Triggers defined for the current show, view the **System Events** page in the **Triggers** page. A page similar to the following will appear (this example includes already defined event actions):

Startup Action	
Startup Command	<input type="text" value="Macro 1; Cue 101 Go"/> Macro 1; Cue 101 Go
DMX Input Actions	
DMX Input Restore Command	<input type="text" value="Channel * Disable"/> Channel * Disable
DMX Input Loss Command	<input type="text" value="Channel * Enable"/> Channel * Enable
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

In this example, the CueServer has been configured to execute Macro 1 and to begin running Cue 101 when the CueServer is turned on.

Also, if a DMX Input source is attached to CueServer, it will disable the influence of all channels on the output, which will allow the DMX Input to pass through to the DMX Output unchanged. Likewise, if the DMX Input source is removed, all channels will be re-enabled, allowing the show sequence started by Cue 101 to resume normal output.

Each of the System Event fields are described below:

Startup Command

This field holds the CueScript commands that will be executed when the CueServer starts up. A start up is whenever the CueServer is powered on, reboots, or when a new show file is loaded into CueServer's memory.

DMX Input Restore Command

This field holds the CueScript commands that will be executed when a DMX Input source is attached to the CueServer. This command will execute after the Startup Command if DMX Input was present at the time the CueServer starts up.

DMX Input Loss Command

This field holds the CueScript commands that will be executed whenever a DMX Input signal was being received but then it is lost or disconnected.

CueScript Command Language

CueServer uses a command language called CueScript as the basis of nearly all of CueServer's automation capabilities. Using CueScript, advanced logic can be added to a CueServer show enabling the system programmer to orchestrate lighting cues with button presses, contact closure inputs, communication via the serial port and/or Ethernet port, digital outputs, prompts on the LCD display, MIDI commands, and much more.

Using CueScript

Executing Commands

On many of CueServer's interactive web-pages a command line appears at the bottom of the page. Enter in a CueScript command (like "Channel 5 at 33%" or "Record Cue 7") and CueServer performs the requested task.

Although many of CueServer's web pages provide web buttons that perform tasks like setting and releasing DMX channels, changing playback faders and recording cues -- all of these tasks and more can be performed by entering CueScript commands on the command line.

CueScript commands can also be sent to CueServer via Ethernet, the serial port and MIDI Input port, allowing external devices to be able to remotely control CueServer. Additionally, custom web pages can be authored with HTML or Flash to create fully custom front-ends (like an interactive touchscreen interface) for CueServer by sending CueScript commands from a remote web browser window.

Automation with Events and Actions

CueServer defines a wide array of **events** that occur when certain actions occur within the system, such as cues executing, buttons being pressed, contacts being closed, a specific time of the day occurs, timecode markers pass, a sunrise or sunset occurs and much more.

Each of these events can be given an **action**. An action is a series of one or more CueScript statements that cause CueServer to perform a task. CueServer can be told to execute a cue, set lighting levels, illuminate an LED indicator on a button, close a relay, send a serial or MIDI message, customize the labels on the LCD display and more.

The ability to assign any arbitrary action to a wide variety of system-defined events makes CueServer extremely powerful and flexible. CueServer can execute very simple shows without much automation, but it's the event/action model that allows the system programmer to fully customize CueServer for a particular application - as simple or complex as necessary.

CueScript Basics

CueScript is a language that is designed to be easy to understand and efficient for speedy typing and/or transmission. A system programmer will typically use CueScript while interacting with CueServer and will write CueScript statements to provide automation actions for special CueServer events.

Command Syntax

To make it easy to understand, CueScript uses simple human readable nouns, verbs and objects. These pieces are put together into commands such as `Time 5`, which sets the current fade-time to 5 seconds.

Multiple commands can be strung together to make more complex requests. For example, to change the fade time and set a DMX channel to 50% at the same time, the command `Time 5; Channel 3 at 50` is used. Note that a semicolon (;) was used to separate the two commands. Using semicolons are recommended to make multi-command CueScript statements more readable, but they are entirely optional.

White spaces in a command (spaces, tabs, etc.) are ignored by CueServer and are used to simply make the commands more readable. The command from above may be rewritten without spaces like `Time5;Channel3at50`

Also, to make CueScript more efficient to type and/or send, most CueScript command words may be abbreviated. For example, the `Time` command may be abbreviated as just `T` since no other command starts with the letter T. For example, the previous example may be abbreviated as `T5;C3A50`.

Only a few commands can be abbreviated as a single letter. For instance, the `Cue` command shares the same first letter as the `Channel` command. As documented in the descriptions of each of these commands, the shortest abbreviation for `Channel` is `C`, but the shortest abbreviation for `Cue` is `Cu`. However, some commands also have aliases – the `Cue` command can also be invoked by the single letter `Q`. The command `Cue 1 Go` may be abbreviated as `Q1G`.

Command Context

CueServer keeps track of the “context” of the currently executing string of CueScript commands, which allows multiple commands which operate on a single object to be split into separate requests.

When the user types `Channel 1 At 100`, the user is actually executing two separate commands. The first command, `Channel 1` tells CueServer to select DMX channel 1. The second command, `At 100` tells CueServer to set the currently selected objects (DMX channel 1) to 100%.

The selected objects (in this case, DMX channel 1) is part of the saved command context.

If the user then enters the command “`At 75`”, CueServer still has DMX channel 1 selected, so channel 1 will be set to 75%.

The command context stores the selected objects (channels, buttons, outputs, etc.), which playback fader is chosen, timing parameters such as fade and follow times and more.

The command contexts for the interactive web pages, the execution of event actions and the execution of CueScript commands coming in from the serial port, Ethernet and MIDI are all kept separate, allowing several concurrent actions to be running at the same time without interfering with each other.

Levels

The `AT` command and several other objects (like arrays) set levels. Levels are an expression of a quantity from lowest possible value (zero) to highest possible value (full). CueServer allows levels to be expressed in three primary ways, by percentage (the default), or by decimal or hexadecimal notation.

Percentage

By default, when setting DMX channel values, levels are specified by percentage numbers (0, 1, 2, ... 98, 99, 100).

For example, to turn a channel completely off, the command “`Channel 1 At 0`” may be used. To turn a channel completely on, the command “`Channel 1 At 100`” may be used. Any percentage number in-between 0 and 100 can set a channel to the corresponding level.

For convenience, a percent sign (%) may be added to the number for clarity. For example, “`Channel 1 At 50%`”. Using the percent sign is **optional**.

Also for convenience, when specifying a level of 100%, either a value of “100” can be entered or “FL” can be used (meaning “Full”).

Decimal

In some instances, it may be appropriate to use decimal numbers to set DMX channel values (such as when setting levels for moving lights, matching colors or adjusting a level more precisely than percentage levels allow).

Decimal numbers use values from 0 to 255 to specify the range from zero to full.

To use decimal numbers while specifying levels, use a pound sign before the level. For example, “`Channel 1 At #253`”.

Decimal numbers may be used in arrays, such as “`Fixture 1 At {#255, #192, #134}`”.

Hexadecimal

In some instances, it may be appropriate to use hexadecimal numbers to set DMX channel values (such as when setting levels for moving lights, matching colors or adjusting a level more precisely than percentage levels allow).

Hexadecimal numbers use digits 0 through 9 and A through F and values from 00 to FF to specify the complete range from zero to full.

To use hexadecimal numbers while specifying levels, use a dollar sign before the level. For example, “Channel 1 At \$A5”.

Hexadecimal numbers may be used in arrays, such as “Fixture 1 At {\$FF, \$C0, \$86}”.

Note that when specifying hexadecimal numbers to CueServer, always use 2 digits. For example, use \$00, \$01, \$02, not \$0, \$1, \$2, for the single-digit hexadecimal values.

Binary (On/Off)

Some devices being controlled by CueServer only have two states, on and off. In order to simplify their operation, the CueScript language has two extra values named `on` and `off`. These are used as a convenience to mean the same as 0% and 100%.

Any place that a percentage value can be used in a command, the `On` and `Off` commands can be used instead. For example, “Channel 1 On”, “Button 2 Off”, “Group 3 On”, “Fixture 4+5 Off”, “Output * On” are all valid binary-value commands.

Variables

The CueScript language supports user-defined variables in commands. A variable is temporary placeholder for information. Each variable is referenced by name and the variable contains a value. Commands can assign values to variables and then variables can be used in commands where the value of the variable is substituted in place of the variable name.

For example, a variable could be used to hold the desired level of a lighting channel. Each night at sunset a timer executes which sets the level of the channel to the value of the variable. By changing the value of the variable, the channel will be set to a different level each night.

Variables can have any name, as long as the name is made up of only letters and numbers. For example, possible variable names include “x”, “MyVariable” and “Thing123”. Spaces and punctuation characters are not valid in variable names.

The following sections describe in detail how variables are used.

Assigning and Using Variables

In CueScript, variables are assigned values by using the following syntax:

```
"MyNumber" = 35
"MyCommand" = "Cue 1 Go"
"Orange" = "{FL, 50, 0}"
"Greeting" = "Hello"
```

Variables can be assigned virtually any value, either a number or an arbitrary string of characters. Use quotes around values, except that quotes are optional when assigning a simple numerical value.

After a variable has been assigned a value, that variable can be used in any CueScript command. When a variable appears in a command (surrounded by double-curly-braces), that variable's current value is substituted in place of the variable.

For example:

```
Channel 1 At {{MyNumber}}
Playback 1; {{MyCommand}}
Fixture 1 At {{Orange}}
"{{Greeting}}~0
```

Is equivalent to:

```
Channel 1 At 35
Playback 1; Cue 1 Go
Fixture 1 At {FL, 50, 0}
"Hello"~0
```

Variables are not case sensitive. This means that the variable `Orange` is the same as `ORANGE`, `orange`, and `oRaNgE`.

Variable names are limited to a maximum of 31 characters. Variable values are limited to a maximum of 47 characters. Longer names and/or values will be cut off (truncated) when the maximum space has been exceeded.

Note that when a CueScript command contains one or more references to variables (such as `{{Greeting}}`), all of the variable references are replaced with their values before the command is executed. This means that if a single CueScript command string tries to assign a variable and then use it in the same command, the variable reference will use the previous value of the variable.

System Variables

There are several built-in “system variables” that affect the CueServer hardware or internal part of the CueServer operating system. All of these variables start with the underscore character (`_`). Take care not to define your own variables that start with an underscore, as they might conflict with a system variable.

For example, the brightness of the LCD backlight can be adjusted by using the “`_backlight`” variable:

```
"_backlight" = 25
```

This command sets the brightness of the LCD Display (on the CueServer Pro only) backlight to 25%.

See the Appendix on System Variables for a complete listing of user accessible system variables.

System Functions

There are several built-in “system functions” that can be used to generate a value or return the operating condition of part of the CueServer. All of these functions start with the underscore character (`_`). Take care not to define your own variables that start with an underscore, as they might conflict with a system function.

For example, the a system function is available that can generate a random number. This function is accessed by using the “`_rand`” function:

```
Cue {{_rand(1,10)}} Go
```

This instance of the `_rand` function is replaced with a random number from 1 through 10, creating a statement “`Cue x Go`”, where x is the randomly chosen number.

See the Appendix on System Functions for a complete listing of available system functions.

CueScript Commands

At

Syntax: `At (level)`

Abbreviation: `A` or `@`

Description: Sets the level of the selected object(s). Use the `At` command to set lighting levels, output states, button LEDs, etc. `At` can work on one or more selected objects at a time. `At` can set all selected objects to the same value or it can use an array of values (see the next section about the `At` command with array values).

Examples: `At 33`
Sets the level of whatever object(s) (channels, buttons, outputs, etc.) are selected to 33%.

`Channel 7 At 50`
Selects channel 7 and sets it to 50%

`Channel 21>30 At FL`
Selects channels 21 through 30 and sets them to 100%

`Channel 2 At #255`
Selects channel 2 and sets it to decimal value 255

`Channel 3 At $E3`
Selects channel 3 and sets it to hexadecimal value \$E3

`Group 31 At 80`
Selects the channels in Group 31 and sets them to 80%

`Fixture 2 At 85`
Selects the channels for Fixture 2 and sets them to 85%

`Button 1>4 At 1`
Selects the LEDs for buttons 1 through 4 and turns them on

`Output 1+8 At 0`
Selects the digital outputs 1 and 8 and turns them off

`C1AFL`
Abbreviation for Channel 1 At FL

`C1@50`
Abbreviation for Channel 1 At 50%

At (with array value)

Syntax: **At** (*array*)

Abbreviation: **A** or **@**

Description: Like the normal form of the **At** command, but an uses an array of values instead of a single value. Each value in the array is used to set the value of sequential objects. If more objects are selected than values are listed in the array, then the array values repeat.

Examples: Channel 1>3 **At** {25, 50, 75}
Selects channels 1 through 3 and sets them to 25, 50 and 75% respectively

Fixture 1 **At** {33, 50, 66}
Selects the channels of Fixture 1 and sets them to 33, 50 and 66% respectively

Channel 1>10 **At** {0, 100}
Selects channels 1 through 10 and sets them to 0, 100, 0, 100, 0, 100, 0, 100, 0 and 100% respectively

Channel 1+2 **At** {1, 2, 3, 4, 5}
Selects channels 1 and 2 and sets them to 1 and 2% respectively

Channel 1>5 **At** {\$80, #255}
Selects channels 1 through 5 and sets them to hexadecimal \$80 and decimal 255, repeating respectively

Button 1>8 **At** {0, 1}
Selects buttons 1 through 8 and sets the odd buttons off and the even buttons on

F1@{100, 50, 0}
Abbreviation for Fixture 1 at {100, 50, 0}

At (with relative value)

Syntax: `At [+/-] (level)`

Abbreviation: `A` or `@`

Description: Similar to the normal form of the `At` command, but uses a relative offset value instead of an absolute value to set a DMX channel to. Use the `At +/-` command to increase or decrease the level of a DMX channel in increments. Although the `At +/-` command defaults to using percentage values, either decimal or hexadecimal values may also be used with `#` and `$` operators.

Note that when adding or subtracting an offset to selected DMX channels, values do not wrap from Full to Zero or vice versa. For example, if a channel was at 95% and the command `At + 10` was given, the new value would be pinned to 100% (Full).

Also note that relative offsets work with the current fade time. For example, when adding 10% to a selected channel, the new value would normally appear immediately, unless the fade time was set to a longer duration. In this case, the result would be a fade to the new offset value.

Examples:

`Channel 1 At + 10`

Selects DMX channel 1 and sets its level to 10% higher than it was previously

`Channel 10>20 At - 10`

Selects channels 10 through 20 and sets their levels to 10% lower than they were previously

`Channel 1 At + #10`

Selects channel 1 and sets its level to decimal 1 higher than it was previously

`Channel 1 At - $1A`

Selects channel 1 and sets its level to hexadecimal \$1A lower than it was previously

`Time 5; Channel 1 At + 10`

Sets the fade time to 5 seconds, then selects channel 1 and begins a fade to a value 10% higher than it was previously

`At + 5`

Adds 5% to the currently selected DMX channels (without changing the selection)

`C1A+5`

Abbreviation for Channel 1 At + 5

At Cue

Syntax: `At Cue (cue number)`
 `At Cue (cue number) [/ (level)]`

Abbreviation: `AQ` or `@Q`

Description: Similar to the normal form of the `At` command, but recalls the levels stored in a cue to assign to the currently selected channels (or buttons, outputs, etc.). Use the `At Cue` command to recall parts of a pre-recorded look (similarly to how Palettes are used on many lighting consoles, or how Presets are used in architectural control systems). Each selected channel (or button, output, etc.) receives its value from the corresponding (same numbered) channel in the specified cue.

Optionally, a level can be specified (by appending a slash / and a level to the end of the cue number) that will be used to scale the retrieved levels from the cue. For example, to recall levels at 50% of those stored in Cue 1, the command “Group 2 At Cue 1/50” can be used.

Examples: `Channel 5 At Cue 1`
 Selects channel 5 and assigns it to the value of channel 5 as recorded in cue 1

`Channel 1>10 At Cue 1`
 Selects channels 1 through 10 and assigns them to the values of channels 1 through 10 as recorded in cue 1

`Fixture 3 At Cue 1`
 Selects the channels contained in fixture 3 and assigns them to the same channel numbers' values as recorded in cue 1

`Group 4 At Cue 1`
 Selects the channels in group 4 and assigns them to the same channel numbers' values as recorded in cue 1

`Button 1 At Cue 2`
 Selects button 1 and assigns it to the value of channel 1 as recorded in cue 2

`Output 1 At Cue 2`
 Selects output 1 and assigns it to the value of channel 1 as recorded in cue 2

`Group 2 At Cue 1/50`
 Selects the channels in group 2 and assigns them to values 50% of the original brightness as recorded in cue 1.

