

CueServer 2 User's Manual

18.12.21 — Last update: 2018/12/21

Interactive Technologies, Inc.

Distributor:



Phone: 407-857-8770

Fax: 407-857-8771

Email: sales@techni-lux.com

www.techni-lux.com

Table of Contents

Getting Started	7
CueServer Studio 2	8
Navigator Window	9
Toolbar	12
Working With Shows	14
Working With Offline Shows	17
Working With Remote CueServers	20
Setting Network Parameters	22
Setting Clock Parameters	26
Identifying CueServers	32
Updating Firmware	33
Editor Window	35
Live	37
Stage	38
Playbacks	41
Status	44
Front Panel	45
Variables	46
CPU Info	47
System Log	48
Resources	49
Cues	50
Cue Types	52
Adding Cues	53
Cue Properties	54
Cue Contents	56
Capturing DMX Snapshots	58
Capturing DMX Streams	60
Clearing Cue Contents	63
Cue Rules	64
Deleting Cues	65
Hardware	66
Models	67
CS-900 CueServer 2 Pro	69
CS-920 CueServer 2 Mini	71

CS-940 CueServer 2 DIN.....	73
Power Input.....	74
Ethernet Ports.....	76
Ethernet Protocols.....	78
sACN (Streaming ACN) Protocol.....	79
Art-Net Protocol.....	80
KiNET Protocol.....	81
CueScript Protocol.....	82
CueStation Protocol.....	83
HTTP Protocol.....	84
DHCP Protocol.....	85
NTP Protocol.....	86
DMX Ports.....	87
DMX Modules.....	89
Audio Ports.....	91
Supported Audio File Formats.....	92
WAV Sample Formats.....	93
USB Ports.....	95
LCD Display.....	96
LCD Display Modes.....	97
LCD Status Options.....	98
LCD Menu Functions.....	100
Function Buttons.....	101
Contact Closures.....	102
Digital Outputs.....	103
Serial Ports.....	105
Memory Card.....	107
Reset Button.....	109
Self-Test Function.....	110
Using CueServer.....	118
DMX Triggers.....	119
Enter/Exit Range Trigger.....	121
Submaster Control Trigger.....	123
Act on Changes Trigger.....	125
CueScript Language.....	127
CueScript Overview.....	128
Executing Commands.....	129
CueScript via UDP.....	131

CueScript via HTTP	132
CueScript via Serial	133
Command Syntax	135
Expressions	136
Operators	137
Variables	139
Grouping	141
Command Context	142
Levels	143
Strings	145
Selection Commands	148
Button	150
Channel	153
Contact	156
Group	158
Indicator	160
Output	163
Playback	165
Station	168
Universe	170
Selection Operators (+, -, >, ~)	172
Using Wildcards	174
Action Commands	175
Assign (=)	176
Audio	178
At	180
Clear	185
Cue	187
Disable	189
Enable	191
Fade	193
Follow	195
Go	197
Input	199
Join	200
Length	203
Link	205
Lock	207
Log	209

Macro	210
Off	212
Offset.....	213
On	215
Park.....	216
Preset.....	218
Press	221
Random.....	223
Reboot.....	225
Record.....	226
Record Cue	227
Record Group.....	230
Record Stream	231
Record Stop	234
Release	235
Reset.....	238
Set.....	239
SMPTE	241
Stack	244
Start	246
Stop.....	247
Time	248
Toggle	250
Unpark.....	252
Update.....	254
Update Cue	255
Update Group.....	258
Update Stream	260
Update Stop	263
Unlock	264
Wait	266
Write.....	269
Zone	271
Logic Commands	273
Break.....	274
If..Then..Else	275
System Variables	278
Internals	284

Web Server.....	285
Environment Variables.....	286
CGI API	287
exe.cgi.....	288
get.cgi.....	290
Button Values [bv]	291
Command Context [cc]	292
CPU SysInfo [cpu]	294
Cue Stack Info [csi]	295
DMX Input [in]	297
DMX Output [out].....	299
Extended Command Context [ecc].....	301
Extended Playback Info [epi]	303
Fade Engine Data [fed].....	305
Group Level [grp].....	308
Hardwired DMX Input [hdi].....	309
Network Info [net]	310
Ping [ping]	312
Playback Info [pi]	314
Playback Values [p*].....	316
Preset Zone Info [pzi]	318
Record Stream Info [rs]	320
System Log [log].....	321
System Status [ss].....	322
Time Info [ti]	324
Time Status [ts]	325
Variables [var]	327
Zone Data [zones]	328
pcmd.cgi.....	330
set.cgi.....	331
Audio Properties [audio]	332
LCD Properties [lcd]	333
Network Properties [net]	335
Time Properties [time]	337
Station Color Properties [stcol]	339
Show File Format.....	341
Directory Structure.....	342
Configuration Files.....	345
show.cfg.....	346

Resource Structures	351
Cue Resource	352
DMXTrigger Resource	356
Group Resource	358
Marco Resource	359
Hardware Model Identifiers	360
Autodiscovery	362
Release Notes.....	364
Release v3.0.0 [Pre-Release].....	365
Release v2.1.2 [Pre-Release].....	367
Release v2.1.1 [May 18, 2018].....	368
Release v2.1.0 [May 16, 2018].....	369
Release v2.0.4 [March 14, 2018].....	371
Release v2.0.3 [February 14, 2018]	373
Release v2.0.2 [January 22, 2018].....	374
Release v2.0.1 [November 3, 2017]	376
Release v2.0.0 [October 24, 2017].....	377
Release v1.5.5 [October 28, 2016].....	385
Release v1.5.4 [September 8, 2016]	386
Release v1.5.3 [August 9, 2016]	387
Release v1.5.2 [July 25, 2016]	388
Release v1.5.1 [July 19, 2016]	389
Release v1.5.0 [June 3, 2016].....	390
Release v1.4.3 [April 18, 2016]	393
Release v1.4.2 [March 17, 2016].....	394
Release v1.4.1 [February 24, 2016]	396
Release v1.4.0 [January 21, 2016].....	397
Release v1.3.0 [November 11, 2015]	400
Release v1.2.0 [July 24, 2015]	402
Release v1.1.0 [May 22, 2015].....	404
Release v1.0.8 [April 27, 2015]	405
Release v1.0.7 [April 7, 2015]	407
Release v1.0.6 [March 13, 2015].....	409
Release v1.0.5 [March 11, 2015].....	410
Release v1.0.4 [February 9, 2015]	412
Release v1.0.3 [January 22, 2015].....	414
Release v1.0.2 [January 9, 2015].....	415
Release v1.0.1 [December 23, 2014]	416

Release v1.0.0 [December 18, 2014]	417
Legal Notices	418

Getting Started

Welcome to CueServer 2.


Sections are being added to this User's Manual on a regular basis.

Current Version

Note that on May 18, 2018, version **2.1.1** of CueServer Studio was released. Please download this latest version here:

- [CueServer 2 Downloads](#)

CueServer Studio can be downloaded as a .dmg file for Macs and a .exe file for Windows.

Whenever you update to a new version of CueServer Studio, it is likely that you will also need to update the firmware in your CueServer. If a firmware update is needed, a yellow caution icon () will appear next to the CueServer's firmware version in the Navigator window. To update your CueServer, choose the *Update Firmware...* menu command in the *CueServer* menu to update your device.

CueServer Studio 2

CueServer Studio 2 is the desktop application used to program, configure, locate and operate CueServer 2 devices. It is available for both Mac OS X and Windows. You can download the current version of CueServer Studio 2 here:

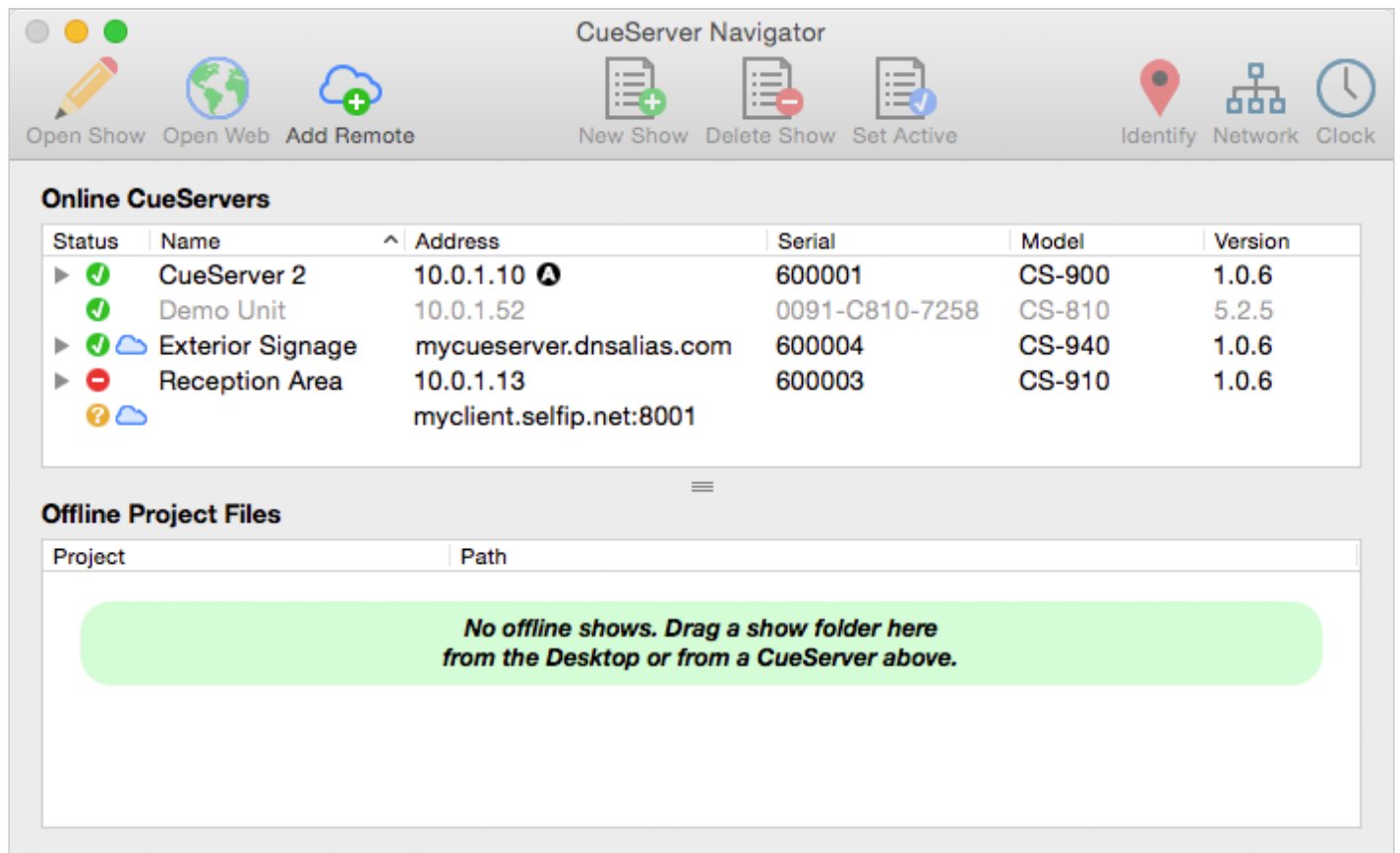
- <http://interactive-online.com/cueserver2/downloads>



Navigator Window

Overview

The *Navigator Window* appears when CueServer Studio opens. Use the Navigator Window to view available CueServers, manage basic settings, change active shows, identify individual devices, update firmware and more.



The screenshot shows the CueServer Navigator window with a toolbar at the top containing icons for Open Show, Open Web, Add Remote, New Show, Delete Show, Set Active, Identify, Network, and Clock. The main area is divided into two panes. The top pane, titled "Online CueServers", contains a table with the following data:


Status	Name	Address	Serial	Model	Version
▶ ✓	CueServer 2	10.0.1.10	600001	CS-900	1.0.6
▶ ✓	Demo Unit	10.0.1.52	0091-C810-7258	CS-810	5.2.5
▶ ✓	Exterior Signage	mycueserver.dnsalias.com	600004	CS-940	1.0.6
▶ -	Reception Area	10.0.1.13	600003	CS-910	1.0.6
▶ ?		myclient.selfip.net:8001			

The bottom pane, titled "Offline Project Files", contains a table with columns for Project and Path. A green message box in the center of this pane reads: "No offline shows. Drag a show folder here from the Desktop or from a CueServer above."




The top pane of this window displays both local and remote CueServers along with their online status, name, address, model and firmware version. The bottom pane is used for working with offline project files.

Working With Online CueServer Devices



The Navigator Window constantly scans the local network and displays any CueServers that are available. These devices will automatically appear in the upper list and will have a green status icon (✓).


Remote CueServers can also be added to the upper list manually. These CueServers will appear with a cloud icon () as part of their status. See [Working With Remote CueServers](#) for more information.

The Status column shows various icons depending on the current state of a device in the list:

-  The CueServer is online.
-  The CueServer is being contacted.
-  The CueServer is offline.


For CueServer devices that are configured to use separate LANs for management and lighting data, the Address column will show which LAN port on the CueServer was used to connect to the device. These icons only appear when the CueServer is configured to use Dual-LANs:

-  LAN Interface A
-  LAN Interface B

If a CueServer is discovered, but it is not reachable on the local network because of a mismatch between the computer's local subnet and the CueServer's IP address, then a warning icon () will appear next to its address. Try using [Network Settings](#) to change the CueServer's IP address to one that is reachable on the local network.

Older CS-800 series CueServers will also appear in the list of online devices. CueServer Studio 2 can not directly edit these devices, and they appear in the list in gray text. If they are opened, CueServer Studio will simply open the device's web interface.

Editing Online CueServers

Double clicking a CueServer or clicking on the Open Show icon () opens that CueServer's Editor Window, which is used to program and configure the CueServer. See the [Editor Window](#) section for more information.

Opening the listbox under a CueServer reveals the available and active show file in the CueServer. Options are available to manage the active show, and to create new, delete and rename shows. See [Working With Shows](#) for more information.



Working With Offline Show Files

The bottom pane of the window is used as a working area to hold offline show files.

This pane makes it easy to open and edit show files that are on the local computer, or to copy shows between a CueServer and the local computer.


See the section on [Working With Offline Shows](#) for more details.

Setting Network or Clock Parameters

When a CueServer is selected, its Network and Clock parameters can be set using options from the CueServer menu, or by right-clicking (or control-clicking) the CueServer to get a contextual menu. Also, a Network button () and a Clock button () are available in the toolbar for easy access to these functions.

See the sections on [Setting Network Parameters](#) or [Setting Clock Parameters](#) for more information.

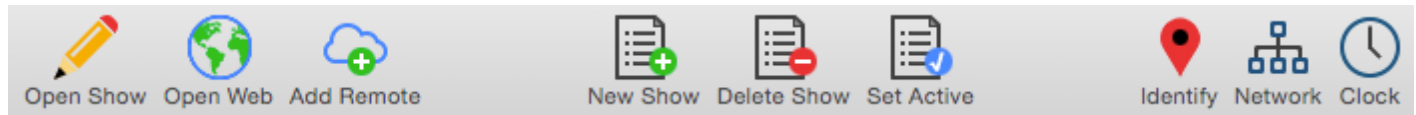
Maintenance

If the firmware of a CueServer is out-of-date, a warning icon () will appear next to its firmware version. See the [Updating Firmware](#) section for details.

If there are multiple CueServers on the network at the same time, it can sometimes be useful to identify which CueServer is which. See the [Identifying CueServers](#) section to learn how to activate the Identify function.

Toolbar

The toolbar in the Navigator Window contains several controls for managing CueServers.



Each of the toolbar items are described below:



Open Show

Opens the currently selected CueServer's Editor Window. The Editor Window is used for programming and configuration of a CueServer.



Open Web

Opens the currently selected CueServer's web page in the default web browser.



Add Remote

Displays a dialog window that allows a remote CueServer to be added to the Navigator Window.

This option is used to add CueServers that are not available on the local network, and are published on the Internet via a router's port-forwarding settings. See [Working With Remote CueServers](#) for more information.



New Show

Creates a new show file for the selected CueServer.



Delete Show

Removes the selected show file from a CueServer. Please note that the currently active show file cannot be deleted.



Set Active

Makes the selected show file the *active show*. The active show appears in the list in bold with a blue checkmark besides it.



Identify

Activates the selected CueServer's *Identify Mode*. When a CueServer is in Identify Mode, its LCD Display and Power LED will flash. Use this feature to help identify which CueServer is which in a complicated setup with multiple CueServer devices. See [Identifying CueServers](#) for more information.

**Network**

Displays a dialog window that allows the network settings of the selected CueServer to be changed. Use this option to change the IP Address, DHCP setting, and Device Name of a CueServer.

**Clock**

Displays a dialog window that allows the clock settings of the selected CueServer to be changed. Use this option to change the time zone, automatic and/or manual time settings of a CueServer.

Working With Shows

About Shows

All of the programming and configuration in a CueServer is stored in a *show file*. CueServer show files contain Cues, Groups, Macros, Sounds, Web Pages, Stations, Timers, Rules, Configuration Data and more. The memory card in CueServer can hold one or more show files, however only one show can be active at a time.


The shows available on a CueServer's memory card are displayed by opening the hierarchical list under the CueServer in the Navigator Window.

The screenshot shows the CueServer Navigator window. At the top, there are several icons for actions: Open Show (pencil), Open Web (globe), Add Remote (cloud with plus), New Show (document with plus), Delete Show (document with minus), Set Active (document with checkmark), Identify (location pin), Network (hierarchy), and Clock (clock). Below these icons is a table of Online CueServers. The table has columns for Status, Name, Address, Serial, Model, and Version. The first entry is CueServer 2 at 10.0.1.10, which is expanded to show three shows: Attraction, Demo Show (bolded and with a blue checkmark icon), and Example. The second entry is Demo Unit at 10.0.1.52, and the third is Exterior Signage at mycueserver.dnsalias.com. Below the table is a section for Offline Project Files, which is currently empty and contains a message: "No offline shows. Drag a show folder here from the Desktop or from a CueServer above."


Status	Name	Address	Serial	Model	Version
▼ ✓	CueServer 2	10.0.1.10	600001	CS-900	1.0.6
	Attraction				
	Demo Show				
	Example				
✓	Demo Unit	10.0.1.52	0091-C810-7258	CS-810	5.2.5
▶ ✓	Exterior Signage	mycueserver.dnsalias.com	600004	CS-940	1.0.6

Offline Project Files

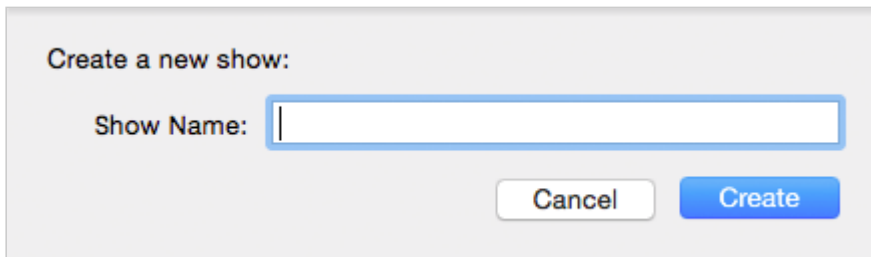
Project	Path
No offline shows. Drag a show folder here from the Desktop or from a CueServer above.	

In the above example, the device named CueServer 2 contains three shows. The show marked in bold and with the blue checkmark icon () next to it is the currently active show in the CueServer.

Creating a New Show

To create a new show, click on the New Show toolbar item (). You can also find the **New Show** command in the contextual menu or the CueServer menu.

A window will appear asking for a new show name:




Create a new show:

Show Name:

Cancel Create

Enter a unique show name and press **Create** to create the new show.

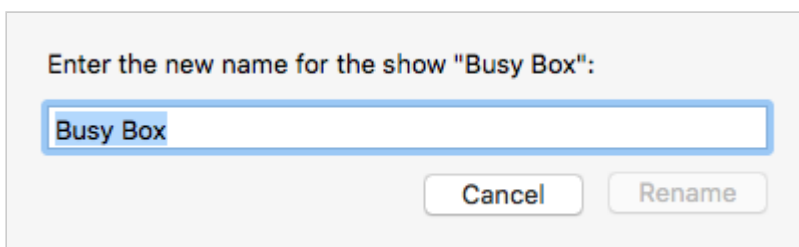
Changing the Active Show

To change the currently active show, click on a show file and then choose the **Set Active Show** menu item or click on the Set Active toolbar item (). You can also find the **Set Active Show** command in the contextual menu or the CueServer menu.

Renaming a Show

To rename a show, right-click on the show and choose **Rename Show** from the contextual menu. You can also find the **Rename Show** command in the CueServer menu.

A dialog window will appear that allows you to rename the show:




Enter the new name for the show "Busy Box":

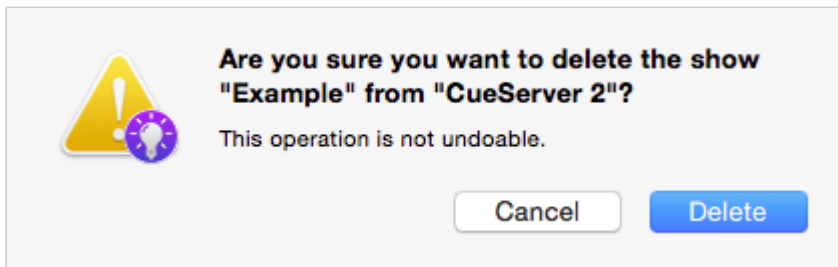
Busy Box

Cancel Rename


Deleting a Show

To delete a show, click on the show file and then click on the Delete Show toolbar item (). You can also find the **Delete Show** command in the contextual menu or the CueServer menu.

A confirmation dialog will appear:

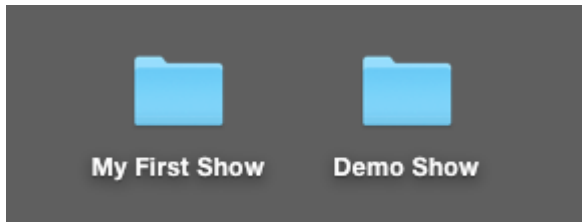


To proceed with deleting the show, choose the **Delete** button.

 You cannot delete the currently active show. If you want to delete the active show, first switch to another show (or create a new one).

Working With Offline Shows

A *Show File* is a directory that contains the data stored in the show. The contents of the Show File directory are individual binary files and subdirectories for each object in the show, including Cues, Macros, Rules, Timers, Sounds, Web Content and more.



Since a Show File is actually a directory, it can't be opened on the computer like a regular data file. If you double-click on a Show File directory on your desktop, it will just open like any regular folder. Because of this, CueServer Studio has tools for working with Show File directories that make it easier to edit them.

Online CueServers

Status	Name	Address	Serial	Model	Version
▼ ✓	CueServer 2	10.0.1.10	600001	CS-900	1.0.6
	Attraction				
	✓ Demo Show				
	Example				
✓	Demo Unit	10.0.1.52	0091-C810-7258	CS-810	5.2.5
▶ ✓	Exterior Signage	mycueserver.dnsalias.com	600004	CS-940	1.0.6

Offline Project Files

Project	Path
<p><i>No offline shows. Drag a show folder here from the Desktop or from a CueServer above.</i></p>	

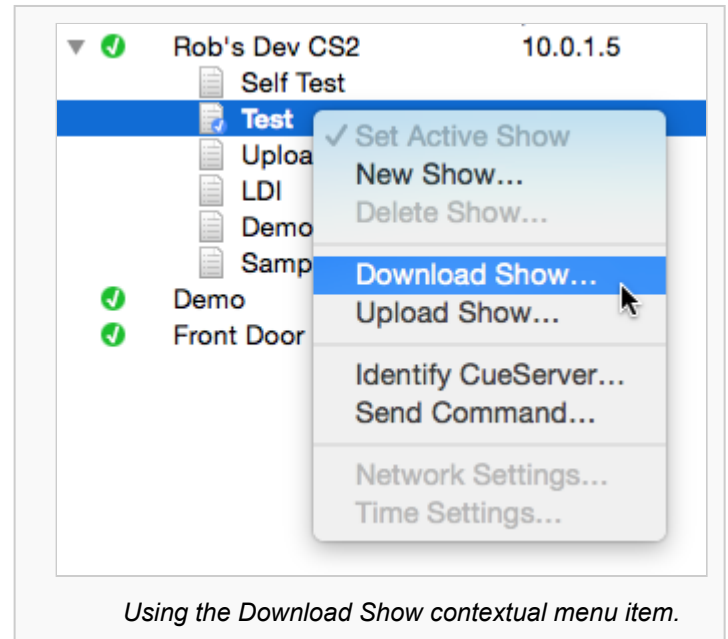
Downloading a Show File from CueServer to Computer

There are several ways to download a show file from a CueServer to the computer.

Option 1: Use the *Download Show...* menu item available in the CueServer menu.

Option 2: Use the *Download Show...* contextual menu item available by right clicking (or control-clicking) on the show file in the CueServer.

When using either Option 1 or 2, a standard file chooser dialog will appear asking where to place the downloaded show file. Once a destination folder is chosen, CueServer Studio will download the show file into the location chosen.



Option 3: Drag the show file directly from the CueServer in the top panel to the *Offline Shows* panel at the bottom of the window.

When dragging a show from the online panel to the offline panel, CueServer Studio will automatically download the show file from the CueServer to the computer's desktop and add the item to the offline projects list.

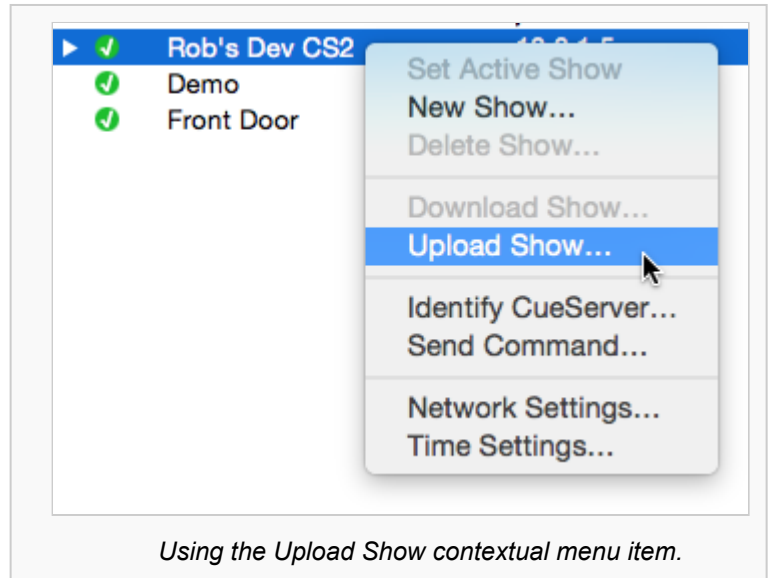
Uploading a Show File from Computer to CueServer

There are several ways to upload a show file from a computer to a CueServer.

Option 1: Use the *Upload Show...* menu item available in the CueServer menu.

Option 2: Use the *Upload Show...* contextual menu item available by right clicking (or control-clicking) on a CueServer.

When using either Option 1 or 2, a standard file chooser dialog will appear asking to choose the show file to upload. Once a show file is chosen, CueServer Studio will upload the show file to the selected CueServer.



Option 3: Drag a show folder directly from the *Offline Shows* panel at the bottom of the window to an online CueServer.

Option 4: Drag a show folder directly from the computer's Desktop to an online CueServer.

When dragging a show from the offline panel or Desktop to an online CueServer, CueServer Studio will automatically upload the show file from the computer to the CueServer device.

Creating An Offline Show

To create a show file for offline editing, first click in the *Offline Project Files* list to select it.


Then, click the *New Show* toolbar item ().

A standard file save dialog window will appear, asking for a name and location to save the new show file.

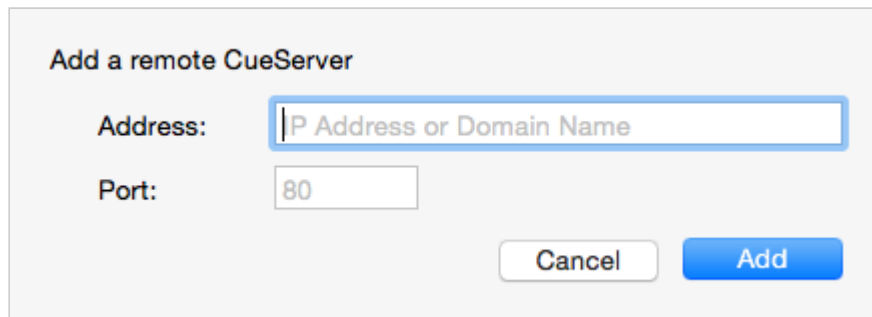
Once the name and location are given, CueServer Studio will create the new show file and add it to the *Offline Project Files* list so the offline show file can be opened and edited.

Working With Remote CueServers

Adding a Remote CueServer

To add a CueServer to the Navigator Window that is “across the Internet” (i.e., not on the local network), choose **Add Remote CueServer...** from the CueServer Menu, or click the **Add Remote** button () in the toolbar.

The **Add Remote CueServer** window will appear:



The screenshot shows a dialog box titled "Add a remote CueServer". It contains two input fields: "Address:" with a text box containing the placeholder "IP Address or Domain Name", and "Port:" with a text box containing the value "80". At the bottom of the dialog, there are two buttons: "Cancel" and "Add".