`C1@Q1`
 Abbreviation for Channel 1 At Cue 1

`F1>5AQ1`
 Abbreviation for Fixture 1 > 5 At Cue 1

At Input

Syntax: `At Input`

Abbreviation: `AIN` or `@IN`

Description: Similar to the normal form of the `At` command, but recalls channel levels that are currently present on the DMX Input jack. Use the `At Input` command to grab or take a snapshot of the DMX Input and store those values into the currently selected playback fader.

The `At Input` command is useful for capturing the current look (or part of the current look) from the DMX Input port. For example, if Playback 1 is in override mode or the DMX Input port is disabled (see the `Input Disable` command), then the DMX Input would not be visible at the CueServer's output. Then, if the command "`Channel * At Input`" is issued, then CueServer would select all channels of Playback 1 and set their levels to the levels currently present at the DMX Input port. This would produce a "snapshot" of the scene being inputted by the external controller. Additionally, a fade time could be set, which would perform the same function, but slowly fade to the snapshot scene.

Another example of the `At Input` command would be to place a command like "`Playback 1 Channel 1>512 At Input`" in the DMX Input Loss event. In this case, whenever the CueServer stops receiving DMX, it will copy the last seen DMX Input levels into Playback 1, which will have the effect of "holding" the last scene sent by the external console.

Examples:

`Channel 1>512 At Input`
Selects all channels of the current playback fader and sets their levels to the current levels at the DMX Input port

`Time 5; Channel * At Input`
Sets the fade time to 5 seconds, selects all channels in the current playback fader and then fades to the levels currently available at the DMX Input port

`Playback 2 Channel 24 At Input`
Sets channel 24 of the Playback 2 to the current input level of channel 24 on the DMX Input port

`Group 3 At Input`
Selects the channels in Group 3 and sets them to the levels currently available at the DMX Input port

`F7AIN`
Abbreviation for Fixture 7 At Input

At Playback

Syntax: `At Playback (playback number)`

Abbreviation: `AP` or `@P`

Description: Similar to the normal form of the `At` command, but recalls channel levels that are currently present in one of the four playback faders. Use the `At Playback` command to grab or take a snapshot of one of the playback faders and store those values into the currently selected playback fader.

The `At Playback` command is useful for capturing the current look (or part of the current look) from one of the playback faders. For example, if Playback 1 contains lighting levels and you want to copy them to Playback 4, the command “`Playback 4 Channel 1>20 At Playback 1`” can be executed.

Examples: `Channel 1>512 At Playback 3`
Selects all channels of the current playback fader and sets their levels to the levels currently in Playback 3

`Time 5; Channel * At Playback 2`
Sets the fade time to 5 seconds, selects all channels in the current playback fader and then fades to the levels currently in Playback 2

`Playback 2 Channel 24 At Playback 1`
Sets channel 24 of the Playback 2 to the current input level of channel 24 in Playback 1

`Fixture 5 At Playback 4`
Selects the channels in Fixture 5 and sets them to the levels of Fixture 5 in Playback 4

`U9AP1`
Abbreviation for Group 9 At Playback 1

Break

Syntax: `Break`

Abbreviation: `BR`

Description: Stops executing the current command string. Any commands that appear after the `Break` command will not be executed.

The `Break` command is useful for dynamically inserting into a command with a variable. For example, a Timer Event's action may be specified as "`{{MyEnable}} Cue 1 Go`". If the variable `MyEnable` is empty when the timer is triggered, then Cue 1 will begin running. If the variable is assigned the value "`Break`", then when the timer is triggered, the cue will not run.

Examples: `Output 1 On Break Output 2 On`
Turns on Output 1, the rest of the command line is not executed

`{{MyEnable}} Macro 1`
Will run Macro 1 if MyEnable variable is empty, will not run Macro 1 if MyEnable variable contains "Break"

`Channel 1 At FL {{DoLED}} Button 1 On`
Sets Channel 1 to Full, also illuminates Button 1's indicator LED if the DoLED variable is empty, doesn't turn on the LED if the DoLED variable contains "Break".

Button

Syntax: `Button (range)`

Abbreviation: `B`

Description: Selects one or more function buttons (the pushbuttons on the front of the CS-800, or connected CueStation buttons). Use the `Button` command to select which buttons will have their LED indicator level set or change enable states, etc.

When specifying button numbers, one of two different methods can be used. Buttons are naturally numbered from 1 to 512. Alternatively, buttons can be referred to by a combination of station number and button number. In the latter method, the station and button numbers are separated by a decimal point. Each station is assumed to have 8 buttons, so Buttons 1 to 8 can be specified as 1.1 to 1.8. Buttons 9 to 16 are equivalent to 2.1 to 2.8. Buttons 17 to 24 are equivalent to 3.1 to 3.8, and so on. When working with many external button stations, this numbering scheme may be easier to work with.

Examples:

`Button 1`
Selects button 1

`Button 1 At FL`
Selects button 1 and turns its LED indicator on

`Button 1>5 On`
Selects buttons 1 through 5 and turns their LED indicators on

`Button * At 0`
Selects all buttons and turns their LED indicators off

`Button 3.4 Off`
Selects button 20 (the 4th button on station 3) and turns its LED indicator off

`Button 2.8 At #235`
Selects button 16 (the 8th button on station 2) and sets its LED indicator value to decimal value 235, which on button station hardware that supports RGB indicators displays as a Fast Flashing Magenta.

`Button 2 Enable`
Selects button 2 and enables its action

`Button 3+4 Disable`
Selects buttons 3 and 4 and disables their actions

`B1@1`
Abbreviation for Button 1 At 1

`B2ENA`
Abbreviation for Button 2 Enable

Note:

The LED indicators on buttons can be set to any value from 0 to 100% (or 0 to 255 in decimal). Several special values cause the LEDs to blink in different patterns, and on hardware that supports RGB indicators, the values between 0 and 255 select different combinations of color, intensity and flashing pattern. The following table shows the basic values used for selecting flash patterns on single-color LED indicators:

- Value 0 = OFF
- Value 1 = ON
- Value 2 = Slow Flash
- Value 3 = Slow Flash, Reverse
- Value 4 = Fast Flash
- Value 5 = Fast Flash, Reverse
- Value 6 = Wink
- Value 7 = Wink, Reverse
- Value 8 to 100 = ON

See the documentation for the RGB button stations for a listing of decimal values (0 to 255) and what colors, intensity and flashing patterns they correspond to.

Channel

Syntax: Channel (*range*)

Abbreviation: C

Description: Selects one or more DMX Channels. Use the Channel command to select which DMX channels will be used to set levels, be released, be recorded into a group or cue and more.

Examples: Channel 1 At 100
Selects channel 1 and sets it to 100%

Channel 1>5 At 75
Selects channels 1 through 5 and sets them to 75%

Channel 3+5+7 At 50
Selects channels 3, 5 and 7 and sets them to 50%

Channel * At FL
Selects ALL 512 channels and sets their levels to 100% (Full)

Channel 5 Release
Selects channel 5 and releases it

Channel 1>10; Record Group 1
Selects channels 1 through 10 and records the selected channels into Group 1

Channel 1>10; Record Cue \$5
Selects channels 1 through 10 and records only the selected channels into Cue 5

C1AFL
Abbreviation for Channel 1 at Full

C1>10@50
Abbreviation for Channel 1 through 10 at 50%

Clear

Syntax: `Clear`

Abbreviation: `CL`

Description: Clears the selected playback fader. All of the fader's channels are released (except for parked channels), and the fade time, follow time, link cue and next cue are all cleared.

See also the `Release` command to selectively release only specific channels and see also the `Reset` command, which entirely resets all playback faders and the command interpreter.

Examples: `Clear`
Clears the currently selected playback fader

`Playback 2 Clear`
Selects Playback Fader 2 and clears it

`P3CL`
Abbreviation for Playback 3 Clear

Contact

Syntax: `Contact (range)`

Abbreviation: `Co`

Description: Selects one or more of the contact closure inputs. Use the `Contact` command to select which contacts will be enabled or disabled. See the `Enable` and `Disable` commands for additional information.

Examples: `Contact 1`
Selects contact 1

`Contact 2 Enable`
Selects contact 2 and enables its action

`Contact 3+4 Disable`
Selects contacts 3 and 4 and disables their actions

`Contact * Enable`
Selects ALL contacts and enables their actions

`CO1ENA`
Abbreviation for Contact 1 Enable

Cue

Syntax: `Cue (cue number)`

Abbreviation: `CU` or `Q`

Description: Selects the specified cue number as the “next cue” in the selected playback fader’s cue stack and loads the timing parameters (fade time, follow time) from the cue into the playback fader. Use the `Cue` command to prepare to execute a specific cue with the `Go` command. Specify any cue number from 0.1 through 4999.9.

When using the `Cue` command with the `Go` command, keep in mind that the `Cue` command queues the next cue along with its timing parameters -- and then the `Go` command executes the next cue with the playback’s settings. This allows you to load a cue and then change some of its parameters before executing it. For example, “`Cue 1 Fade 5 Go`” would run Cue 1 with a fade time of 5 seconds regardless of what fade time was recorded in the cue.

Examples:

`Cue 1`
Selects Cue 1 to be the next cue executed when a Go command is issued

`Cue 2 Go`
Selects Cue 2 as the next cue and runs it

`Cue 9.5 Go`
Selects Cue 9.5 as the next cue and runs it

`Cue 10 Fade 12.5 Go`
Runs Cue 10 with a fade time of 12.5

`Cue 101.2 Fade 5 Follow 12 Link 100 Go`
Runs Cue 101.2 with a fade time of 5 seconds, a follow time of 12 seconds and a link back to Cue 100

`Cue 0`
Clears the next cue from the playback fader

`Q1G`
Abbreviation for Cue 1 Go

Delete

Syntax: Delete (*object*)

Abbreviation: DEL

Description: Deletes a cue or a group. Use the delete command to remove a cue or group from the current show.

Examples: Delete Cue 1
Deletes Cue 1 from the Cue List

Delete Group 2
Deletes Group 2 from the Group List

DELQ1
Abbreviation for Delete Cue 1

DELU2
Abbreviation for Delete Group 2

Device

- Syntax: Device (*range*)
- Abbreviation: DEV or !
- Description: Filters subsequent commands for execution based on the CueServer's assigned Device ID. Use the `Device` command to specify which CueServer(s) on a network should be processing certain commands or to make the same configuration operate differently on different CueServers.

Each CueServer can be assigned a Device ID in the **Hardware Setup > Network Settings** page. While a command is executing, the `Device` command instructs CueServer if the following commands should be executed based on the unit's assigned Device ID.

- Examples: `Device 1; Cue 1 Go`
When this string of commands is executed, the "Device 1" part indicates that the following commands should only be executed if the CueServer's Device ID is set to 1. In this example, "Cue 1 Go" will only be executed if the CueServer is Device 1.

`Device 1>5 + 10>15; Cue 1 Go`
This command enables the following commands if the CueServer has a Device ID from 1 through 5 or 10 through 15.

`Device *; Cue 1 Go`
This command enables ALL device IDs, then executes "Cue 1 Go".

`Device 1; Button 1 At FL; Device 2+3; Button 2 At FL;`
If this unit's Device ID is 1, Button 1 will illuminate. If this unit's Device ID is 2 or 3, Button 2 will illuminate.

By using the `Device` command in conjunction with the command broadcast feature (using the `*` command), specific CueServers may be addressed by command strings broadcast across the network.

- Examples: `"Cue 1 Go" *`
All CueServers on the network will receive the command "Cue 1 Go". Each of these units will execute Cue 1 simultaneously.

`"Device 7; Cue 1 Go" *`
All CueServers on the network will receive the command "Device 7; Cue 1 Go". Only CueServers with their Device ID set to 7 will perform the "Cue 1 Go" part of the command. In effect, this command instructed CueServer #7 to execute Cue 1.

`"Dev 1; Out 1 @ FL; Dev 2; Out 2 @ FL; Dev *; Out 3 @ FL" *`
This command would make CueServer #1 turn on it's Output 1 and CueServer #2 turn on it's Output 2 and all CueServers on the network turn on their Output 3.

Disable

Syntax: `Disable`

Abbreviation: `DIS`

Description: Disables the currently selected Channels, Buttons or Contacts. Use the `Disable` command to temporarily disable the contribution of a fader channel or the action associated with the pressing of a button or closing of a contact. This feature can be used to suspend the output of channels in a playback fader or to “lock-out” certain actions or events from occurring at specific times in a show.

Disabling Channels

If a channel or group of channels in a playback fader are disabled, they no longer affect the output even though they continue to store their value, perform crossfades and are affected by commands and cues. Disable channels in a playback fader to temporarily suspend its contribution to CueServer’s output. To resume inclusion of a channel in CueServer’s output, use the `Enable` command.

Disabling Buttons and Contacts

If a button or contact input is disabled, it no longer responds to being pressed or being closed. Disable buttons or contacts to temporarily prevent additional button presses or contact closures from executing their assigned commands. To resume normal operation of a button or contact, use the `Enable` command.

Examples: `Channel 1 Disable`
Selects channel 1 of the current playback fader and disables it

`Channel 10>30 Disable`
Selects channels 10 through 30 of the current playback fader and disables them

`Playback 3; Channel * Disable`
Selects Playback 3, then selects all channels in Playback 3 and disables them

`Button 1 Disable`
Selects button 1 and disables its action

`Contact 1>8 Disable`
Selects contacts 1 through 8 and disables their actions

`P4C*DIS`
*Abbreviation for Playback 4 Channel * Disable*

`B1DIS`
Abbreviation for Button 1 Disable

Enable

Syntax: Enable

Abbreviation: ENA

Description: Enables the currently selected Channels, Buttons or Contacts. Use the `Enable` command to re-enable channels in a playback fader or the action associated with the pressing of a button or closing of a contact. This feature can be used to reverse the effects of the `Disable` command.

Examples: Channel 1 Enable
Selects channel 1 of the current playback fader and enables it

Channel 10>30 Enable
Selects channels 10 through 30 of the current playback fader and enables them

Playback 3; Channel * Enable
Selects Playback 3, then selects all channels in Playback 3 and enables them

Button 1 Enable
Selects button 1 and enables its action

Contact 1>8 Enable
Selects contacts 1 through 8 and enables their actions

P4C*ENA
*Abbreviation for Playback 4 Channel * Enable*

B1ENA
Abbreviation for Button 1 Enable

Fade

Syntax: `Fade (time or split time)`

Abbreviation: `FA`

Description: Changes the currently selected playback fader's cue crossfade time. Use the `Fade` command to modify the fade time that the next cue on the playback fader's stack will use when the next `Go` command is issued. Also, the `Fade` time is used as the default fade time recorded into a cue when a command such as "`Record Cue 1`" is used.

Times from 0 to 6500 seconds (about 1.8 hours) may be specified in 0.1 second increments. Times may optionally contain a decimal digit.

A single time or split time may be used to specify the fade time. Split times are two time values separated by a "/", for example `1.5/3`. In this example, channels fading up will fade in 1.5 seconds, and channels fading down will fade in 3 seconds. If only a single time is specified, both the up and down fading channels will fade in the same time.

Examples:

`Cue 1; Fade 30; Go`

Loads Cue 1 as the next cue, changes the fader's crossfade time to 30 seconds and then executes the cue (the cue will crossfade in 30 seconds regardless of what fade time is recorded in the cue)

`Cue 1; Fade 1.5/3; Go`

Loads Cue 1 as the next cue, changes the fader's crossfade time to 1.5 seconds for rising channels and 3 seconds for falling channels and then executes the cue (the cue will crossfade in 1.5/3 seconds regardless of what fade time is recorded in the cue)

`Fade 10; Record Cue 7`

Sets the fade time to 10 seconds and then records Cue 7 (which will be recorded with a 10 second fade time)

`FA30`

Abbreviation for Fade 30

Fixture

Syntax: `Fixture (fixture number)`

Abbreviation: `F`

Description: Selects the DMX channels of the given fixture number. Use the Fixture command to select a fixture to set the levels for. Presently, CueServer defaults to a fixed patch of 170 “fixtures”, each with 3 channels, starting at channel 1, which is the typical setup for addressing LED-type RGB fixtures.

Examples: `Fixture 1`
Selects fixture 1 by selecting DMX channels 1 through 3

`Fixture 2+4`
Selects fixtures 2 and 4 by selecting DMX channels 4 through 6 and 10 through 12

`Fixture 3 At 100`
Selects fixture 3 (channels 7 through 9) and sets all three levels to 100%

`Fixture 5 At {100, 25, 75}`
Selects fixture 5 and sets its levels to 100, 25, 75%, which for an RGB fixture is a light purple

`Fixture 1>10 At {FL, FL, 0}`
Selects fixtures 1 through 10 and sets their RGB levels to a bright yellow

`Fixture * At FL`
Selects ALL fixtures and sets their levels to 100% (Full)

`F1@FL`
Abbreviation for Fixture 1 At Full

Follow

Syntax: `Follow (time)`

Abbreviation: `FO`

Description: Sets the amount of time after the next `Go` before the playback fader will automatically issue another `Go` to advance to the next cue. Use `Follow` to specify how long a cue will remain active before it automatically advances to the next cue.

Examples: `Follow 10`
Sets the follow time to 10 seconds

`Follow 5; Go`
Sets the follow time to 5 seconds and then executes the next cue (which will remain active for 5 seconds before automatically advancing to the next cue)

`Follow 15; Record Cue 5`
Sets the follow time to 15 seconds and then records Cue 5 (with a follow time of 15)

`FO15`
Abbreviation for Follow 15

Follow Clear

Syntax: `Follow Clear`

Abbreviation: `FOCL`

Description: Clears the follow timer in the current playback fader. If the follow timer of the current playback is counting down (which will cause an automatic follow to the next cue when it reaches zero), the `Follow Clear` command can be used to clear this timer - which will cause the auto follow to not be taken.

Examples: `Follow Clear`
Clears the follow timer from the current playback fader

`Playback 2 Follow Clear`
Selects playback 1 and then clears the follow timer from playback 2

`P3FOCL`
Abbreviation for Playback 3 Follow Clear

Go

Syntax: Go

Abbreviation: G

Description: Causes the next cue in the current playback fader's cue stack to execute with the current timing parameters and then loads the next cue in the stack into the next cue area.

Examples: Go
Executes the next cue in the current playback fader's cue stack

Cue 12 Go
Loads Cue 12 as the next cue in the playback fader and then executes it

Cue 33; Time 5; Go
Loads Cue 33 as the next cue, changes the fade time to 5 seconds and then executes it

G
Abbreviation for Go

Q1G
Abbreviation for Cue 1 Go

Group

Syntax: `Group (group number)`

Abbreviation: `GR` or `U`

Description: Recalls a set of selected channels or specifies which group to record. Use the `Group` command to work with groups of DMX channels.

Examples: `Group 4`
Recalls the selected channels recorded into Group 4

`Group 5 At 33`
Recalls the selected channels in Group 5 and sets their levels to 33%

`Group 1+3+7 At 75`
Recalls the selected channels in Groups 1, 3 and 7 and sets their levels to 75%

`Group 6 At {0, FL}`
Recalls the selected channels in Group 6 and sets every-other level to 0 and 100%, respectively

`Group 7 Release`
Recalls the selected channels in Group 7 and releases control of those channels

`Record Group 8`
Records the currently selected channels into Group 8

`U4`
Abbreviation for Group 4

`U5@33`
Abbreviation for Group 5 at 33%

HTP (deprecated)

Syntax: `HTP`

Abbreviation: *none*

Description: In previous versions of CueServer software, this command set a playback fader into a mode that performed the same function of the `Merge` command. For best compatibility in the future, use the `Merge` command instead.

IF ... THEN ... [ELSE ...] ENDIF

Syntax: IF *(boolean)* THEN *(statements)* ENDIF
 IF *(boolean)* THEN *(statements)* ELSE *(statements)* ENDIF

Abbreviation: *none*

Description: Allows for the conditional execution of CueScript statements, based on the value of a boolean value.

The boolean value can be a variable that contains a simple value that is either true or false. Any value or variable that is empty or 0 (zero) is considered false, all other values are considered true.

If the boolean value is true, then the statements following THEN and before ELSE or ENDIF are executed.

The IF ... THEN block may optionally contain an ELSE statement. If the boolean value is false, then the statements following ELSE and before ENDIF are executed.

Any additional CueScript statements that occur after the ENDIF are executed normally.

Examples: IF {{enabled}} THEN Cue 1 Go ENDIF
 Executes "Cue 1 Go" only if "enabled" is true (not empty or zero)

 IF {{showMode}} THEN Macro 1 ELSE Macro 2 ENDIF
 Executes "Macro 1" if "showMode" is true and "Macro 2" if "showMode" is false

 IF {{test}} THEN Macro 99 ENDIF Cue 1 Go
 Executes "Macro 99" if "test" is true and then always executes "Cue 1 Go" regardless of the value of "test"

Input [Enable | Disable]

Syntax: `Input [Enable | Disable]`

Abbreviation: `IN [ENA | DIS]`

Description: Controls the influence of the DMX Input port on the playback faders. By executing `Input Disable`, the influence of the DMX Input on the playback faders can be disabled. This influence can be re-enabled by executing `Input Enable`.

When the DMX Input port is disabled, it continues to receive DMX values, but those values do not enter the stack of playback faders. The DMX Input continues to be visible in the Input tab of the Stage view, operate DMX Input Triggers, available to the `At Input` command and others.

Examples: `Input Disable`
Disables the influence of the DMX Input on the playback faders

`Input Enable`
Enables the influence of the DMX Input on the playback faders

`INDIS`
Abbreviation for Input Disable

`INENA`
Abbreviation for Input Enable

Input Update

Syntax: `Input Update`

Abbreviation: `INUP`

Description: Forces all DMX Input Triggers to execute their appropriate action commands when CueServer receives the next DMX packet from an external source.

Use `Input Update` to make sure all DMX Input triggers are “up-to-date” when switching modes, changing overrides or in other situations where the normal automatic triggering of DMX Input Triggers is not sufficient.

Examples: `Input Update`
Execute the actions of all DMX Input Triggers.

`INUP`
Abbreviation for Input Update

Join

Syntax: `Join (station number)`

Abbreviation: *none*

Description: Joins the selected stations with the specified stations. Use the `Join` command to logically connect stations together so that a button press on one station activates the same button one or more other stations. Use with the `Station` and `Unjoin` commands.

Note that joined stations cascade, meaning that if Station 1 is joined with Station 3 and Station 3 is Joined with Stations 5 and 7, then a press on any of these stations will activate the button on all four stations.

Examples: `Station 1 Join 2`
Joins actions on Station 2 with Station 1

`Station 1+3+5 Join 10`
Joins actions on Station 10 with those on Stations 1, 3 and 5

Link

Syntax: `Link (cue number)`

Abbreviation: `L`

Description: Sets the cue number of the cue that should follow the current cue. Use the `Link` command to override the normal sequential execution of cues in the Cue List. If a cue does not specify a link cue, the next cue in the list will follow the current cue. If a cue has a link to another cue in the Cue List, then the linked cue will be the next cue to execute after the current one.

Examples: `Link 101`
Sets the link cue to Cue 101

`Link 1; Go`
Changes the next cue's link cue to Cue 1 and then executes the next cue in the stack. A subsequent Go will execute Cue 1.

`Cue 10; Link 5; Go`
Loads Cue 10 as the next cue, changes the cue's link to Cue 5, then executes Cue 10. A subsequent Go will execute Cue 5.

`L101`
Abbreviation for Link 101

Log

Syntax: `Log (string)`

Abbreviation: *none*

Description: Writes a message to the System Log page. Useful for indicating that a certain event has occurred. Logged messages are recorded into the system log with the current date and time.

Note: The System Log page can be viewed by choosing the **System Log** link in the **Hardware Setup** page.

Examples: `Log "Show Started Manually"`
Adds the message "Show Started Manually" to the system log

`Log "Variable x = {{x}}"`
Adds the message "Variable x = ???" to the system log, with ??? replaced by the actual value of x (see the section on CueScript variables)

LTP (deprecated)

Syntax: `LTP`

Abbreviation: *none*

Description: In previous versions of CueServer software, this command set a playback fader into a mode that performed the same function of the `Override` command. For best compatibility in the future, use the `Override` command instead.

Macro

Syntax: Macro (*macro number*)

Abbreviation: M

Description: Executes the specified macro. Use the Macro command to temporarily suspend the current command string while CueServer executes the commands contained in the specified macro. When the macro's commands are completed, command execution resumes with the original command.

Examples: Macro 7
Executes Macro 7. Assuming that Macro 7 contained the commands "Playback 3; Cue 7 Go", then by executing Macro 7, cue number 7 would begin executing in playback fader number 3.

Macro 2; At FL
Assuming Macro 2 contains the commands "Button 2 + 4 + 6 + 8", this command selects the even-numbered front-panel buttons and turns their LED indicators On.

M100
Abbreviation for Macro 100

M2@FL
Abbreviation for same example #2 above

Merge

Syntax: Merge

Abbreviation: ME

Description: Changes the selected playback fader to Merge Mode. When a playback fader is in Merge mode, each of its active channel values are combined with the channel values of the output stages that come before it using a “Highest Takes Precedence” (HTP) strategy. Each playback fader may be in one of Merge, Override or Scale modes. All playback faders default to Merge mode on power-up.

For example, if Playback 3 is in Merge mode, then any channel levels that are brought up in Playback 3 will be combined with any values coming from the DMX input or Playback 1 or 2. The result of a Merge between playback faders is that the highest level takes precedence. To cancel the effects of a channel, use the Release command.

By default, all playback faders are in Merge mode. If a channel in Playback 1 has a level of 33% and the same channel in Playback 2 has a level of 75% and Playback 3 has a level of 50% and Playback 4 has a level of 66%, then the resulting output on that channel would be 75%, which is the highest level of each of the playback faders.

Note that the Merge command is the same as the HTP command, which was present in earlier versions of CueServer software. For best compatibility in the future, use Merge instead of HTP.

See the chapter about DMX Output for additional information about the Merge, Override and Scale modes.

Examples: Merge
Changes the currently selected playback fader to Merge Mode

Playback 1 Merge
Selects Playback 1 and changes it to Merge Mode

P4ME
Abbreviation for Playback 4 Merge

Next

Syntax: `Next`

Abbreviation: `++`

Description: Shifts the current selection to the next object(s). The `Next` command operates on Channels, Fixtures, Buttons, Contacts and Outputs.

For example, if Button 1 is currently selected, choosing `Next` will increment to Button 2. If multiple objects are selected, they are all incremented. For example, if Channels 1+4+7 are selected, after a `Next`, Channels 2+5+8 will be selected. Blocks of objects are treated as a single entity that are moved forward. For example, if Fixtures 1>5 are selected, then after a `Next`, Fixtures 6>10 will be selected.

Also see the `Previous` command.

Examples: `Channel 1 At 10 Next At 20 Next At 30`
Selects channel 1 and sets it to 10%, then selects channel 2 and sets it to 20% then selects channel 3 and sets it to 30%

`Fixture 5 At FL Next At 75 Next At 50`
Selects the channels of Fixture 5 and sets them to 100%, then selects the channels of Fixture 6 and sets them to 75%, then selects the channels of Fixture 7 and sets them to 50%

`Button 1>4 On Next Off`
Selects Buttons 1 through 4 and sets their indicator LEDs on, then selects Buttons 5 through 8 and sets their indicator LEDs off

Off

Syntax: `Off`

Abbreviation: *none*

Description: A special CueScript token that has the same meaning as “At 0%”. Optionally use `off` in instances where it is more syntactically correct (i.e.: it reads better). For example, instead of the command “`Button 1 At 0`”, you can use the command “`Button 1 Off`”.

Examples: `Button 1 Off`
Selects Button 1 and then sets its LED indicator's level to 0% (or “Off”)

`Channel 3 Off`
Selects DMX Channel 3 and sets its output level to 0% (or “Off”)

`Group 5 Off`
Selects the channels included in Group 5 and sets their output levels to 0% (or “Off”)

`Fixture 7 Off`
Selects the channels included in Fixture 7 and sets their output levels to 0% (or “Off”)

On

Syntax: `On`

Abbreviation: *none*

Description: A special CueScript token that has the same meaning as “At 100%”. Optionally use `on` in instances where it is more syntactically correct (i.e.: it reads better). For example, instead of the command “`Button 1 At 100`”, you can use the command “`Button 1 On`”.

Examples: `Button 1 On`
Selects Button 1 and then sets its LED indicator's level to 100% (or “On”)

`Channel 3 On`
Selects DMX Channel 3 and sets its output level to 100% (or “On”)

`Group 5 On`
Selects the channels included in Group 5 and sets their output levels to 100% (or “On”)

`Fixture 7 On`
Selects the channels included in Fixture 7 and sets their output levels to 100% (or “On”)

Output

Syntax: Output *(range)*

Abbreviation: 0

Description: Selects one or more digital outputs. Use the `Output` command to select which outputs' levels will be set.

Examples: Output 1
 Selects output 1

 Output 1 At 0
 Selects output 1 and turns it off

 Output 1>5 At FL
 Selects output 1 through 5 and turns them on

 Output 2 At 4
 Selects output 2 and sets it value to a fast flashing pattern

 Output * At FL
 Selects ALL outputs and turns them on

 O1@FL
 Abbreviation for Output 1 At Full

 O1ENA
 Abbreviation for Output 1 Enable

Note:

Each digital output can be set to any value from 0 to 100%. Several special values cause the outputs to flash in different patterns. The following table shows which special values produce certain flashing patterns.

- Value 0 = OFF
- Value 1 = ON
- Value 2 = Slow Flash
- Value 3 = Slow Flash, Reverse
- Value 4 = Fast Flash
- Value 5 = Fast Flash, Reverse
- Value 6 = Wink
- Value 7 = Wink, Reverse
- Value 8 through 100 = ON

Override

Syntax: `Override`

Abbreviation: `OV`

Description: Changes the selected playback fader to Override Mode. When a playback fader is in Override mode, each of its active channel values take precedence over the channel values of the output stages that come before it. Each playback fader may be in one of Merge, Override or Scale modes.

For example, if Playback 2 is in Override mode, then any channel levels that are brought up in Playback 2 will take precedence over any values coming from the DMX input or Playback 1. To cancel the effects of a channel, use the `Release` command.

A useful function of playback faders in Override mode is to allow for temporary overrides of CueServer's output. For example, if Playback 4 is in Override mode, then the rest of the playback faders can be used to control a show. Then, in the case where emergency lighting is needed, a cue can be played in Playback 4, which will entirely override the levels coming from earlier output stages. When the show needs to be resumed, Playback 4 can be released (with the `Release` command), which would remove Playback 4's influence on the output.

Note that the `Override` command is the same as the `LTP` command, which was present in earlier versions of CueServer software. For best compatibility in the future, use `Override` instead of `LTP`.

See the chapter about DMX Output for additional information about the Merge, Override and Scale modes.

Examples: `Override`
Changes the currently selected playback fader to Override Mode

`Playback 2 Override`
Selects Playback 2 and changes it to Override Mode

`P1OV`
Abbreviation for Playback 1 Override

Park

Syntax: `Park`

Abbreviation: *none*

Description: Parks the currently selected DMX channels in the currently selected playback fader. When a channel is parked, it is no longer affected by commands or any type of cue playback. For example, when one or more channels are parked, their output levels become frozen and do not change when a cue is executed or commands like `Channel`, `At` or `Release` are used, or when a streaming DMX cue is playing. The `Park` command is useful for implementing room partitioning, conditional overrides or other zoning features.

Channel parking occurs only within the currently selected playback fader.

Use the `Unpark` command to reverse the effects of parking a channel.

Examples:

`Channel 1 Park`
Selects channel 1 and parks it

`Channel 1>10 Park`
Selects channels 1 through 10 and parks them

`Group 3 Park`
Selects the channels included in Group 3 and parks them

Playback

Syntax: `Playback (playback number)`

Abbreviation: `P`

Description: Selects one of the four playback faders as the target for all subsequent commands. Use the `Playback` command to change which of the playback faders is currently selected. Playback fader numbers from 1 to 4 may be used.

Use the `Playback` command with the `At` command to set the submaster level for a playback fader. For example, `Playback 1 At 50` sets the submaster of playback 1 to 50%. When changing submaster levels, CueServer uses the current Fade Time to ramp the submaster level up or down.