The fields in this window are described below:

Address


This field can accept either an IP Address (for example: 50.167.102.1), or a domain name (such as: mycueserver.dnsalias.com).

Port

This field is used to specify the *port number* of the remote CueServer. If left blank, the default port 80 will be used. Valid port numbers range from 1 to 65535.

To add a remote CueServer (after the fields are filled out properly), click **Add**.




Viewing Remote CueServers in the Navigator Window

Once a Remote CueServer has been added to the Navigator Window, it will appear in the CueServer list with a small cloud icon () next to the status icon. For example:

Status	Name	Address	Serial	Model
▶  	CueServer 2	mycueserver.dnsalias.com	600004	CS-940

The cloud icon shows that the CueServer in the list is a Remote CueServer.

The following icons can appear in the status column for Remote CueServers:

-  The CueServer is online.
-  The CueServer is being contacted.
-  The CueServer is offline.




Remote CueServers that connect properly are automatically saved in the application's preferences. Each time the application is launched, the added Remote CueServers will re-appear. If an added Remote CueServer cannot be contacted, it will not be saved in the preferences.

Removing Remote CueServers from the Navigator Window

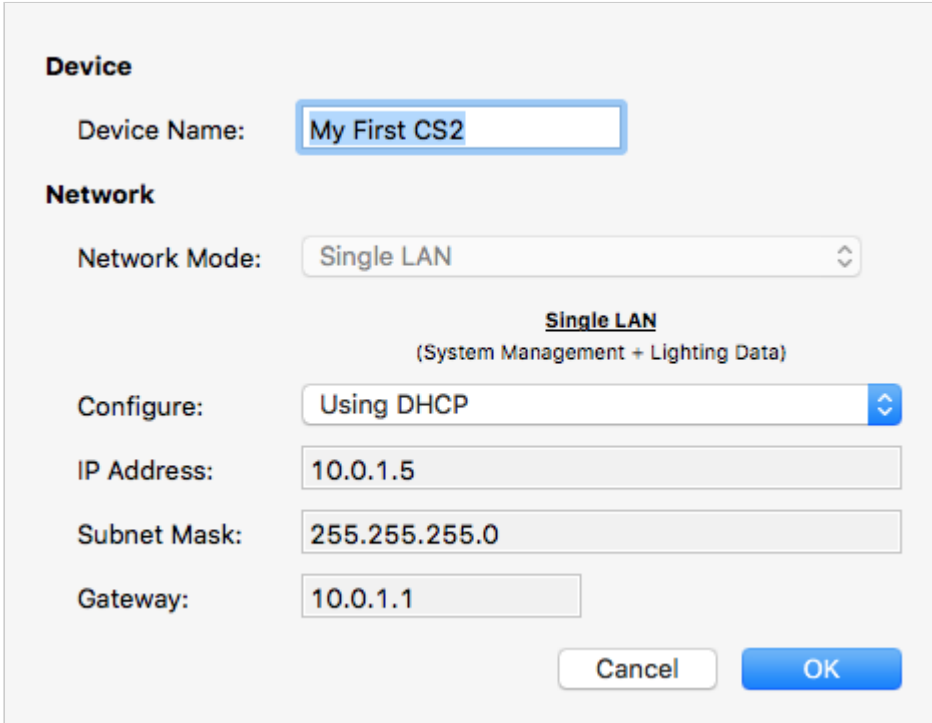
Simply select the Remote CueServer, and then press the **Delete** key on your keyboard.

Setting Network Parameters

When a CueServer is selected in the Navigator Window, its various network parameters can be changed by clicking on the Network Toolbar Item (), or by selecting the *Network Settings...* menu item in the CueServer menu.

These parameters include the device's network name, DHCP settings, IP Address, Subnet and Gateway addresses.

A dialog window similar to the following will appear:



Device

Device Name:

Network

Network Mode:

Single LAN
(System Management + Lighting Data)

Configure:

IP Address:

Subnet Mask:

Gateway:

Device Name

This is the name of the device on the network (sometimes called the *hostname*). The device name can be set to any practical name that can be used to identify the CueServer on the network.

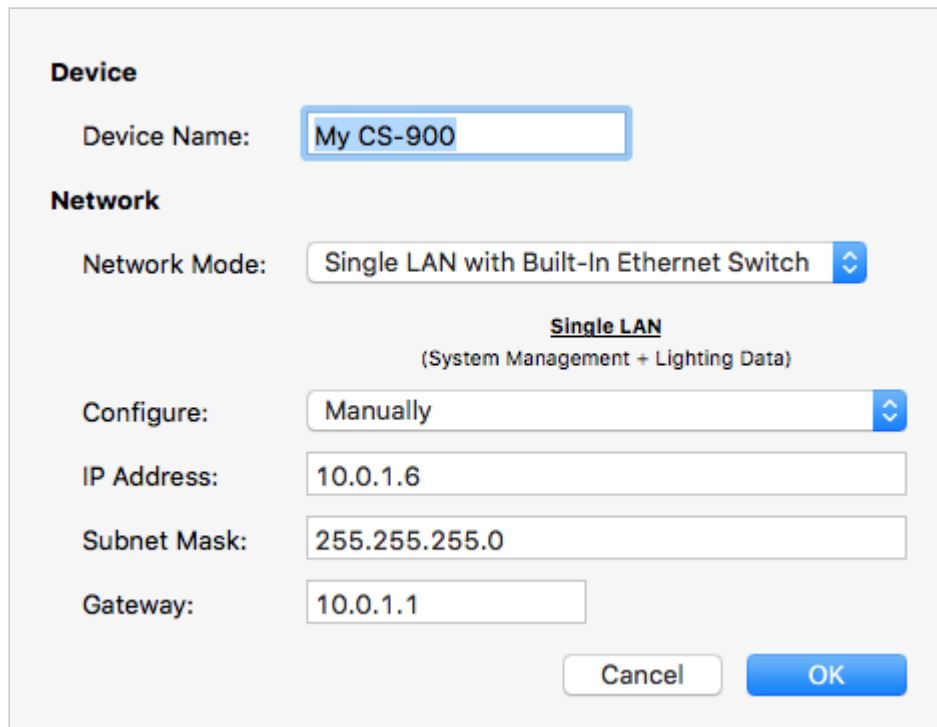
Network Mode

On CueServers with only a single Ethernet jack, this option is fixed to “Single LAN”.

On CueServers with two Ethernet jacks, two different options are available in this menu:

Option 1: Single LAN with Built-In Ethernet Switch

When this mode is selected (which is the factory default), the two Ethernet jacks are simply two ports of a built-in Ethernet switch, both of which are connected to the CueServer. In this configuration, either one of the two jacks can be connected to the local network, and the other jack can be used as an extra port for connecting a laptop, DMX node, CueStation Hub or any other network device. All of the CueServer management and lighting control data flows over this single LAN connection.



The screenshot shows a configuration dialog box for the CueServer. It is divided into two main sections: "Device" and "Network".

- Device:** The "Device Name" field contains the text "My CS-900".
- Network:** The "Network Mode" dropdown menu is set to "Single LAN with Built-In Ethernet Switch". Below this, the text "Single LAN" is displayed in bold, followed by "(System Management + Lighting Data)" in a smaller font.
- The "Configure:" dropdown menu is set to "Manually".
- The "IP Address:" field contains "10.0.1.6".
- The "Subnet Mask:" field contains "255.255.255.0".
- The "Gateway:" field contains "10.0.1.1".

At the bottom of the dialog, there are two buttons: "Cancel" and "OK".

Option 2: Dual LANs with Separate IP Addresses

When this mode is selected, the two Ethernet jacks become two separate LAN ports, each of which have their own network settings. The jack marked “A” is used to connect to a device management network. LAN “A” can be used for device discovery, configuration editing, and has access to the web interface. The jack marked “B” will have a different IP address and is used for lighting control data. LAN “B” is where DMX-over-Ethernet protocols such as sACN, Art-Net, and KiNET are flowing. LAN “B” can also be used for device discovery, configuration editing, and has access to the web interface.

Device		
Device Name:	<input type="text" value="My CS-900"/>	
Network		
Network Mode:	<input type="text" value="Dual LANs with Separate IP Addresses"/>	
	LAN A (System Management)	LAN B (Lighting Data)
Configure:	<input type="text" value="Manually"/>	<input type="text" value="Manually"/>
IP Address:	<input type="text" value="10.0.1.6"/>	<input type="text" value="192.168.1.123"/>
Subnet Mask:	<input type="text" value="255.255.255.0"/>	<input type="text" value="255.255.255.0"/>
Gateway:	<input type="text" value="10.0.1.1"/>	
<input type="button" value="Cancel"/> <input type="button" value="OK"/>		

! Please note that when changing the network mode, the device will need to reboot for the changes to take effect. Please remember to reconfigure the physical network connections when changing modes, especially if the mode is being changed from a Dual LAN to a Single LAN configuration. In this case, it's likely that there were two separate networks attached to the two ports on the CueServer and after the mode is changed the built-in Ethernet switch would attempt to bridge these separate networks into one, which will certainly cause unintended network problems.

Network Address

CueServer allows the Network Address to be set manually or automatically. If the CueServer is configured for Dual LANs, then each network (A and B) can have it's own network settings.

If the CueServer is on a network with a DHCP server or Router (which is common in buildings, offices and home networks), this setting can be set to *Using DHCP*.

Using DHCP

When *Using DHCP* is chosen, the IP Address fields become disabled. This is because the CueServer will fetch these address parameters from the network automatically. There is no need to set these parameters manually when using DHCP.

Manually


When *Manually* is chosen, the IP Address fields can be entered with a static IP Address, Subnet and Gateway address.

It is best to use this option if the CueServer is not connected to a network, or if the network is known to not have a DHCP server or Router, or if a specific network configuration is desired that uses a static address for the CueServer.



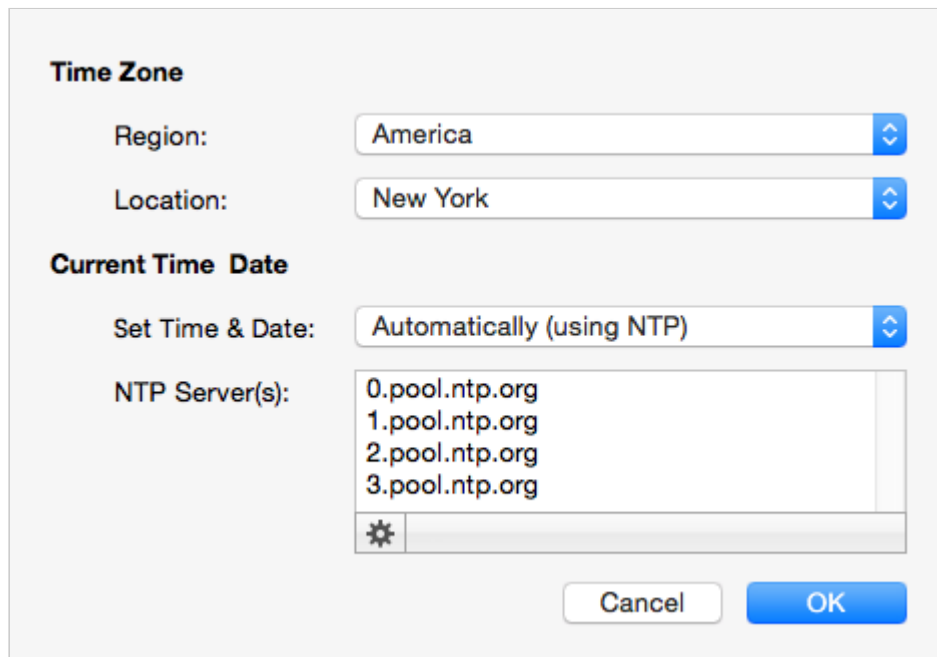
Please note that if the CueServer is configured to use Dual LANs, the second network (LAN "B") does not have a field for setting a default gateway. This is because all network traffic that would need to use a gateway to reach the external Internet will flow through LAN "A".

Setting Clock Parameters

When a CueServer is selected in the Navigator Window, it's various clock parameters can be changed by clicking on the Clock Toolbar Item (), or by selecting the *Time Settings...* menu item in the CueServer menu.

These parameters include the timezone the unit is located within, network time protocol (NTP) server configuration, and/or manual date and time settings.

A dialog window similar to the following will appear:



Time Zone

Region:

Location:

Current Time Date

Set Time & Date:

NTP Server(s):

- 0.pool.ntp.org
- 1.pool.ntp.org
- 2.pool.ntp.org
- 3.pool.ntp.org

Timezone

The top section of this window allows the timezone of the CueServer to be set. Use the *Region* menu first to select a general region from around the globe. Options exist for America, Asia, Australia, Canada, Europe, Pacific, US and others.

Once a region is chosen, use the *Location* menu to choose a specific timezone location within the region.

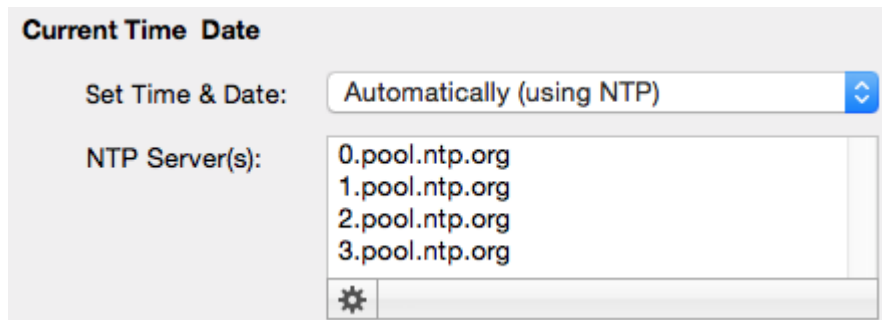
CueServer's timezone database is derived from the standard Linux distribution and includes over 400 distinct regional locations. See the [timezone listing](#) for a complete list of available timezones.

Current Time & Date


CueServer allows the Time and Date to be set manually or automatically. If the CueServer has a network connection where it can reach the Internet, or if the network has a network time server, then the Set Time & Date option can be set to *Automatically*.

Automatically Set Time & Date

When *Automatically* is chosen, a text field appears that allows one or more NTP time server addresses to be entered. Put one time server per line.

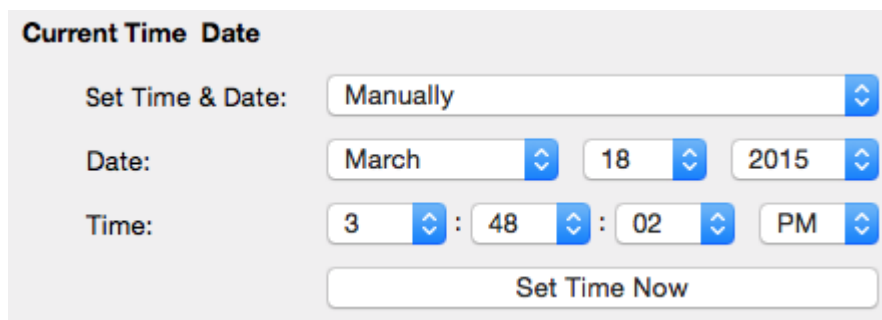


The screenshot shows the 'Current Time Date' settings panel. The 'Set Time & Date:' dropdown menu is set to 'Automatically (using NTP)'. Below it, the 'NTP Server(s):' text area contains a list of four NTP server addresses: 0.pool.ntp.org, 1.pool.ntp.org, 2.pool.ntp.org, and 3.pool.ntp.org. A gear icon is visible at the bottom left of the text area, indicating a menu for selecting default NTP servers.

The gear button () can be clicked to pop up a menu that includes several popular choices of publicly available Network Time (NTP) Servers. Choosing one of these options will automatically fill the server list with one of these sets of options.

Manually Set Time & Date

When *Manually* is chosen, the time and date can be set manually.



The screenshot shows the 'Current Time Date' settings panel with 'Manually' selected in the 'Set Time & Date:' dropdown. The 'Date:' section consists of three dropdown menus for month, day, and year, currently showing 'March', '18', and '2015'. The 'Time:' section consists of four dropdown menus for hour, minute, second, and AM/PM, currently showing '3', '48', '02', and 'PM'. A 'Set Time Now' button is located at the bottom of the panel.

Use the popup menus to choose the Time and Date. Before any of the menus are clicked, they show the current time of the computer. Once a menu is clicked, the time and date can be adjusted independently from the computer. Once the desired time is chosen, click on the **Set Time Now** button to set the clock in the CueServer.

Available Timezones

The following list shows the time zones available for CueServer.

Africa/Abidjan	Africa/Nairobi	America/Grenada	DumontDUrville
Africa/Accra	Africa/Ndjamena	America/Guadeloupe	Antarctica/Macquarie
Africa/Addis Ababa	Africa/Niamey	America/Guatemala	Antarctica/Mawson
Africa/Algiers	Africa/Nouakchott	America/Guyana	Antarctica/McMurdo
Africa/Asmara	Africa/Ouagadougou	America/Halifax	Antarctica/Palmer
Africa/Asmera	Africa/Porto-Novo	America/Havana	Antarctica/Rothera
Africa/Bamako	Africa/Sao Tome	America/Hermosillo	Antarctica/South Pole
Africa/Bangui	Africa/Timbuktu	America/Indianapolis	Antarctica/Syowa
Africa/Banjul	Africa/Tripoli	America/Inuvik	Antarctica/Troll
Africa/Bissau	Africa/Tunis	America/Jamaica	Antarctica/Vostok
Africa/Blantyre	Africa/Windhoek	America/Juneau	Asia/Aden
Africa/Brazzaville	America/Anchorage	America/La Paz	Asia/Almaty
Africa/Bujumbura	America/Anguilla	America/Lima	Asia/Amman
Africa/Cairo	America/Antigua	America/Los Angeles	Asia/Anadyr
Africa/Casablanca	America/Aruba	America/Louisville	Asia/Aqtou
Africa/Ceuta	America/Bahia	America/Martinique	Asia/Aqtobe
Africa/Conakry	America/Barbados	America/Mendoza	Asia/Ashgabat
Africa/Dakar	America/Belize	America/Mexico City	Asia/Ashkhabad
Africa/Dares Salaam	America/Bogota	America/Monterrey	Asia/Baghdad
Africa/Djibouti	America/Boise	America/Montreal	Asia/Bahrain
Africa/Douala	America/Buenos Aires	America/Nassau	Asia/Baku
Africa/El Aaiun	America/Cambridge Bay	America/New York	Asia/Bangkok
Africa/Freetown	America/Campo Grande	America/Nome	Asia/Beirut
Africa/Gaborone	America/Cancun	America/Panama	Asia/Bishkek
Africa/Harare	America/Caracas	America/Phoenix	Asia/Brunei
Africa/Johannesburg	America/Catamarca	America/Port-au-Prince	Asia/Calcutta
Africa/Juba	America/Cayenne	America/Puerto Rico	Asia/Choibalsan
Africa/Kampala	America/Cayman	America/Regina	Asia/Chongqing
Africa/Khartoum	America/Chicago	America/Santiago	Asia/Chungking
Africa/Kigali	America/Chihuahua	America/Santo Domingo	Asia/Colombo
Africa/Kinshasa	America/Coral Harbour	America/Sao Paulo	Asia/Dacca
Africa/Lagos	America/Cordoba	America/St Johns	Asia/Damascus
Africa/Libreville	America/Costa Rica	America/St Kitts	Asia/Dhaka
Africa/Lome	America/Cuiaba	America/St Lucia	Asia/Dili
Africa/Luanda	America/Curacao	America/St Thomas	Asia/Dubai
Africa/Lubumbashi	America/Danmarkshavn	America/St Vincent	Asia/Dushanbe
Africa/Lusaka	America/Denver	America/Tijuana	Asia/Gaza
Africa/Malabo	America/Detroit	America/Toronto	Asia/Harbin
Africa/Maputo	America/Dominica	America/Vancouver	Asia/Hebron
Africa/Maseru	America/Edmonton	America/Winnipeg	Asia/Ho Chi Minh
Africa/Mbabane	America/Eirunepe	Antarctica/Casey	Asia/Hong Kong
Africa/Mogadishu	America/El Salvador	Antarctica/Davis	Asia/Hovd
Africa/Monrovia	America/Fort Wayne	Antarctica/	Asia/Irkutsk


Asia/Istanbul	Asia/Ulan Bator	Saskatchewan	Europe/Chisinau
Asia/Jakarta	Asia/Urumqi	Canada/Eastern	Europe/Copenhagen
Asia/Jayapura	Asia/Ust-Nera	Canada/Mountain	Europe/Dublin
Asia/Jerusalem	Asia/Vientiane	Canada/Newfoundland	Europe/Gibraltar
Asia/Kabul	Asia/Vladivostok	Canada/Pacific	Europe/Guernsey
Asia/Kamchatka	Asia/Yakutsk	Canada/Saskatchewan	Europe/Helsinki
Asia/Karachi	Asia/Yekaterinburg	Canada/Yukon	Europe/Isle of Man
Asia/Kashgar	Asia/Yerevan	Chile/Continental	Europe/Istanbul
Asia/Kathmandu	Atlantic/Azores	Chile/EasterIsland	Europe/Jersey
Asia/Katmandu	Atlantic/Bermuda	Etc/GMT	Europe/Kaliningrad
Asia/Khandyga	Atlantic/Canary	Etc/GMT+1	Europe/Kiev
Asia/Kolkata	Atlantic/Cape Verde	Etc/GMT+2	Europe/Lisbon
Asia/Krasnoyarsk	Atlantic/Faeroe	Etc/GMT+3	Europe/Ljubljana
Asia/Kuala Lumpur	Atlantic/Faroe	Etc/GMT+4	Europe/London
Asia/Kuching	Atlantic/Jan Mayen	Etc/GMT+5	Europe/Luxembourg
Asia/Kuwait	Atlantic/Madeira	Etc/GMT+6	Europe/Madrid
Asia/Macao	Atlantic/Reykjavik	Etc/GMT+7	Europe/Malta
Asia/Macau	Atlantic/South Georgia	Etc/GMT+8	Europe/Mariehamn
Asia/Magadan	Atlantic/St Helena	Etc/GMT+9	Europe/Minsk
Asia/Makassar	Atlantic/Stanley	Etc/GMT+10	Europe/Monaco
Asia/Manila	Australia/ACT	Etc/GMT+11	Europe/Moscow
Asia/Muscat	Australia/Adelaide	Etc/GMT+12	Europe/Nicosia
Asia/Nicosia	Australia/Brisbane	Etc/GMT-1	Europe/Oslo
Asia/Novokuznetsk	Australia/Broken Hill	Etc/GMT-2	Europe/Paris
Asia/Novosibirsk	Australia/Canberra	Etc/GMT-3	Europe/Podgorica
Asia/Omsk	Australia/Currie	Etc/GMT-4	Europe/Prague
Asia/Oral	Australia/Darwin	Etc/GMT-5	Europe/Riga
Asia/Phnom Penh	Australia/Eucla	Etc/GMT-6	Europe/Rome
Asia/Pontianak	Australia/Hobart	Etc/GMT-7	Europe/Samara
Asia/Pyongyang	Australia/LHI	Etc/GMT-8	Europe/San Marino
Asia/Qatar	Australia/Lindeman	Etc/GMT-9	Europe/Sarajevo
Asia/Qyzylorda	Australia/Lord Howe	Etc/GMT-10	Europe/Simferopol
Asia/Rangoon	Australia/Melbourne	Etc/GMT-11	Europe/Skopje
Asia/Riyadh	Australia/NSW	Etc/GMT-12	Europe/Sofia
Asia/Saigon	Australia/North	Etc/Greenwich	Europe/Stockholm
Asia/Sakhalin	Australia/Perth	Etc/UCT	Europe/Tallinn
Asia/Samarkand	Australia/Queensland	Etc/UTC	Europe/Tirane
Asia/Seoul	Australia/South	Etc/Universal	Europe/Tiraspol
Asia/Shanghai	Australia/Sydney	Etc/Zulu	Europe/Uzhgorod
Asia/Singapore	Australia/Tasmania	Europe/Amsterdam	Europe/Vaduz
Asia/Taipei	Australia/Victoria	Europe/Andorra	Europe/Vatican
Asia/Tashkent	Australia/West	Europe/Athens	Europe/Vienna
Asia/Tbilisi	Australia/Yancowinna	Europe/Belfast	Europe/Vilnius
Asia/Tehran	Brazil/Acre	Europe/Belgrade	Europe/Volgograd
Asia/Tel Aviv	Brazil/DeNoronha	Europe/Berlin	Europe/Warsaw
Asia/Thimbu	Brazil/East	Europe/Bratislava	Europe/Zagreb
Asia/Thimphu	Brazil/West	Europe/Brussels	Europe/Zaporozhye
Asia/Tokyo	Canada/Atlantic	Europe/Bucharest	Europe/Zurich
Asia/Ujung Pandang	Canada/Central	Europe/Budapest	Indian/Antananarivo
Asia/Ulaanbaatar	Canada/East-	Europe/Busingen	Indian/Chagos

Indian/Christmas	Pacific/Efate	Pacific/Nauru	Pacific/Wake
Indian/Cocos	Pacific/Enderbury	Pacific/Niue	Pacific/Wallis
Indian/Comoro	Pacific/Fakaofu	Pacific/Norfolk	Pacific/Yap
Indian/Kerguelen	Pacific/Fiji	Pacific/Noumea	US/Alaska
Indian/Mahe	Pacific/Funafuti	Pacific/Pago Pago	US/Aleutian
Indian/Maldives	Pacific/Galapagos	Pacific/Palau	US/Arizona
Indian/Mauritius	Pacific/Gambier	Pacific/Pitcairn	US/Central
Indian/Mayotte	Pacific/Guadalcanal	Pacific/Pohnpei	US/East-Indiana
Indian/Reunion	Pacific/Guam	Pacific/Ponape	US/Eastern
Mexico/BajaNorte	Pacific/Honolulu	Pacific/Port Moresby	US/Hawaii
Mexico/BajaSur	Pacific/Johnston	Pacific/Rarotonga	US/Indiana-Starke
Mexico/General	Pacific/Kiritimati	Pacific/Saipan	US/Michigan
Pacific/Apia	Pacific/Kosrae	Pacific/Samoa	US/Mountain
Pacific/Auckland	Pacific/Kwajalein	Pacific/Tahiti	US/Pacific
Pacific/Chatham	Pacific/Majuro	Pacific/Tarawa	US/Pacific-New
Pacific/Chuuk	Pacific/Marquesas	Pacific/Tongatapu	US/Samoa
Pacific/Easter	Pacific/Midway	Pacific/Truk	

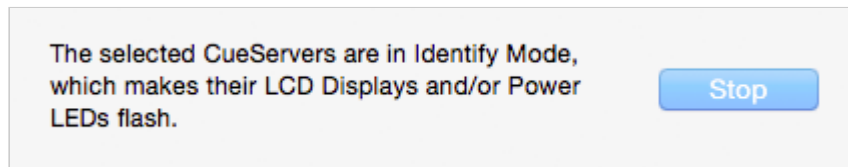
Identifying CueServers

When working with multiple CueServers, sometimes it may be useful to be able to positively identify which CueServer is which.

A CueServer's *Identify Mode* can be activated, which causes its LCD Display and Power LED to flash. This function makes it easy to match a CueServer listed in the Navigator Window with a physical device on the network.

To activate the Identify Mode, select a CueServer in the list, then choose the **Identify...** item in the CueServer Menu, or click on the Identify toolbar icon ().


The CueServer will begin flashing, and the following window will appear:



To exit the Identify Mode, click on the **Stop** button.

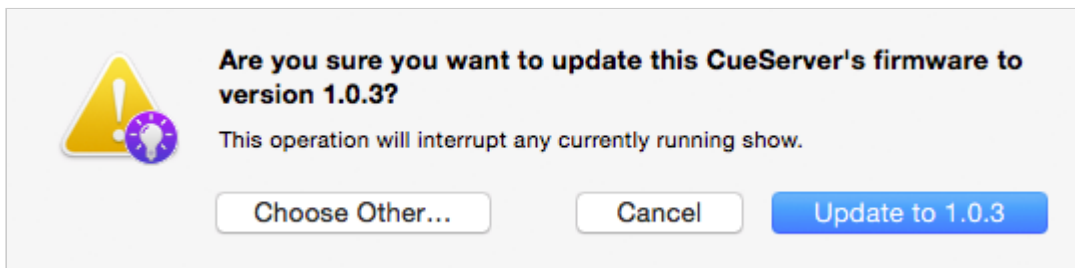
Updating Firmware

When new features or bug fixes become available for CueServer 2, a new version of CueServer Studio will be released. With each software release, CueServer Studio will check to make sure that the CueServer devices have the most up-to-date software version.

If a CueServer's firmware is out of date, it will appear in the Navigator Window with a warning icon () in front of the firmware version number.

CueServer Studio can update the firmware in connected CueServers by choosing the **Update Firmware...** menu item in the CueServer menu.