Examples: `Playback 1`
Selects playback fader number 1

`Playback 2 Channel 1 @ FL`
Selects playback 2, then sets DMX channel 1 to 100% (in playback fader 2)

`Playback 3 Release; Playback 4 Go`
Releases the selected channels in playback 3, then executes the next cue in playback 4

`Playback 4 At 33`
Sets the submaster level of Playback 4 to 33%, using the current fade time (if any)

`P3Z`
Abbreviation for Playback 3 Release

Previous

Syntax: `Previous`

Abbreviation: `PREV` or `--`

Description: Shifts the current selection to the previous object(s). The `Previous` command operates on Channels, Fixtures, Buttons, Contacts and Outputs.

For example, if Button 4 is currently selected, choosing `Previous` will decrement to Button 3. If multiple objects are selected, they are all decremented. For example, if Channels 2+5+8 are selected, after a `Previous`, Channels 1+4+7 will be selected. Blocks of objects are treated as a single entity that are moved backwards. For example, if Fixtures 6>10 are selected, then after a `Previous`, Fixtures 1>5 will be selected.

Also see the `Next` command.

Examples: `Channel 5 At 10 Previous At 20 Previous At 30`
Selects channel 5 and sets it to 10%, then selects channel 4 and sets it to 20% then selects channel 3 and sets it to 30%

`Fixture 7 At FL Previous At 75 Previous At 50`
Selects the channels of Fixture 7 and sets them to 100%, then selects the channels of Fixture 6 and sets them to 75%, then selects the channels of Fixture 5 and sets them to 50%

`Button 5>8 On Previous Off`
Selects Buttons 5 through 8 and sets their indicator LEDs on, then selects Buttons 1 through 4 and sets their indicator LEDs off

Reboot

Syntax: `Reboot`

Abbreviation: *none*

Description: Reboots the CueServer. All running shows will immediately be stopped and the CueServer will reboot as if it had been turned off and back on.

Examples: `Reboot`
Immediately reboots the CueServer

Record Cue

Syntax: Record Cue *[modifier] (cue number)*

Abbreviation: RQ

Description: Records a standard (static) cue into memory with the given cue number. Any cue number from 0.1 through 4999.9 may be used.

When the Record Cue command is executed, a cue is stored with the current channel values being output from CueServer with the current Fade Time, Follow Time and Link Cue.

By default, all 512 channels being output from CueServer are recorded into the new cue. To record only a selected subset of channels, the dollar-sign (\$) may be used as a modifier before the cue number to indicate that only the currently selected channel values be recorded into the cue. Additionally, to record a cue with ***no*** channel values, the pound sign (#) may be used as a modifier before the cue number to indicate that no DMX channel values should be recorded into the cue.

Examples: Record Cue 1
Records the current DMX channel levels and timing parameters into Cue 1

Fade 10 Follow 20 Record Cue 1.5
Records the current DMX channel levels into Cue 1.5 with a Fade Time of 10s and a Follow Time of 20s

Channel 1>10 Record Cue \$2
Selects channels 1 through 10 and then records Cue 2 with only channels 1 through 10 included in the cue

Group 1 Record Cue \$7.5
Selects channels in Group 1 and then records Cue 7.5 with only Group 1's channels included in the cue

Record Cue #3
Records Cue 3 without any DMX channels included in it

RQ1
Abbreviation for Record Cue 1

Record Group

Syntax: `Record Group (group number)`

Abbreviation: `RU`

Description: Records a channel group into memory with the given group number. When a group is recorded, the currently selected set of channels is stored in the group. Any group number from 1 to 999 may be used.

A group contains a collection of channels. You can use a group any time you want to recall a specific set of channels. See the `GROUP` command to recall and use the channels stored in a group.

Examples: `Record Group 1`
Records the currently selected channels into Group 1

`Channel 1+3+5+7 Record Group 2`
Selects channels 1, 3, 5 and 7 and records Group 2 containing those channels

`Fixture 1>3 Record Group 3`
Selects Fixtures 1 through 3 and then records Group 3 containing those channels

`RU1`
Abbreviation for Record Group 1

Record Stop

Syntax: `Record Stop`

Abbreviation: `RSTO`

Description: Stops any stream that is currently being recorded. See the `Record Stream` command for additional details about stream recording.

Examples: `Record Stop`
Stops any stream that is currently being recorded

`RSTO`
Abbreviation for Record Stop

Record Stream

Syntax: `Record Stream (cue number)`

Abbreviation: `RSTR`

Description: Records a streaming DMX cue into memory with the given cue number. When a streaming cue is recorded, CueServer records the live DMX input like a tape recorder for identical playback later. Any cue number from 0.1 to 4999.9 may be used. Streaming cues are stored in the same cue list as standard cues.

Streams can be recorded from CueServer's web pages or by using this CueScript commands. Although the Record Stream command only takes one parameter (the cue number), there are several other parameters that affect stream recording which can be set by setting the value of system variables.

The trigger channel used for automatic starting and stopping of stream recording is stored in the system variable "`_triggerchannel`". This variable can be set to the desired trigger channel (or zero for no trigger channel) before beginning a stream recording.

The automatic stopping of stream recording at a specific time may be specified by the "`_streamlength`" system variable. Set this variable to a number of seconds (accurate to 0.1s) or to zero for unlimited recording duration.

Depending on the values of these parameters, when the Record Stream command is issued, a streaming cue may begin recording immediately, or CueServer will wait until the trigger channel rises above zero, or CueServer will wait for DMX Input to be present.

Streaming cue recording will stop when the trigger channel falls back to zero (if specified), or when the stream duration is reached (if specified), or if the DMX Input is removed, or if the `Record Stop` command is received, or if available memory is exhausted.

Examples: `Record Stream 1`
Begins recording Streaming Cue 1 with the current system parameters

```
"_triggerchannel"="512"
```

```
"_streamlength"="15.5"
```

`Record Stream 2`

Sets the trigger channel to channel 512, then sets the maximum stream length to 15.5 seconds, then begins recording streaming cue 2

`RSTR1`

Abbreviation for Record Stream 1

Release

Syntax: `Release`

Abbreviation: `REL` or `Z`

Description: If DMX channels are selected, releases the selected channels in the current playback fader and then clears the selection. If no DMX channels are selected, releases all channels in the current playback fader. If a streaming cue is playing back, aborts the playback. When channels are released, they no longer contribute to the DMX output of the playback fader in any way. Parked channels are not affected by the `Release` command.

See also the `Clear` command, which unconditionally releases all channels in a playback fader and clears the playback's parameters. Also see the `Reset` command, which reinitializes all playbacks and the command interpreter.

Examples: `Release`
Releases the selected channels in the current playback fader (or all channels if no channels were selected)

`Release Release`
Regardless of whether there were channels selected or not, releases all channels in the current playback fader

`Channel 5+7 Release`
Selects channels 5 and 7, then releases these channels (clearing the selection)

`Group 3 Release`
Selects the channels in Group 3 and then releases them

`C1Z`
Abbreviation for Channel 1 Release

`ZZ`
Abbreviation for Release Release

Reset

Syntax: `Reset`

Abbreviation: *none*

Description: Resets **all** playback faders and the command interpreter their default values. As a result, all DMX playback is stopped, selections are cleared and Playback 1 is selected. Parked channels are also cleared by this command. CueServer is returned to its “power-on” state. Timecode and the command queue are also reset by this command.

Examples: `Reset`
Resets CueServer to its “power-on” state

Scale

Syntax: `Scale`

Abbreviation: `SC`

Description: Changes the selected playback fader to Scale Mode. When a playback fader is in scale mode, it uses its active channel values to proportionally scale the values of the output stages that come before it. Using the Scale mode, the programmer can create inhibitive submasters or a grand master for the system. Each playback fader may be in one of Merge, Override or Scale modes.

For example, if Playback 1 is in Scale mode, then it scales the DMX input by the percentage of each level in Playback 1. If Playback 2 is in Scale mode, then it scales the result of the DMX input and Playback 1. If Playback 4 is in Scale mode, then it scales the result of the DMX input and Playback 1 through 3. To cancel the effects of a channel, use the `Release` command.

If Channel 1 of Playback 4 is set to 50% and Playback 4 is set to Scale mode, then it will scale the levels of the output stages that come before Playback 4 (the DMX input and Playbacks 1 through 3). If the output of CueServer through Playback 3 was 60%, then Playback 4 would scale it by 50%, resulting in an output from Playback 4 of 30%.

See the chapter about DMX Output for additional information about the Merge, Override and Scale modes.

Examples: `Scale`
Changes the currently selected playback fader to Scale Mode

`Playback 3 Scale`
Selects Playback 3 and changes it to Scale Mode

`P4SC`
Abbreviation for Playback 4 Scale

Self

Syntax: `Self`

Abbreviation: `SE`

Description: Refers to the Button or Contact object that executed the current command. Use `Self` to operate on the current object without knowing which object initiated the command. `Self` is useful for setting LED indicators, or enabling and disabling the object.

The `Self` command works within Macros and other subroutines. When used outside of the context of a button or contact, `Self` has no value and performs no function.

Examples: `Self On`
Turns on the LED indicator of the button or contact that was pressed

`Self At 4`
Sets the value of the pressed button or contact to 4

`Self Disable`
Disables the button or contact that was pressed

`SEON`
Abbreviation for Self On

SMPTE

Syntax: SMPTE (Reset | Start | Stop | Clear | *(timecode)*)

Abbreviation: SMPTE (RESET | STA | STO | CL | *(timecode)*)

Description: The SMPTE commands manually control the internal generation of SMPTE Timecode. Commands are available to reset, start, stop, clear and to set the timecode to an arbitrary time.

While timecode is being generated internally, actions in the Timecode Event Trigger List will be executed at their specified times.

Examples: SMPTE Reset
Stops timecode generation and resets the timecode position to "00:00:00.00"

SMPTE Start
Begins or resumes internal generation of timecode from the current time position

SMPTE Stop
Pauses internal generation of timecode at the current time position

SMPTE Clear
Stops timecode generation and returns the timecode position to none "--:--:--.--"

SMPTE "01:23:45.30"
Sets the timecode position to a user-specified time (in hours, minutes, seconds and frames), in this example 1 hour, 23 minutes, 45 seconds and the 30th frame.

Start

Syntax: `Start`

Abbreviation: `STA`

Description: Resumes normal timing operation of the currently selected playback fader. Use the `Start` command to reverse the operation of the `Stop` command. `Start` resumes any fades that are paused, and resumes any follow timers that were suspended. See the `Stop` command for additional details.

Examples: `Start`
Resumes any halted fades and/or follow timers for the currently selected playback fader

`STA`
Abbreviation for Start

Station

Syntax: `Station (range)`

Abbreviation: `STAT`

Description: Selects a station for use with the `Join` and `Unjoin` commands. Can be used with the `+`, `-` and `>` operators to extend the selection.

Examples: `Station 1 Join 2`
Selects Station 1 and then Joins it with Station 2

`Station 1>10 + 20 Join 30`
Selects Stations 1 through 10 and 20 and then Joins them with Station 30

Stop

Syntax: Stop

Abbreviation: STO

Description: Suspends timing operation of the selected playback fader. If a cue was in the process of fading, the fade is immediately halted and if a follow timer was running, it is halted. The `Start` command can be used to resume the fade and follow.

If a new cue is executed (using the `Go` command) while the playback fader is stopped, the cue's levels appear on stage immediately (without observing any fade time) and the follow timer is disabled. This feature is useful for loading and editing cues that have fade and/or follow times that would otherwise make it difficult to edit the cue's levels.

Examples: Stop
Halts any fades and/or follow times currently executing on the selected playback fader, and causes the playback fader to operate without any timing functions until a subsequent Start command resumes normal operation

STO
Abbreviation for Stop

Time

Syntax: `Time` (*time or split time*)

Abbreviation: `T`

Description: Specifies the time in which DMX channels move towards their destination values when set with the `At` command. A fade time of 0 (zero) indicates that all channels will immediately snap to their specified values. See also the `Fade` command.

Times from 0 to 6500 seconds (about 1.8 hours) may be specified in 0.1 second increments. Times may optionally contain a decimal digit.

A single time or split time may be used to specify the fade time. Split times are two time values separated by a “/”, for example `1.5/3`. In this example, channels fading up will fade in 1.5 seconds, and channels fading down will fade in 3 seconds. If only a single time is specified, both the up and down fading channels will fade in the same time.

Examples: `Time 5`
Sets the live fade time to 5 seconds

`Time 70.5`
Sets the live fade time to 1 minute, 10.5 seconds

`Time 10; Channel 1 At 50`
Sets the live fade time to 10 seconds, then causes DMX channel 1 to fade to 50% over 10 seconds

`Time 1.5/3`
Sets the live fade time for rising channels to 1.5 seconds and falling channels to 3 seconds.

`T5`
Abbreviation for Time 5

Toggle

Syntax: `Toggle (level)`

Abbreviation: `TOG`

Description: Toggles the level of a channel, fixture, group, button, output or submaster between 0 (zero) and the specified level.

If the level of the selected object is equal to the specified level, it is set to 0 (zero), otherwise it is set to the specified level.

For example, to toggle a channel on and off, then perform a “Channel 1 Toggle FL”. Or, to toggle a channel between zero and 50%, perform a “Channel 1 Toggle 50”.

Examples: `Channel 1 Toggle FL`
Toggles channel 1 between 0 (zero) and FL (full)

`Group 1 Toggle 33`
Toggles the channels in group 1 between 0 (zero) and 33%

`Button 1 Toggle 1`
Toggles the LED indicator on button 1 between 0 (zero, off) and 1 (on)

`Playback 1 Toggle FL`
Toggles the output of playback 1 between 0 (zero) and FL (full)

`O1TOGFL`
Abbreviation for Output 1 Toggle FL

Unjoin

Syntax: `Unjoin (station number)`

Abbreviation: *none*

Description: Removes the connection between the selected stations and the specified stations. Use the `Unjoin` command to reverse the effect of a previous `Join` between stations. Use with the `Station` and `Join` commands.

Examples: `Station 1 Unjoin 2`
Removes the connection between Station 2 and Station 1

`Station 1+3+5 Unjoin 10`
Removes the connections between Station 10 with those on Stations 1, 3 and 5

Unpark

Syntax: `Unpark`

Abbreviation: *none*

Description: Unparks the currently selected DMX channels in the currently selected playback fader. Unparking one or more channels reverses the effect of parking a channel.

See the `Park` command for additional information.

Examples: `Channel 1 Unpark`
Selects channel 1 and unparks it

`Channel 1>10 Unpark`
Selects channels 1 through 10 and unparks them

`Group 3 Unpark`
Selects the channels included in Group 3 and unparks them

Update Cue

Syntax: `Update Cue [option] (cue number)`

Abbreviation: `UQ`

Description: Updates the specified cue. Use the `Update` command to update the DMX channel levels of a cue without affecting any of the other parameters of the cue (such as the cue's name, fade time, follow time, link, action, etc.).

The dollar-sign (\$) option may be used before the cue number to indicate that only the selected channel values be re-recorded into the cue. In this mode, non-selected channels will be removed from the cue.

The tilde (~) option may be used before the cue number to indicate that the currently selected channels should be updated in the cue without updating any other channels that may have already existed in the cue.

Examples: `Update Cue 1`
Re-records the current channel levels into Cue 1

`Channel 1>512@FL; Update Cue 2`
Changes Cue 2 to have all channels set to 100% (while maintaining the cue's name, timing, aciton, etc.)

`Channel 5; Update Cue $3`
Re-records the DMX channel data for Cue 3, but only for channel 5

`Group 1; Update Cue ~4`
Updates only the channels in Group 1 in Cue 4 without disturbing any other channels in Cue 4 that may have previously existed

`UQ1`
Abbreviation for Update Cue 1

Wait

Syntax: `wait (Delay) | clear`

Abbreviation: `w`

Description: Inserts a delay into command execution. Use the `wait` command to cause CueServer to temporarily suspend execution of the current command for the specified number of seconds.

When the `wait` command is encountered, CueServer stores the remainder of the command in a special Wait Queue. When the corresponding timer expires, the queued command is executed.

Also, the command `wait clear` may be executed, which clears **all** currently waiting commands in the Wait Queue.

Examples: `Button 1 On; Wait 10; Button 1 Off`
Turns the LED indicator on Button 1 on, then 10 seconds later, the LED turns off

`Channel 1 At 33; Wait 2.5; At 66; Wait 2.5; At FL`
Selects channel 1 and set it to 33%, then waits 2.5 seconds and sets it to 66%, then waits 2.5 more seconds and sets it to 100%

`Cue 1 Go; Wait 60; Release`
Begins playing Cue 1, then 60 seconds later the playback fader is released

`Wait Clear`
Removes all currently waiting commands from the Wait Queue

`C1AFLW1A0`
Abbreviation for Channel 1 At 100% Wait 1 At 0

Note: CueServer's Wait Queue for delayed commands may contain up to eight (8) simultaneously suspended commands. This means that only 8 command-delay timers may be active at a time. If this limit is exceeded, CueServer will post a system log message indicating the limitation has been met.

In the above examples, only 3 of the 8 queue entries are used (with 10, 2.5, and 60 second timers, respectively) if all three example commands are executed at the same time. The second 2.5 second timer does not become active until the first 2.5 second timer expires, so it does not require additional queue resources until the first 2.5 second timer is no longer in use.

+ (and)

Syntax: + (*range*)

Abbreviation: *none*

Description: Used to extend the current selection. Use the + command to add additional objects to the selection while specifying a range of objects. The add command can be used in-line with other ranges or completely by itself in order to add additional objects to the current selection.

Examples: Channel 1 + 5
Selects channels 1 and 5

Button 1>3 + 6>8
Selects buttons 1 through 3 and 6 through 8

Output 1+3+5+7
Selects the odd numbered outputs

+200
Adds object 200 to the previously selected group of objects

- (except)

Syntax: - (*range*)

Abbreviation: *none*

Description: Used to remove objects from the current selection. Use the - command to exclude objects from the selection while specifying a range of objects. The except command can be used in-line with other ranges or completely by itself in order to remove objects from the current selection.

Examples: Channel 1>10 - 5
Selects channels 1 through 4 and 6 through 10

Button 1>8 - 3>5
Selects buttons 1 through 2 and 6 through 8

Output 1>8 - 2 - 4 - 6 - 8
Selects the odd numbered outputs

-33
Subtracts object 33 from the previously selected group of objects

> (through)

Syntax: > (*number*)

Abbreviation: *none*

Description: Used to specify a range of objects. Use the > command to indicate multiple sequential objects at once. The through command can be mixed with + and - commands.

Examples: Channel 33 > 96
Selects channels 33 through 96

Button 1 > 8 at 0
Selects buttons 1 through 8 and turns their LEDs off

Output 1 > 3 + 6 > 8 @ FL
Selects outputs 1 through 3 and 6 through 8 and turns them on

* (wildcard)

Syntax: (*object*) *

Abbreviation: *none*

Description: Used to select all objects of a given type. Use the * character to specify that all objects of the chosen type should be selected. Works with Buttons, Channels, Contacts, Devices, Fixtures, Outputs and Stations.

Examples: Button * At 1
Selects ALL buttons and turns their LED indicators on.

Channel * At FL
Selects ALL channels and sets their levels to 100% (Full)

Contact * Enable
Selects ALL contacts and enables their actions

Device *
Selects ALL devices for operation on subsequent commands

Fixture * At 50
Selects ALL fixtures and sets their levels to 50%

Output * At 0
Selects ALL outputs and turns them off

Button * - 3 At 0
Selects ALL buttons except for Button 3 and turns their LED indicators off.

Channel * - 10 > 20 At 33
Selects ALL channels except for channels 10 through 20 and sets their DMX levels to 33%.

“ ” * (command broadcast)

Syntax: “(string)” *

Description: The string/asterisk command is used to broadcast a command string to all CueServers on the local Ethernet network (including the CueServer that the command was executed from). Use this command to make all CueServers in a networked environment perform the same action at the same time.

Use the command broadcast feature in conjunction with the `Device` command to sent commands to specific CueServers and/or groups of CueServers.

Note: Strings can either be enclosed in double-quotes (“”) or in single quotes (‘’).

Examples: “Cue 1 Go” *
Sends the command “Cue 1 Go” to all CueServers on the network. All CueServers will begin executing Cue 1 at the same time.

“Device 3; Cue 1 Go” *
Sends the command string “Device 3; Cue 1 Go” to all CueServers on the network. Although all CueServers will receive the command string, “Device 3” will cause the remainder of the string to only be performed by any CueServer with a Device ID of 3. In effect, this command is telling CueServer #3 to execute Cue 1.

“Device 1>6; Output 1 At 0” *
Sends the command string “Device 1>6; Output 1 At 0” to all CueServers on the network. Only CueServers with Device IDs of 1 through 6 will set their Output 1 off.

“ ”~ (string store)

Syntax: “(string)”~(location)

Description: The string/tilde command is used to write a character string into an I/O location in CueServer. Store strings to make changes to the LCD display, send strings to the serial or MIDI ports and even send messages via UDP packets.

Depending on the value of the *location* parameter, you can store strings into one of several special CueServer “locations”. The following list describes the available string locations.

Note: Strings can either be enclosed in double-quotes (“”) or in single quotes (‘ ’).

Location 0 (LCD User String)

Writes the string to the User String location for display on the LCD. The User String can be positioned in any of the four quadrants of the LCD display (see the LCD Setup section).

Location 1 (LCD Line 1 Override)

Writes the string to Line 1 of the LCD display. This location overrides any other data being shown on Line 1 of the LCD. Write an empty string to this location to remove the override.

Location 2 (LCD Line 2 Override)

Writes the string to Line 2 of the LCD display. This location overrides any other data being shown on Line 2 of the LCD. Write an empty string to this location to remove the override.

Location 3 (Ethernet UDP Broadcast)

Sends the string via a UDP packet out the CueServer’s Ethernet port. By default the packet is sent to the CueServer Multicast address (239.255.204.2) on port 52737. Both of these parameters can be changed by the `_udpip` and `_udpport` system variables. See the **Ethernet** chapter of this manual for additional details.

Location 4 (Serial Port)

Sends the string via CueServer’s external RS-232 serial port. See the **Serial Port** chapter of this manual for additional details.

Location 5 (MIDI Port)

Sends the string via CueServer’s MIDI Output port. See the **MIDI** chapter of this manual for additional details.

Location 6 (Telnet Port)

Sends the string to the current Telnet session (if one is currently active).

Once a string has been encountered in a CueScript statement, it may be stored in multiple locations by adding additional tilde/location pairs in the command. For example, "Hello"~0~3~4 will write the string "Hello" on the LCD screen, send it via Ethernet and out the serial port at the same time.

The dollar-sign (\$) character can be used in strings to toggle between ASCII text mode and hexadecimal data mode within the string. By default, all strings are treated as ASCII text. Each time a dollar-sign is encountered in the string, the mode toggles between ASCII and Hex data. For example, the string "Testing \$313233" would be written as "Testing 123" (since byte 31h is ASCII "1", 32h is "2" and 33h is "3"). Also, two dollar-signs in a row is simply interpreted as a single dollar-sign. For example, "Price = \$\$3.00" will be written as "Price = \$3.00".

Examples:

"Hello"~0

Writes "Hello" to the User String area of the LCD display (by default, in the lower-left corner of the display)

"Building Lighting Override Enabled"~1

Overrides Line 1 of the LCD completely with the specified string

"~1

Removes the override from Line 1 of the LCD display

"Testing\$00"~3

Sends the ASCII text "Testing", followed by a null-byte to the Ethernet multicast address

"\$FF01\$Start"~4

Sends hexadecimal bytes FF and 01 to the serial port, followed by the ASCII text "Start"

"\$903C7F"~5

Sends three hexadecimal bytes 903C7F to the MIDI Output port, which is the MIDI command for Middle-C Note On at Maximum Velocity

"Hello"~0~3~4

Displays "Hello" on the LCD display, and sends "Hello" to the Ethernet port and Serial Port

“ ” = (variable assignment)

Syntax: *“(variable name)” = (value)*

Description: The string/equals command is used to store the value of either a user-defined or system-defined variable. Variables are holding places for information (numerical values, strings, commands, etc.). After a variable has been assigned a value, it may be used in CueScript commands to automatically substitute the current value of the variable.

See also the Variable Substitution command (`{{ }}`) below.

Note: Strings can either be enclosed in double-quotes (“”) or in single quotes (‘’).

Examples: `“x” = 3`
Assigns variable x the value 3

`“TimerCommand” = “Cue 1 Go”`
Assigns variable TimerCommand the value Cue 1 Go

`“_backlight” = 25`
Assigns the system-defined variable _backlight the value 25 (which sets the LCD backlight to 25% brightness)

{{ }} (variable substitution)

Syntax: { {(variable name)} }

Description: The variable substitution command is used to recall the value of a previously stored user-defined or system-defined variable. When a variable name enclosed in angle brackets (greater-than and less-than symbols) is encountered in a CueScript command, CueServer substitutes the variable's current value before executing the command.

See also the Variable Assignment command (" " =) above.

Examples:

```
Cue {{CueNumber}} Go
```

Executes the cue specified by the value of the variable CueNumber (for example, "CueNumber" = 1)

```
Channel 1 At {{MyLevel}}
```

Selects channel 1 and assigns its DMX level to the value contained in the variable MyLevel (for example, "MyLevel"="50%")

```
Fixture 1+3+5 At {{Orange}}
```

Selects the channels of fixtures 1, 3 and 5 and assigns them to the DMX levels specified by the variable Orange (for example, "Orange" = "{FL, 50, 0}")

```
"Main Cue = {{CueNumber}}"~0
```

Displays "Main Cue = 1" on the LCD Display (in position 0), assuming that "CueNumber" = 1

;
(semicolon)

Syntax: ;

Description: The semicolon command is used to help visually separate multiple CueScript commands, but it is **optional** since it performs no other operation. The semicolon command is particularly useful if abbreviations are being used in CueScript command strings as it makes the command more readable.

Examples: Channel 1 @ FL; Channel 2 @ 50
Sets channel 1 to 100% and then channel 2 to 50%

C1@FL;C2@50
Same as last example, but using abbreviations

Group 3; Time 0; At FL; Time 5; At 0
Selects the channels of group 3, then sets the fade time to immediate (0), then sets all of the channels to 100%, then changes the fade time to 5 seconds, then begins fading the channels down to 0.

U3;T0;@FL;T5;@0
Same as the previous example, but using command abbreviations

U3T0@FLT5@0
Same as the previous example, but without the semicolon separators (although this example works without the optional semicolons, it is more difficult to read is is only recommended if command-length needs to be kept to an absolute minimum)

DMX-512

CueServer includes two DMX-512 ports, one for DMX input and one for DMX output.

DMX Output

The DMX Output port is a standard 5-pin Female XLR jack. This port provides DMX output from the CueServer and is intended to be connected to lighting instruments that operate from a DMX signal. When channel values are set, cues are played or DMX input is passing through CueServer, the resulting lighting levels are transmitted from the DMX Output port.

In typical installations, the DMX Output from the CueServer will be connected to one or more DMX controlled lighting fixtures, or possibly to the input of a DMX splitter or opto-isolator for further distribution around a facility.

DMX Input

The DMX Input port is a standard 5-pin Male XLR jack. This port provides a way to send DMX lighting levels into the CueServer. CueServer can use DMX Input for a variety of purposes:

- DMX levels can be passed through directly from a console to the fixtures.
- DMX levels can be modified while being passed through from a console to the fixtures.
- DMX levels can be captured and stored in a static scene for playback later.
- Streaming DMX levels can be recorded into a “Streaming Cue” for precise show playback recorded from another console.
- CueServer can dynamically act as a console backup by choosing when to override some or all DMX levels.
- CueServer can insert emergency lighting cues in-line with an external DMX stream.

In normal operation, CueServer passes the DMX Input through to the DMX Output port. Once channel levels are set or cues are played back, CueServer begins to override the lighting levels that are present on the DMX Input port. The method of overriding channel levels can be set to either Highest Level Takes Priority (HTP) or by Latest Level Takes Priority (LTP) mode. CueServer can also select channels to return to pass-through operation.

Example: To override a channel or group of channels, execute a command such as “Channel 1>10 At 50”. Or to override channels with a scene, execute a cue by issuing a command like “Cue 2 Go”.

To return overridden channels to their original pass-through state, execute the Release command. The Release command works on the selected channels. See the description of Release for more details.

The DMX Input port is also used when the user wishes to record a snapshot of the incoming DMX levels into a cue in CueServer. When recording a cue, whatever levels that are present at the CueServer’s DMX Output port when the cue is recorded are captured in the cue.

Example: To record a scene from a console into CueServer, attach the console’s DMX Output to CueServer’s DMX Input. Then, bring up the scene on the console. The levels will pass-through CueServer to the lighting fixtures. Then, record the cue in CueServer.

CueServer is also capable of recording streaming DMX input from a console. When a Streaming Cue is recorded, every piece of DMX data on the DMX Input port is saved in CueServer’s flash memory for exact playback later.

Example: To record a Streaming Cue from another console, attach the console’s DMX Output to CueServer’s DMX Input. Then, from the Cue List page, click on Add New Cue. Then, select the Streaming Cue type from the New Cue page before proceeding.

For performing backup or emergency lighting looks, any number of CueServer's triggers can be programmed to execute a cue or bring up channel levels for a variety of events. Then, other triggers can be used to Release the levels to return operation back to its pass-through operation.

Example: Contact 1 can be programmed with "Cue 1 Go" as the contact-closed event and "Release" as the contact-opened event. Then a latching button or keylock switch can be attached to Contact 1's terminal block connections to implement a simple lighting override system. Attach the contact to a security system output or power-loss monitor to make the operation fully automatic.

Contact Closure Inputs

CueServer provides eight contact closure inputs that are designed to be connected to external switches, buttons, relays or other devices that close a circuit when they activate. These “contact-closures” are detected by CueServer and can be used to trigger the execution of CueScript statements. These statements can execute cues, bring up lighting levels, illuminate the function button LED indicators, change the LCD display, send serial strings, and much more.

In typical applications, the contact-closure inputs are connected to external buttons or switches that run cues. A simple button station can be attached that gives the end-user the ability to recall lighting cues from CueServer without needing to go to the CueServer itself.

Sometimes, it is desirable to use the contact-closure inputs in conjunction with the low-voltage outputs in order to provide LED feedback on each of the buttons of a simple button station.

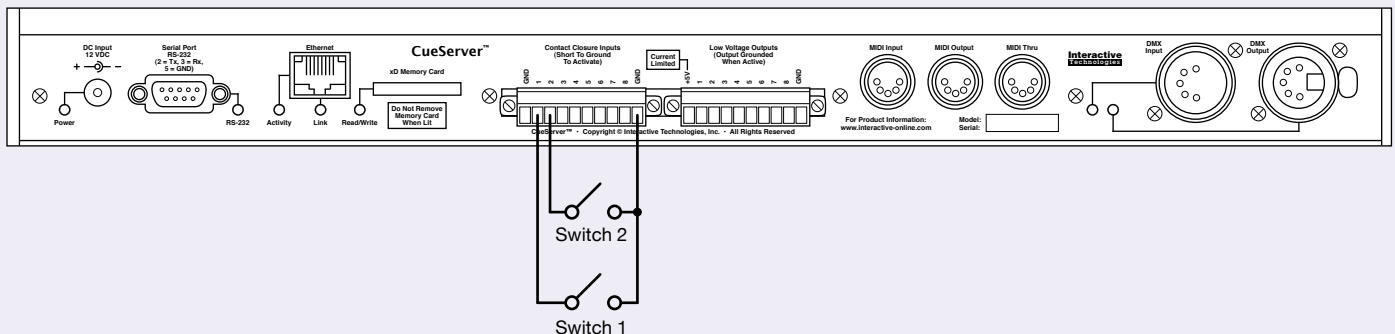
The following list shows some of the possible devices that can be attached to CueServer’s contact-closure port:

- Switches
- Buttons
- Relay Outputs
- Foot Petals
- Keyswitches
- Limit Switches
- Motion Detectors
- Floor Switches
- Photocells

Contact Closure Connections

Up to eight external switches can be attached to CueServer’s contact closure port. Refer to the following example for the proper way to connect switch-type devices to CueServer.

Example: To connect switches or other contact devices to CueServer’s contact closure inputs, connect one side of the switch to the desired contact input (1-8) and the other side of the switch to one of CueServer’s ground pins (GND).



Contact Closure Events

CueServer can execute CueScript commands with either a contact closure input is closed or opened. Since CueScript commands can effect nearly any aspect of CueServer, the ability to execute arbitrary commands for contact-closed and contact-opened events provides for a high degree of flexibility for the system installer.