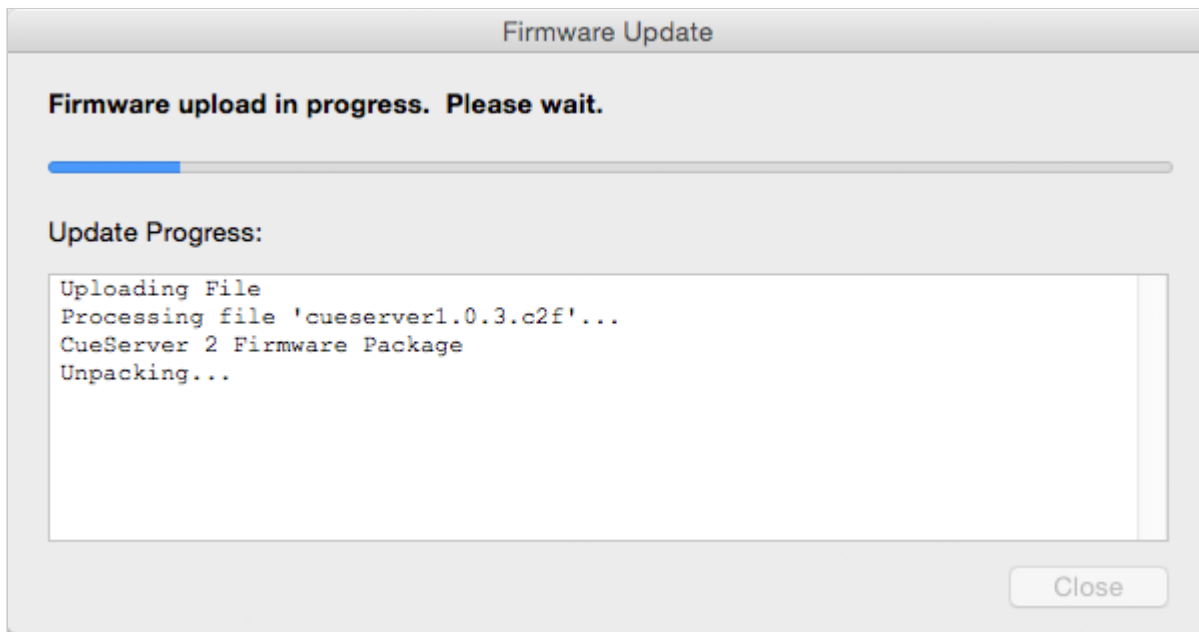
The following dialog window appears:



In this example, CueServer Studio is recommending that the device be upgraded to version 1.0.3. This firmware image is embedded in the CueServer Studio application itself. Simply click on the **Update** button to perform the update.

If you want to update the CueServer to a different version of firmware, click on the **Choose Other...** button. A file chooser window will appear that will allow a different firmware version to be loaded. CueServer firmware files have the file extension **.c2f**.

When the firmware update process is running, a progress window appears:

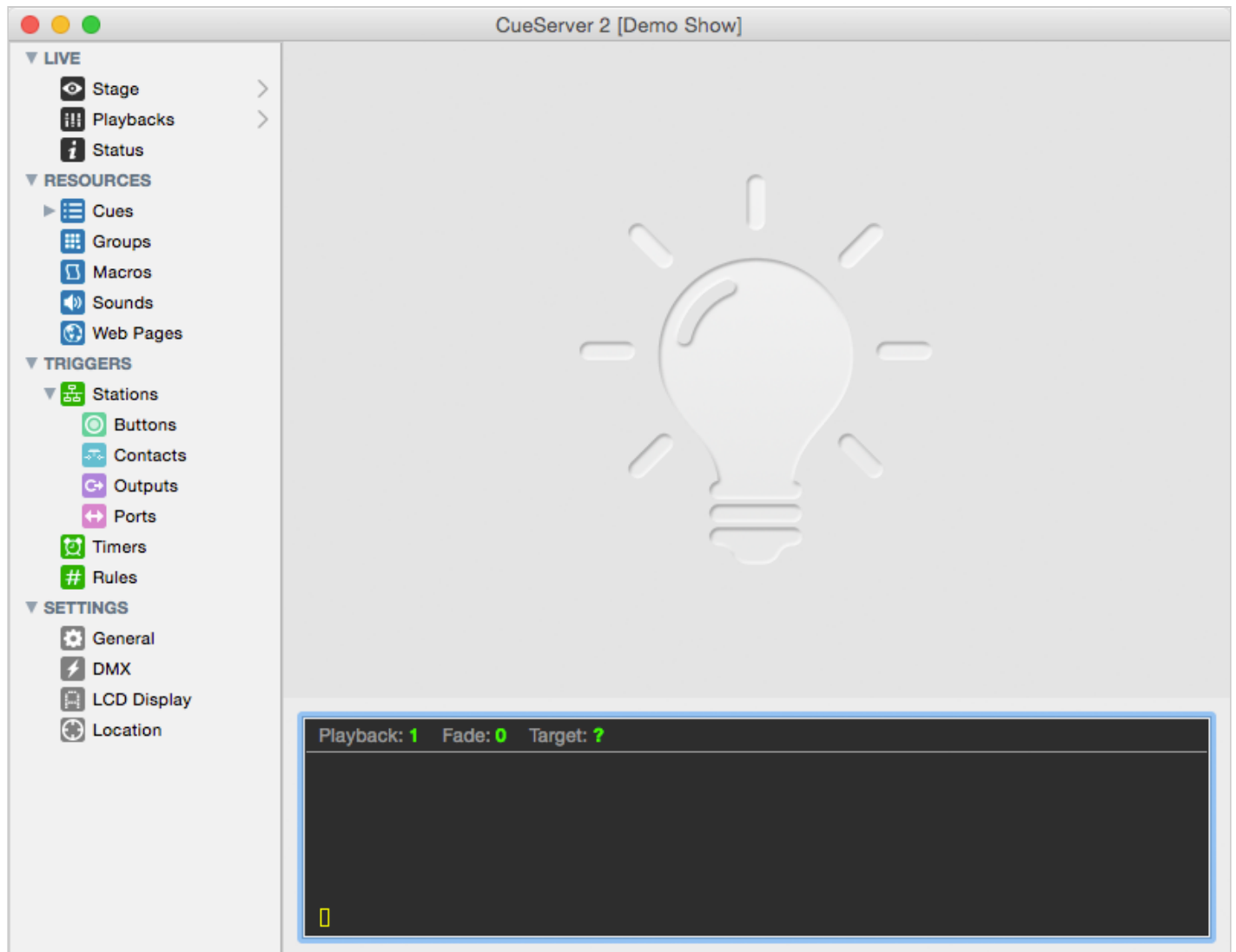


The progress of the update is shown in the window and on the LCD screen of the CueServer. When the update is complete, the CueServer will reboot and the **Done** button can be clicked to dismiss the window.

Editor Window

Overview

The *Editor Window* is the primary window used to interact with, program and configure CueServer.



Use the Editor Window to view the “live” operation, edit resources and triggers, and set various configuration properties of a CueServer show.

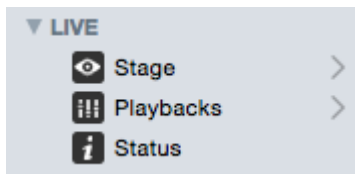
The panel on the left of the window contains numerous views into the CueServer, such as Stage, Cues, and Location. The following manual sections describe the details of each of these CueServer editor views:

- [Live](#) – live views of CueServer operation
 - [Stage](#) – for viewing DMX channels
 - [Playbacks](#) – for viewing playback faders
 - [Status](#) – for viewing the front-panel of the CueServer
- [Resources](#) – various content types for CueServer projects
 - [Cues](#) – scenes and timeline based streams
 - Groups – definitions of groups of channels
 - Macros – user-defined scripts
 - Sounds – audio clips
 - Web Pages – custom web pages for the project
- Triggers – definitions for incoming system events
 - Stations – setup for stations, buttons, contact-closures and more
 - Timers – setup for timers
 - Rules – a global list of rules
- Settings – system preferences
 - General – general purpose settings
 - DMX – DMX related settings
 - LCD Display – customization of the LCD display
 - Location – location settings for astronomical time

The panel at the bottom of the window is a live command line that allows the user to directly enter CueScript commands to cause the CueServer to perform operations. Note that this command line is only visible if you are editing the active show file in an “online” CueServer.

Live

The *Live* section of the navigator contains views that show the Stage, Playback Operation, and System Status of the CueServer. Each of these views show dynamic screens that are updating “live” as the CueServer is performing it’s operations.



The following sections describe these views in more detail:

- [Stage](#) – for viewing DMX channels
- [Playbacks](#) – for viewing playback faders
- [Status](#) – for viewing the front-panel of the CueServer

Stage

Overview

The *Stage View* shows the output channels of the CueServer. This view is arranged in a grid of channels. Controls within the window change the visible layer of the channel grid between the device's Output, one of the Playbacks, or the Input. Various colors indicate the source of each channel value and/or the state of the channel.

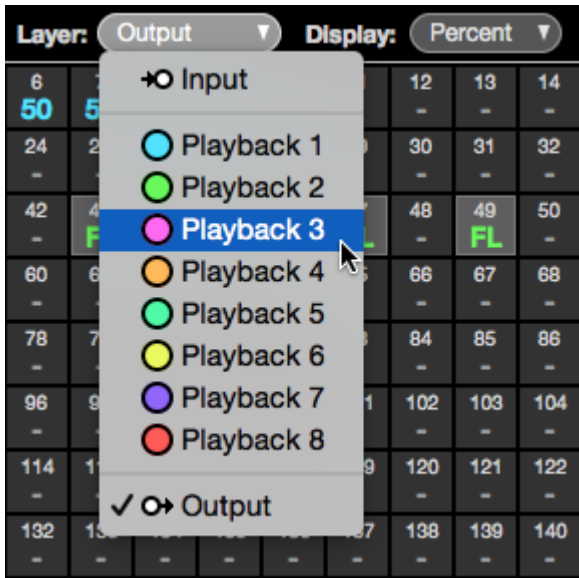
View:		All Universes		Layer:		Output		Display:		Percent							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
50	50	50	50	50	50	50	50	50	50	-	-	-	-	-	-	-	-
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
-	-	-	-	FL	-	FL	-	FL	-	FL	-	FL	-	-	-	-	-
55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

While cues are running and/or channels are fading, they will update live within this view. The channels are colored to match the display color for each Playback Fader. In the example above, the Blue channels are coming from Playback 1 and the Green channels are coming from Playback 2.

The area behind the odd-numbered channels from 41 through 49 are shaded in Gray to indicate that these channels are currently selected.

Choosing the View Layer

Use the **Layer** popup menu to choose which layer of the DMX composition is being shown:

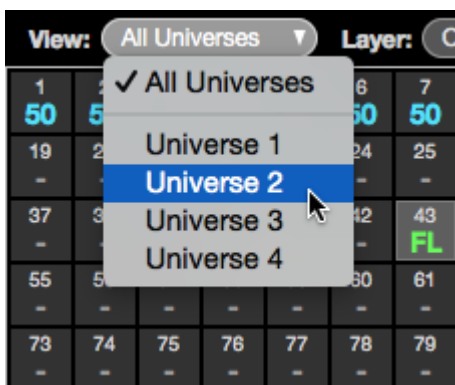


The view options are:

- **Input** – This view shows any DMX values that are being input into the device.
- **Playback** – This view shows DMX values that are present in a specific Playback Fader. The colored circle shows the color of the channels for the given Playback Fader.
- **Output** – This view shows the final composite DMX values that are being output from the device.

Choosing the Visible Universes

Use the **View** popup menu to choose which universe(s) are being shown:



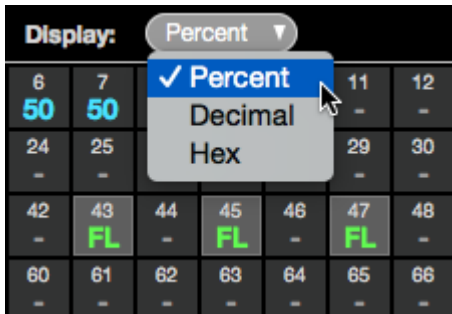
The view options are:

- **All Universes** – This view shows all Universes in one continuous table.

- **Universe n** – This view focuses the display on only the chosen Universe.
-

Choosing a Display Mode

Use the **Display** popup menu to choose how the values in the channel grid are shown:




The display options are:

- **Percent** – This mode shows channel levels as a percentage. Values range from 0 to 99, and then FL (meaning Full, or 100%).
- **Decimal** – This mode shows channel levels in decimal format. Values range from 0 to 255.
- **Hexadecimal** – This mode shows channel levels in hexadecimal format. Values range from 00 to FF.

Playbacks

Overview




The *Playbacks View* shows the current state and properties of the Playback Fader layers of the CueServer. This view is arranged in a stack of Playbacks. Each Playback has three panes, the left-hand pane shows what is currently loaded in the Playback, the center pane shows what's coming up next, and the right-hand pane shows additional properties for the Playback. While cues are running and/or channels are fading, bar graphs appear that show the progress of the cues, fades, streams, etc.

Playback 1		
Cue: 30 "Breakbeat" Stream Time: 00:00:04.78 	Next Cue: 99 "Dim Blue"	Output Normal
Playback 2 ACTIVE		
Cue: 3 "Blue" Fade: 1.5s  Follow: 4.5s 	Next Cue: 1 "Red" Fade: 5, Follow: 8	Submaster: 75% 
Playback 3		
Active Channels	No Next Cue	Fader Stopped Mode: Scale

In the example above, Playback 1 is currently playing back Cue 30, which is a streaming cue called "Breakbeat". It is currently 4.78 seconds into the stream. The next cue in Playback 1 is Cue 99, which is called "Dim Blue". Playback 2 is the active playback, it is currently fading into Cue 3 "Blue". The fade has 1.5 seconds remaining, and a follow timer is running with 4.5 seconds remaining. The next cue in Playback 2 is Cue 1 "Red", and that cue will have a Fade Time of 5 seconds, and a Follow Timer of 8 seconds. Also, Playback 2's submaster has been lowered to 75%. Finally, Playback 3 has manually set "active" DMX channels in it and no next cue. Playback 2 is "stopped", meaning that fade and follow timing is disabled, and it's layer mode is set to "Scale".

The Current Pane (Left Side)

The pane on the left-hand side of each Playback shows what is currently loaded in the Playback.

- **Empty** – Shown if the playback has no active channels. An empty Playback has no effect on the DMX output.
 - **Active Channels** – Shown when the Playback has active channels (not originating from a Cue).
 - **Cue (n)** – Shown when the Playback is loaded with the channels from a particular Cue.
 - **Cue (n) + Changes** – Shown when the Playback was loaded with a Cue, and then manual channel values were changed.
 - **Fade (time)** – Shown when the Playback is actively fading channels. A green progress bar () shows the fade time remaining.
 - **Follow (time)** – Shown when the Playback is counting down to an auto-follow event. A blue progress bar () shows the follow time remaining.
 - **Stream (time)** – Shown when a Streaming Cue is being played back. An orange progress bar () shows the stream time remaining.
-

The Next Pane (Center)


The panel in the center of each Playback shows what is queued to be “next”.

- **Next Cue (n)** – Shown if the Playback has a *next cue* that will execute upon a Go command or auto-follow.
 - **Fade (time)** – Shown to indicate the fade time of the *next cue*.
 - **Follow (time)** – Shown to indicate the follow time of the *next cue*.
 - **Link (n)** – Shown to indicate the link of the *next cue*.
 - **No Next Cue** – Shown if the Playback does not have a *next cue*.
-

The Properties Pane (Right Side)

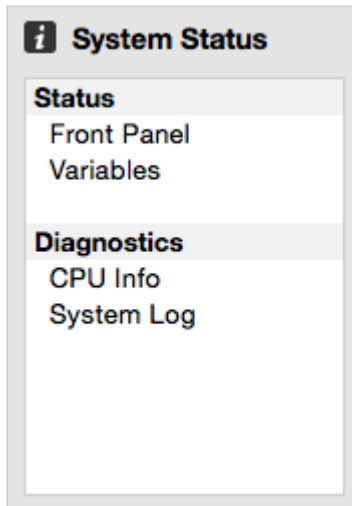
The panel on the right-hand side of each Playback shows additional properties for the Playback Fader.

- **Output Normal** – Shown if the Playback has no overridden properties. All values are normal.
- **Stack (name)** – Shown if the Playback has a cue stack assigned to it.
- **Fader Stopped** – Shown in Red color when the Playback is stopped. A stopped Playback has its timing overridden, meaning that setting channel levels or executing cues always appear immediately (they do not fade), the follow timer does not run, and streaming cues are paused.
- **Channels Parked** – Shown in Red color when channels in the Playback are parked. Parked channels retain their current values and cannot be modified by executing cues or by using the Channel, At, Release, or Clear commands. Parked channels must either be Unparked, or the CueServer can be Reset.

- **Submaster (*level*)** – Shown when the Playback’s submaster level is not at 100%. A pink progress bar () shows the submaster percentage.
- **Mode (*mode*)** – Shown if the Playback’s combine mode is set to anything other than the default “Merge” mode. Options include **Override**, **Scale**, and **Pin**.


Status

The *Status* page provides several views that show live status of various CueServer subsystems.



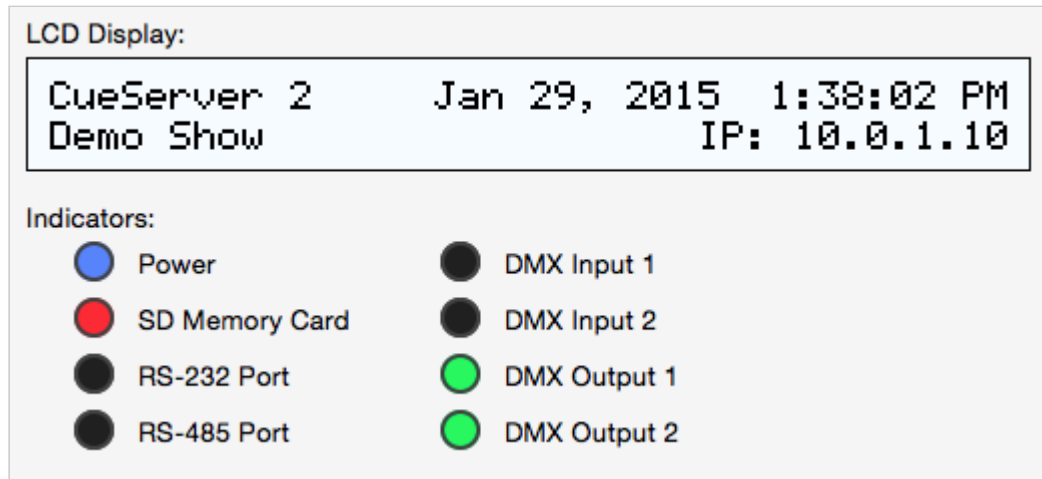
The following status views are available:

- [Front Panel](#) – a live view of the front-panel of the CueServer.
- [Variables](#) – a live listing of user-defined variables.
- [CPU Info](#) – a live view of the hardware status.
- [System Log](#) – the current system log.

Note that if any of the status views has an important condition that needs to be shown to the user, the caution icon () will appear to the right of the corresponding line in the list of status views.

Front Panel

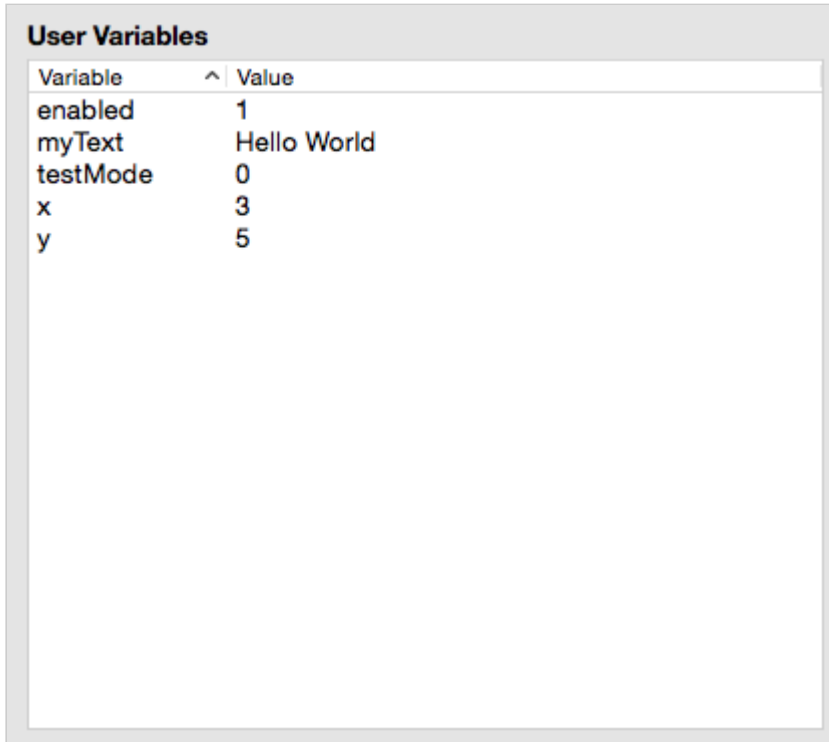
The *Front Panel View* shows the current state of the physical CueServer. The CueServer's LCD display and LED indicators are visible in this view.



As the LCD display and/or LED indicators on the physical CueServer changes, they are updated live on this view.

Variables

The *Variables View* shows any currently defined user variables.



The screenshot shows a window titled "User Variables" containing a table with two columns: "Variable" and "Value". The table lists five variables: "enabled" with value "1", "myText" with value "Hello World", "testMode" with value "0", "x" with value "3", and "y" with value "5".

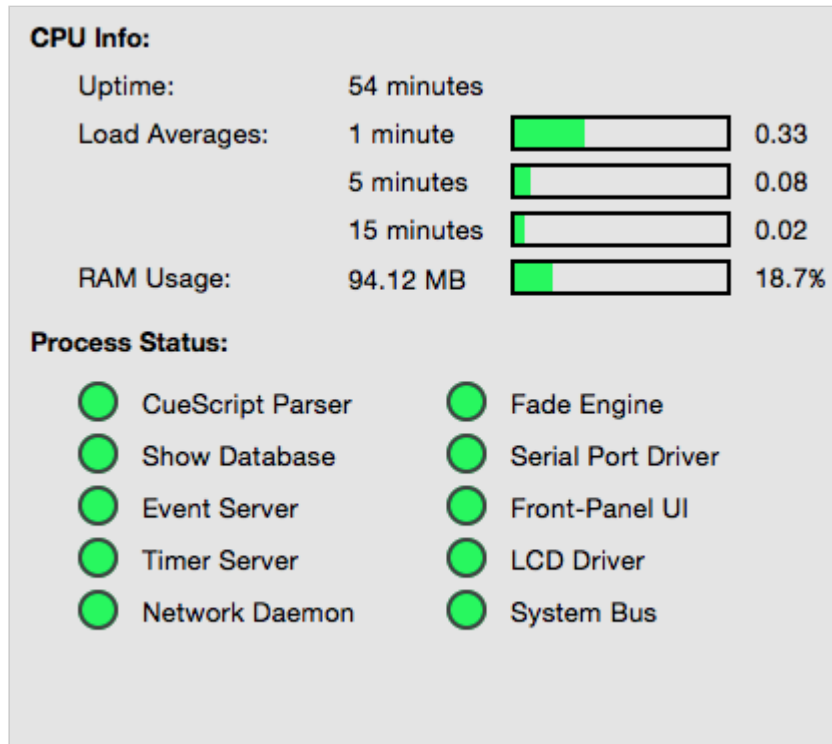
Variable	Value
enabled	1
myText	Hello World
testMode	0
x	3
y	5

Whenever any CueScript statements are used to define or update the value of a user variable, this view will show those values “live”.

For more information about using variables in scripts, see the [Variables](#) section of the CueScript Language chapter.


CPU Info

The *CPU Info View* shows the status of the CueServer hardware.



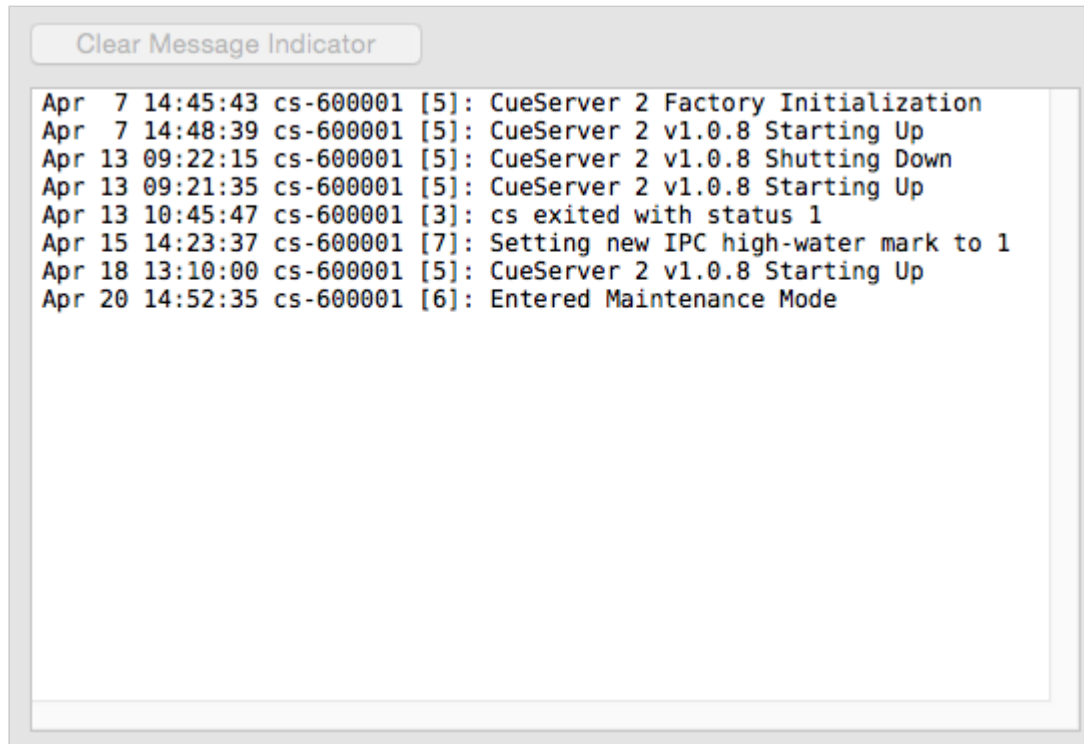
The following information is displayed:

- **Uptime** – shows the number of days, hours, and minutes since the CueServer was powered-on.
- **Load Averages** – shows the CPU load, averaged over the last 1, 5 and 15 minutes.
- **RAM Usage** – shows how much system RAM is being used. Note that this is not the memory on the SD Card.
- **Process Status** – shows the running state of each of CueServer's internal processes. Green means that the service is running, Red means that an error has occurred.

Note that if any of the processes in the CPU Info view require attention, a warning icon () will appear next to the CPU Info line in the status list.

System Log

The *System Log* shows internal system messages posted by CueServer's operating system and related software.




Most messages in the System Log are only useful for diagnosing problems, however other informational messages can appear in the System Log as well.

For instance, the System Log shows each time the system is rebooted.

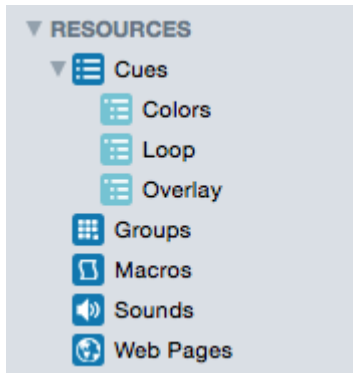
Also, user-defined messages can be added to the System Log by using the [Log](#) CueScript command.

When a message is added to the System Log that indicates a serious condition, the Power LED will begin to blink. This is called the "Message Indicator". It means that the System Log contains an important message. To clear this indication, click on the "Clear Message Indicator" button.

If a new important message is currently showing, a warning icon () will appear next to the System Log line in the status list.

Resources

The *Resources* section of the navigator contains views that edit Cues, Groups, Macros, Sounds and Web Pages in the CueServer project.



The following sections describe these views in more detail:

- [Cues](#) – scenes and timeline based streams
- Groups – definitions of groups of channels
- Macros – user-defined scripts
- Sounds – audio clips
- Web Pages – custom web pages for the project

Cues

Overview

The *Cues* editor shows the Cue List, and allows for the creation, capture, modification and removal of cues from the project.

The screenshot displays the Cue List editor interface. At the top, there is a 'Cue List' header with a menu icon. Below it is a table with the following data:

Number	Name	Timing	Link	Action
1	Solid Red	5 (8)		
2	Solid Green	5 (8)		
3	Solid Blue	5 (8)	1	
10	44th Street	00:00:09.85	85	
20	Bossa Lounger	00:00:07.47	85	
30	Breakbeat	00:00:09.75	85	

Below the table, there are controls for adding (+), removing (-), and configuring (gear icon) cues, with a total of 14 cues shown. The interface then transitions to a detailed editor for the selected cue (Number 2).

The editor has three tabs: 'General', 'Contents', and 'Capture'. The 'General' tab is active, showing the following properties:

- Number: 2
- Name: Solid Green
- Fade: 5 (with a menu icon)
- Follow: 8
- Link: Next Cue

Below the properties is a 'Rules' section with a single rule:

```
WHENEVER This Cue Is Executed
THEN Perform Script B1OFF
```

At the bottom right of the editor, there are 'Revert' and 'Apply' buttons.

The Cues Editor is divided into several sections. The top panel shows the list of Cues. Click on a cue to have it appear in the lower panel. Once selected, a Cue's properties, rules, and contents can be viewed or modified.