To program the events for a particular contact closure input, go to the following page within CueServer:

- **Main > Triggers > Contact Closures**

See the section on Contact Closure Triggers for detailed explanation of the various programming features available to each contact closure input.

Binary Cue Select Feature

An advanced feature of the contact closure port is to allow an external device such as a Programmable Logic Controller (PLC) to be able to use a binary input to CueServer to remotely execute cues.

Note: This is an advanced feature that, if enabled, causes the contact closure inputs to bypass their normal mode of operation. It is highly recommended not to enable this feature unless the user is familiar with its operation.

About the Binary Cue Select Feature

Using the Binary Cue Select Feature allows a binary input on the contact closure port to directly execute up to 255 individual cues. In this mode, the pins on the contact closure port are interpreted as bits of a binary number (contact 1 is the least-significant bit and contact 8 is the most-significant bit).

When the numerical value of the input pins is non-zero, then CueServer automatically executes the cue with the same number as the binary value.

Each of the contacts represents the following binary values. When multiple contacts are closed simultaneously, the values are added together in order to arrive at the total numerical value.

- Contact 1 = 1
- Contact 2 = 2
- Contact 3 = 4
- Contact 4 = 8
- Contact 5 = 16
- Contact 6 = 32
- Contact 7 = 64
- Contact 8 = 128

Example: If contacts 1, 3 and 6 are closed simultaneously, Cue 37 is executed.

$$1 + 4 + 32 = 37$$

Enabling the Binary Cue Select Feature

The Binary Cue Select feature can be enabled on a pin-by-pin basis for each of the eight contact closure inputs. By default, none of the contacts have this feature enabled. If all eight bits are enabled, then the external device can select any cue from 1 to 255. If less than eight bits are enabled, then only those bits contribute to the binary value of the cue number (each bit still represents the same numerical value).

To enable or disable the Binary Cue Select feature for each of the contact closure pins, go to the following page within CueServer:

- **Main > Hardware Setup > Port Settings > Contact Closures**

A page similar to this one will appear:

Contact Closure Input Port

Binary Cue Select Pins
Automatically executes cues from binary
input on selected contact closure input pins

1 (1) **2 (2)** **3 (4)** **4 (8)** **5 (16)** **6 (32)** **7 (64)** **8 (128)**

Place a check mark next to each contact that should be included in the Binary Cue Select function.

Digital Outputs

CueServer provides eight Digital “Low-Voltage” Outputs that are designed to allow CueServer to directly illuminate LEDs, operate relays, sound buzzers and other similar “low-voltage” devices.

The outputs are activated by CueScript commands, which means that nearly any event in CueServer (cues executing, buttons being pressed, contacts being closed, timers occurring, etc.) can operate on the eight outputs.

Controlling the Low-Voltage Outputs

To turn on or off an output, use the `Output` command. The `Output` command works in a manner similar to the `Channel` command. An output is set to a level from 0 to FULL.

Setting an output to FULL (100%) turns the output On. Setting an output to zero (0%) turns the output off. Most other values in-between also turn the output On, except for several special values, which cause the specified output to flash in different patterns.

The following table shows the possible values of each output and how it affects it’s state:

- Value 0 = OFF
- Value 1 = ON
- Value 2 = Slow Flash
- Value 3 = Slow Flash, Reverse
- Value 4 = Fast Flash
- Value 5 = Fast Flash, Reverse
- Value 6 = Wink
- Value 7 = Wink, Reverse
- Value 8 to 100 = ON

Example: The following examples of CueScript commands show how to control the outputs:

```
Output 1 At FL  
Turn output 1 on.
```

```
Output 3+5+7 @ 0  
Turn outputs 3, 5 and 7 off.
```

```
Output 1>4 @ 4  
Make outputs 1 through 4 flash quickly.
```

```
Output 1>8 @ 0; Output 3 @ 1  
Turn off all outputs and then immediately turn on output 1.
```

Low-Voltage Output Connections

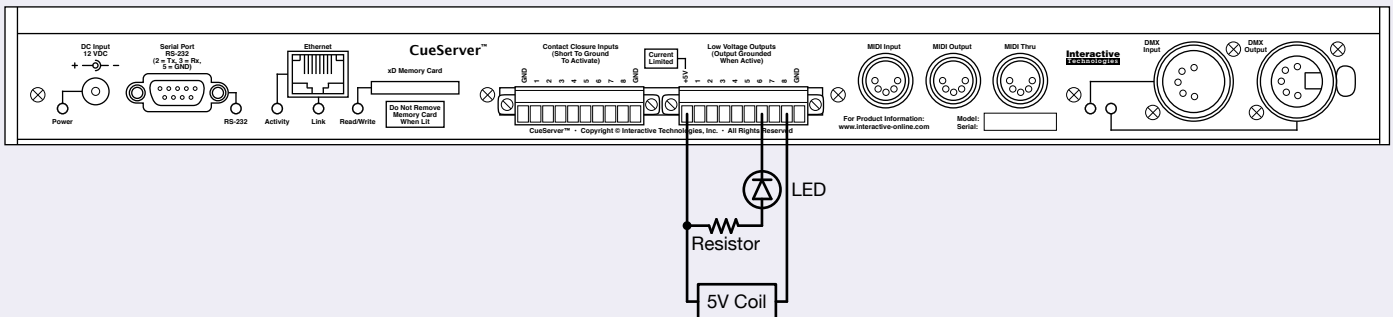
LEDs, lamps, relays, sounders and other low-voltage devices can be directly attached to the CueServer Low-Voltage Output pins.

Each of the eight outputs from CueServer are transistor-based “pull-to-ground” type outputs. This means that when an output is “ON”, the corresponding output pin is connected to ground. When an output is “OFF”, the corresponding pin acts as if nothing is connected to it.

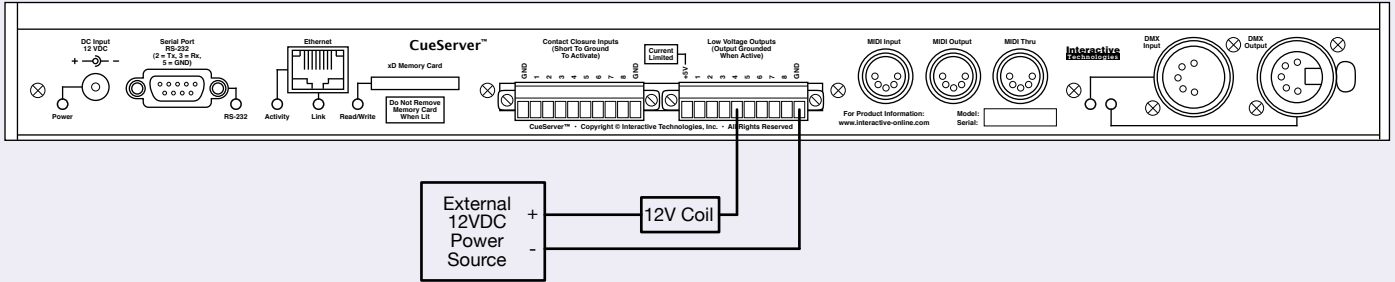
To connect a low-voltage device to CueServer, consider that the negative (ground) side of the device is the side being switched on and off. The other side of the device is connected to the positive voltage source. This allows devices with various voltage requirements (5V, 12V, 24V, etc.) to be connected to CueServer.

Refer to the following examples which show how various low-voltage devices can be connected to CueServer:

Example: To directly connect an LED, 5V Relay or other 5V powered low-voltage device, follow the following circuit diagram. CueServer can supply enough current at 5V DC to power most small-signal devices.



Example: To use a low-voltage device with a voltage requirement other than 5V or that requires additional current to operate, use an external power source. The positive side of the device connects to the positive voltage of the power supply. The negative side of the device connects to one of the CueServer output pins. Finally, the grounds of the power supply and CueServer are connected.



Serial Port

The serial port on CueServer can send and receive standard RS-232 serial data. External devices can send arbitrary CueScript commands to CueServer and CueServer may send any data string out the serial port.

Many external devices use serial data to monitor, operate, control or program their behavior. Devices that provide building automation, lighting control, security, video, HVAC, data logging and much more can be attached to CueServer's serial port. CueServer can then send these devices strings that operate those devices, or those devices may be programmed to send CueServer special strings that cause CueServer to take specific actions. The possibilities for integration with other systems is nearly unlimited.

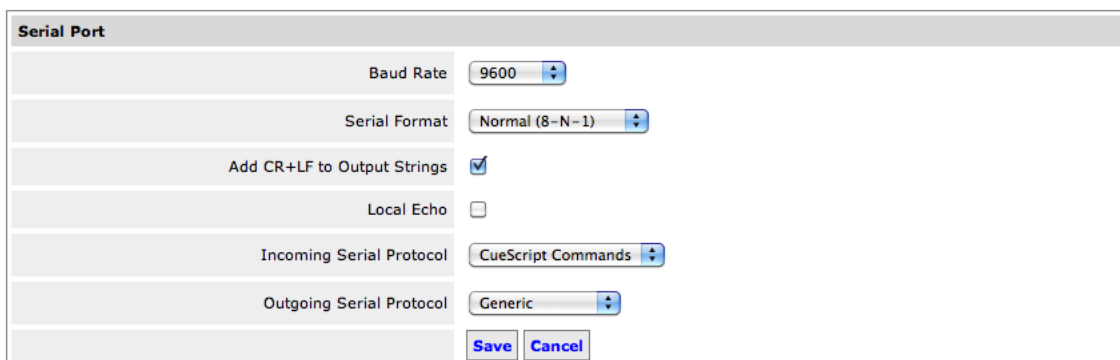
Serial Port Settings

The serial port on CueServer has several configuration options affecting the serial rate and formatting options, and both incoming and outgoing protocol options.

To set the serial port's parameters, go to the following page within CueServer:

- **[Main](#) > [System Preferences](#) > [Port Settings](#) > [Serial Port \(RS-232\)](#)**

On this page is are several options that affect the serial port:



Serial Port	
Baud Rate	9600
Serial Format	Normal (8-N-1)
Add CR+LF to Output Strings	<input checked="" type="checkbox"/>
Local Echo	<input type="checkbox"/>
Incoming Serial Protocol	CueScript Commands
Outgoing Serial Protocol	Generic
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Baud Rate

Choose the desired serial port baud rate using this pop-up menu. Many popular rates are available from 300 bps to 230400 bps.

Serial Format

The standard (default) serial format is "Normal (8-N-1)". This pop-up menu has several choices for other non-standard formats. Choose the one that suits your application.

Add CR+LF to Output Strings

This check box, when enabled, instructs CueServer to append a Carriage Return (CR) and Line Feed (LF) characters to the end of each string transmitted from the serial port using the ~4 command.

Local Echo

This check box, when enabled, instructs CueServer to repeat (echo) every incoming character back out to the sending device. This function is useful if you are using CueServer with a serial terminal program (such as HyperTerminal).

Incoming Serial Protocol

This pop-up menu chooses one of several ways for CueServer to automatically handle incoming serial data:

- **Ignore**
All incoming serial data is ignored.
- **CueScript Commands**
All incoming serial data is interpreted as CueScript commands. When a CueScript statement is received in [square brackets], it will be executed.
- **CueStation Hub**
All incoming serial data is handled as communications with a CueStation Hub.
- **ASCII Mapped Macros**
Every serial character received is interpreted as a request to execute a macro. The ASCII value of each character is used to determine which macro number to execute. For instance, if the capital letter 'A' is received, which has an ASCII value of 65, CueServer will attempt to execute Macro 65. This mode is useful for triggering macros from a simple keyboard accessory attached to CueServer's serial port.
- **Save ETC Unison Data**
All incoming serial data is handled as ETC Unison messages. Information about which presets are active and which partitions are opened or closed is all saved to CueScript variables.

Outgoing Serial Protocol

This pop-up menu chooses one of several ways for CueServer to automatically handle outgoing serial data:

- **Generic**
All outgoing serial data is not formatted in any special way. Send strings out the serial port by using the ~4 command.
- **CueStation Hub**
Outgoing serial data is used for communication with a CueStation Hub. CueServer automatically sends commands to the Hub when button events or indicator states change.

Receiving Commands via the Serial Port

If the serial port's Incoming Serial Protocol mode is set to "CueScript Commands", CueServer will continuously monitor incoming data at the serial port. If it detects a CueScript command, the command is executed. Nearly any external device capable of sending serial data can send CueScript commands to CueServer.

To send CueScript statements to CueServer, the command must be transmitted in ASCII text and enclosed in square-brackets ([and]).

CueServer ignores all incoming serial data until it receives an open-bracket ([). Once the open-bracket is received, it accumulates the ASCII bytes it receives until it encounters a close-bracket (]). Once the close bracket arrives, then the entire CueScript statement received is executed.

Example: The following examples show a few valid CueScript commands that may be sent to CueServer's serial port:

```
[Cue 7 Go]
```

Loads Cue 7 into the playback fader and executes it

```
[Playback 3; Release; Channel 1 @ 50]
```

Releases any currently live channels and then sets channel 1 to 50% of Playback 3

```
[Button 1>4 at 0; Button 3 at FL]
```

Turns off the LED indicator on buttons 1 through 4 and then immediately turns on the LED indicator on button 3

Sending Serial Strings

CueServer can send serial strings to devices attached to CueServer via the serial port. Serial strings are sent as part of CueServer's CueScript command language, which allows strings to be sent as a result of a cue executing, a button being pressed, a timer event occurring and much more.

To send a serial string, use the Store (~) command in a CueScript statement. First a string is specified (which may contain ASCII and/or hexadecimal values). Then the Store operator (~) is included in the command, followed by the destination of the store. For sending serial strings, the destination is 4. For example, to send the message `Testing`, use the command `"Testing"~4`.

When sending serial strings, the strings are interpreted as ASCII text by default. In order to send hexadecimal data, insert a dollar-sign (\$) in the string. Each dollar-sign in the string toggles the mode between ASCII and hexadecimal. To send a dollar-sign in the ASCII string, include the dollar-sign twice (\$\$).

Example: Here are a few examples of serial strings being sent via CueScript:

```
"Hello"~4  
Send Hello to the serial port.
```

```
"$C31F20"~4  
Send the hexadecimal data 0xC31F20 to the serial port.
```

```
"Price = $$2.99"~4  
Send Price = $2.99 to the serial port.
```

```
"$A531$Serial Data$FF"~4  
Send the hexadecimal data 0xA531 followed by ASCII "Serial Data" followed by 0xFF to the serial port.
```

Ethernet

CueServer is based largely on its Ethernet port. The majority of system programming, monitoring and operation occurs through CueServer's built-in web server.

CueServer's Ethernet port can also be used to send generic UDP messages to other devices on the network and can receive CueScript packets from other devices. This allows other devices in a system to remotely control CueServer and/or have CueServer remotely control other devices. CueServer can be involved in a coordinated and/or synchronized show system all via Ethernet.

Receiving Messages via UDP Packets

CueServer listens for incoming UDP packets on port 52737 (0xCE01). When received, CueServer interprets the data in the packet as a CueScript command in simple ASCII text format. The command is executed by CueServer as soon as it is received. The incoming command can be any valid CueScript statements.

Virtually any Ethernet-enabled device could generate CueScript commands and send them to CueServer via UDP packets. For instance, a PC can remotely control CueServer via its Ethernet connection. A PLC (Programmable Logic Controller) can send messages to CueServer. A custom application on a tablet PC or PDA could wirelessly send commands to CueServer via wireless Ethernet. The possibilities are nearly endless.

To send a UDP packet to CueServer, use the operating system of the sending device to send a UDP-type message to the IP Address of the CueServer on port 52737 (0xCE01).

Example: Here are a few examples of valid CueScript commands that could be sent to CueServer via UDP packets:

```
Cue 1 Go  
Loads Cue 1 into the playback fader and executes it.
```

```
Time 5; Channel 1>10 @ FL  
Fade channels 1 through 10 to Full in 5 seconds.
```

```
Output 7 At 0; Output 8 At 1  
Turn output 7 off, turn output 8 on.
```

Sending Messages via UDP Packets

CueServer can generate messages and send them via UDP packets to other Ethernet enabled devices as part of CueServer's CueScript command language, which allows packets to be sent as a result of a cue executing, a button being pressed, a timer event occurring and much more.

To send a message via UDP, use the Store (~) command in a CueScript statement. First a string is specified (which may contain ASCII and/or hexadecimal values). Then the Store operator (~) is included in the command, followed by the destination of the store. For sending UDP messages, the destination is 3. For example, to send the message `Hello`, use the command `"Hello"~3`.