For details about different aspects of creating and modifying cues, see the following topics:

- [Cue Types](#) – discusses the differences between normal and streaming cues.
- [Adding Cues](#) – to learn how to add cues to a project.
- [Cue Properties](#) – for a description of the various properties of a cue.
- [Cue Contents](#) – to see how the contents of a cue are displayed.
- [Cue Rules](#) – for how to add automation rules to a cue.

Cue Types

There are two cue types available to CueServer.

Normal Cues

A “normal” cue is similar to the type of cue used on traditional lighting consoles. A cue of this type stores a single scene (or part of a scene).

In CueServer, a normal cue stores an array of DMX channel values, which will be recalled when the cue is executed. The cue may contain *all*, *some*, or *none* of the available DMX channels in the system. Normal cues have extra parameters such as fade and follow times, an optional linked cue, and automation rules.

Generally speaking, when playing back (executing) normal cues, the output of the CueServer will crossfade to a new scene. Again, a normal cue may only include *some* of the DMX channels, so only part of a scene may be affected by playing back a normal cue.


Streaming Cues

A “streaming” cue is a different type of cue that stores DMX channels and their changes over a period of time.

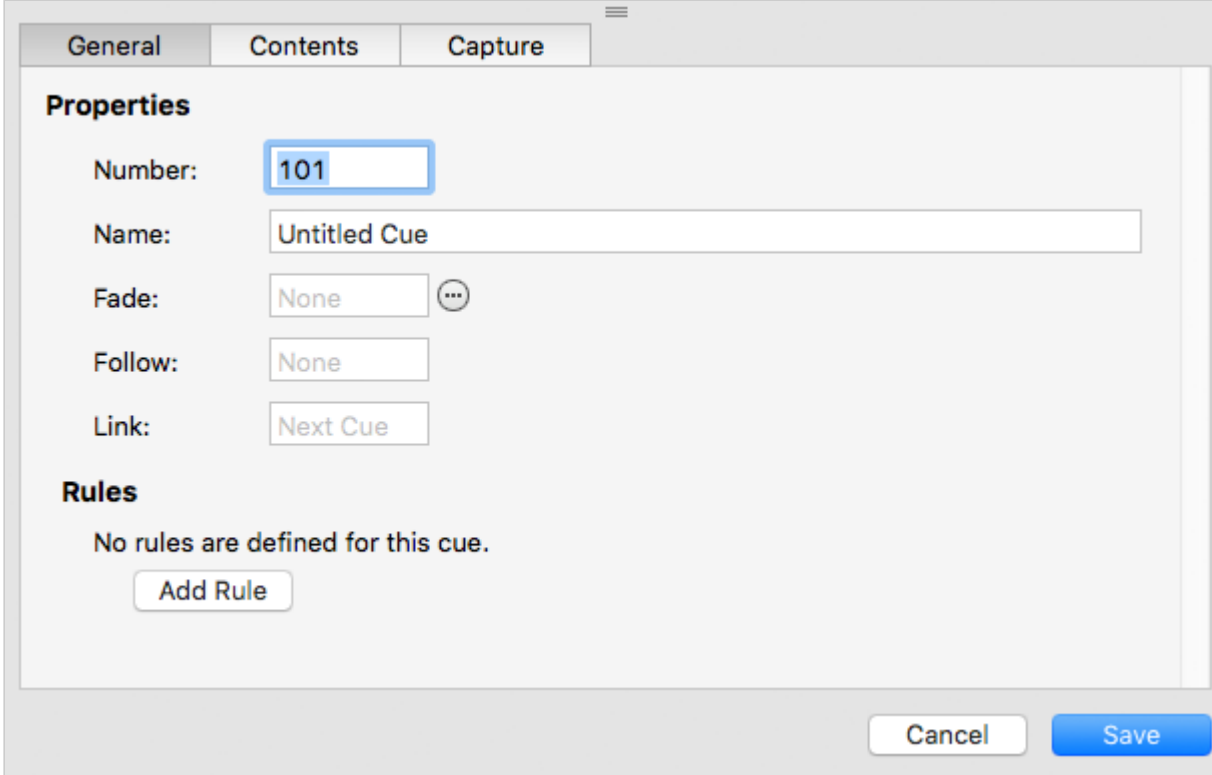
An analogy can be made between a streaming cue in CueServer, and a “tape recorder” for audio. When a streaming cue is captured in CueServer, every change to a DMX channel during the capture is saved. Then, when the streaming cue is played back, the changes occur in real-time just the same way that it was recorded.

Streaming cues have extra parameters such as playback mode, follow time, an optional linked cue, and automation rules.

Adding Cues

To add a new cue to the cue list, click the plus button () at the lower-left corner of the cue list. Or, choose the **New Cue...** item from the File menu.

A new empty cue will appear in the window:



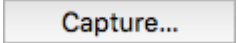
The screenshot shows a dialog box with three tabs: "General", "Contents", and "Capture". The "General" tab is selected. The dialog is titled "Properties" and contains the following fields:

- Number: 101
- Name: Untitled Cue
- Fade: None (with a menu icon)
- Follow: None
- Link: Next Cue

Below these fields is a section titled "Rules" with the text "No rules are defined for this cue." and an "Add Rule" button. At the bottom right of the dialog are "Cancel" and "Save" buttons.

CueServer Studio will automatically create the cue with the next available cue number already chosen. This number can be changed before saving the cue to use a different cue number.

The cue's name, fade and follow times, link and rules can all be set by clicking into these fields.

The newly created cue does not have any channels recorded into it. To add content to this cue, click on the Capture button (). See the [Cue Contents](#) section for information about how to capture scenes and/or streams into cues.

Cue Properties

Each cue has a number of properties that may be edited:

Number

Number:

By convention, every cue in a cue list has a number. Valid cue numbers range from 0 through 999999. Optionally, up to two digits can be used after a decimal point (for example, Cue 1.23).

Once a cue is recorded, it's number can be changed by entering a new number into this field.

Name

Name:

A cue may be given a descriptive name.

Fade (normal cues only)

Fade: ⋮

A normal cue has a fade time (expressed in seconds) that is used to specify how quickly the cue's channels will crossfade from their previous values to the ones recorded in the cue. Fade times from 0 (no fade) to 86400 seconds (24 hours) may be specified.

Fade times can be split into separate times for channels fading up and channels fading down, and delays can be introduced to the up-fading and down-fading channels.


	Delay	Fade
Rising Channels:	0	3.5
Falling Channels:	1.5	7

Cancel OK

Fade details window.

Click on the More button (⋮) next to the fade field to display a window to enter advanced fade time parameters.

Mode (streaming cues only)

Mode: 

A streaming cue can be set to play back with one of four modes:

- **None** – When the stream finishes, playback stops and the last channel values remain active.
- **Loop** – When the stream reaches its last frame, it will seamlessly loop back to its beginning.
- **Follow** – When the stream finishes, the next cue automatically follows.
- **Release** – When the stream finishes, channels in the stream are released.

Follow

Follow:

Cues have an *auto follow timer* that begins when the cue is executed, as specified by this field (in seconds). When the timer expires, the playback fader automatically executes a *Go* to advance to the next cue in the cue list (or whatever cue the current cue is linked to).

This field can be left blank to allow cues to advance in regular numerical order.

Cue Contents

Each cue may contain DMX channels, or streaming data, or may be empty.

The contents of the cue is displayed in the **Contents** section of the Cue Editor panel.

One of three types of content will be displayed:

Normal DMX Channels

1024 Channels															Capture...	
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
0	70	0	0	70	0	0	70	0	0	70	0	0	70	0	0	
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
70	0	0	70	0	0	70	0	0	70	0	0	70	0	0	70	

Normal cues contain a single snapshot of DMX channels. The cue might have been recorded with all DMX channels in it, or only a subset of available channels (selected channels).

When a cue with DMX channels is executed, those channel values will appear in the active playback fader. If the cue has a fade time of zero (no fade time), the channel values will appear immediately. If the cue has a fade time, then the channels will crossfade from their previous values to the ones in the cue.

To capture a DMX snapshot, see the section [Capturing DMX Snapshots](#).

Streaming Cue Data

1024 Channels of Streaming Data (18.30 seconds, 300.4KB)		Capture...	

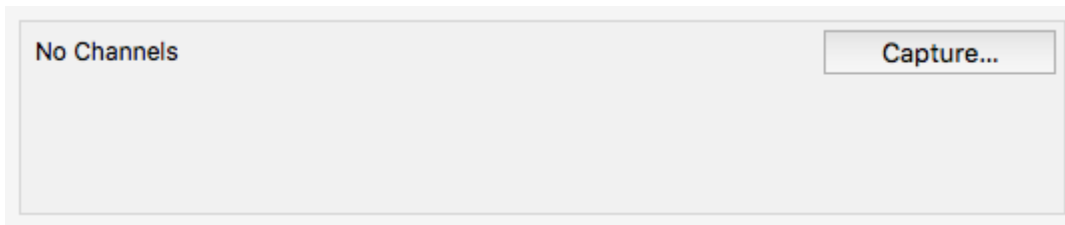
A streaming cue contains a recording of DMX data over a period of time.

When a cue with streaming DMX data is executed, the recorded channel data plays back over time matching the changes that were occurring when it was recorded.

Recording and playing back streaming cues is similar to using a tape recorder to store and then play back an audio recording. Streaming cues do a similar thing with DMX lighting data.

To capture a DMX stream, see the section [Capturing DMX Streams](#).

Empty Cues

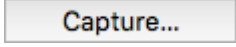


A cue can be recorded with **no** DMX channels. This type of cue does not directly affect any DMX channels when it is executed.

An empty cue will still observe its follow timing and it will also evaluate any rules in the cue, but it will not change any DMX channel values.

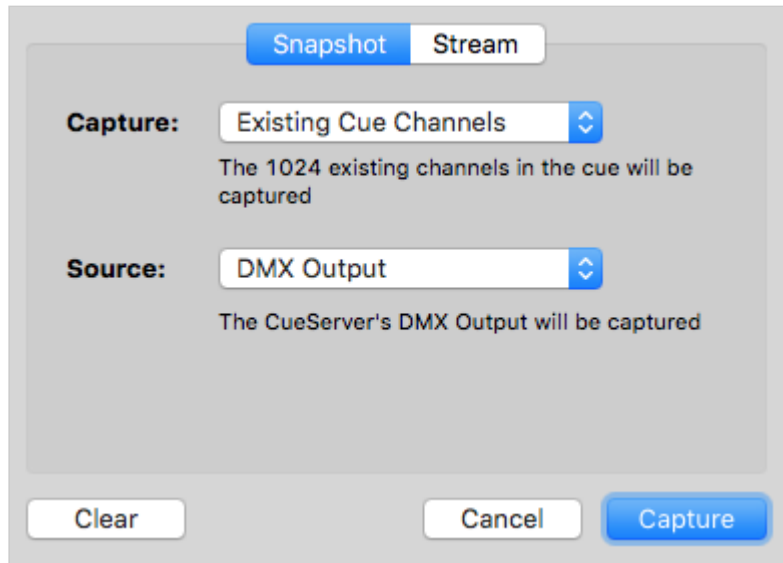
To clear a cue's contents (creating an empty cue), see the section [Clearing Cue Contents](#).

Capturing Content into a Cue

To record, change or clear the contents of a cue, click on the Capture button ().

Capturing DMX Snapshots

The **Snapshot** tab of the Capture window is used to capture a single snapshot of DMX channels into a cue:



This window has pop-up menus for choosing what DMX channels will be captured into the cue and from what source the channels will be captured.

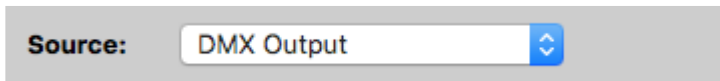
Capture Menu



This menu has several options to specify which channels will be recorded into the cue:

- **All Channels** – Every channel in the system will be recorded. If the CueServer is configured with two universes of DMX then 1,024 channels will be captured.
- **Active Channels** – Only channels that have non-zero values will be recorded into the cue.
- **Selected Channels** – Only the currently selected channels will be recorded into the cue. The selected channels are the ones previously selected using the [Channel](#) and [Group](#) commands.
- **Existing Cue Channels** – Only the channels that are currently recorded in the cue will be re-recorded. If the cue previously contained channels 101 through 199, then those channels are the only ones that will be re-recorded.

Source Menu



This menu has several options to specify the source of the DMX channel values that will be recorded into the cue:

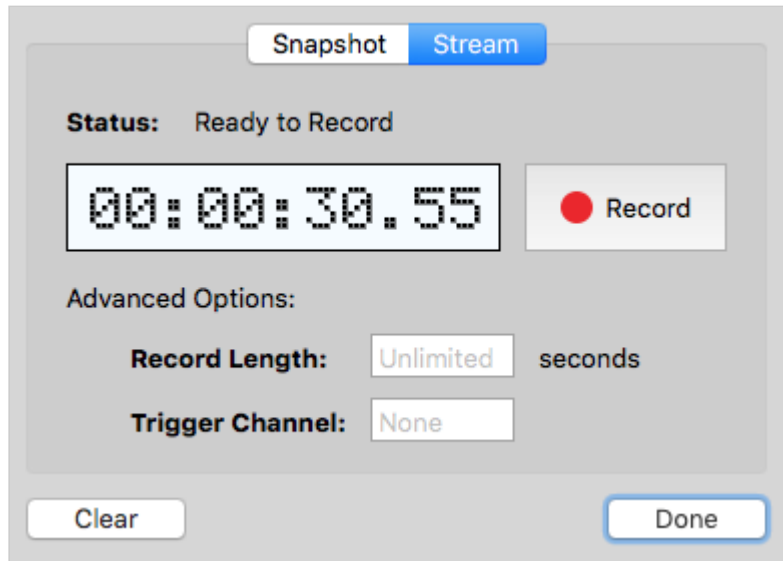
- **DMX Input** – The DMX channel values being input into the CueServer will be recorded. None of the values in the playbacks or being output will be recorded.
- **Playback *n*** – The DMX channel values in Playback *n* will be recorded. Neither the DMX input or output will be recorded.
- **DMX Output** – The DMX channel values being output from the CueServer will be recorded. This is the default option.

Capturing Channel Values

Once the appropriate options have been selected, click on the **Capture** button to record the current DMX values into the cue. The Capture window will close and the cue will be updated to show the newly captured channels.

Capturing DMX Streams

The **Stream** tab of the Capture window is used to capture a stream of changing DMX channel values into the cue.



This window has controls for starting/stopping the stream recording and additional advanced options for controlling the length or external triggering of the stream recording.

Recording Controls



At the top of the window, there is a time display readout and a **Record** button.

The time display shows the current duration of stream recording that is in the cue. For new cues, this will show `00:00:00.00`. For cues with existing streaming data, the cue's current duration will appear in this display.

To start recording, press the **Record** button. The button will change to **Stop** and the time display will begin counting. Recording of DMX channel values will continue until the **Stop** button is pressed.

Record Length Option

Record Length: seconds

This field can be used to limit the length of the recording to a specific number of seconds. Any number of seconds may be entered down to 1/100th second precision.

When a Record Length has been specified, the stream recording will automatically stop after the length has been reached.

If this field is empty, recording will continue until the **Stop** button is pressed.

Trigger Channel Option

Trigger Channel:

This field can be used to specify a channel number that the CueServer should watch to automatically start and stop the stream recording. When the input channel rises above zero, the recording will start. Then, when the input channel falls back to zero, the recording will stop.

The typical use for this feature is to allow the external console that is sending DMX data to be able to start and stop the CueServer's recording by raising and lowering this "trigger channel". Any channel can be chosen, but it is typical to use a channel that is not being used by a dimmer or fixture.

When a trigger channel is specified, press the **Record** button to begin waiting for the trigger channel to rise above zero. As long as the trigger channel is being received as zero, the time display will wait to start recording:

Status: Waiting for Trigger Channel

As soon as the external console raises the trigger channel above zero, the recording will begin automatically. Then, when the trigger channel falls back to zero, recording will stop.

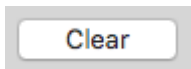


CueServer very precisely monitors the value of the trigger channel and will begin recording a stream on the very first DMX frame that has a non-zero trigger channel value. The recording continues until the trigger channel becomes zero again. The last frame recorded is the frame received *just before* a frame arrives with a zero value trigger channel.

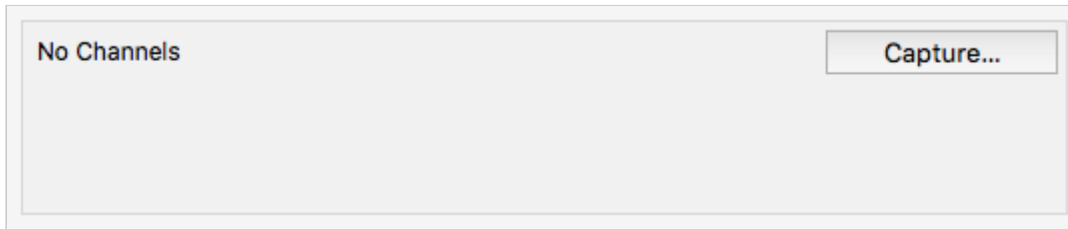
Clearing Cue Contents

Sometimes it may be desirable to create a cue that does not have any DMX channel values recorded in it. This is called an empty cue. An empty cue can be useful to provide additional timing steps in a list of cues, or that may have automation rules without affecting DMX channels, etc.

A cue with channel values can be cleared by clicking on the **Capture** button, and then clicking on **Clear** in the lower left corner of the capture window.



After **Clear** is clicked, the capture window will close and the cue will show that it no longer contains any channel values:



Cue Rules


Rules can be added to a cue to allow it to automate certain tasks when it is executed.


The rules for a cue might look like this:

Rules

WHENEVER This Cue Is Executed
AND The Time Is After 2 : 00 : 00 PM
THEN Perform Script Indicator 1 On


WHENEVER This Cue Is Executed
AND The Time Is Before 2 : 00 : 00 PM
THEN Perform Script Indicator 2 On

To add a rule to a cue, click on the “plus” button ().

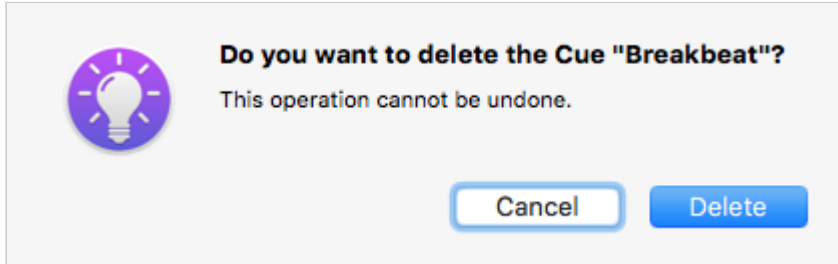
Then, click on the various “bubble” buttons () in the rule to build an event, conditions and action that the rule will execute.

For more information about building rules, see the [Rules](#) topic.

Deleting Cues

To remove a cue from the cue list, click the minus button () at the lower-left corner of the cue list.

A confirmation dialog will appear:



After confirmation of the delete operation, the cue will be removed from the cue list.

 You can also use the **Delete** or **Backspace** keys on your keyboard. To avoid the confirmation dialog, you can hold the **Option** or **Alt** key.

Hardware

Overview

This chapter describes the models of CueServer 2 available along with the various physical features, differences between models, specifications and explanation of indicators and displays.

For a description of the available models of CueServer 2, see these sections:

- [Models](#)
 - [CS-900 CueServer 2 Pro](#)
 - [CS-940 CueServer 2 DIN](#)

For explanations and specifications for the physical features of CueServer 2, see these sections:

- [Power Input](#)
- [Ethernet Ports](#)
- [DMX Ports](#)
- [Audio Ports](#)
- [USB Ports](#)
- [LCD Display](#)
- [Function Buttons](#)
- [Contact Closures](#)
- [Digital Outputs](#)
- [Serial Ports](#)
- [Memory Card](#)
- [Reset Button](#)

Models

There are currently two CueServer 2 models available:

CS-900 CueServer 2 Pro



The CueServer 2 Pro is housed in an enclosure with removable brackets suitable for either 19" rack-mounting or desktop use. It features dual LAN ports, four field-replaceable DMX module slots and customizable front-panel button caps. See the [CS-900 CueServer 2 Pro](#) section for more details for this model.

CS-940 CueServer 2 DIN



The CueServer 2 DIN is housed in an enclosure with replaceable side brackets suitable for DIN-Rail or surface mounting. It features a single LAN port, two DMX input ports and two DMX output ports. See the [CS-940 CueServer 2 DIN](#) section for more details for this model.

CS-900 CueServer 2 Pro

The CueServer 2 Pro (CS-900) is housed in a sturdy 1U rack-mount enclosure with removable brackets.



CueServer 2 Pro features dual LAN ports for splitting Ethernet-based lighting and management data onto separate networks if desired.

CueServer 2 Pro boasts an innovative modular DMX port system. Four bi-directional DMX ports on the back of the unit are user-configurable with any of seven available port modules. These interchangeable modules allow CueServer 2 Pro to be customized for different installation environments eliminating the need for external adaptors.

The front-panel of CueServer 2 Pro has eight customizable function buttons. Each button has fully controllable RGB backlighting and field-replaceable legends for project personalization. A navigation keypad is used to operate the onboard LCD menu for basic system settings, show selection, and macro execution.

Features

- Completely self-contained lighting playback, architectural processor, and DMX fade engine
- Seamless handling of Cue Lists, Presets, and Streams
- Control of up to 32 universes of DMX or 32 independent playback timelines
- Dynamic patching of up to 16,384 channels to 128 separate sACN, Art-Net, or KiNET universes
- Dual Ethernet ports for separate configurable lighting data and management networks
- Flexible module-based bi-directional DMX ports for custom jack configurations
- Front-panel configurable function buttons with RGB backlighting and field-replaceable legends
- Creation of lighting scenes directly or capture from external sources
- Powerful CueScript scripting language
- Real-Time clock with astronomical and calendar events
- Built-in web server for hosting custom interactive web pages
- Multi-show storage on removable microSD memory card
- System integration via Ethernet, Serial, Digital I/O, and Audio
- Compatible with CueStation buttons and CueTouch touchscreens
- Easy interfacing with Crestron, AMX, Vantage, Control 4, Medialon, Savant and other automation systems

- Native programming environment for both Mac and Windows
- 1U rack-mounted enclosure with removable brackets

CS-920 CueServer 2 Mini

The CueServer 2 Mini (CS-920) is the smallest of the CueServer 2 models and is housed in a rugged anodized aluminum enclosure suitable for desktop use or panel, DIN, or truss mounting using optional bracket kits.

The CueServer 2 Mini can output shows utilizing up to 16,384 channels and features two built-in modular DMX slots that are user-configurable with any of seven available port modules. These interchangeable modules allow CueServer 2 Mini to be customized for different installation environments eliminating the need for external adapters.



CueServer 2 Mini also features two user-definable function buttons with RGB indicators, two contact closure inputs, two low-voltage digital outputs, a serial port, and stereo audio output.

Features

- Completely self-contained lighting playback, architectural processor, and DMX fade engine
- Seamless handling of Cue Lists, Presets, and Streams
- Control of up to 32 universes of DMX or 32 independent playback timelines
- Dynamic patching of up to 16,384 channels to 128 separate sACN, Art-Net, or KiNET universes
- Flexible module-based bi-directional DMX ports for custom jack configurations
- Front-panel configurable function buttons with RGB indicator LEDs
- Creation of lighting scenes directly or capture from external sources
- Powerful CueScript scripting language
- Real-Time clock with astronomical and calendar events
- Built-in web server for hosting custom interactive web pages
- Multi-show storage on removable microSD memory card
- System integration via Ethernet, Serial, Digital I/O, and Audio
- Compatible with CueStation buttons and CueTouch touchscreens
- Easy interfacing with Crestron, AMX, Vantage, Control 4, Medialon, Savant and other automation systems

- Native programming environment for both Mac and Windows
- Small anodized aluminum enclosure with optional brackets

CS-940 CueServer 2 DIN

The CueServer 2 DIN (CS-940) is housed in an enclosure suitable for DIN rail, surface, or panel mounting. The DIN rail brackets accommodate standard 35mm rail. Mounting flanges are included for surface or panel mounting.

Connections for power, DMX, contact closures, and digital outputs are made using removable terminal blocks across the top edge of the unit. Ethernet, USB, and stereo audio are connected along the bottom edge.



CueServer 2 DIN's front panel has eight customizable function buttons. Each button has a fully controllable RGB indicator. A navigation joystick is used to operate the onboard LCD menu for basic system settings, show selection, and macro execution.

Features

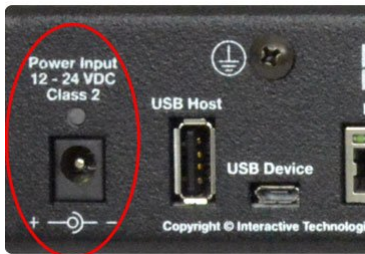
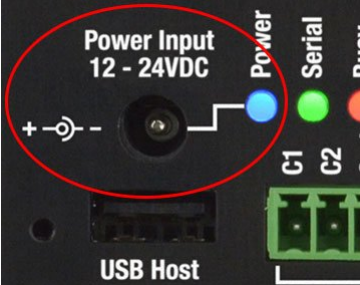
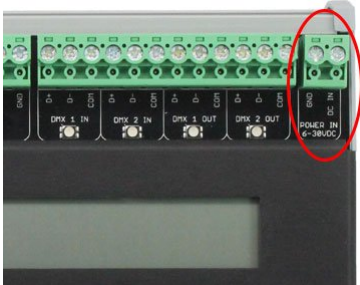
- Completely self-contained lighting playback, architectural processor, and DMX fade engine
- Seamless handling of Cue Lists, Presets, and Streams
- Control of up to 32 universes of DMX or 32 independent playback timelines
- Dynamic patching of up to 16,384 channels to 128 separate sACN, Art-Net, or KiNET universes
- Front-panel configurable function buttons with RGB indicator LEDs
- Creation of lighting scenes directly or capture from external sources
- Powerful CueScript scripting language
- Real-Time clock with astronomical and calendar events
- Built-in web server for hosting custom interactive web pages
- Multi-show storage on removable microSD memory card
- System integration via Ethernet, Serial, Digital I/O, and Audio
- Compatible with CueStation buttons and CueTouch touchscreens
- Easy interfacing with Crestron, AMX, Vantage, Control 4, Medialon, Savant and other automation systems
- Native programming environment for both Mac and Windows
- Standard DIN-Rail mounting or surface/panel mounting

Power Input



All models of CueServer 2 can be powered by a 12 to 24 VDC Class 2 input.

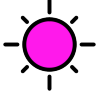
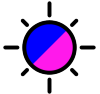

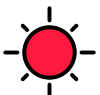

Although the power requirements are similar for the various models, their connectors and current requirements are different. The rack-mounted CS-900 and the miniature CS-920 both have a standard DC power input jack. The surface-mounted CS-940 uses screw terminals that are suitable for hardwire connections to DC power.

Specifications

	CS-900	CS-920	CS-940
Power Input	12-24 VDC	12-24 VDC	12-24 VDC
Minimum Power Supply Wattage	9 Watts	7 Watts	8 Watts
Connector	2.1mm DC Power Jack	2.1mm DC Power Jack	Screw Terminals
			
Pinout	Center = DC Input (V+) Barrel = Ground	Center = DC Input (V+) Barrel = Ground	1 = DC Input (V+) 2 = Ground

Indicators

Color & Pattern	Description
 Solid Blue	Power in on, all systems normal
 Slowly Flashing Blue	Device is in the process of starting up

 Slowly Flashing Magenta	Device is in Bootloader Mode (contact Technical Support)
 Slowly Alternating Blue/Magenta	The System Log has a new message
 Quickly Alternating Blue/Magenta	The System Log has an <i>important</i> new message
 Slowly Flashing Red	Device has shut down (must power cycle to reboot)
 Off	Device has no power

Ethernet Ports


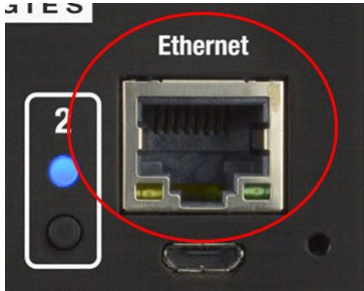
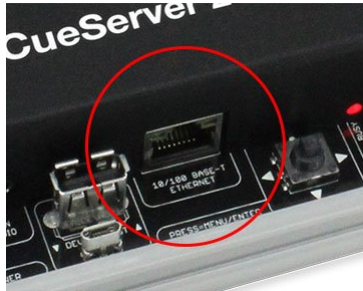
CueServer 2 is an Ethernet-based product. An Ethernet connection to a local network is required to program CueServer using the [CueServer Studio 2](#) software. Additionally, Ethernet is required if a DMX-over-Ethernet protocol (such as [sACN](#), [Art-Net](#), or [KiNET](#)) is going to be used to input or output DMX over Ethernet, or to connect to button stations, use the CuePad iOS application or to remotely manage the device. Only if the CueServer has already been programmed and no Ethernet protocols are needed to run the show can the CueServer be used without a network connection.

The rack-mounted CS-900 has two Ethernet ports, the miniature CS-920 and surface-mounted CS-940 only have a single Ethernet port.