By default, CueServer sends all outbound UDP packets to port 52737 (0xCE01) at IP Address 239.255.204.2, which is a multicast address. Any device on the local network can subscribe to this address and receive CueServer generated messages.

When sending strings via UDP, the strings are interpreted as ASCII text by default. In order to send hexadecimal data, insert a dollar-sign (\$) in the string. Each dollar-sign in the string toggles the mode between ASCII and hexadecimal. To send a dollar-sign in the ASCII string, include the dollar-sign twice (\$\$).

Example: Here are a few examples of UDP messages being sent to external devices via CueScript statements:

```
"Hello"~3  
Send Hello to the UDP multicast address.
```

```
"$F100C20E"~3  
Send the hexadecimal data 0xF100C20E to the UDP multicast address.
```

```
"Apples $$1.00 each"~3  
Send Apples $1.00 each to the UDP multicast address.
```

```
"$EE$Testing$00"~3  
Send the hexadecimal data 0xEE followed by ASCII Testing followed by 0x00 to the UDP multicast address.
```

The default IP Address and port number used to broadcast UDP messages can be changed by setting the values of the `_udpip` and `_udpport` system variables. See the System Variables section for details.

MIDI

CueServer provides standard MIDI ports (Input, Output and Thru) for connection to external devices such as timecode synchronizers, show control devices, personal computers, musical sequencers, musical instruments and even possibly other CueServer devices.

The MIDI ports on CueServer are designed to make it easy to use timecode to trigger events in CueServer and also provide a means of controlling CueServer from another MIDI device or allowing CueServer to send control commands to other MIDI devices. CueServer can also act as a MIDI-to-DMX converter.

MIDI Input

CueServer will respond to a variety of MIDI commands, including MIDI Timecode and related messages, Note On and Off, MIDI Reset and some special System Exclusive messages. Each of these types of MIDI Input commands are described in the following sections.

MIDI Timecode

CueServer can synchronize show playback with a MIDI Timecode (MTC) stream. When CueServer begins receiving MTC, it decodes the timecode data and uses its Timecode Event List to execute CueScript commands at specific times during the playback of an external show.

Using this technique, CueServer can be synchronized with a Video playback device, an Audio track or any other show system that may produce MTC signals. Since any arbitrary CueScript commands can be executed at any timecode event, CueServer can play cues, set lighting levels, change outputs, send serial strings and anything else CueServer can do as a result of encountering specific times in the input timecode.

CueServer can also be synchronized to SMPTE timecode by using a SMPTE-to-MIDI converter (such as the JL Cooper PPS-2). Devices such as these take a SMPTE signal and translate it into MTC, which can be fed into CueServer.

Note: The current timecode can be displayed on CueServer's front LCD display. See the section on customizing the LCD for details.

Programming Timecode Events

To set up a list of timecode based events, go to the following page within CueServer:

- [Main](#) > [Triggers](#) > [Timecode Events](#)

A page similar to this one will appear:

Timecode (click to edit)	Name	Command
00:00:02.00	Reset Show	Release
00:01:01.00	Start Show	Cue 1 Go
00:01:16.21	Sunrise	Cue 21 Go
00:01:49.10	Bird Flyby	Playback 2; Cue 101 Go
00:02:21.08	Rain Shower	Output 6 @ FL
00:03:04.29	Sunset	Output 6 @ 0; Cue 301 Go
00:03:30.00	End	Cue 320 Go

[Add New Timecode Event](#)

The Timecode Event List allows the programmer to specify exact times (Hours/Minutes/Seconds/Frames) that a specific event should occur. When CueServer detects that a timecode event should execute, it performs the specified CueScript commands for that event.

To add a new timecode event, click on the Add New Timecode Event button. A timecode event editor page will appear:

Timecode Event Details	
Timecode	00 : 01 : 01 . 00
Event Name	Start Show
Command String	Cue 1 Go
	Cue 1 Go
	<input type="button" value="Save"/> <input type="button" value="Delete"/> <input type="button" value="Cancel"/>

Timecode

Specify the absolute time this timecode event should occur. Timecodes are specified in Hours (0-24), Minutes (0-59), Seconds (0-59) and Frames (0-30).

Event Name

The event name is for the programmer's convenience. It is used to give a descriptive name of the event to make it easier to identify the intended purpose of the event.

Command String

Specify the CueScript statements that should be executed by CueServer when this time occurs.

CueServer will also respond to MIDI Full-Frame messages which set the current timecode to any specific point in time. Some devices send these messages when the Audio or Video program is being cued or shuttled. CueServer will update its internal timecode clock when receiving these messages, but no events will be executed from the Timecode Event List.

Note: CueServer works internally with 30fps timecode. If CueServer is receiving 24fps, 25fps or 30/drop timecode and an event in its list has a time specified with a frame time larger than the current format, CueServer will execute that event the moment that the input timecode wraps around to the next whole-second time.

MIDI Reset Messages

CueServer responds to MIDI Reset messages by resetting its timecode registers. The current timecode counter will be set to “no time”.

The MIDI Reset message has the following format:

[FF]

[FF]

This 1-byte MIDI Status message instructs all connected MIDI devices to perform a Reset.

CueServer System Exclusive Messages

CueServer will respond to a special System Exclusive message designed to make it possible for an external MIDI device to send any arbitrary CueScript commands to CueServer. This makes it possible for an external show control device, sequencer, PC or musical instrument to run cues, set lighting levels and much more via MIDI commands.

CueServer System Exclusive messages have the following format:

[F0][00][43][59][channel][cueserver_command_string][00][F7]

[F0][00][43][59]

This is the 4-byte header of the CueServer message. It is a MIDI System Exclusive message that identifies the data as being CueScript commands.

[channel]

This byte is the MIDI channel. It must match the channel number set up in the CueServer MIDI configuration. This byte's value is one less than the MIDI channel (00 = Channel 1, 01 = Channel 2, 0F = Channel 16, etc.)

[cueserver_command_string]

This field contains a variable length number of ASCII bytes that represent a CueScript statement. For example, “Cue 1 Go”.

[00][F7]

This is the 2-byte ending of the CueServer System Exclusive message.

Note On Messages

External MIDI devices can directly set DMX output levels of CueServer by sending Note On messages.

A Note On message has the following format:

[9x] [channel] [value]

[9x]

This is the 1-byte Note On MIDI status message. The x in the byte indicates which MIDI Channel the message is being sent (90 = Channel 1, 91 = Channel 2, 9F = Channel 16, etc.). CueServer's MIDI Channel is set up in the Port Settings page of the Hardware Setup page.

[channel]

This value is typically the note number for the Note On command. CueServer interprets this value as the DMX channel number to set. Legal values are from 0 to 127, which correspond to DMX channels 1 to 128.

[value]

This value is typically the velocity for the Note On command. CueServer interprets this value as the lighting intensity of the channel. Legal values are from 0 to 127, which correspond to intensities from OFF to FULL.

Note Off Messages

External MIDI devices can directly release DMX output levels of CueServer by sending Note Off messages. Releasing a channel is different than setting it's level to zero. Releasing a channel removes it's influence from the output lighting levels.

A Note Off message has the following format:

[8x] [channel] [velocity]

[8x]

This is the 1-byte Note Off MIDI status message. The x in the byte indicates which MIDI Channel the message is being sent (80 = Channel 1, 81 = Channel 2, 8F = Channel 16, etc.). CueServer's MIDI Channel is set up in the Port Settings page of the Hardware Setup page.

[channel]

This value is typically the note number for the Note Off command. CueServer interprets this value as the DMX channel number to release. Legal values are from 0 to 127, which correspond to DMX channels 1 to 128.

[velocity]

This value is typically the velocity for the Note Off command. CueServer ignores this value. Releasing a channel has no corresponding velocity or time.

MIDI Output

CueServer can send nearly any type of MIDI command to other external MIDI devices through its MIDI Output port.

Commands may be sent to other devices such as show control processors, time synchronizers, sequencers, musical instruments, PCs and even other CueServer devices.

Since MIDI commands are sent to the MIDI Output port via CueScript commands, any action or event within CueServer can produce one or more MIDI messages to send to other devices, providing a very flexible and powerful means to coordinate external MIDI devices with a show running in CueServer.

Sending MIDI Commands

CueServer can send any arbitrary byte string to the MIDI Output port, allowing the programmer to send nearly any MIDI Command as necessary.

MIDI uses hexadecimal byte values (from $\$00$ to $\$FF$) to send commands to other devices. For instance, to send a command to play Middle C at maximum velocity on MIDI Channel 1, the bytes $\$903C7F$ are sent. It is beyond the scope of this manual to list the entire MIDI command set, but there are many online resources available that list the available MIDI commands.

To send a MIDI command string, use the Store (~) command in a CueScript statement. First a string is specified (which may contain ASCII and/or hexadecimal values). Then the Store operator (~) is included in the command, followed by the destination of the store. For sending MIDI command strings, the destination is 5. For example, to send the message $\$903C7F$, use the command `“ $\$903C7F$ ”~5`.

When sending MIDI command strings, the strings are interpreted as ASCII text by default. In order to send hexadecimal data, insert a dollar-sign (\$) in the string. Each dollar-sign in the string toggles the mode between ASCII and hexadecimal. To send a dollar-sign in the ASCII string, include the dollar-sign twice (\$\$).

Here are a few examples of MIDI command strings being sent via CueScript:

Example: The following examples of CueScript commands show how to send MIDI commands out the MIDI Output port:

```$903C7F``~5`  
*Send Note On (Channel 1, Middle C, Velocity 127).*

```$C133``~5`  
Send Program Change (Channel 2, Program 51).

```$F07F00010105060708F7``~5`  
*Send Full-Frame System Exclusive message to cue slave devices to 05:06:07:08 on MIDI Channel 1.*