Units with two Ethernet ports can be configured in one of two modes. The first mode is to provide only a single network on both ports using a built-in Ethernet Switch. The second mode is to separate the two ports into different LANs, with management data on LAN A and lighting data on LAN B. In this second mode, each port will have separate IP addresses.

See the [Ethernet Protocols](#) section for a description of the protocols supported by CueServer.

Specifications

	CS-900	CS-920	CS-940
Ethernet Ports	2	1	1
Network Mode(s)	Single Network with Built-In Switch or Two Separate LANs	Single Network	Single Network
Factory Default Settings	Single Network DHCP Enabled Fallback IP 10.0.1.234	DHCP Enabled Fallback IP 10.0.1.234	DHCP Enabled Fallback IP 10.0.1.234
			

Indicators

Left LED	Description
 Off	No Link, bad cable, or no connection on opposite end
 Solid Green	Ethernet link is established
Right LED	Description
 Off	No Link, bad cable, or no connection on opposite end
 Solid Amber	No data activity
 Flashing Amber	Data activity

Ethernet Protocols

CueServer 2 supports several *Ethernet Protocols* for the transmission and reception of DMX lighting control data, remote control of the CueServer, synchronization of network time, web services and more.

See the following sections for additional information about each Ethernet Protocol as implemented by CueServer:

- [sACN \(Streaming ACN\) Protocol](#)
- [Art-Net Protocol](#)
- [KiNET Protocol](#)
- [CueScript Protocol](#)
- [CueStation Protocol](#)
- [HTTP Protocol](#)
- [DHCP Protocol](#)
- [NTP Protocol](#)

sACN (Streaming ACN) Protocol

sACN (or Streaming ACN) is a preferred method of sending and receiving DMX-over-Ethernet to and/or from a CueServer.

The following table lists the general specifications for the CueServer implementation of sACN:

- Compliant with ANSI Standard E1.31-2009 (sACN)
- May send and/or receive up to 128 universes of sACN simultaneously
- May send and/or receive sACN packets with an arbitrary universe number between 1 and 63999
- sACN is sent at a maximum rate of 40Hz for each universe when channel values are changing
- sACN is sent at a minimum rate of 1Hz for each universe when channel values are static
- Supports the sending of user-defined priority levels for each universe
- Received sACN is merged with hardwired DMX designated for the same universe
- Ignores incoming data marked as Preview Data
- Performs immediate stream termination when a Stream Terminated packet is received
- Maintains proper sequence number transmission separately for each output universe
- Sends the universe's name as the sACN source name for each universe
- Ignores packets with start code 0xDD (used for slot-by-slot priority)
- Each universe times out after 2.5 seconds when no packets for that universe are not received
- CueServer does not receive its own sACN output

For more information about sACN, please visit the **ESTA Technical Standards Program** at tsp.esta.org.

Art-Net Protocol

Art-Net is a method of sending and receiving DMX-over-Ethernet to and/or from a CueServer.

Art-Net is owned and copyright by Artistic Licence Holdings Ltd. Artistic Licence has published the specification and made it available for anyone to use on a royalty-free basis.

The following table lists the general specifications for the CueServer implementation of Art-Net:

- Compliant with the Art-Net 3 Specification by Artistic License
- May send and/or receive up to 128 universes of Art-Net simultaneously
- May send and/or receive Art-Net packets with an arbitrary port address (network/sub-net/universe) from 0:0:0 thru 127:F:F
- Art-Net is sent at a maximum rate of 40Hz for each universe when channel values are changing
- Art-Net is sent at a minimum rate of 1Hz for each universe when channel values are static
- Art-Net may be sent to the limited broadcast address, directed broadcast address or unicast to a specific IP address
- Received Art-Net is merged with hardwired DMX designated for the same universe
- Maintains proper sequence number transmission separately for each output universe
- Each universe times out after 6 seconds when no packets for that universe are not received
- CueServer's implementation of Art-Net does not yet support "automatic" IP configuration via the ArtPoll method
- CueServer does not receive its own Art-Net output

For more information about Art-Net, please visit the **Art-Net Home Page** at art-net.org.uk.

KiNET Protocol

KiNET is an alternate method of sending DMX lighting control values to from a CueServer to lighting fixtures and/or power supplies that support the proprietary Philips/Color Kinetics KiNET protocol.

The following table lists the general specifications for the CueServer implementation of KiNET:

- Compliant with the v1 and v2 versions of the KiNET protocol
- May send and/or receive up to 128 universes of KiNET simultaneously
- A universe outputting KiNET v1 may be sent to any arbitrary IP address
- A universe outputting KiNET v2 may specify a port number and DMX range
- KiNET is sent at a maximum rate of 40Hz for each universe when channel values are changing
- KiNET is sent at a minimum rate of 1Hz for each universe when channel values are static

For more information about KiNET, please visit **Color Kinetics** at colorkinetics.com.

Interactive Technologies is a KiNET licensee by Philips/ColorKinetics.

CueScript Protocol

CueScript Protocol is a method of sending CueScript commands to a CueServer over Ethernet.

CueServer listens for incoming UDP packets on port 52737 that contain a valid CueScript command string.

CueScript packets may be sent to CueServer by:

- Unicast to the CueServer's IP Address
- Multicast to the CueServer group address 239.255.204.2

The payload of the packet can be any valid CueScript command, such as:

- Cue 1 Go
- Q1G
- Macro 7
- M7
- Channel 1>10 At FL
- C1>10AFL
- Button 1 On; Wait 3.5; Button 1 Off
- If ('x' = 1) Then Playback 7 Clear

CueStation Protocol

CueStation Protocol is the method of communication between a CueServer and the CueStation Hub.

The following table lists the general specifications for the CueServer implementation of CueStation protocol:

- Can communicate with one or more CueStation Hubs
- Supports HUB IDs from 1..254 for unique identification of multiple hubs on a single network
- Uses the CueStation Multicast group 239.255.204.3
- Uses multicast traffic only for configuration-free setup

HTTP Protocol

Hypertext Transfer Protocol (HTTP) is a network protocol for “hypermedia information systems”. HTTP is the foundation of data communication for the World Wide Web.

CueServer uses HTTP for a variety of purposes.

CueServer uses its embedded HTTP web server to allow custom web pages to be served by the active project file. This allows a project to be set up in CueServer that includes its own customized web based content. A project in CueServer can host a “home page” that acts as a landing page for the project, and nearly any other pages, images, documents, and other content as necessary. CueServer also has the ability to interact with this web content, making the web pages interact live with the running CueServer lighting control and automation.

CueServer uses HTTP to communicate with CueServer Studio. All of the transactions between CueServer Studio and the CueServer device are occurring over HTTP. This allows CueServer Studio to be able to remotely control a CueServer using nothing other than TCP Port 80 access over the Internet.

CueServer uses HTTP to communicate with various companion “apps”, such as CuePad and our touchscreen options.

CueServer exposes an open Application Programming Interface (API) through HTTP for software developers to use to interact with the device. Custom applications can be written in nearly any computer language that communicates with CueServer via HTTP.

For more information about HTTP, please visit the * Hypertext Transfer Protocol Wikipedia Page* at wikipedia.org/wiki/Hypertext_Transfer_Protocol.

DHCP Protocol

Dynamic Host Configuration Protocol (DHCP) is a network protocol used to automatically configure devices on the network. With DHCP, devices request IP addresses and networking parameters automatically from a DHCP server, reducing the need for a network administrator or a user to configure these settings manually.

CueServer can optionally use DHCP to automatically set its network parameters (such as IP Address, Subnet, Gateway, etc.) without requiring the user to adjust these settings manually.

By default from the factory, CueServer has DHCP turned on, which means that it will attempt to find a DHCP server and automatically configure itself as the CueServer is powered on. CueServer can be configured to have DHCP turned off, which would allow manually assigned (static) network parameters to be used.

Some CueServer models may be configured to have more than one LAN connection. On models configured this way, DHCP may be used separately on each LAN.

For more information about DHCP, please visit the **Dynamic Host Configuration Protocol Wikipedia Page** at wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol.

NTP Protocol

Network Time Protocol (NTP) is a networking protocol for clock synchronization between computer systems over Ethernet.

CueServer uses NTP to keep its clock as accurate as possible without requiring the user to adjust the time manually.

NTP is intended to synchronize computers to within a few milliseconds of Coordinated Universal Time (UTC). It is an algorithm to select accurate time servers and is designed to mitigate the effects of variable network latency. NTP can usually maintain time to within tens of milliseconds over the public Internet, and can achieve better than one millisecond accuracy in local area networks under ideal conditions. Asymmetric routes and network congestion can cause errors of 100 ms or more.

Because a CueServer that intends to use NTP to synchronize its time must be able to reach a NTP Server via Ethernet, this function can only work if the CueServer is on a network that has access to the Internet, or the facility must have an NTP Server on its local network.



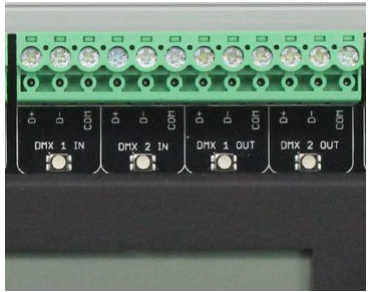
For more information about NTP, please visit the **Network Time Protocol Wikipedia Page** at wikipedia.org/wiki/Network_Time_Protocol.

DMX Ports


In addition to being able to transmit and receive DMX-over-Ethernet, CueServer also has built-in DMX ports for hard-wired DMX connections to fixtures, dimmers, consoles and virtually any other DMX compatible devices.












The rack-mounted CS-900 has four replaceable DMX module slots, and the miniature CS-920 has two replaceable DMX module slots, each of which can accept any of seven available DMX modules for input or output of DMX. The surface-mounted CS-940 has two DMX input ports and two DMX output ports that are available on the unit's pluggable terminal block strips.

Specifications

	CS-900	CS-920	CS-940
DMX Ports	4	2	4
Bi-Directional Ports	4	2	–
Fixed Input Ports	–	–	2
Fixed Output Ports	–	–	2
Replacable DMX Modules	Yes	Yes	No
			

Indicators

Description	CS-900 or CS-920	CS-940
Port Disabled	 Off	 Off

DMX Output (Active)	 Solid Green	 Solid Green
DMX Input (Active)	 Solid Blue	 Solid Green
DMX Input (No Input)	 Solid Magenta	 Off
Bad Universe	 Slowly Flashing Yellow	 Slowly Flashing Yellow
Bad Port Direction	n/a	 Quickly Flashing Red
Error Condition	 Solid Red	 Solid Red

DMX Modules

The CS-900 CueServer 2 Pro uses a unique field-replaceable DMX module system for allowing the DMX ports to be customized for each individual project's needs.

The CS-900 ships with four blank plates covering the module slots. Optional modules can be purchased and installed into each of these slots. Using CueServer Studio, each slot can be configured as a DMX input or output.

The following list shows the available modules:

Module	Description	DMX Pinout
	MOD-X5F 5-Pin Female XLR	1 – Common 2 – Data - 3 – Data + 4 – NC 5 – NC
	MOD-X5M 5-Pin Male XLR	1 – Common 2 – Data - 3 – Data + 4 – NC 5 – NC
	MOD-X3F 3-Pin Female XLR	1 – Common 2 – Data - 3 – Data +

	<p>MOD-X5M 3-Pin Male XLR</p>	<p>1 – Common 2 – Data - 3 – Data +</p>
	<p>MOD-RJ45 Ethercon RJ45</p>	<p>1 – Data + (White/Orange) 2 – Data – (Orange) 3 – NC (White/Green) 4 – NC (Blue) 5 – NC (White/Blue) 6 – NC (Green) 7 – Common (White/Brown) 8 – Common (Brown) * Colors use T-568B Standard</p>
	<p>MOD-TB-ST Screw Terminals</p>	<p>1 – Data - 2 – Data + 3 – Common</p>
	<p>MOD-TB-IDC IDC Terminals</p>	<p>1 – Data - 2 – Data + 3 – Common</p>




Audio Ports

CueServer 2 has built-in stereo audio.

The stereo output is able to play sound effects, music and other audio clips in response to CueScript commands triggered by the active show. At this time, the stereo input is not implemented in software and is reserved for future use.

See the section [Supported Audio File Formats](#) for a listing of what types of audio files that CueServer 2 can play.

Specifications

	CS-900	CS-920	CS-940
Audio Input	1	–	1
Audio Output	1	1	1
Connector	1/8" Phono Plug	1/8" Phono Plug	1/8" Phono Plug
			

Supported Audio File Formats

CueServer 2 supports the playback of the following common popular audio file formats:

File Extension	Description
<code>.aif .aifc .aiff .aiffc</code>	Audio Interchange File Format (Apple)
<code>.au .snd</code>	Unix Audio File (Sun, NeXT, UNIX)
<code>.flac</code>	Free Lossless Audio (Open-Source)
<code>.mp2 .mp3</code>	MPEG Audio File (MP3)
<code>.ogg .vorbis</code>	Ogg Vorbis Audio File (Open-Source)
<code>.wav</code>	Waveform Audio File (See WAV Sample Formats)

CueServer 2 also contains partial support for the following formats (use cautiously as these have not been tested):

```
8svx amb amr-nb amr-wb anb avr awb cdda cdr cvs cvsd cvu dat dvms gsm gsrt hcom htk ima
ircam lpc lpc10 maud nist prc sds sln smp sndfile sndr sndt sou sox sph txw vms voc vox
wavpcm wv wve xa
```

CueServer 2 **DOES NOT** support the following audio file formats:

File Extension	Description
<code>.m4a .m4p</code>	iTunes Advanced Audio Coding (AAC)

WAV Sample Formats

The following sample formats are supported by CueServer 2's WAV audio playback:

- S8
- U8
- S16_LE
- S16_BE
- U16_LE
- U16_BE
- S24_LE
- S24_BE
- U24_LE
- U24_BE
- S32_LE
- S32_BE
- U32_LE
- U32_BE
- FLOAT_LE
- FLOAT_BE
- FLOAT64_LE
- FLOAT64_BE
- IEC958_SUBFRAME_LE
- IEC958_SUBFRAME_BE
- MU_LAW
- A_LAW
- IMA_ADPCM
- MPEG
- GSM
- SPECIAL
- S24_3LE
- S24_3BE
- U24_3LE
- U24_3BE
- S20_3LE
- S20_3BE
- U20_3LE

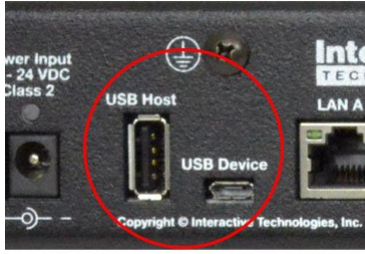


- U20_3BE
- S18_3LE
- S18_3BE
- U18_3LE

USB Ports

CueServer 2 has both USB Host and USB Device ports.

At this time, the USB Host port is only used as an alternative way to apply firmware updates to the device. The USB Device port is not used and is reserved for future use.

Specifications

	CS-900	CS-920	CS-940
USB Host Ports	1 (Type A)	1 (Type A)	1 (Type A)
USB Device Ports	1 (Micro B)	1 (Micro B)	1 (Micro B)
			

LCD Display

Some CueServer 2 models have a front-panel LCD Display with navigation buttons. This display is used to see the overall operational status of the device and can be used to adjust a small subset of settings and perform basic diagnostics.

The LCD Display features include:

- System Status (Default)
- Main Menu
 - Execute Macros
 - Change the Active Show
 - Adjust the Clock
 - Adjust Network Settings
 - View DMX Channels
 - Display System Information
 - Perform a Self Test

See the [LCD Display Modes](#) section for a complete description of the LCD functions available.

Specifications

	CS-900	CS-920	CS-940
LCD Display	2-Line x 40 Character Black on White	n/a	2-Line x 40 Character Black on White
Navigation	5-Way Backlit Button Pad	n/a	5-Way Micro Joystick
			

LCD Display Modes

The LCD Display has two modes of operation, the Status Display and the Menu Display:

Status Display

By default, while the CueServer is running normally, the Status Display will be visible. This display typically shows the device's name, the current time, and the device's IP address. Although this information is typically shown, the Status Display is entirely customizable by the active show, so the Status Display might appear differently.

```
CueServer 2      Jan 1, 2016  9:42:00 AM
                  IP: 10.0.1.5
```

For a list of available Status Display customizations, see the [LCD Status Options](#) section.

Menu Display

By pressing the **Enter** button of the navigation switch, the LCD will change to its Menu Display. This display will show a list of **Main Menu** choices. Use the **Up** and **Down** buttons to scroll through the list of available choices. To activate a choice, press **Enter** or **Right**. To return to the previous display, press the **Left** button.

```
Main > [1] Macros
        [2] Shows
        [3] Clock Settings
        [4] Network Settings
        [5] DMX Menu
        [6] System Information
        [7] Self Test
```

See the [LCD Menu Functions](#) section for details about each available menu function.

LCD Status Options

The Status Display of the LCD is divided into four quadrants, the top-left, top-right, bottom-left, and bottom-right. Each of these four quadrants can be customized to show a different piece of information about the status of the device or show.

The default Status Display for CueServer 2 has the following quadrant layout:

Device Name	Long Date + 12-Hour Time
Show Name	IP Address

The following table shows the available options and how they appear on the LCD display:

Status Type	Example	Description
Device Name	CueServer 2	The assigned name of the device.
Show Name	My First Show	The name of the active show.
Show Directory	/shows/My First Show/	The file system directory of the active show.
IP Address	IP: 10.0.1.234	The IP Address of the device. If the device is configured with multiple network LANs, the display will alternate between both networks' IP Addresses. If the Ethernet jack is unplugged, a Ø symbol will appear in front of the address.
Timecode	TC: 00:00:00:00	The current timecode within the system.
IO Status	[C:--*--*--][O:-----*-]	The current status of the built-in contact closures and digital outputs. A * symbol indicates that a contact is closed or an output is active.
CPU Load	CPU: 7%	The current 5-minute average CPU load percentage.
Long Date + 12-Hour Time	Feb 15, 2017 10:34:56 PM	The long date and 12-hour time combined.
Short Date + 12-Hour Time	2/15/17 10:34:56 PM	The short date and 12-hour time combined.
Long Date + 24-Hour Time	Feb 15, 2017 22:34:56	The long date and 24-hour time combined.
Short Date + 24-Hour Time	2/15/17 22:34:56	The short date and 24-hour time combined.
Long Date	Feb 15, 2017	The long date.

Short Date	2/15/17	The short date.
12-Hour Time	10:34:56 PM	The 12-hour time.
24-Hour Time	22:34:56	The 24-hour time.

These settings can be changed by choosing the *Settings > LCD Display* from within CueServer Studio.

LCD Menu Functions

The Main Menu of the LCD appears when the **Enter** or **Right** button of the navigation switch are pressed. Use the **Up** and **Down** buttons to select a menu option, then press **Enter** or **Right** to choose the menu option. To exit a menu option, press **Left**.

The following table shows the **Main Menu** options:

Menu Option	Function
Macros	Displays a list of the available Macros in the current show that have the “Show in LCD Display” option selected. Any macro shown in the list can be activated.
Shows	Displays a list of the available Shows on the memory card. This menu also shows and allows the currently active show to be changed.
Clock Settings	Displays and allows adjustments to the current time and date, including time zone.
Network Settings	Displays and allows adjustments to the current network settings, including DHCP.
DMX Menu	Displays a list of DMX related functions.
System Information	Displays information about the hardware and firmware revisions, including license information.
Self Test	Runs the built-in hardware self test routine. Two confirmation screens will appear before allowing the self test to run. See the Self Test section of this manual for instructions. Warning: Executing the self test will interrupt the currently running show.

Function Buttons

CueServer 2 provides up to eight customizable front-panel function buttons. These buttons can be customized for the needs of a particular application, for example, they can run presets, start shows, change modes, show operating status, and more.

Each button can be programmed with any of the available CueScript actions and/or rules to automate any kind of action in CueServer. Each function button includes full RGB backlighting that is also controlled by the CueScript programming.

The CS-900 CueServer 2 Pro features removable button caps. Optional blank caps are available that allow for the insertion of printed transparency films to customize the legends for each button.

Specifications

	CS-900	CS-920	CS-940
Function Buttons	8	2	8
Backlighting	Full RGB	Full RGB	Full RGB
Replaceable Legends	Yes, Optional	No	No
			

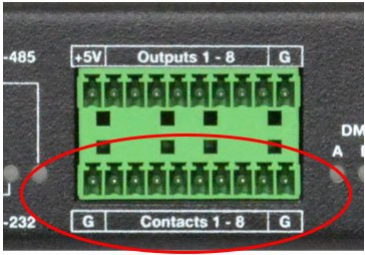
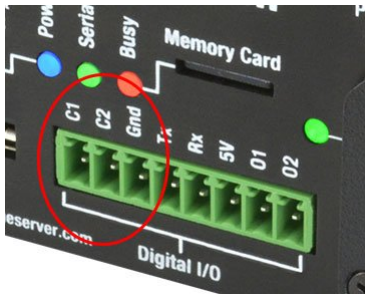
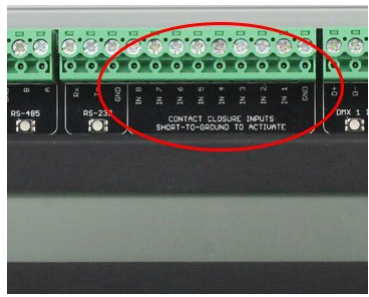
Contact Closures

CueServer 2 provides up to eight low-voltage contact-closure inputs. These inputs are designed for connecting switches, buttons, motion detectors, sensors, and most anything that makes an electrical connection between two conductors.

Each contact-closure input can be programmed with any of the available CueScript actions and/or rules to automate any kind of action in CueServer.

Each input floats up to a voltage around 3.3V DC through a weak pull-up resistor. When that input is shorted-to-ground, the CueServer device detects this drop in voltage as a “contact closure”.

Specifications

	CS-900	CS-920	CS-940
Contact Closure Inputs	8	2	8
Connector	10-Position Pluggable Terminal Block	8-Position Pluggable Terminal Block (shared with other I/O)	12-Position Pluggable Terminal Block (shared with RS-232 port)
			
Pinout	<ul style="list-style-type: none"> 1 = Common (Ground) 2 = Contact 1 3 = Contact 2 4 = Contact 3 5 = Contact 4 6 = Contact 5 7 = Contact 6 8 = Contact 7 9 = Contact 8 10 = Common (Ground) 	<ul style="list-style-type: none"> 1 = Contact 1 2 = Contact 2 3 = Common (Ground) 4 = RS-232 Transmit 5 = RS-232 Receive 6 = Aux +5VDC Output 7 = Output 1 8 = Output 2 	<ul style="list-style-type: none"> 1 = Common (Ground) 2 = Contact 1 3 = Contact 2 4 = Contact 3 5 = Contact 4 6 = Contact 5 7 = Contact 6 8 = Contact 7 9 = Contact 8 10 = RS-232 Common 11 = RS-232 Transmit 12 = RS-232 Receive

Digital Outputs

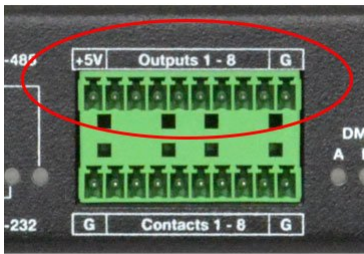
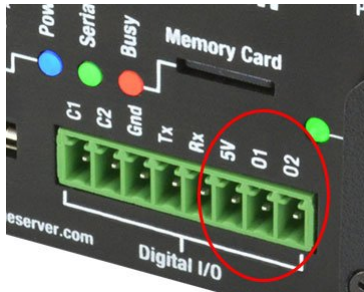
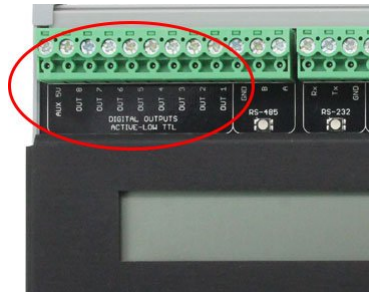
CueServer 2 provides up to eight low-voltage digital outputs. These outputs are designed for connecting LED indicators, small relays, buzzers, pilot lights, and most anything that can be powered from a small DC voltage.

Each digital output can be turned on or off as a response to any of the available CueScript actions and/or rules.

Each output is a TTL “short-to-ground” output. When the output is “on”, its pin shorted-to-ground. When an output is “off”, its pin is an open connection. This means that the output is used to make a complete circuit through the connected accessory device (like an indicator or relay) to ground; the other side of the accessory needs to be connected to a positive voltage.

For convenience, an auxiliary 5VDC output is provided with the digital outputs. This voltage source can optionally be used to provide power to LED indicators, small relays, etc. The maximum current available at the *Aux +5VDC Output* is 200mA.

Specifications

	CS-900	CS-920	CS-940
Digital Outputs	8	2	8
Connector	10-Position Pluggable Terminal Block	8-Position Pluggable Terminal Block (shared with other I/O)	12-Position Pluggable Terminal Block (shared with RS-485 port)
			
Pinout	<ul style="list-style-type: none"> 1 = Aux +5VDC Output 2 = Output 1 3 = Output 2 4 = Output 3 5 = Output 4 	<ul style="list-style-type: none"> 1 = Contact 1 2 = Contact 2 3 = Common (Ground) 4 = RS-232 Transmit 5 = RS-232 Receive 	<ul style="list-style-type: none"> 1 = RS-485 “A” 2 = RS-485 “B” 3 = RS-485 Common 4 = Output 1 5 = Output 2

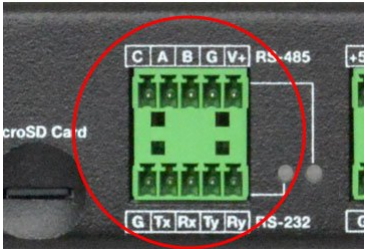
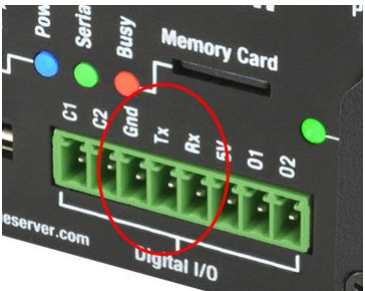
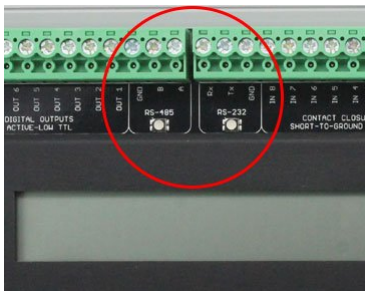
	6 = Output 5 7 = Output 6 8 = Output 7 9 = Output 8 10 = Common (Ground)	6 = Aux +5VDC Output 7 = Output 1 8 = Output 2	6 = Output 3 7 = Output 4 8 = Output 5 9 = Output 6 10 = Output 7 11 = Output 8 12 = Aux +5VDC Output
--	--	--	---

Serial Ports

CueServer 2 provides up to two serial ports, one RS-232 port and optionally one RS-485 port. These ports are designed to be used to interface with 3rd party devices such as video projectors, automation systems, security panels, motorized window coverings, and most anything else that has either an RS-232 or RS-485 serial interface.





Each serial port can be programmed to output strings of ASCII and/or binary data via CueScript actions and/or rules. Each serial port can be configured to receive and execute CueScript commands or to work with CueStation Hub protocol. Triggering on custom input strings has not been implemented yet.

Specifications

	CS-900	CS-920	CS-940
RS-232 Ports	1	1	1
RS-485 Ports	1	–	1
Connector	2× 5-Position Pluggable Terminal Blocks	8-Position Pluggable Terminal Block (shared with other I/O)	2× 12-Position Pluggable Terminal Blocks (shared with contact closures and digital outputs)
			
Pinout	<p>Top Row</p> <ul style="list-style-type: none"> 1 = RS-485 Common 2 = RS-485 “A” 3 = RS-485 “B” 4 = Ground 5 = V+ <p>Bottom Row</p> <ul style="list-style-type: none"> 1 = RS-232 Common 2 = RS-232 Transmit 3 = RS-232 Receive 	<ul style="list-style-type: none"> 1 = Contact 1 2 = Contact 2 3 = Common (Ground) 4 = RS-232 Transmit 5 = RS-232 Receive 6 = Aux +5VDC Output 7 = Output 1 8 = Output 2 	<p>Contact Closure Terminals</p> <ul style="list-style-type: none"> 1-9 = Contact Closures 10 = RS-232 Common 11 = RS-232 Transmit 12 = RS-232 Receive <p>Digital Output Terminals</p> <ul style="list-style-type: none"> 1 = RS-485 “A” 2 = RS-485 “B” 3 = RS-485 Common 4-12 = Digital Outputs

	4 = Aux Transmit (Do Not Use) 5 = Aux Receive (Do Not Use)		
--	---	--	--

Indicators

Color & Pattern	Description
 Off	No serial data input or output
 Quickly Flashing Green	Transmitting and/or receiving data
 Quickly Flashing Yellow	Received unexpected data (possibly bad protocol)
 Quickly Flashing Red	Received poorly framed data bytes (possibly wrong baud rate, or breaks in data)

Memory Card

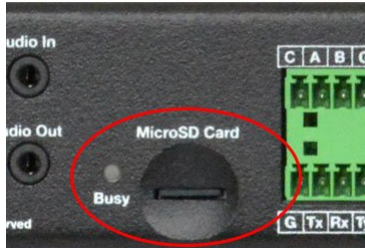


CueServer 2 uses a *microSD* memory card for storage of show files. Units ship from the factory with Class 10 8GB cards pre-installed.

At this time, CueServer 2 does not support hot-swapping cards while the system is running. This means that a card must be inserted when the device is turned on, and must remain inserted while it is running. To switch cards, please power the device off before changing cards.






We have received reports of CueServer 2 show failures on units that have Class 4 or lower cards. All CueServer 2 units ship from the factory with Class 10 (or higher) cards installed. If you plan to use your own card, please make sure that it is Class 10 (or higher) as specified below.

Specifications

	CS-900	CS-920	CS-940
Card Type	microSD	microSD	microSD
File System Format	SDHC/FAT32	SDHC/FAT32	SDHC/FAT32
Speed Class	Class 10 (or higher) or UHS Class 1 (U1) (or higher)	Class 10 (or higher) or UHS Class 1 (U1) (or higher)	Class 10 (or higher) or UHS Class 1 (U1) (or higher)
Maximum Card Size	2TB	2TB	2TB
			

Indicators

Color & Pattern	Description
-----------------	-------------

 Off	No card inserted
 Solid Red	Card is mounted and in use by system (do not remove)
 Quickly Flashing Red	Card did not mount properly




Reset Button

CueServer 2 has a “Reset” button.

The reset button is used to force the unit into bootloader mode if the button is held down during power up. The bootloader mode is for factory use only. There is no need to attempt to use bootloader mode except as instructed by Technical Support.

The reset button has no function while the unit is running.

Specifications

	CS-900	CS-920	CS-940
Location	“Pinhole” on Rear Panel	“Pinhole” on Front Panel	Internal, Under Cover
			

Self-Test Function

CueServer 2 has a built-in Self-Test function that tests nearly every subsystem and circuit path in the product. This function can be used if there is a suspicion that the CueServer hardware has a physical fault.

! Do not enter the Self-Test mode while a show is in progress. The show will be halted and the DMX output from the device will switch to a test pattern. Also, the only way to exit the Self-Test mode is to power-cycle the device.

This function can be accessed by selecting the “Self Test” menu item from the LCD Display. To enter the Self-Test mode, press the **Enter** button to show the main menu, then scroll down to the “Self Test” item and press **Enter** again. A confirmation dialog will appear on the LCD screen. Move the cursor to the right and press **Enter** again. A second confirmation dialog will appear. Again, move the cursor to the right and press **Enter**. These confirmations appear because the Self-Test function causes the CueServer to halt any currently running show.

When the Self-Test starts, a display similar to the following will appear on the LCD:

```
KEYS | DI | BUS | RTC | DMX | SER |
---- | -- | OK | OK | OK | OK |
```

While the Self-Test is running, the following functions are performed:

- Continuous display of front-panel switch inputs
- Continuous display of contact closure inputs
- Display of system bus status
- Display of real-time clock status
- Display of DMX loopback test
- Display of Serial Port loopback test
- PCB Indicator test
- Digital Output test

The following sections describe each of these tests and the display in detail.

Indicator and Digital Output Test

While in the Self-Test mode, all of the indicators on the device illuminate and slowly cycle through various colors.

Function Button Indicators

Since the function button indicators have the capability of illuminating in full 24-bit RGB colors, they will demonstrate this by slowly crossfading through the entire RGB spectrum in order: Red, Yellow, Green, Cyan, Blue, Magenta. If any of the button indicators is faulty, they will not match the color of the other button indicators.

General Purpose Indicators

The remaining indicators on the product can only illuminate in primary colors. As the Self-Test is running, they will periodically jump from color to color. On the CS-900, the indicators will show six colors (Red, Yellow, Green, Cyan, Blue, Magenta). On the CS-940, all indicators except the power indicator will show three colors (Red, Yellow, Green) and the power indicator will show three different colors (Red, Magenta, Blue). If any of these patterns are not followed, then one of the indicators may be faulty.

Digital Outputs

The eight digital outputs slowly cycle through one output being on at a time in order from 1 thru 8. If more than one output is on or if an output is skipped in the pattern, the output may be faulty.

Manual Testing Mode

When any of the front-panel buttons and/or contact closure inputs is pressed/closed, then the corresponding function button indicator will illuminate White and the corresponding digital output will turn on and a corresponding general indicator will illuminate. In this mode, all other indicators and outputs will turn off. After the button or contact is released, the regular cycling pattern will be resumed after three seconds. This “manual” mode of testing is useful for diagnosing a broken switch or indicator.

The following table shows how each button, indicator, contact and output is connected in manual testing mode:

Function Button, Contact Closure, or Digital Output	Indicator (CS-900)	Indicator (CS-940)
1	DMX Port D	Power/Status
2	DMX Port C	RS-485
3	DMX Port B	RS-232
4	DMX Port A	Memory Card

5	RS-485	DMX 1 In
6	RS-232	DMX 2 In
7	Memory Card	DMX 1 Out
8	Power/Status	DMX 2 Out

Keyboard and LCD Display Test



The section of the display marked **KEYS** shows which keys on the front-panel are currently depressed. The following table shows the possible values that are shown for this portion of the test:

KEYS	Meaning
----	No keys are pressed. This is the normal “good” display when no keys are pressed.
UP	The navigation UP button is pressed. Pressing UP also dims the LCD display’s backlight to 50% to test software control of the backlight.
DOWN	The navigation DOWN button is pressed. Pressing DOWN also sets the LCD display’s contrast to 50% to test software control of the display contrast.
RGHT	The navigation RIGHT button is pressed. Pressing RIGHT also advances to the next self-test page, such as one of the DMX detail pages. Continue pressing LEFT and RIGHT to circle around the available self-test pages.
LEFT	The navigation LEFT button is pressed. Pressing LEFT also advances to the previous self-test page, such as one of the DMX detail pages. Continue pressing LEFT and RIGHT to circle around the available self-test pages.
ENTR	The navigation ENTER button is pressed. Pressing ENTER also clears any errors for the DMX and Serial loopback tests.
1 .. 8	One of the front-panel function buttons is pressed. The number of the button is displayed. Also, the indicator on above the button illuminates White and the corresponding digital output is turned on.
XXXX	A four-digit hexadecimal number will appear if more than one button is pressed simultaneously. The hexadecimal number represents the sum of individual “bits” that correspond to each button that is pressed. The following are the bit values of each button: 0001 = Up 0002 = Down 0004 = Right 0008 = Left

0010 = Enter
 0100 = Function 1
 0200 = Function 2
 0400 = Function 3
 0800 = Function 4
 1000 = Function 5
 2000 = Function 6
 4000 = Function 7
 8000 = Function 8

Digital Input (Contact Closure) Test



The section of the display marked **DI** shows which digital inputs (contact closures) are currently closed. The following table shows the possible values that are shown for this portion of the test:

DI	Meaning
--	No contacts are closed. This is the normal “good” display when no contacts are closed.
1 .. 8	One of the contact closure inputs is closed. The number of the contact input is displayed. Also, the corresponding digital output is turned on and the indicator above the corresponding button illuminates White.
XX	A two-digit hexadecimal number will appear if more than one contact is closed simultaneously. The hexadecimal number represents the sum of individual “bits” that correspond to each contact that is closed. The following are the bit values of each button: 01 = Contact 1 02 = Contact 2 04 = Contact 3 08 = Contact 4 10 = Contact 5 20 = Contact 6 40 = Contact 7 80 = Contact 8

System Bus Test

BUS
OK

The section of the display marked **BUS** shows the result of testing the peripherals on the internal system bus. The following table shows the possible values that are shown for this portion of the test:

BUS	Meaning
OK	All tests have passed.
XXX	<p>A three-digit hexadecimal number will flash if one or more system bus tests failed. The hexadecimal number represents the sum of individual “bits” that correspond to each failed test. The following are the bit values of each test:</p> <ul style="list-style-type: none"> 001 = Board ID Error 002 = NVRAM Error 004 = LCD Contrast Error 008 = Digital IO Error 010 = Audio Codec Error 020 = Keyboard Switch Error 040 = Keyboard Indicator 1-4 Error 080 = Keyboard Indicator 5-8 Error 100 = PCB Indicator A Error 200 = PCB Indicator B Error 400 = Real-Time Clock Error

Real-Time Clock Test

RTC
OK

The section of the display marked **RTC** shows the result of testing the real-time clock circuitry. The following table shows the possible values that are shown for this portion of the test:

RTC	Meaning
OK	All tests have passed.
OSC	The clock oscillator has failed.

BAT The clock backup battery has failed.

DMX Transceiver Loopback Test

DMX
OK

While the Self-Test is running, a DMX output test signal is generated by DMX Ports B & D (on the CS-900) and DMX 1/2 Outputs (on the CS-940). A loopback to a corresponding input port is used so the Self-Test function can verify that the data sent out of the output port is what is received by the input port. The following table shows which output port should be looped with which input port.

CS-900	From (Output)	To (Input)
Loopback 1	Port B	Port A
Loopback 2	Port D	Port C
CS-940	From (Output)	To (Input)
Loopback 1	DMX 1 Out	DMX 1 In
Loopback 2	DMX 2 Out	DMX 2 In

The section of the display marked **DMX** shows the result of testing the DMX Input/Output circuitry. The following table shows the possible values that are shown for this portion of the test:

DMX	Meaning
OK	All tests have passed.
1	DMX Loopback 1 have errors.
2	DMX Loopback 2 have errors.
1+2	Both DMX Loopbacks have errors.

Use the **RIGHT** and **LEFT** navigation buttons to display either the DMX 1 or DMX 2 pages on the LCD to view additional details about the DMX loopback tests.

The extra LCD pages that show detailed DMX loopback information appear similar to the following:

DMX		C1,2,3		MIS		CHN		ST		DAT		OV
1		000000		0		512		1		0		0

The number directly under **DMX** shows which loopback (1 or 2) that you are viewing. The hexadecimal six digits under **C1,2,3** show the current DMX values for channels 1, 2 and 3 (these channels are copied throughout the entire 512 channels of the output). The value under **MIS** shows how many “missed” packets have been not read on the input that were sent on the output (which should be 0). The value under **CHN** shows how many DMX channels are being received (which should be 512). The value under **ST** shows how many unique start codes are being received (which should be 1). The value under **DAT** shows how many individual data errors have been detected (which should be 0). The value under **OV** shows how many packets received had more than 512 channels in it (which should be 0).

If any of the fields on this display counts up to a number higher than can be displayed, the field will show a ******* to signify that more than 999 events were counted.

To reset all of the error counters back to zeros, press the **Enter** button on the navigation switch.

Serial Loopback Test

SER
OK

While the Self-Test is running, a test signal is generated at the RS-232 Tx pin. A loopback to the RS-232 Rx pin is used so the Self-Test function can verify that the data sent out of the port is what is received by the input. The following table shows which output port should be looped with which input port.

CS-900	From (Output)	To (Input)
Loopback	RS-232 Tx (Pin 2)	RS-232 Rx (Pin 3)
CS-940	From (Output)	To (Input)
Loopback	RS-232 Tx (Pin 11)	RS-232 Rx (Pin 12)

The section of the display marked **SER** shows the result of testing the Serial Port circuitry. The following table shows the possible values that are shown for this portion of the test:

SER	Meaning
OK	All tests have passed.
ERR	A serial port loopback test failed.

Using CueServer

This section of the manual is under construction. This section will provide detailed information about specific programming topics for CueServer.

Presently, this section contains the following subsections:

- [DMX Triggers](#)

DMX Triggers

CueServer offers the ability to trigger actions or events based on the live incoming value of DMX channels.

DMX Triggers are configured using the **DMX** section of the **TRIGGERS** group within CueServer Studio.





When chosen, the top of the editor panel will show the DMX Triggers listing:


DMX Triggers		
Number	Name	Details
1	Show Start	Channel: 501, Range: 128...255, Rules: 2
2	Intensity Control	Channel: 502, Submaster: 3
3	Screen Position	Channel: 503, Act: Write COM1 "A5\$IFF", Delay: 250

+ - ⚙️ 3 DMX triggers

Each column of the list is described below:

- **Number** – The numerical order of the list.
- **Name** – The name given to each DMX Trigger.
- **Details** – A summary of the properties of each DMX Trigger.


The  and  buttons at the bottom of the list will add a new trigger, or remove a selected trigger.

The  button displays a menu of options, including:

- **Duplicate DMX Trigger...** – Used to make a copy of an existing trigger.
- **Refresh** – Used to reload the list of DMX triggers.

DMX Trigger Types

When creating or editing a DMX Trigger, a type menu appears that chooses what type of behavior the trigger will have:

Type: 

There are three types (or modes) of DMX Triggers. Each type of trigger requires different properties and exhibits a different behavior. The available trigger types include:

- [Enter/Exit Range](#) – A trigger based on a DMX channel either entering or exiting a specific range of values.
- [Submaster Control](#) – A trigger that automatically links the given DMX channel value to a Playback fader's submaster.
- [Act on Change](#) – A trigger that performs a CueScript action whenever a DMX channel value changes.

The following sections describe each of these DMX Trigger types.

Enter/Exit Range Trigger

When a DMX Trigger is set to the **Enter/Exit Range** type, various rules can be added to the trigger that fire whenever the channel value either enters or exits a particular range of values.

This type of trigger is best used to activate certain events within CueServer based on an input channel being raised or lowered into or out of a range of values.

When editing an **Enter/Exit Range** trigger, the editor panel will appear similar to this example:

Properties

Number:

Name:

Trigger

Channel:

Type:

Range: thru

Rules:

- +
 -
 WHENEVER This DMX Trigger Enters Range
 THEN Execute Cue in playback
- +
 -
 WHENEVER This DMX Trigger Exits Range
 THEN Clear Playback

The following properties can be set for an **Enter/Exit Range** trigger:

- **Properties**
 - **Number** – The numerical order of the trigger in the list. This number can be changed to replace an existing trigger or to organize triggers numerically.
 - **Name** – A name for the trigger. This field can be freely set to any text.
- **Trigger**

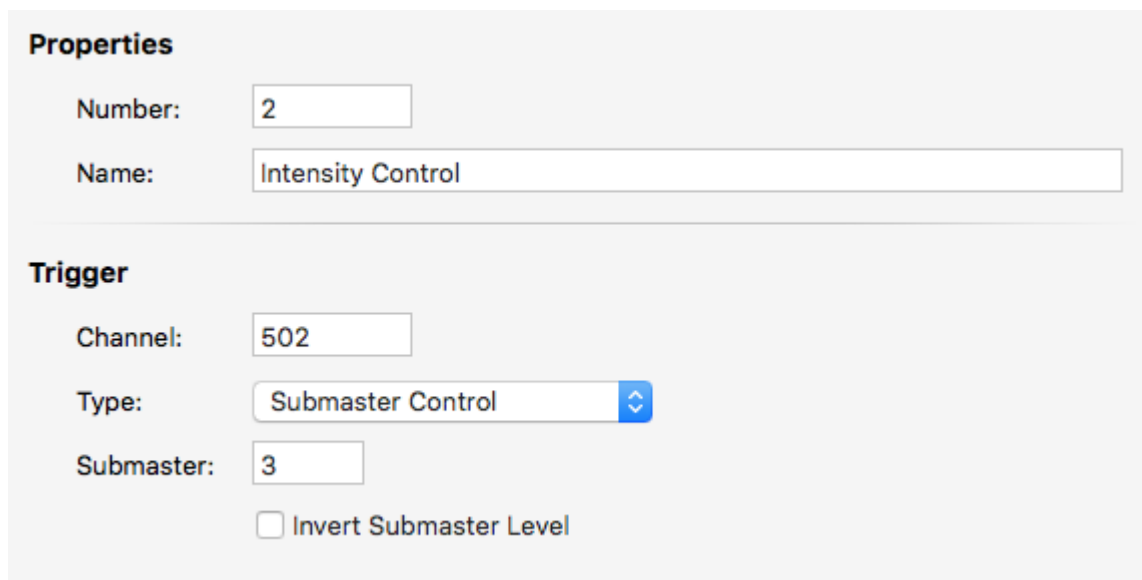
- **Channel** – The channel that is being observed for the trigger. A value from 1 to 16384 may be entered into this field. The channel number corresponds to the *global* channel number within CueServer, not a local channel number in a single universe.
- **Type** – The type of the DMX Trigger. Set to **Enter/Exit Range** for this type of trigger.
- **Range** – A range of *decimal* DMX values for this trigger. Each field may be from 0 to 255.
- **Rules** – One or more rules configured to trigger whenever the input DMX channel either enters or exits that range. Each rule may contain any actions or conditions as permitted by rules.

Submaster Control Trigger

When a DMX Trigger is set to the **Submaster Control** type, the incoming DMX channel value is directly mapped to control the Submaster of a Playback Fader.

This type of trigger is best used when it is desired to have an external console directly control the Submaster level of one of CueServer's Playback Faders.

When editing an **Submaster Control** trigger, the editor panel will appear similar to this example:



The screenshot shows a web-based editor panel for a Submaster Control Trigger. It is divided into two main sections: 'Properties' and 'Trigger'. In the 'Properties' section, there is a 'Number' field with the value '2' and a 'Name' field with the value 'Intensity Control'. In the 'Trigger' section, there is a 'Channel' field with the value '502', a 'Type' dropdown menu set to 'Submaster Control', and a 'Submaster' field with the value '3'. Below the 'Submaster' field is a checkbox labeled 'Invert Submaster Level' which is currently unchecked.

The following properties can be set for an **Submaster Control** trigger:

- **Properties**
 - **Number** – The numerical order of the trigger in the list. This number can be changed to replace an existing trigger or to organize triggers numerically.
 - **Name** – A name for the trigger. This field can be freely set to any text.
- **Trigger**
 - **Channel** – The channel that is being observed for the trigger. A value from 1 to 16384 may be entered into this field. The channel number corresponds to the *global* channel number within CueServer, not a local channel number in a single universe.
 - **Type** – The type of the DMX Trigger. Set to **Submaster Control** for this type of trigger.
 - **Submaster** – The number of the Playback Fader's Submaster to control. A value from 1 to 32 may be in this field.

- **Invert Submaster Level** – Normally, the DMX value is passed directly to the Submaster value. When this checkbox is selected, the Submaster value will be inverted from the DMX value. For example, when the DMX value is at zero, the Submaster will be at full.

Act on Changes Trigger

When a DMX Trigger is set to the **Act on Changes** type, any time the incoming DMX value changes, a CueScript action is executed.

This type of trigger is best used to create custom actions that take the input value of a DMX channel and perform some kind of operation upon it. For example, a trigger of this type can output a string via a serial port or by UDP messages that passes the DMX channel value to another device. Another example would be to control a group of channels from the input of a single channel.

When editing an **Act on Changes** trigger, the editor panel will appear similar to this example:

The screenshot shows a configuration window for an 'Act on Changes' trigger. It is divided into two sections: 'Properties' and 'Trigger'. In the 'Properties' section, the 'Number' is set to 3 and the 'Name' is 'Screen Position'. In the 'Trigger' section, the 'Channel' is 503, the 'Type' is 'Act on Change', the 'Action' is 'Write COM1 "A5\\$\1FF"', and the 'Delay' is 250 milliseconds.

Properties	
Number:	3
Name:	Screen Position

Trigger	
Channel:	503
Type:	Act on Change
Action:	Write COM1 "A5\\$\1FF"
Delay:	250 milliseconds

The following properties can be set for an **Act on Changes** trigger:

- **Properties**
 - **Number** – The numerical order of the trigger in the list. This number can be changed to replace an existing trigger or to organize triggers numerically.
 - **Name** – A name for the trigger. This field can be freely set to any text.
- **Trigger**
 - **Channel** – The channel that is being observed for the trigger. A value from 1 to 16384 may be entered into this field. The channel number corresponds to the *global* channel number within CueServer, not a local channel number in a single universe.
 - **Type** – The type of the DMX Trigger. Set to **Act on Changes** for this type of trigger.

- **Action** – A CueScript action that is executed each time the input DMX value changes. See below for examples.
- **Delay** – Each time this trigger is activated by a change in DMX value, this delay temporarily *disarms* the trigger from firing again. After the delay expires, the trigger is re-armed and will fire again if the channel value has changed since the previous triggering. This is particularly useful when using this trigger type to send serial strings to external devices that cannot handle updates as fast as DMX is coming into the CueServer (approximately 40Hz).



Note: String Substitution Codes can be used to retrieve the DMX value and/or other live information into strings when using this type of DMX Trigger. See the section on [Strings](#) for more information.

Examples

Controlling Groups

To create a scenario where a group of channels in CueServer is controlled by an input DMX channel, string substitution can be used. First assign a variable to the DMX value, and then use that variable to adjust the level of the group. The CueScript action would be:

```
"level" = "#\#1"; Group 1 at `level`
```

Controlling Window Blinds

Popular window blind systems use a RS-232 serial protocol to communicate with the motors. To create a DMX trigger that takes the value of a DMX channel and passes it to the window blind controller, a string such as "`FCF0FF<group-address><motor-id>FB<position>FFFF<checksum>`" must be sent out the serial port. To do this, the CueScript action would be:

```
Write COM1 "FCF0FF808080ABCDEFFB\lFFFF\lS"
```


CueScript Language

CueServer uses a command language called **CueScript** as the basis of nearly all of CueServer's control and automation capabilities. You will use CueScript to make CueServer perform actions. If you need CueServer to start playing a cue, you can enter `Cue 1 Go` on the command line. If you want CueServer to fade up a DMX channel, enter `Time 5 Channel 1 At FL`.

Not only can CueScript be used to enter live commands into CueServer, but CueScript is used throughout the system to perform all kinds of automation tasks. Advanced logic can be added to a CueServer project using CueScript to orchestrate lighting cues with button presses, contact closure inputs, serial port strings, LCD messages, digital outputs, and much more.

CueScript was created with the following in mind:

- It must be easy to use – the language reads easily in English.
- It must be familiar to lighting professionals – commands like `Group 1 Release` are very “console-like”.
- It has a short-hand abbreviation system to make it easier to type – Instead of typing `Channel 1 At 100`, you can type `C1A100`.

The following sections describe the language in more detail.

CueScript Overview

The following topics describe the details of the language:

- [Executing Commands](#)
- [Command Syntax](#)
- [Expressions](#)
- [Command Context](#)
- [Levels](#)

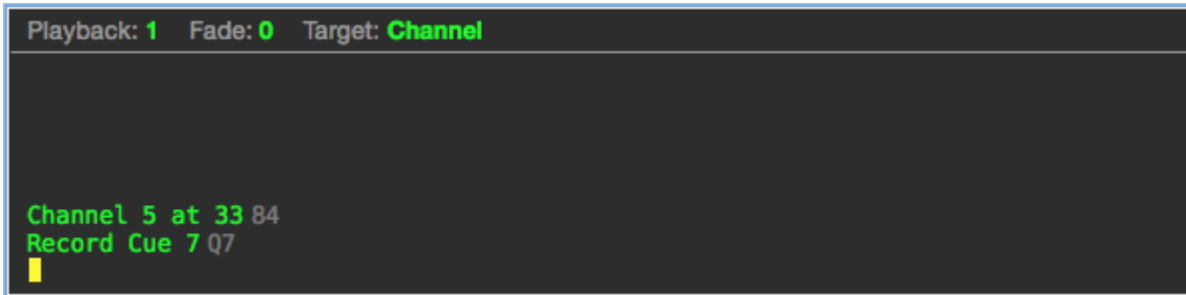
The specific commands available are detailed in the following sections:

- [Selection Commands](#)
- [Action Commands](#)
- [Logic Commands](#)

Executing Commands

There are several places where CueScript commands are used within the system.

Command Line



When working with CueServer Studio on a live CueServer, a command line appears at the bottom of the window. This command line allows CueScript commands to be executed at any time.

Enter a CueScript command (like `Channel 5 at 33` or `Record Cue 7`) and CueServer performs the requested task. Whenever a command is entered, the CueServer replies with a value (which is shown in gray text after the command).

Rules



CueServer uses the concept of *rules* to define automation tasks throughout the system. Using CueServer Studio, you can define global rules that are always being monitored for triggering, or you can assign local rules to individual cues, buttons, contacts, and other objects within the system.

A rule takes the form of **Whenever something happens ... Then do something**

One of the options in the *then do something* clause of a rule is to perform a CueScript. In the example above, *whenever this button is pressed, execute the command Cue 1 Go*.

Actions

Time of Day

Type:

Time: : : AM PM

Action:

Some objects in the system (such as Timers and Macros) are programmed with CueScript actions.

When editing a Timer or Macro, an action field appears, allowing a CueScript command to be entered as the object's action. Whenever the Timer or Macro is triggered, the programmed action is performed.

External Commands

CueScript commands can also be sent to CueServer from an external source by one of the methods listed below:

- [CueScript via UDP](#)
- [CueScript via HTTP](#)
- [CueScript via Serial](#)

CueScript via UDP

CueServer allows external CueScript commands to be sent to it via UDP packets.

There are two methods that can be used to send UDP packets to CueServer:

- **Unicast Method:** A UDP packet containing one or more CueScript commands can be *unicast* directly to the IP Address of the CueServer on port 52737. Using this method, only the specific CueServer sent the packet will execute the CueScript.
- **Multicast Method:** A UDP packet containing one or more CueScript commands can be *multicast* to the CueScript Group Address (239.255.204.2) on port 52737. Using this method, **all** CueServers on the local network will receive and execute the CueScript.

The contents of the UDP packet are simply the CueScript string that is to be executed by the CueServer.

Examples:

- Cue 1 Go
- Button 1.5 On
- Macro 7
- P3CL; Q5G; B1ON

CueScript via HTTP

CueServer allows external CueScript commands to be sent to it via HTTP protocol (a simple URL request).

Built into CueServer is a web server that allows CueScript commands to be executed by receiving them in a special URL. The typical format of this URL is:

```
http://<ip-of-CueServer>/exe.cgi?cmd=<command>&<optional-parameters>
```

For example, the following URL will execute the command `Cue 1 Go`:

```
/exe.cgi?cmd=Cue+1+Go
```

Additional details about the `exe.cgi` URL is available in [CGI API](#) section of this manual.

CueScript via Serial

CueServer allows external CueScript commands to be sent to it via RS-232 and/or RS-485 serial strings.

A serial port on CueServer is configured by going into the *Stations* section and then editing the Built-In Station (Station 0). The serial ports available to CueServer will appear in the list of *Ports*.

Port 1 (RS-232, COM1)

Name:	<input type="text" value="RS-232 Port"/>
Protocol:	<input type="text" value="CueScript in [Brackets]"/> ▾
Baud Rate:	<input type="text" value="9,600 bps"/> ▾
Data Format:	<input type="text" value="8-N-1 (Default)"/> ▾
	<input type="checkbox"/> Echo Received Characters
	<input type="checkbox"/> Reply with CueScript result

Other external station types that include serial ports would also be configured by choosing them in the *Stations* list and then clicking on one of the *Ports* that appears.

From the *Port Configuration* panel that appears there are two protocols that can be chosen that allow CueScript commands to be sent to the CueServer via a serial port:

- **CueScript in [Brackets]:** When this protocol is chosen, a CueScript string can be sent to this serial port as long as it is enclosed by “square brackets”. For example, `[Cue 1 Go]`. Any characters received outside of the brackets will be ignored. After the opening bracket, the command is received and buffered until a closing bracket is received. As soon as the closing bracket is received, the command string is executed. If a command is being received and then another open-bracket is received, then any accumulated characters in the receive buffer are cleared and a new string will begin to be received.
- **CueScript with CR/LF:** When this protocol is chosen, a CueScript string can be sent to this serial port with a terminating carriage-return (0×0D) and/or line-feed (0×0A). For example, `Cue 1 Go,` followed by either a 0×0D or 0×0A (or both) characters. All characters received on the serial port will

be accumulated until a carriage-return or line feed is received. As soon as one of those terminating characters are received, the command string is executed.

The Baud Rate and Data Format selected for the serial port must match that of the transmitting device for the CueScript commands to be received properly.

If the “Echo Received Characters” box is checked, then every character received by the port will be re-transmitted (“echoed”) back to the sender.

If the “Reply with CueScript result” box is checked, then the result of the CueScript string will be transmitted back to the sender.

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.

White spaces in a command (spaces, tabs, new lines, etc.) are ignored by CueServer and are used to simply make the commands more readable. Also, the semicolon (;) can optionally be used between commands on a single line to make commands more readable. CueScript is not case-sensitive, meaning that it doesn't matter if you use upper or lower case letters in a command. All of the following commands are equivalent:

- `Time 5 Channel 3 at 50`
- `time5channel3at50`
- `Time 5; Channel 3 at 50`
- `Time 5`
`Channel 3 at 50`

Using Abbreviations

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`, `Channel` as `C` and `At` as `@`. For example, the previous example may be abbreviated as:

- `T5;C3@50`

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 command, 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`. Therefore, the command `Cue 1 Go` may be abbreviated as `Q1G`.

Expressions

An expression is a combination of symbols including numbers, operators, variables and groupings that are used to specify a mathematical function. Expressions result in a numerical value.