```$FF``~5`  
Send MIDI Reset command to all connected devices.

```$F000435900$Cue 1 Go$00F7``~5`  
*Send CueServer System Exclusive message "Cue 1 Go" to MIDI Channel 1.*



# Appendix A: CueScript Command Summary

The following table is a quick reference of the commands in the CueScript programming language used by CueServer.

Both simple and complex command sequences can be executed by CueServer by simply typing them into one of CueServer's web pages, or as a result of one of CueServer's triggers (such as timers, buttons, contact closures, cues, and more) or from external sources (such as custom web pages, Flash objects, LCD touchscreens or via the serial, MIDI or Ethernet ports).

For a complete description of each command, please refer to the CueScript Command Language chapter.

Command	Description	Examples
At <i>[unit] (value)</i> A or @	Sets the value of selected object(s). Works with Channels, Fixtures, Groups, Buttons and Outputs. By default, channel values are expressed in percentages. Decimal values can be specified with the # character, hexadecimal values can be specified with the \$ character. The FL expression is a shortcut for 100%.	Channel 1 At 50 Fixture 5>8 At 75 Group 3 At FL Button 4 At #255 Output 5 At \$FF Time 5 Channel 1>10 At 33
At <i>(array of values)</i> A or @	Sets the values of selected object(s) to an array of values. Similar to the regular At command, but multiple values are given in an array, which are assigned to each selected Channel, Button or Output in a sequential pattern.	Channel 1>3 At {10,20,30} Fixture 4 At {0,50,FL} Group 10 At {#255,0,\$80}
At <i>(+/-) [unit] (offset)</i> A or @	Sets the values of selected object(s) to an offset (+/-) from their current values. Similar to the regular At command, but values are assigned as a relative offset from the current value(s). All values are pinned to 0 and 100% (i.e. the values do not wrap around).	Channel 1 At +5 Channel 5>8 At -10 Group 3 At +25 Fixture 4 At -50 Channel 1 At +#10 Channel 2 At -\$0C
At Cue <i>(cue number)</i> [/ (level)] AQ or @Q	Sets the values of selected object(s) to data stored in a cue. Useful for recalling only parts of a cue into the current playback. An optional level may be specified that scales the recalled levels from the cue.	Channel 1>10 At Cue 7 Fixture 3 At Cue 101 Group 4 Time 10 At Cue 99 Button 1 At Cue 1 Group 2 At Cue 1/50
At Input AIN	Sets the values of selected object(s) to the current levels available on the DMX Input port. Useful for taking a snapshot of the DMX Input.	Channel 1>512 At Input Fixture 1+3+5 At Input Output 1 At Input

Command	Description	Examples
At Playback ( <i>number</i> ) AP	Sets the values of selected object(s) to the current levels in one of the four playback faders.	Channel 1>512 At <u>Playback 4</u> Group 3 At <u>Playback 2</u>
Break BR	Stops executing CueScript commands that follow the Break command. Useful for use with variables to conditionally execute commands.	{{MyEnable}} Cue 101 Go  * <i>The MyEnable variable may be empty or contain the "Break" command</i>
Button ( <i>range</i> ) B	Selects one or more buttons for setting the indicator LED or enable states.	<u>Button 1</u> At FL <u>Button 5</u> >8 On Button 3.4 At #235 <u>Button 1+3+5</u> Enable
Channel ( <i>range</i> ) C	Selects one or more channels. Used for setting and releasing channel levels, enabling, disabling, parking, unparking and other similar functions.	<u>Channel 1</u> At 50 <u>Channel 10</u> >25 At 75 <u>Channel 10+20+30</u> Release <u>Channel *</u> Disable <u>Channel 1</u> >4+21>24 Park
Clear CL	Clears the selected playback fader. All channels (except parked channels) will be released, cue list information and timing values will be reset.	Playback 3 <u>Clear</u>
Contact ( <i>range</i> ) CO	Selects one or more contacts for setting their enable states.	<u>Contact 1</u> Disable <u>Contact 2+5</u> Enable
Cue ( <i>cue number</i> ) CU or Q	Specifies the cue to be executed by the next Go command.	<u>Cue 1</u> <u>Cue 2</u> Go <u>Cue 10.5</u> Fade 7 Go
Delete ( <i>object</i> ) DEL	Deletes Cues or Groups.	<u>Delete Cue 1</u> <u>Delete Group 2</u>
Device ( <i>range</i> ) DEV or !	Targets one or more CueServers (by Device ID) for receiving CueScript commands. Typically used in conjunction with the ""* syntax for broadcasting messages to remote CueServers over the network.	" <u>Device 7</u> Cue 1 Go"* " <u>Device 5</u> >8 Output 1 On"*
Disable DIS	Disables currently selected Channels, Buttons or Contacts.	Channel 1>10 <u>Disable</u> Button 4 <u>Disable</u> Contact * <u>Disable</u>
Enable ENA	Enables currently selected Channels, Buttons or Contacts	Channel 1>10 <u>Enable</u> Button 4 <u>Enable</u> Contact * <u>Enable</u>
Fade ( <i>time</i> )	Changes the currently selected playback fader's cue crossfade time. Times from 0 to 6500 seconds may be specified in 0.1 second increments. Split fade times are expressed with a "/" separator.	<u>Fade 5</u> Cue 1 <u>Fade 10</u> Go Cue 5 <u>Fade 2.5/5</u> Go

Command	Description	Examples
Fixture ( <i>range</i> ) F	Selects one or more fixture's channels. The Fixture command works similarly to the Channel command, but it selects all channels of a fixture at a time.	<u>Fixture 1</u> At 50 <u>Fixture 2+4+6+8</u> At 75 <u>Fixture 1&gt;5</u> At {50,FL,0} <u>Fixture 3</u> Park
Follow ( <i>time</i> ) FO	Changes the currently selected playback fader's cue auto-follow time. Times from 0 to 6500 seconds may be specified in 0.1 second increments.	<u>Follow 10</u> Cue 1 <u>Follow 15</u> Go
Follow Clear FOCL	Clears the follow timer from the current playback fader.	<u>Follow Clear</u> Playback 2 <u>Follow Clear</u>
Go G	Causes the next cue in the current playback fader to execute.	<u>Go</u> Cue 1 <u>Go</u> Cue Fade 5 <u>Go</u>
Group ( <i>group number</i> ) GR or U	Recalls a set of channels saved in a group. Can be used similarly to the Channel or Fixture commands.	<u>Group 1</u> <u>Group 2</u> At 33 <u>Group 3</u> Release
HTP	<i>Deprecated. Use Merge instead.</i>	
IF ... THEN ... [ELSE ...] ENDIF	Conditionally executes statements depending on a boolean value. May optionally contain an ELSE statement for execution if the boolean is false.	<u>IF</u> {{enabled}} <u>THEN</u> Cue 1 <u>Go</u> <u>ENDIF</u>  <u>IF</u> {{showMode}} <u>THEN</u> Macro 1 <u>ELSE</u> Macro 2 <u>ENDIF</u>
Input ( <i>enable state</i> ) IN	Enables or disables the influence of the DMX Input port on the playback faders.	<u>Input Enable</u> <u>Input Disable</u>
Input Update INUP	Forces all DMX Input Triggers to execute their action commands.	<u>Input Update</u>
Join ( <i>station number</i> )	Logically joins the selected station(s) with the station specified by the Join command. See also Unjoin.	Station 1 <u>Join 2</u> Station 1>10 <u>Join 15</u>
Link ( <i>cue number</i> ) L	Changes the cue that will follow the current cue. By default, cues execute in numerical order. This command can allow any cue to follow the current cue.	<u>Link 1</u> Cue 1 <u>Link 101</u> Go
Log ( <i>string</i> )	Adds the given string to the System Log.	<u>Log "Hello World!"</u> <u>Log "Variable x = {{x}}"</u>
LTP	<i>Deprecated. Use Override instead.</i>	
Macro ( <i>macro number</i> ) M	Executes a user-defined macro.	<u>Macro 1</u>

Command	Description	Examples
Merge ME	Changes the currently selected playback fader's DMX combine mode to "merge", which combines the fader's levels with the previous levels in a "highest-takes-precedence" (HTP) manner.	<u>Merge</u> Playback 3 <u>Merge</u>
Next ++	Shifts the current selection to the next logical group of objects.	<u>Next</u> At FL Channel 1 At 10 <u>Next</u> At 20 Fixture 1 At FL <u>Next</u> At 50
Off	A special CueScript token that is equivalent to the command At 0. Can be used to set the value of any object.	Channel 1 <u>Off</u> Output 2+4 <u>Off</u> Button 1>5 <u>Off</u>
On	A special CueScript token that is equivalent to the command At FL. Can be used to set the value of any object.	Channel 1 <u>On</u> Output 2+4 <u>On</u> Button 1>5 <u>On</u>
Output ( <i>range</i> ) O	Selects one or more outputs for setting their output value.	<u>Output 1</u> At FL <u>Output 2+4+6</u> At 0 <u>Output 1&gt;8</u> Off
Override OV	Changes the currently selected playback fader's DMX combine mode to "override". When a fader is in override mode, any active channels in the fader replace the levels of other faders before it.	<u>Override</u> Playback 2 <u>Override</u>
Park	Parks the currently selected DMX channels. When a channel is parked, it is no longer affected by setting or releasing channels or cue playback. Parked channels become "frozen" at their current level until they are unparked.	Channel 1 <u>Park</u> Channel 101>200 <u>Park</u> Fixture 3 <u>Park</u> Group 7 <u>Park</u>
Playback ( <i>number</i> ) P	Selects one of the four playback faders as the target for all subsequent commands that operate on a fader.	<u>Playback 1</u> <u>Playback 2</u> Override <u>Playback 3</u> Cue 1 Go <u>Playback 4</u> Clear Playback 2 At 50
Previous PREV or --	Shifts the current selection to the previous logical group of objects.	<u>Previous</u> At 0 Channel 2 At 20 <u>Previous</u> At 10 Fixture 2 At 50 <u>Previous</u> At FL
Reboot	Reboots the CueServer. All show playback is immediately interrupted.	<u>Reboot</u>
Record Cue <i>[option] (cue number)</i> RQ	Records a cue into memory with the given cue number. Any cue number from 0.1 through 6499.9 may be used. By default, all 512 channels are recorded into the cue. To record only selected channels, use the '\$' option. To record no channels, use the '#' option.	<u>Record Cue 1</u> <u>Record Cue 101.5</u> Channel 1>10 <u>Record Cue \$5</u> <u>Record Cue #7</u>

Command	Description	Examples
Record Group (group number) RU	Records a group into memory with the currently selected channels included in the group.	<u>Record Group 1</u> Channel 1+3+5+7 <u>Record Group 2</u>
Record Stop RSTO	Stops any streaming cue that is currently being recorded.	<u>Record Stop</u>
Record Stream (cue number) RSTR	Begins recording a streaming DMX cue into memory with the given cue number. Any cue number from 0.1 through 4999.9 may be used. See the detailed description of Record Stream for details about its advanced options.	<u>Record Stream 101</u>  "_triggerchannel"=512; "_streamlength"=20.5; <u>Record Stream 102</u>
Release REL or Z	Removes the influence of the currently selected DMX channels from the currently selected playback fader. If no channels are selected, the Release command releases all channels in the playback fader.	<u>Release</u> Channel 1>10 <u>Release</u> Playback 2 Group 3 <u>Release</u>
Reset	Resets all four playback faders and the CueScript command interpreter to power-on values. Also resets Timecode and the command queue.	<u>Reset</u>
Scale SC	Changes the currently selected playback fader's DMX combine mode to "scale". When a fader is in scale mode, any active channels in the fader are used as a proportional scaling factor for values coming from the previous fader. Use scale mode to create grand masters or submasters.	<u>Scale</u> Playback 4 <u>Scale</u>
Self SE	Refers to the Button or Contact that resulted in the execution of the command.	<u>Self On</u> <u>Self At 4</u> <u>Self Disable</u>
SMPTE Reset SMPTE Start SMPTE Stop SMPTE Clear SMPTE "hh:mm:ss.ff"	Manages the internal generation of timecode. Allows the timecode generator to be reset to zero, started, stopped, cleared or sets the timecode to a user-specified value.	<u>SMPTE Reset</u> <u>SMPTE Start</u> <u>SMPTE Stop</u> <u>SMPTE Clear</u> <u>SMPTE "01:23:45.30"</u>
Start STA	Resumes normal timing operation of the current playback fader. Start resumes any crossfades, streaming cues or auto-follow timers that have been paused by the Stop command.	<u>Start</u> Playback 4 <u>Start</u>
Station (range)	Selects one or more stations for joining or unjoining with other stations.	<u>Station 1 Join 2</u> <u>Station 1&gt;10 Join 15</u>

Command	Description	Examples
Stop STO	Suspends the timing operation of the current playback fader. Any crossfades, streaming cues or auto-follow timers are paused. This function is useful for loading cues that are part of chases for editing.	<u>Stop</u> Playback 3 <u>Stop</u>
Time ( <i>time</i> ) T	Specifies the crossfade time of any channel values that are set with the <code>At</code> command. The <code>Time</code> command does not affect cue playback. Any time from 0 through 6500 may be specified in 0.1 second increments. A time of 0 (zero) indicates that values snap immediately to their desired levels.	<u>Time 5</u> <u>Time 10</u> Channel 1>10 <u>At FL</u> <u>Time 1.5</u> Fixture 3 <u>At 50</u>
Toggle ( <i>level</i> ) TOG	Toggles the level or value of the selected object(s) between 0 (zero) and the specified level.	Channel 1 <u>Toggle FL</u> Group 1 <u>Toggle 33</u> Button 1 <u>Toggle 1</u> Playback 1 <u>Toggle 100</u>
Unjoin ( <i>station number</i> )	Removes the logical connection between stations as created by the <code>Join</code> command.	Station 1 <u>Unjoin 2</u> Station 1>10 <u>Unjoin 15</u>
Unpark	Unparks the currently selected DMX channels. See the <code>Park</code> command for more details.	Channel 1 <u>Unpark</u> Channel 101>200 <u>Unpark</u> Fixture 3 <u>Unpark</u> Group 7 <u>Unpark</u>
Update Cue <i>[option] (cue number)</i> UP	Updates the DMX levels in a cue. Similar to recording a cue, but does not affect the cue name, timing parameters, link or command string. To update using only selected channels, use the '\$' option. To update with no channels, use the '#' option. To merge only selected channels, use the '~' option.	<u>Update Cue 1</u> Channel * <u>At FL Update Cue 2</u> Channel 5 <u>Update Cue \$3</u> <u>Update Cue #4</u> Group 1 <u>Update Cue ~5</u>
Wait ( <i>time</i> ) W	Inserts a delay into command execution. Any commands that appear after the <code>wait</code> command will be placed in a special Wait Queue for execution in the future.	<u>Wait 30</u> Cue 1 <u>Go</u> Button 1 <u>On Wait 5 Off</u> Cue 1 <u>Go Wait 1 Go Wait 1 Go</u>
Wait Clear	Clears <b>all</b> currently waiting commands in the Wait Queue.	<u>Wait Clear</u>
+ ( <i>and</i> )	Used to extend the current selection.	<u>+10</u> Channel 1 <u>+3</u> Button 1>4 <u>+8</u>
- ( <i>except</i> )	Used to remove objects from the current selection.	<u>-5</u> Channel 1>10 <u>-4</u> Button 1>4 <u>-2</u>



Command	Description	Examples
> ( <i>through</i> )	Used to specify a range of objects.	<u>&gt;10</u> Channel 101 <u>&gt;201</u> Output 1 <u>&gt;4</u>
* ( <i>wildcard</i> )	Used to select all objects of a given type.	Channel <u>_</u> At FL Button <u>*</u> On Output <u>*</u> Off Device <u>_</u> Cue 1 Go
"(command string)"*	Used to broadcast CueScript commands to other CueServers on the local network.	<u>"Cue 1 Go"*</u> <u>"Output 1&gt;8 Off"*</u> <u>"Device 7 Release"*</u>
"(string)"~ ( <i>string location</i> )	Stores a given string into a special location. Used to write to the LCD Display, serial ports, MIDI port, and Ethernet broadcast packets. See the command's documentation for details.	<u>"Hello World!"~0</u> <u>"Begin Show"~4</u>
"(variable)"="(value)"	Stores a value into a variable. Variables may be user-defined or system defined. The quotes around the value are optional if the value is a number.	<u>"x"=3</u> <u>"myCueNumber"=10.5</u> <u>"myPrompt"="Press Start"</u> <u>"_backlight"=50</u>
{{(variable)}}}	Substitutes the value of a variable or system function into the command line.	Group <u>{{x}}</u> At 33 Cue <u>{{myCueNumber}}</u> Go <u>{{myPrompt}}</u> "~0 Log "Light = <u>{{_backlight}}</u> " Cue <u>{{_rand(1,10)}}</u> Go
;	The semicolon (;) is for visually separating parts of a long string of CueScript commands to help make CueScript more readable. CueServer ignores semicolons in commands.	Group 1 At 33; <u> Group 2 At 50</u>  Channel 1 <u>&gt;10</u> ; Record Cue \$101  Button 1 On; <u> Wait 10</u> ; Off



---

# Appendix B: System Variables

---

There are several globally-defined system variables that affect the CueServer hardware or specific areas of the CueServer operating system. The user can change these variables to customize some aspects of CueServer's behavior.

See the section on CueScript Variables for a complete description of how both user-defined and system variables are used in the CueScript language.

In general, the following variables can be set using the following syntax:

```
"_backlight"=50
```

This example shows the variable "\_backlight" set to the value 50, which will set the LCD backlight on the CueServer Pro to 50% brightness.

The following is a listing of system variables:

## **\_backlight**

This is the intensity of the LCD display backlight (on CueServer Pro only). Possible values are from 0 to 100 (FL). Keep in mind that CueServer Pro uses white LED backlighting for the LCD Display, which has a considerably longer life than EL backlighting.

## **\_dmxinport**

This system variable affects the DMX Input port hardware. When set to 1 (default), the port behaves normally, allowing a standard DMX-512 signal to be received. When set to 0, the port's circuitry is shut down, causing no signal to be received.

## **\_dmxoutput**

This system variable affects the DMX Output port hardware. When set to 1 (default), the port behaves normally, outputting a standard DMX-512 signal. When set to 0, the port stops transmitting DMX, causing any connected fixtures to assume that the DMX cable has been disconnected. When set to 2, the port's internal fail-safe relay de-energizes, causing the DMX Input port to become electrically connected to the DMX Output port, while disconnecting CueServer's internal circuitry (the fail-safe relay function is only available on the CS-800 model). While the DMX Output port is disabled (in either mode 0 or 2, CueServer continues to internally run its shows and perform all other operations.

## **\_fixturesize**

This is the default number of channels in a fixture, as used by the `Fixture` command. This variable is normally set to 3 to accommodate RGB-type fixtures, however any value from 1 to 255 may be specified.

## **\_streamlength**

This is the default recording duration of a stream in seconds (accurate to 0.1s). By default, this value is 0 (meaning that the default is unlimited recording time). This variable can be useful when attempting to automate the stream recording process via CueScript commands.

**\_triggerchannel**

This is the default DMX Input channel used to trigger the automatic start and stop of stream recording. If set to any channel from 1 through 512, then when stream recording is armed, CueServer will wait until the specified channel rises above 0 to automatically start recording. Then, when the specified channel falls back to 0, stream recording is automatically stopped. If the trigger channel is set to zero, then this feature is disabled.

**\_udpip**

This system variable holds the IP Address that will be used to broadcast packets when the ~3 store command is used. See the Store command for additional information.

**\_udpport**

This system variable holds the Port Number that will be used to broadcast packets when the ~3 store command is used. See the Store command for additional information.

---

# Appendix C: System Functions

---

There are several functions that return values that can be used in CueScript statements.

See the section on CueScript Variables for a complete description of how both user-defined and system variables are used in the CueScript language.

In general, the following functions can be used using the following syntax:

```
{{_rand(1,10)}}
```

This example shows the function “\_rand” used to randomly pick a number from 1 to 10 and substitute it into the CueScript statement.

This syntax is typically used as part of a more complex statement, such as:

```
Cue {{_rand(1,10)}} Go
```

This example uses the “\_rand” function to randomly pick a cue number from 1 to 10 and execute it.

The following is a listing of system functions:

## **\_button(x)**

Returns the current value of Button x. This is the value of the LED indicator, as set by using the Button CueScript command.

## **\_dmxout(x)**

Returns the current value of the DMX output channel x.

## **\_output(x)**

Returns the current value of Output x. This is the value of the Output port, as set by using the Output CueScript command.

## **\_rand(x, y)**

This system function randomly picks a number from x through y.



---

# Appendix D: Warranty Information

---

## LIMITED WARRANTY

Interactive Technologies, Inc. (Interactive) warrants to You that, for a period of one year (the “Warranty Period”), your Interactive Product will be substantially free of defects in materials and workmanship under normal use. Your exclusive remedy and Interactive’s entire liability under this warranty will be for Interactive at its option to repair or replace the Product. This limited warranty extends only to the original purchaser. Proof of purchase may be required to obtain warranty coverage.

If the Product proves defective during the Warranty Period call Interactive Technical Support in order to obtain a Return Authorization Number, if applicable. If You are requested to return the Product, mark the Return Authorization Number clearly on the outside of the package. You are responsible for shipping defective Products to Interactive. Interactive pays for UPS Ground shipping from Interactive back to You only. Customers located outside of the United States of America and Canada are responsible for all shipping and handling charges.

ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE ARE LIMITED TO THE DURATION OF THE WARRANTY PERIOD. ALL OTHER EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF NON-INFRINGEMENT, ARE DISCLAIMED. Some jurisdictions do not allow limitations on how long an implied warranty lasts, so the above limitation may not apply to You. This warranty gives You specific legal rights, and You may also have other rights which vary by jurisdiction.

This warranty does not apply if the Product (a) has been altered, except by Interactive, (b) has not been installed, operated, repaired, or maintained in accordance with instructions supplied by Interactive, or (c) has been subjected to abnormal physical or electrical stress, misuse, negligence, or accident. In addition, due to the continual development of new techniques for intruding upon and attacking networks, Interactive does not warrant that the Product will be free of vulnerability to intrusion or attack.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL INTERACTIVE BE LIABLE FOR ANY LOST DATA, REVENUE OR PROFIT, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, REGARDLESS OF THE THEORY OF LIABILITY (INCLUDING NEGLIGENCE), ARISING OUT OF OR RELATED TO THE USE OF OR INABILITY TO USE THE PRODUCT (INCLUDING ANY SOFTWARE), EVEN IF INTERACTIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT WILL INTERACTIVE’S LIABILITY EXCEED THE AMOUNT PAID BY YOU FOR THE PRODUCT. The foregoing limitations will apply even if any warranty or remedy provided under this Agreement fails of its essential purpose. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to You.

Please direct all inquiries to: Interactive Technologies, Inc., 5040 Magnolia Creek Drive, Cumming, GA 30028.