The following are examples of expressions:

- 5
- 3 + 7
- 'x' + 4
- ('x' + 5) - 'y'
- (3 + (('x' - 'y') * 12)) - 1
- 'x' > 9
- ('x' > 3) and ('y' < 5)
- (('myShow' + 1) > 5) or 'maintenanceMode'

These examples show the use of operators (such as +, -, >, and *and*), variables (such as 'x', 'y', and 'maintenanceMode') and groupings (using parenthesis).

The following sections explain each of these expression components in detail:

- [Operators](#)
- [Variables](#)
- [Grouping](#)

Operators

The CueScript language allows for operators to be used in expressions. Operators are symbols that appear in-between two values that “operate” on those values. Common operators include mathematical functions such as **+** and **-** for addition and subtraction, and boolean functions such as **And**, and **Or**.

Mathematical Operators

The following operators are mathematic, meaning that they perform functions on numbers:

Operator	Function	Example	Result
+	Addition	3 + 5	8
-	Subtraction	5 - 3	2
*	Multiplication	3 * 7	21
/	Division	18 / 3	6

Concatenation Operator

The following operator performs its function on strings. Shared with the Addition Operator, if either side of the “+” is a string, the result will be a string:

Operator	Function	Example	Result
+	Concatenation	“Cue” + “Server” “CS” + 2 3 + “ is a crowd”	“CueServer” “CS2” “3 is a crowd”

Boolean Operators

The following operators are boolean, meaning that they compare two values in a true or false context. Note that the result of a boolean operator will always be either 0 (meaning false) or 1 (meaning true).

Operator	Function	Examples	Result
==	Equal	5 == 5 3 == 5	1 0

!=	Not Equal	5 != 5 3 != 5	0 1
>	Greater Than	5 > 3 3 > 5	1 0
>=	Greater Than or Equal	5 >= 3 4 >= 4 2 >= 7	1 1 0
<	Less Than	3 < 5 5 < 3	1 0
<=	Less Than or Equal	3 <= 5 4 <= 4 7 <= 2	1 1 0
and	Logical And	0 and 0 0 and 1 1 and 0 1 and 1	0 0 0 1
or	Logical Or	0 or 0 0 or 1 1 or 0 1 or 1	0 1 1 1



It is important to note that when using boolean operators, any value that is zero is interpreted to mean “false”, and any value that is non-zero is interpreted to mean “true”. Given that any non-zero value is “true”, then the expression **5 and 3** would evaluate to **1**, because both sides of the **and** are both true.

Variables

A variable is a symbol that holds and represents a value. Variable symbols are names such as *x*, *MyVariable*, or *lcd.backlight*. Variables can hold numbers (such as 3 or 12.7) or strings (such as *Hello World*).

CueServer uses two different kinds of variables: User Variables and System Variables. User variables can be any combination of printable letters, numbers, the underscore (_) or hyphen (-). System variables are similar, but must contain a dot (.) character. The dot character is how the CueServer distinguishes between User and System variables.

Assigning Values to Variables

There are two ways to assign a value to a variable. The first is with the [Assign](#) command. Here are a few examples:

```
"x" = 3
"MyVariable" = 42
"Message" = "Hello World"
"y" = ('x' + 3)
"caption" = "Press " + 'y' + " to Start"
```

The first line assigns the number 3 to the variable *x*. The second assigns the number 42 to the variable *MyVariable*. The third assigns the string *Hello World* to the variable *Message*. The fourth assigns the result of the expression 'x'+3 to the variable *y*. The last example assigns a string made of several smaller strings ("Press ", followed by the value of variable 'y', followed by " to Start") into the variable *caption*.

The second way to assign a value to a variable is with the [Set](#) command. Here are a few examples:

```
Set x 3
Set MyVariable 42
Set Message "Hello World"
Set y ('x' + 3)
Set caption ("Press " + 'y' + " to Start")
```

These examples are the same as above, except that the [Set](#) command is used instead of using the [Assign](#) command.

Using Variable Values

To use variables in CueScript commands, enclose the variable name in single quotes (`'MyVariable'`).

For example, using the variable values set from above, the following variable substitutions would be made:

<code>Cue 'x' Go</code>	Executes Cue 3
<code>Macro 'MyVariable'</code>	Runs Macro 42
<code>Set lcd.top 'Message'</code>	Displays "Hello World" on the top line of the LCD
<code>Log 'y'</code>	Writes "6" to the System Log
<code>WRITE COM1 'caption'</code>	Sends "Press 6 to Start" to the RS-232 port

Using Variable Values as Commands

To use variables values in CueScript as commands, enclose the variable name in accent quotes (``myCommand``).

The following example script shows how to assign a string that happens to be valid CueScript to the variable `myCommand`. On the second line of script, if the variable `x` is greater than 3, then the commands in `myCommand` will be executed.

```
"myCommand" = "Cue 1 Go"
If ('x' > 3) Then `myCommand`
```

Using System Variables

Special *System Variables* are used to set the properties of hardware devices, or to change internal behaviors of the CueServer. All system variables include a dot (`.`) in their name, for example `lcd.backlight`, or `universe.priority`.

See the section on [System Variables](#) for a description of how to use these built-in variables.

Grouping

Parenthesis are used for grouping expressions. Expression grouping is useful when multiple expressions are strung together in a line and the normal order of operations must be overridden.

The CueScript, operators are always interpreted from left to right. Parenthesis can be inserted into a command string to force different groupings of expressions to be evaluated in a different order.

The following examples illustrate how to use parenthesis to get different results. For these examples, assume $x = 3$ and $y = 7$.

Expression	Result
$4 + 2 * 3$	18
$4 + (2 * 3)$	10
Channel 'x' + 'y'	Selects channel 3 and channel 7
Channel ('x' + 'y')	Selects channel 10

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 completely 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) are 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.

CueServer uses separate command contexts internally to keep the user who is using the live command line in CueServer Studio operating in a different environment from other asynchronous actions that are occurring elsewhere in the system. For instance, if an external process is sending UDP messages to CueServer, these messages get their own command context so they don't interfere with others using the system. Also, if a timer or button executes in-between when the user selected a channel and set it's level, this process won't be disturbed, because each of these asynchronous actions occur in their own context.

Levels

The `At` command and several other methods 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 four primary ways, by percentage (the default), or by decimal, hexadecimal or binary 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 instance, `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 `Group 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 `Group 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` keywords can be used instead. For example, `Channel 1 On`, `Button 2 Off`, `Group 3 On`, `Output * On` are all valid binary-value commands.

Strings

CueScript commands frequently contain *strings*. A string is a series of zero or more characters enclosed in “double-quotes”.

Examples of strings include: “Hello World”, “Press Stop to Cancel Show Playback”, “button.onColor”, and “My First Show”.

Examples of commands that use strings are [AUDIO](#), [LOAD](#), [LOG](#), [SET](#), [STACK](#), and [WRITE](#).

Special Characters

Sometimes it is necessary to enter special characters that are non-printable or difficult to enter into a string from the keyboard. Examples include carriage returns, linefeeds, tabs, quotation marks, or special hexadecimal characters such as NULL (0×00).

CueServer allows special characters to be entered into strings using *escape sequences* that start with the backslash character (\) followed by a single letter that designates the specific escape character desired. For example, the escape sequence “\n” becomes a new-line character.

Because the backslash is used to mean “escape”, a single backslash can’t be used to put a backslash into a string. If a backslash is needed, use the escape sequence for backslash which is a double-backslash (\\).

The following table shows the supported escape sequences:

Escape Sequence	Hex Value	Character Represented
\a	0x07	Alarm/Bell
\b	0x08	Backspace
\f	0x0C	Formfeed
\n	0x0A	Newline
\r	0x0D	Carriage Return
\t	0x09	Horizontal Tab
\v	0x0B	Vertical Tab
\\	0x5C	Backslash

<code>\'</code>	<code>0x27</code>	Single quotation mark
<code>\"</code>	<code>0x22</code>	Double quotation mark
<code>\xhh</code>	<code>0xhh</code>	Any hexadecimal byte
<code>\nnn</code>	<code>0xoo</code>	Any octal byte

Note that there are two special escape sequences for hexadecimal and octal bytes. The hexadecimal escape sequence is a “backslash-x” followed by exactly two hexadecimal characters (each from 0 thru F) that represents the desired byte. The octal escape sequence is a backslash followed by three digits from 0 thru 7. These three digits represent the desired byte value in octal.



The escape sequences used by CueServer are the same (or very similar) as those used by several popular programming languages, including C, C++, Java, JavaScript, JSON, Objective-C, PHP, Python, and SQL.

Value Substitutions

In addition to the *escape sequences* above that allow special characters to be inserted into a string, CueServer also supports a variety of escape sequences that are used to substitute special *values* into a string.

These values change depending on the context in which the string is being used. CueServer will substitute the escape sequence with the actual value at the time that the string is being used. Each substitution is only valid within the context(s) supported.

The following table shows the supported escape sequences for Value Substitutions:

Escape Sequence	Value Substituted	Valid Contexts
<code>\c</code>	Channel Number (8-bit)	DMX Triggers
<code>\C</code>	Channel Number (16-bit)	DMX Triggers
<code>\l</code>	Channel Value (8-bit)	DMX Triggers
<code>\L</code>	Channel Value (16-bit)	DMX Triggers
<code>\i</code>	Inverted Channel Value (8-bit)	DMX Triggers
<code>\I</code>	Inverted Channel Value (16-bit)	DMX Triggers
<code>\s</code>	Checksum; sum of preceding bytes (8-bit)	Any

<code>\S</code>	Checksum; sum of preceding bytes (16-bit)	Any
<code>*</code>	Reset checksum to zero	Any

By default, each of the values above will be substituted as a binary value. Additional modifiers may be placed between the backslash and the character to change which character format is used to output the value. Supported modifiers are listed below:

Modifier	Result
<code>\$</code>	The value will be output in hexadecimal ASCII characters (0-F). 8-bit values will output exactly 2 characters and 16-bit values will output exactly 4 characters.
<code>#</code>	The value will be output in decimal ASCII characters (0-9). 8-bit values can range from 0-255 and 16-bit values can range from 0-65535.
<code>%</code>	The value will be scaled to a percentage (0-100) output in decimal ASCII characters (0-9)

Examples

```
Write COM1 "Hello World\n"
```

Writes the string *Hello World* followed by a newline to the COM1 serial port.

```
Write COM1 "One\tTwo\Three\tFour"
```

Writes the strings *One*, *Two*, *Three*, and *Four* with tabs in-between each string to the COM1 serial port.

```
Write COM1 "Press \"Start\" to Begin\x00"
```

Writes the string *Press "Start" to Begin* followed by a NULL byte to the COM1 serial port.

```
Log "Channel \#C is set to \#1"
```

Adds a system log message with a string such as "Channel 701 is set to 255".

```
Write "10.0.1.5" "5AA5\#C80\#1\#S"
```

Sends a UDP packet to 10.0.1.5 with a string such as "5AA502BD80FF033D", assuming that the channel is 701 and the level is 255.

Selection Commands

A *selection command* is a type of CueScript command that is used to refer to objects in the system.

Selection commands can be used in conjunction with action commands to perform actions, or selection commands can be used by themselves to query an object's value.

Selecting Objects To Perform Actions

As CueScript is being interpreted by the system, selection commands are used in conjunction with action commands to get things done. First, one or more objects are *selected* by using a selection command, and then one or more *action commands* are used to operate on those selected objects.

For instance, the following CueScript does two things. First, it selects a button. Second, it performs the `On` action.

```
Button 1 On
```

Note that the `On` action turns “on” the currently selected objects, which in this case happens to be Button 1.

The next CueScript selects playback number 3 with a selection command, then the action command `At 75` sets the playback's submaster to 75%.

```
Playback 3 At 75
```

More than one action can be performed on a selected object. The following example shows the selection command `Channel 1` followed by four action commands: `Time 0`, `At 100`, `Time 5`, and `At 0`. In other words, Channel 1 is selected, then the fade time is set to zero, then Channel 1's value is set to 100%, then the fade time is set to 5 seconds, then Channel 1's value is set to 0%.

```
Channel 1 Time 0 At 100 Time 5 At 0
```

Stringing multiple actions together that refer to the same selected object is a powerful way to express compound actions that you want to apply to one or more objects.

Referring To Objects To Determine Their Value

Another powerful way to use *selection commands* is to refer to one or more object to retrieve their value.

For instance, by executing the command:

```
Channel 1
```

CueServer will not only select Channel 1, but it will also reply with the current value of Channel 1.

Being able to ask CueServer the value of an object is very useful for [evaluating expressions](#). Consider the following command:

```
If (Channel 1 > 50) Then Cue 1 Go
```

The **If .. Then** statement is used with the expression **Channel 1 > 50** to make a decision based on the current value of Channel 1. If the value is greater than 50, then **Cue 1 Go** will occur.

All of the [Selection Commands](#), such as [Button](#), [Channel](#), [Contact](#), [Group](#), [Indicator](#), [Output](#), [Playback](#), and [Universe](#) all reply with the current value of their objects.

Referring To Multiple Objects With Different Values

When referring to multiple objects at once, if *all* of the objects have the same value, their shared value will be returned. For instance, if channels 1 through 10 are all set to 50, then the following command will return 50.

```
Channel 1>10
```

But, if the values of channels 1 through 10 have *mixed* values, then the value **-1** will be returned. This special value indicates that the selected objects' values are *mixed*.

Button

Syntax

Command	Description	Return Value
<code>Button <number> [<range...>]</code>	Select one or more buttons	The pressed state of the selected button(s)
<code>Button <station>.<number> [<range...>]</code>	Select one or more buttons on a specific station	The pressed state of the selected button(s)
<code>Button ?</code>	Return the current selection	A selection string

Abbreviation

B

Description

Selecting Buttons

The **Button** command selects one or more buttons in the system. Buttons are typically physical pushbuttons on the front of a CueServer or individual buttons on a connected button station. Use the **Button** command in conjunction with an action command like [At](#), [On](#), [Off](#), [Set](#), [Enable](#) or [Disable](#) to change the properties of one or more buttons. When used alone or in logic expressions, the **Button** command returns the current state of the specified button(s).

Either a single button number can be specified, or a range of buttons can be specified using the various [selection operators](#) like +, -, > and ~.

The wildcard character * can be used as the button number to mean *all* buttons for a particular station.

Working With Stations

When no station number specified, the default station is assumed. The [Station](#) command can be used to change the default station. Unless changed by the [Station](#) command, the default station is typically Station 0, which corresponds to the built-in buttons on the CueServer itself. When a station number is specified as part of the **Button** command, that station number will be used for the selection.

Determining Which Buttons Are Selected

The question mark `?` can be used to ask what the current selection is. A selection string will be returned, which will consist of a single number (like `3`) or a range (like `5>7+9`), or if no buttons are selected, `0` will be returned.

* Note that CueServer treats buttons and indicators very similarly. Buttons have indicators. Buttons are the physical switch that is being pressed by the user and Indicators are the pilot light that shows a button's status. Setting the value of a button or an indicator both sets the indicator's value (turning the indicator on or off). However, getting the value of a button returns the physical switch state (opened or closed), and getting the value of an indicator returns the current state of the indicator (on or off).

Examples

```
Button 1
```

Selects button 1. Future action commands will be directed towards button 1. Also returns `0`, or `1` to indicate if the button is currently unpressed or pressed.

```
Button 1>5 On
```

Turns the LED indicators of buttons 1 thru 5 on.

```
Button 1>3+5>8 Off
```

Turns the LED indicators of buttons 1 through 3 and 5 through 8 off.

```
Button 2.3>5 Enable
```

Enables buttons 3 through 5 on station 2.

```
Button 3.1
```

```
Set Button.OnColor {100,50,0}
```

```
Set Button.Flash 4
```

```
On
```

```
Disable
```

Selects button 1 of station 3, then sets the button's *OnColor* property to Orange (RGB color (100,50,0)), then sets the button's *Flash* property to 4, then turns the LED indicator on, then disables button presses from the button.

```
Button 1.* Off
```

Turns the LED indicators of all buttons on station 1 off.

Station 5

Button 7 Enable

Enables button 7 of station 5.

Button ?

Returns the current button selection in the format of a single number like 3, or a range like 5>7+9.

See Also

- [Selection Operators](#)
- [At](#), [Disable](#), [Enable](#), [Off](#), [On](#), [Set](#)

Channel

Syntax

Command	Description	Return Value
<code>Channel <number> [<range...>]</code>	Select one or more DMX channels	The selected channels' value
<code>Channel <universe>.<local> [<range...>]</code>	Select one or more DMX channels	The selected channels' value
<code>Channel ?</code>	Return the currently selected DMX channels	A selection string

- `<number>`
 - A global channel number from 1 to 16384.
 - This number addresses channels as a continuous block through *all* configured universes.
- `<universe>.<local>`
 - A channel number in “dotted” notation specifies both a universe number from 1 to 128 and a channel from 1 to 512 within that universe.
 - This dotted number addresses channels locally within *each* universe.

Abbreviation

C

Description

Selecting Channels

The **Channel** command selects one or more DMX channels in the currently active playback fader. DMX channels are the individual control levels sent out of the CueServer to operate connected DMX lighting fixtures. Use the **Channel** command in conjunction with an action command like [At](#), [On](#), [Off](#), [Enable](#), [Disable](#), [Park](#), [Unpark](#) or [Release](#) to set channel levels, change the enable or parked state of channels, or release them. When used alone or in logic expressions, the **Channel** command returns the current value of the specified channel(s).

Either a single channel number can be specified, or a range of channels can be specified using the various [selection operators](#) like +, -, > and ~.

The wildcard character `*` can be used as the channel number to mean *all* channels in the active playback fader.

Using Global vs. Local Channel Numbers

The **Channel** command can use either global or local channel numbers. Global channel numbers sequentially number every channel used by all universes in sequential order (typically from `1` up to `16384`). Local channel numbers restart at `1` for each universe and are denoted in a `.` format.

For example, `Channel 3.1` refers to the first channel of universe 3. In global channel numbering (assuming that universes 1 & 2 both have 512 channels each) the same channel would be referred to as `Channel 1025`.

Determining Which Channels Are Selected

The question mark `?` can be used to ask what the current selection is. A selection string will be returned, which will consist of a single number (like `3`) or a range (like `5>7+9`), or if no channels are selected, `0` will be returned.

Examples

```
Channel 1
```

Selects channel 1. Future action commands will be directed towards channel 1. Also returns the channel's current value between `0` and `255`, or `-1` if the channel is released.

```
Channel 1>5 At 33
```

Sets channels 1 through 5 to 33%.

```
Channel 1>3+5>8 On
```

Sets channels 1 through 3 and 5 through 8 to 100%.

```
@Channel 2+5 Park
```

Parks channels 2 and 5.

```
Channel 100
```

```
Time 0
```

```
At 75
```

Time 5

At 0

Selects channel 100, then sets the fade time to 0 (immediate), then sets the channel (100) to 75%, then sets the fade time to 5 (seconds), then sets the channel (100) to 0%.

Channel 33 At #253

Sets channel 33 to decimal value 253.

Channel 44 at \$FA

Sets channel 44 to hexadecimal value \$FA.

Channel 7.1 at FL

Sets channel 1 of universe 7 to 100%.

Channel ?

Returns the current channel selection in the format of a single number like 3, or a range like 5>7+9.

See Also

- [Selection Operators](#)
- [At](#), [Disable](#), [Enable](#), [Off](#), [On](#), [Park](#), [Release](#), [Unpark](#)

Contact

Syntax

Command	Description	Return Value
<code>Contact <number> [<range...>]</code>	Select one or more contacts	The closed state of the selected contact(s)
<code>Contact <station>.<number> [<range...>]</code>	Select one or more contacts on a specific station	The closed state of the selected contact(s)
<code>Contact ?</code>	Return the current selection	A selection string

Abbreviation

CO

Description

Selecting Contacts

The **Contact** command selects one or more contacts in the system. Contacts are typically the hard-wired contact closure inputs on a CueServer or external I/O board. Use the **Contact** command in conjunction with an action command like [Enable](#) or [Disable](#) to change the enabled state of contacts. When used alone or in logic expressions, the **Contact** command returns the current state of the specified contact(s).

Either a single contact number can be specified, or a range of contacts can be specified using the various [selection operators](#) like +, -, > and ~.

The wildcard character * can be used as the contact number to mean *all* contacts for a particular station.

Working With Stations

When no station number specified, the default station is assumed. The [Station](#) command can be used to change the default station. Unless changed by the [Station](#) command, the default station is typically Station 0, which corresponds to the built-in contacts on the CueServer itself. When a station number is specified as part of the **Contact** command, that station number will be used for the selection.

Determining Which Contacts Are Selected

The question mark `?` can be used to ask what the current selection is. A selection string will be returned, which will consist of a single number (like `3`) or a range (like `5>7+9`), or if no contacts are selected, `0` will be returned.

Examples

`Contact 1`

Selects contact 1. Future action commands will be directed towards contact 1. Also returns `0`, or `1` to indicate if the contact is currently opened or closed.

`Contact 1>5 Disable`

Disables processing of events on contacts 1 thru 5.

`Contact 1>3+5>8 Enable`

Enables processing of events on contacts 1 through 3 and 5 through 8.

`Station 5`

`Contact 7 Enable`

Enables contact 7 of station 5.

`Contact ?`

Returns the current contact selection in the format of a single number like `3`, or a range like `5>7+9`.

See Also

- [Selection Operators](#)
- [Disable](#), [Enable](#)

Group

Syntax

Command	Description	Return Value
<code>Group <number> [<range...>]</code>	Select one or more channel groups	The selected channels' value

Abbreviation

GR or U

Description

Selecting Groups

The **Group** command selects one or more DMX channels in the currently active playback fader that were stored in the specified group resource. Use the **Group** command in conjunction with an action command like [At](#), [On](#), [Off](#), [Enable](#), [Disable](#), [Park](#), [Unpark](#) or [Release](#) to set channel levels, change the enable or parked state of channels, or release them. When used alone or in logic expressions, the **Group** command returns the current value of the specified channel(s).

Either a single group number can be specified, or a range of groups can be specified using the various [selection operators](#) like +, -, > and ~.

Examples

```
Group 1
```

Selects the channels in group 1. Future action commands will be directed towards these channels. Also returns the selected channel's current value between 0 and 255, or -1 if the channels are released and/or mixed in value.

```
Group 1+5 At 33
```

Sets the channels in groups 1 and 5 to 33%.

```
@Group 2+5 Park
```

Parks the channels in groups 2 and 5.


```
Group 100
```

```
Time 0
```

```
At 75
```

```
Time 5
```

```
At 0
```

Selects the channels in group 100, then sets the fade time to 0 (immediate), then sets the selected channels to 75%, then sets the fade time to 5 (seconds), then sets the selected channels to 0%.

See Also

- [Selection Operators](#)
- [At](#), [Disable](#), [Enable](#), [Off](#), [On](#), [Park](#), [Release](#), [Unpark](#)

Indicator

Syntax

Command	Description	Return Value
<code>Indicator <number> [<range...>]</code>	Select one or more indicators	The on state of the selected indicator(s)
<code>Indicator <station>.<number> [<range...>]</code>	Select one or more indicators on a specific station	The on state of the selected indicator(s)
<code>Indicator ?</code>	Return the current selection	A selection string

Abbreviation

IND

Description

Selecting Indicators

The **Indicator** command selects one or more indicators in the system. Indicators are typically the LED indicators of pushbuttons on the front of a CueServer or individual indicators on a connected button station. Use the **Indicator** command in conjunction with an action command like [At](#), [On](#), [Off](#) or [Set](#) to change the indication state of one or more indicators. When used alone or in logic expressions, the **Indicator** command returns the current state of the specified indicator(s).

Either a single indicator number can be specified, or a range of indicators can be specified using the various [selection operators](#) like +, -, > and ~.

The wildcard character * can be used as the indicator number to mean *all* indicators for a particular station.

Working With Stations

When no station number specified, the default station is assumed. The [Station](#) command can be used to change the default station. Unless changed by the [Station](#) command, the default station is typically Station 0, which corresponds to the built-in indicators on the CueServer itself. When a station number is specified as part of the **Indicator** command, that station number will be used for the selection.

Determining Which Indicators Are Selected

The question mark `?` can be used to ask what the current selection is. A selection string will be returned, which will consist of a single number (like `3`) or a range (like `5>7+9`), or if no indicators are selected, `0` will be returned.

* Note that CueServer treats buttons and indicators very similarly. Buttons have indicators. Buttons are the physical switch that is being pressed by the user and Indicators are the pilot light that shows a button's status. Setting the value of a button or an indicator both sets the indicator's value (turning the indicator on or off). However, getting the value of a button returns the physical switch state (opened or closed), and getting the value of an indicator returns the current state of the indicator (on or off).

Examples

```
Indicator 1
```

Selects indicator 1. Future action commands will be directed towards indicator 1. Also returns `0`, or `1` to indicate if the indicator is currently off or on.

```
Indicator 1>5 On
```

Turns indicators 1 thru 5 on.

```
Indicator 1>3+5>8 Off
```

Turns indicators 1 through 3 and 5 through 8 off.

```
Indicator 3.1
```

```
Set Indicator.OnColor {100,50,0}
```

```
Set Indicator.Flash 4
```

```
On
```

Selects indicator 1 of station 3, then sets the indicator's *OnColor* property to Orange (RGB color {100,50,0}), then sets the indicator's *Flash* property to 4, then turns the indicator on.

```
Indicator 1.* Off
```

Turns indicators of all buttons on station 1 off.

```
Station 5
```

```
Indicator 7 On
```

Turns indicator 7 of station 5 on.

Indicator ?

Returns the current indicator selection in the format of a single number like 3, or a range like 5>7+9.

See Also

- [Selection Operators](#)
- [At](#), [Off](#), [On](#), [Set](#)

Output

Syntax

Command	Description	Return Value
<code>Output <number> [<range...>]</code>	Select one or more outputs	The state of the selected output(s)
<code>Output <station>.<number> [<range...>]</code>	Select one or more outputs on a specific station	The state of the selected output(s)
<code>Output ?</code>	Return the current selection	A selection string

Abbreviation

o

Description

Selecting Outputs

The **Output** command selects one or more outputs in the system. Outputs are typically the hard-wired digital outputs on a CueServer or external I/O board. Use the **Output** command in conjunction with an action command like [On](#), [Off](#), or [At](#) to change the output state of one or more outputs. When used alone or in logic expressions, the **Output** command returns the current state of the specified output(s).

Either a single output number can be specified, or a range of outputs can be specified using the various [selection operators](#) like +, -, > and ~.

The wildcard character * can be used as the output number to mean *all* outputs for a particular station.

Working With Stations

When no station number specified, the default station is assumed. The [Station](#) command can be used to change the default station. Unless changed by the [Station](#) command, the default station is typically Station 0, which corresponds to the built-in outputs on the CueServer itself. When a station number is specified as part of the **Output** command, that station number will be used for the selection.

Determining Which Outputs Are Selected

The question mark `?` can be used to ask what the current selection is. A selection string will be returned, which will consist of a single number (like `3`) or a range (like `5>7+9`), or if no outputs are selected, `0` will be returned.

Examples

`Output 1`

Selects output 1. Future action commands will be directed towards output 1. Also returns `0`, or `1` to indicate if the output is currently off or on.

`Output 1>5 On`

Turns on outputs 1 thru 5.

`Output 1>3+5>8 Off`

Turns off outputs 1 through 3 and 5 through 8.

`Output 7 At 50`

Sets the level of output 7 to 50% (any non-zero level turns an output on).

`Output ?`

Returns the current output selection in the format of a single number like `3`, or a range like `5>7+9`.

See Also

- [Selection Operators](#)
- [At](#), [Off](#), [On](#)

Playback

Syntax

Command	Description	Return Value
<code>Playback <number> [<range...>]</code>	Change the active playback fader and/or select a range of playbacks	The new playback number
<code>Playback ?</code>	Return the currently active playback fader	The current playback number

Abbreviation

P

Description

Choosing The Active Playback


The **Playback** command changes the currently active playback fader and/or allows a playback fader's properties to be changed. Playback faders are the functional units in the DMX output stack that control the playback of cues, streams and maintain the fade progress and timing of linked cues. Each playback fader operates as an independent *layer* of control in the DMX output stack and has properties that control how each playback layer is merged with the preceding layer, and the overall intensity of the channels in the layer. Use the **Playback** command to change the active playback fader, or in conjunction with an action command like [At](#), [On](#), [Off](#) or [Set](#) to change the playback's submaster level, or to set the layer properties. When used alone or in logic expressions, the **Playback** command returns the currently active playback fader.


Selecting a Range of Playbacks

The **Playback** command works differently from other selection commands. When selecting a range of playbacks (for example using the command `Playback 3>5`), the first playback of the range becomes the "active" playback while all of the playbacks in the range become selected. This is for two reasons, (1) only one playback can be active at a time, but (2) the selected range can be used for working with playbacks with the [At](#), [Clear](#), [Enable](#), and [Disable](#) commands. For example, the command `Playback 1>7 Clear` will clear all seven playbacks, but only playback 1 will become "active".

Determining The Active Playback

The question mark `?` can be used to ask what the currently active playback is. A number will be returned, from `1` to `32` indicating which playback is currently active.

 Many commands operate on the currently active playback fader, such as **Channel**, **Clear**, **Cue**, **Fade**, **Follow**, **Go**, **Group**, **Link**, **Stack**, **Start** and **Stop**. Since each playback fader maintains its own set of DMX channels and properties, such as the current and next cue, fade and follow times, cue link, and more, it is important to make sure that you use the **Playback** command to specify which playback you are targeting when using the above commands.

 Although CueServer can have a maximum of 32 playback faders, your configuration may have fewer, depending on the combination of playbacks and DMX universes that you have chosen. If you select a playback that is not available, the subsequent commands sent to that playback will have no effect.

Examples

```
Playback 1
```

Makes playback 1 active. All future playback related commands will be directed to playback 1.

```
Playback 2 At 75
```

Makes playback 2 active and sets the playback's submaster to 75%.

```
Playback 3
```

```
Cue 1 Go
```

Makes playback 3 active, then executes cue 1 in playback 3.

```
Playback 4
```

```
set Playback.Mode "Override"
```

Makes playback 4 active, then sets the playback's layer mode to "override".

```
Playback 2>4 At 50
```

Makes playback 2 active and sets playback 2, 3, and 4's submaster to 50%.

```
Playback 4>8 Disable
```

Makes playback 4 active and disables playbacks 4 through 8.

See Also

- [At](#), [Off](#), [On](#), [Set](#)

Station

Syntax

Command	Description	Return Value
<code>Station <number></code>	Select one or more stations	The selected stations
<code>Station ?</code>	Return the current selection	The selected stations

Abbreviation

`STAT`

Description

When selecting Buttons, Contacts, Indicators, or Outputs on a connected station, the **Station** command can be used to specify a station number instead of specifying it as part of the Button, Contact, Indicator or Output number. The **Station** command actually sets the default station(s) to be used with any of the other selection commands when a station number is not used with the selection command. See the examples below for clarification.

Two Ways To Select Station Objects

There are two ways to specify a station object. The first is to use dotted-notation with the selection command. For example, to select Button 7 of Station 3, the command `Button 3.7` can be used. The the “dot” is used to separate the station number from the object number.

The **Station** command provides a second way to select station objects. Using this method, the **Station** command is used first to change which station (or stations) are the default, and then use the object command to select the individual object. For example, to select Button 7 of Station 3, the command `Station 3 Button 7` can be used.

Selecting The Same Object On Multiple Stations

The **Station** command allows the same object on multiple stations to be selected at the same time.

For example, to select Button 3 of Stations 1>10, the command `Station 1>10 Button 3` can be used.

Examples

```
Station 3 Contact 4
```

Sets Station 3 as the default station, and then selects Contact 4 on the default station (Station 3).

```
Station 4 Output 1>10
```

Sets Station 4 as the default station, and then selects Outputs 1 through 10 on the default station (Station 4).

```
Station 1>5 Button 8 Off
```

Sets Stations 1 through 5 as the default stations, and then selects Button 8 on the default stations (Stations 1>5), and then turns these button indicators off.

```
Station 4
```

```
Button 1 On
```

```
Button 2 Off
```

```
Button 3 On
```

Sets Station 4 as the default station, then turns Button 1 On, Button 2 Off and Button 3 On (all on Station 4).

See Also

- [Selection Operators](#)
- [Button](#), [Contact](#), [Indicator](#), [Output](#)

Universe

Syntax

Command	Description	Return Value
<code>Universe <number> [<range...>]</code>	Select one or more universes	<i>None</i>
<code>Universe ?</code>	Return the current selection	A selection string

Abbreviation

UNIV

Description

Selecting Universes

The **Universe** command selects one or more universes in the system. Universes are the logical blocks of 512 DMX channels that are sent and/or received by the CueServer across the Ethernet network. Use the **Universe** command in conjunction with an action command like [Enable](#), [Disable](#), or [Set](#) to enable/disable the universe or to change the universe's properties (such as its broadcast priority). When used alone or in logic expressions, the **Universe** command returns `0` or `1` to indicate if the universe is disabled or enabled.

Either a single universe number can be specified, or a range of universes can be specified using the various [selection operators](#) like `+`, `-`, `>` and `~`.

The wildcard character `*` can be used as the universe number to mean *all* universes.

Determining Which Universes Are Selected

The question mark `?` can be used to ask what the current selection is. A selection string will be returned, which will consist of a single number (like `3`) or a range (like `5>7+9`), or if no universes are selected, `0` will be returned.

Examples

```
Universe 1
```

Selects universe 1. Future action commands will be directed towards universe 1. Also returns `0`, or `1` to indicate if the universe is currently disabled or enabled.

```
Universe 3 Disable
```

Disables the transmission of universe 3.

```
Universe 1>3+5>8 Enable
```

Enables universes 1 through 3 and 5 through 8.

```
Universe 2
```

```
Set Universe.Priority 150
```

Sets the “priority” property of Universe 2 to 150.

See Also

- [Selection Operators](#)
- [Disable](#), [Enable](#), [Set](#)

Selection Operators (+, -, >, ~)

Most of the selection commands allow more than one object to be selected at once. To select more than one of a particular object, you can use the **Plus**, **Minus**, **Thru** and **Invert** operators.

These operators work on most of the selection commands, including:

- [Button](#)
- [Channel](#)
- [Contact](#)
- [Group](#)
- [Indicator](#)
- [Output](#)
- [Universe](#)

Plus (+)

Use the **Plus** (+) operator to add additional objects to your selection.

For example:

```
Channel 1+3+5+7
```

Selects channels 1, 3, 5 and 7.

```
Channel 1>10+20>30
```

Selects channels 1 thru 10 and 20 thru 30.

```
Group 1+5
```

Selects the channels in Group 1 and Group 5.

Minus (-)

Use the **Minus** (-) operator to remove objects from your selection.

For example:

```
Channel 1>10-5
```

Selects channels 1 thru 10, except for channel 5 (or, in other words, channels 1 thru 4 and 6 thru 10).

```
Group 1-3
```

Selects the channels in Group 1 that are not in Group 3.

Thru (>)

Use the **Thru (>)** operator to add a range of objects to your selection.

For example:

```
Channel 1>10
```

Selects channels 1 thru 10.

```
Channel 1>50-20>30
```

Selects channels 1 thru 50, except for channels 20 thru 30 (or, in other words, channels 1 thru 19 and 31 thru 50).

Invert (~)

Use the **Invert (~)** operator to invert which objects are selected.

For example:

```
Button 3
```

```
On
```

```
~
```

```
Off
```

Selects button 3, then turns it's indicator on, then inverts the selection (selecting all buttons except for button 3), then turns those indicators off.

Using Wildcards

When selecting objects, a *wildcard* operator is available as a shortcut for selecting *all* objects of a particular type.

The wildcard operator is an asterisk character (*).

This character can be inserted in most places that a selection range is required, which means to select *all* of a particular object.

The following table shows how the wildcard operator can be used:

Command	Result
Channel * Release	Releases all channels in the selected playback
Button * On	Turns on all button indicators on the default station
Button 3.* Off	Turns off all button indicators on Station 3
Output * Off	Turns off all outputs on the default station
Universe * Enable	Enabled all universe outputs

Action Commands

Action commands perform actions. Some action commands operate on the current selection (as set by the Selection Commands), and some action commands perform a global action that does not depend on selected objects.

For example, the **At** command operates on selected channels, buttons, playbacks, outputs and more. In order to properly use the **At** command, one of these objects must be selected first. The following examples show some of the proper uses of **At**:

- `Channel 1 At 75`
- `Button 1>8 At 0`
- `Playback 3 At FL`
- `Group 1+3+5+7 At 95`

Other action commands, such as the **Audio** command do not depend on other objects being selected first. The following examples show how the **Audio** command can be used to start and stop playing sounds.

- `Audio "Chime.wav"`
- `Audio "Breakbeat.mp3"`
- `Audio Stop`

All of the available action commands are detailed in the following sections.

Assign (=)

Syntax

Command	Description	Return Value
<code><variable> = <value></code>	Sets the value of the variable	The value the variable was set to

- `<variable>`
 - A user variable or system variable name. Must be enclosed in double quotes.
- `<value>`
 - A string (a combination of characters enclosed in quotes, such as “Hello World”).
 - A number (a whole number, or a decimal number, such as 123 or 12.7).

Alternate Syntax

See: [Set](#) command.

Description

Setting Values

The **Assign** command sets the value of a variable. The variable can be user defined (such as *xyz*, *LoopCount*, or *IsMyShowEnabled*), or it may be a system variable (such as *button.onColor* or *lcd.backlight*). System variables always contain a “dot” character (.). User variables must not contain a “dot” character, otherwise, they will be interpreted as a system variable, and they will not be stored properly.

User variables can be defined on the fly, simply by assigning a value to a variable. There is no need to pre-define variables.

See the [Assign](#) command for an alternate syntax for assigning variable values.

See the [Variables](#) section for how to use variables in the script language.

See the [System Variables](#) section for a complete list of available system variables.

Examples

```
"x"=3
```

Sets the variable *x* to the number 3.

```
"text"="Hello World"
```

Sets the variable *text* to the string *Hello World*.

```
"lcd.backlight"=25
```

Sets the system variable *lcd.backlight* to 25%.

```
"y"=('x' + 1)
```

Sets the variable *y* to the result of the expression *'x' + 1*.

See Also

- [Set, System Variables](#)

Audio

Syntax

Command	Description	Return Value
<code>Audio "<filename>"</code>	Plays the given sound file	The file name being played
<code>Audio Stop</code>	Stops playing sound immediately	Always returns 0

- `<filename>`
 - The file name of a sound resource loaded into *Sounds*
 - Sound formats recognized include: .aif, .mp3, .ogg, .snd, and .wav

Abbreviation

None

Description

Playing Sounds

The **Audio** command will play a given sound resource file to the Audio Output jack. The sound plays asynchronously, meaning that the command returns immediately while the sound plays in the background until it is finished, or it is interrupted by the **Audio Stop** command.

If a sound was already playing, issuing another **Audio** command will immediately stop playing the previous sound and begin playing the new sound.

Stopping Sounds

The **Audio Stop** command will immediately stop any sound that is currently playing.

Examples

```
Audio "Sound Effect.wav"
```

Begins playing the *Sound Effect.wav* sound resource file.

Audio "Background Music.mp3"

Begins playing the *Background Music.mp3* sound resource file.

Audio Stop

Stops playing sound immediately.

At

Syntax

Command	Description	Return Value
<code>At <value></code>	Set the value of the selected object(s)	The value the object(s) were set to
<code>At Cue <cue></code>	Sets the selected channels to the values in Cue <i>cue</i>	The number of channels set
<code>At Playback <playback></code>	Sets the selected channels to the values in Playback <i>playback</i>	The number of channels set
<code>At Input</code>	Sets the selected channels to the values currently coming from the DMX input	The number of channels set
<code>At Output</code>	Sets the selected channels to the values currently being sent to the DMX output	The number of channels set
<code>At [+/-]<value></code>	Set the value of the selected object(s) to an offset (or delta) from the object(s) current value	The value the object(s) were set to
<code>At ?</code>	Get the value of the selected object(s)	The current value of the selected object(s)

- `<value>`
 - A percentage from 0 to 100. When specifying percentages, the value can optionally be followed by the % sign.
 - A decimal number from #0 to #255. When specifying decimal numbers, the value must be preceded with a # sign.
 - A hexadecimal number from \$00 to \$FF. When specifying hexadecimal numbers, the value must be preceded with a \$ sign.
 - FL (Full) or On can be used as a shortcut that means 100%
 - Off can be used as a shortcut that means 0%
 - A sign + or - may appear before the value to specify an offset (or delta) from the current value.
- `<cue>`
 - Any whole number from 0 to 99999
 - May optionally contain decimal numbers from .00 to .99
- `<playback>`
 - Any whole number from 1 to 32

Abbreviation

A or @

Description

Setting Values

The **At** command sets the currently selected object(s) values. The **At** command can be used with many types of objects, including **Buttons**, **Channels**, **Groups**, **Outputs**, **Playbacks**, and **Presets**.

The following table shows how the **At** command affects each of these object types:

Object	Result Of At Command
Buttons	Turns the button(s) indicator on or off or sets it to a special user value
Channels	The channel(s) are set to the specified value
Groups	The channels in the group are set to the specified value
Outputs	Turns the output(s) on or off
Playbacks	The playback(s) submasters are set to the specified value
Presets	The preset is activated with an intensity of the the specified value

Setting Values With Timing

A playback fader's **Time** parameter will cause the value of **Channels**, **Groups** and **Playbacks** to fade to the desired value. A time of 0 (zero) causes the value to be set immediately. Any non-zero time will cause the value to gradually change at a speed that will cause it to reach the desired value in the number of seconds set by the [Time](#) command.

Recalling Values From A Cue

Using the **At Cue** command allows the data from a given Cue to be used to set the currently selected channel values (instead of specifying a single fixed value). This is useful to only recall parts of a Cue without affecting other channels.

Recalling Values From A Playback

Using the **At Playback** command allows the channels from a given Playback to be used to set the currently selected channel values (instead of specifying a single fixed value). This is useful to copy components of a scene from one playback fader to another.

Pulling Values From the DMX Input

Using the **At Input** command allows the channels from the DMX Input to be used to set the currently selected channel values (instead of specifying a single fixed value). This is useful to copy (or snapshot) the DMX Input into a playback fader.

Pulling Values From the DMX Output

Using the **At Output** command allows the channels from the DMX Output to be used to set the currently selected channel values (instead of specifying a single fixed value). This is useful to copy (or snapshot) the DMX Output into a playback fader.

Using Delta Values

Using the **At +/-** command allows channels to be set to an offset (or delta) from their current value(s). This is useful if the current channel value is not known and an offset value need to be added or subtracted from the current value. For example, an offset can be used to bump a channel value up or down using commands such as `Channel 1 At +5` or `Group 7 At -25`.

Setting Indicators

Using the **At** command with buttons will turn a button's LED indicator on or off. Use `At 0` or `OFF` to turn an indicator off. Use `At FL`, `At 100` or `ON` to turn an indicator on. Values of `1`, `2`, `3`, and `4` will set an indicator to one of the "user colors".

Setting Outputs

Using the **At** command with outputs will turn an output on or off. Use `At 0` or `OFF` to turn an output off. Use `At FL`, `At 100` or `ON` to turn an output on.

Examples

```
Channel 1 At 33
```

Sets the value of channel 1 to 33%.

```
Channel 1>3+5>8 On
```

Sets channels 1 through 3 and 5 through 8 to 100%.

```
Channel 1>10
```

```
Time 5
```

```
At FL
```

Selects channels 1 through 10, then changes the fade time to 5 seconds, then begins fading the channels to 100% (Full).

```
Group 5 On
```

Sets the channels in group 5 to 100%.

```
Channel 100>200 At Cue 44
```

Sets channels 100 through 200 to the channel levels recorded in Cue 44.

```
Group 7 At Playback 8
```

Sets the channels in Group 7 to the channel levels currently in Playback 8.

```
Channel 1>10 At +15
```

Increases the value of Channels 1 thru 10 by 15%.

```
Preset 1 At 33
```

Activates preset 1 with intensity level 33%.

```
Button 1 At FL
```

Turns the indicator for button 1 on.

```
Button 2 At 3
```

Turns the indicator for button 2 to the color specified as "User 3".

```
Output 3 At 0
```

Turns output 3 off.

See Also

- [Button](#), [Contact](#), [Group](#), [Off](#), [On](#), [Output](#), [Playback](#), [Preset](#)

Clear

Syntax

Command	Description	Return Value
<code>Clear</code>	Clears the selected playback fader(s)	The playback(s) cleared

Abbreviation

`CL`

Description

The **Clear** command clears the selected playback fader(s).

Clearing a playback fader has the following effect:

- Releases all DMX channels (including parked channels)
- Removes the current selection
- Sets the current and next cue to *none*
- Aborts any fade or follow timers in progress
- Zeros the current fade, follow and link properties
- Returns the playback's submaster to 100%

! The **Clear** command also clears parked channels. To clear channels without clearing parked channels, use the **Release** command instead.

Examples

```
Playback 1 Clear
```

Clears playback 1.

```
Playback 4>8 Clear
```

Clears playbacks 4 through 8.

See Also

- [Playback](#), [Release](#)

Cue

Syntax

Command	Description	Return Value
<code>Cue <cue number></code>	Sets the active playback fader's next cue	The cue number set
<code>Cue ?</code>	Returns the current cue in the active playback fader	The current cue number

- `<cue number>`
 - Any whole number from 0 to 99999
 - May optionally contain decimal numbers from .00 to .99

Abbreviation

Q or CU

Description

Setting The Next Cue

Use the **Cue** command to set the *next cue* in the active playback fader. Whenever the playback fader receives a **Go** command, it will advance to this next cue. The **Cue** command is frequently used in conjunction with the **Go** command. For instance, the commands `Cue 1 Go` are typically used together, even though they are two distinct commands. The first command, **Cue 1** sets which cue is “next”, and then the **Go** proceeds to execute it.

Setting The Next Cue With Overrides

When the **Cue** command is executed, not only is the next cue number placed into the playback fader, but the cue's fade and follow times and link cue are also loaded into the playback. This allows for manually overriding the timing or link before the **Go** command is executed. For instance, the commands `Cue 1 Fade 5 Follow 10 Go` would first load cue 1 as the next cue for the playback, then the fade time would be changed to 5 seconds, then the follow time would be changed to 10 seconds, and then the cue would be executed with the new timing substituted into place of the default values for the cue.

Working With Cue Stacks

By default, all cues are loaded from the main cue list. Additionally, the show file may contain one or more *cue stacks*. The [Stack](#) command is used to change which cue stack the playback fader is using. Once the stack has been changed on a playback fader, all cues on that playback will be loaded from that cue stack.

Determining The Current Cue

Use the **Cue** command with the question mark (?) to return the current cue number of the active playback fader. For instance, if playback 3 currently executing cue 7, then executing the commands

```
Playback 3 Cue ?
```

 will return 7.

If a playback isn't loaded with a cue, the return value will be negative (less than zero).

Examples

```
Cue 1
```

Sets cue 1 as the next cue in the active playback fader.

```
Cue 7 Go
```

Executes cue 7 in the active playback fader by first loading cue 7 then executing it.

```
Playback 3 Cue 100.5 Go
```

Executes cue 100.5 in playback 3.

```
Cue 999.99 Fade 5 Go
```

Loads cue 999.99, then overrides the fade time to 5 seconds, then executes it.

```
Playback 5
```

```
Stack "Intro"
```

```
Cue 1 Go
```

Sets playback 5 as the active playback, then switches the playback to use the stack named "Intro", then executes Cue 1 from the "Intro" stack.

See Also

- [Fade](#), [Follow](#), [Go](#), [Link](#), [Playback](#), [Stack](#)

Disable

Syntax

Command	Description	Return Value
<code>Disable</code>	Disables the selected object(s)	The number of objects disabled

Abbreviation

`DIS`

Description

The **Disable** command disables the currently selected object(s). The **Disable** command can be used with many types of objects, including **Buttons**, **Channels**, **Contacts**, **Groups**, **Playbacks**, **Stations**, and **Universes**. **Disable** has the opposite effect as **Enable**.

The following table shows the various effect of enabling or disabling an object:

Object	When Enabled	When Disabled
Buttons	Responds to presses normally	Does not trigger any actions
Channels	Channel(s) contribute to playback	Channel(s) do not contribute to playback
Contacts	Responds to closures normally	Does not trigger any actions
Groups	Channel(s) contribute to playback	Channel(s) do not contribute to playback
Playbacks	Playback(s) contribute to output	Playback(s) do not contribute to output
Stations	Station(s) operate normally	Station(s) do not trigger any actions
Universes	Normal broadcast from universe	No broadcast from universe



*Disabling or Locking a Button, Contact or Station both prevent events from being triggered when physically operated, however a *Disabled* Button, Contact or Station still shows its normal indicator states, where a *Locked* object shows its *Locked* indicator state.*

Examples

`Button 1 Disable`

Disables button 1.

`Channel 1>10 Disable`

Disables channels 1 thru 10.

`Playback 3 Disable`

Disables playback 3.

`Station 7 Disable`

Disables station 7.

`Universe 1+7 Disable`

Disables universes 1 and 7.

See Also

[Button](#), [Channel](#), [Contact](#), [Group](#), [Playback](#), [Station](#), [Universe](#)

Enable

Syntax

Command	Description	Return Value
<code>Enable</code>	Enables the selected object(s)	The number of objects enabled

Abbreviation

`ENA`

Description

The **Enable** command enables the currently selected object(s). The **Enable** command can be used with many types of objects, including **Buttons**, **Channels**, **Contacts**, **Groups**, **Playbacks**, **Stations**, and **Universes**. **Enable** has the opposite effect as **Disable**.

The following table shows the various effect of enabling or disabling an object:

Object	When Enabled	When Disabled
Buttons	Responds to presses normally	Does not trigger any actions
Channels	Channel(s) contribute to playback	Channel(s) do not contribute to playback
Contacts	Responds to closures normally	Does not trigger any actions
Groups	Channel(s) contribute to playback	Channel(s) do not contribute to playback
Playbacks	Playback(s) contribute to output	Playback(s) do not contribute to output
Stations	Station(s) operate normally	Station(s) do not trigger any actions
Universes	Normal broadcast from universe	No broadcast from universe



*Disabling or Locking a Button, Contact or Station both prevent events from being triggered when physically operated, however a **Disabled** Button, Contact or Station still shows its normal indicator states, where a **Locked** object shows its **Locked** indicator state.*

Examples

`Button 1 Enable`

Enables button 1.

`Channel 1>10 Enable`

Enables channels 1 thru 10.

`Playback 3 Enable`

Enables playback 3.

`Station 7 Enable`

Enables station 7.

`Universe 1+7 Enable`

Enables universes 1 and 7.

See Also

[Button](#), [Channel](#), [Contact](#), [Group](#), [Playback](#), [Station](#), [Universe](#)

Fade

Syntax

Command	Description	Return Value
<code>Fade <cue fade time></code>	Sets the active playback fader's cue fade time	The cue fade time set
<code>Fade ?</code>	Returns the current cue fade time of the active playback fader	The current cue fade time

- `<cue fade time>`
 - A decimal number of seconds (optionally using decimal digits for fractions of seconds)
 - `0` means no fade time (or the channels are set immediately without fading)
 - Optionally may include a slash (`/`) which indicates a split (up/down) fade
 - Optionally may include a dash (`-`) which indicates a delayed fade

Abbreviation

`FA`

Description

Setting The Cue Fade Time

Use the **Fade** command to set the *cue fade time* for the active playback fader. This time is used to crossfade the channels of the next cue whenever the **Go** command is executed. The cue fade time is automatically set by cues being loaded into the playback fader, but the **Fade** command can be used to override the cue fade time.

Using Split Fade Times

A *split fade time* is used when it is desired to have channels that are fading up occur at a different rate than channels fading down. To specify a split fade time, use a slash character in between two fade times. For instance, the command `Fade 3.5/7.5` will cause any channel that is fading up to occur in 3.5 seconds, and any channel that is fading down to occur in 7.5 seconds.

Using Fade Delays

Normally, whenever a cue is executed, the fade begins immediately. A delay can be inserted that would cause the fade to be delayed before starting to change value. To specify a fade delay, use a delay time and dash character before the fade time. For instance, the command `Fade 5.5-10` will cause the fade to delay 5.5 seconds before beginning a 10 second fade.

Using Both Fade Delays And Split Fade Timing

Both fade delays and split fades can be combined. For instance, the command `Fade 1-2/3-4` would cause any channels fading up to be delayed 1 second before fading over 2 seconds, while the downward fading channels would be delayed 3 seconds before fading over 4 seconds.

Determining The Current Cue Fade Time

Use the **Fade** command with the question mark (?) to return the current cue fade time. A cue fade time such as `7.21` or `12/3` will be returned.



Note that the *Cue Fade Time* is different from the *Global Fade Time*. The cue fade time effects the **Go** command. The global fade time effects the **At** command. The cue fade time is set with the **Fade** command.

Examples

```
Fade 1
```

Sets the cue fade time to 1 second.

```
Fade 1.35/7.2
```

Sets the cue fade time to 1.35 seconds for upward fading channels and 7.2 seconds for downward fading channels.

```
Cue 22 Fade 5 Go
```

Loads cue 22, then overrides it's fade time to 5 seconds before executing it.

See Also

- [Cue, Go, Time](#)

Follow

Syntax

Command	Description	Return Value
<code>Follow <cue follow time></code>	Sets the active playback fader's cue follow time	The cue follow time set
<code>Follow Clear</code>	Clears the active playback fader's cue follow time	<i>None</i>
<code>Follow ?</code>	Returns the current cue follow time of the active playback fader	The current cue follow time

- `<cue follow time>`
 - A decimal number of seconds (optionally using decimal digits for fractions of seconds)
 - `0` means no follow time

Abbreviation

`FO`

Description

Setting The Cue Follow Time

Use the **Follow** command to set the *cue follow time* for the active playback fader. Whenever a **Go** occurs, this time is used to start a timer that will automatically execute another **Go** as soon as the timer expires. The cue follow time is automatically set by cues being loaded into the playback fader, but the **Follow** command can be used to override the cue follow time.

Clearing The Cue Follow Time

Sometimes, it may be useful to cancel the follow timer in a playback fader. Use the **Follow Clear** command to clear any currently running follow timer in the active playback fader.

Determining The Current Cue Follow Time

Use the **Follow** command with the question mark (?) to return the current cue follow time. A cue follow time such as `1` or `7.21` will be returned.

Examples

`Follow 1`

Sets the cue follow time to 1 second.

`Cue 22 Follow 5 Go`

Loads cue 22, then overrides it's follow time to 5 seconds before executing it.

`Follow Clear`

Clears any currently running follow timer in the active playback fader.

See Also

- [Cue](#), [Go](#)

Go

Syntax

Command	Description	Return Value
Go	Executes the next cue in the active playback fader	The cue number executed

Abbreviation

G

Description

Going To The Next Cue

Use the **Go** command to execute the *next cue* in the active playback fader. The next cue is typically the cue with the next numerically higher number in the cue list, but the next cue can be overridden by an optional link in the cue or executing the [Cue](#) command.

After each **Go** occurs, the properties of the next cue are loaded into the playback fader. These properties include the next cue's fade and follow times and the cue's link.

Timing For Normal Cues

When the next cue is executed with the **Go** command, the playback's Fade time is used to crossfade to the channels recorded in the cue. The playback's Follow time is used to start a timer that, when expired, will automatically "follow" to the next cue by automatically executing another **Go**.

It is important to note that a cue's Fade and Follow times are started at the same time. For instance, if a cue has a fade of 3 seconds and a follow of 4 seconds, then the fade will complete 3 seconds after the cue started, and the follow will occur 4 seconds after the cue started (or 1 second after the fade completes). This means that if the follow time is shorter than the fade time, the fade will not fully complete before the follow occurs.

Timing For Streaming Cues

Streaming cues do not have a fade time, but they do use a follow time. If a follow time is specified, the stream will only play until the follow time is reached and then will automatically “follow” to the next cue by automatically executing another **Go**.

Streams have various playback modes that affects what action is taken when the end of the stream is reached. These modes include Follow, Loop, Hold and Blackout. Please refer to the section about [streaming cues](#) for more details.

Links

If a cue has an optional link, then when it is loaded into a playback fader, the playback's Link property is set. Whenever a **Go** occurs on that playback, if the playback has a link set, then instead of advancing to the next sequential cue, the linked cue will be loaded.

Examples

`Go`

Advances to the next cue in the active playback fader.

`Cue 7 Go`

Executes cue 7 in the active playback fader by first loading cue 7 then executing it.

`Playback 3 Cue 100.5 Go`

Executes cue 100.5 in playback 3.

See Also

- [Cue](#), [Fade](#), [Follow](#), [Link](#), [Playback](#), [Stack](#)

Input

Syntax

Command	Description	Return Value
<code>Input Disable</code>	Disables the DMX Input into the Playback Faders	0
<code>Input Enable</code>	Enables the DMX Input into the Playback Faders	1
<code>Input ?</code>	Returns the current enable state of the DMX Input	0 or 1

Abbreviation

IN

Description

The **Input** command is used to either enable or disable the DMX Input layer into the Playback Fader stack. By disabling the DMX Input, no incoming DMX channels from Ethernet or hardwired DMX will flow into the Playback Fader stack. Use the **Input Disable** command to ignore DMX Input. Use the **Input Enable** command to resume the reception of DMX Input.

Use the **Input ?** command to return the current enable state of DMX Input.

Examples

```
Input Disable
```

Disables the DMX Input into the Playback Fader stack.

```
Input Enable
```

Resumes normal DMX Input into the Playback Fader stack.

```
Input ?
```

Returns either 0 or 1, indicating if the DMX Input is currently disabled or enabled.

Join

Syntax

Command	Description	Return Value
<code>Join <join group></code>	Sets the join group of the current zone	The join group number
<code>Join Clear</code>	Sets the join group of the current zone to 0	0
<code>Join ?</code>	Returns the current zone's join group	The join group number

- `<join group>`
 - A numeric ID from 0 through 32.

Abbreviation

None

Description

Use the **Join** command to set the *Join Group Number* for the selected zone.

Each Zone can be assigned a Join Group. By default each Zone starts in Group 0 (zero). When a Zone is a member of a non-zero Join Group, that zone becomes logically “joined” with all other Zones that are members of the same group.

When Zones are joined together by having the same Join Group Number, the channels of all Zones in the group are merged together. Also, any Preset number that is activated a Zone will be also activated in all joined zones.

For example, a project has three zones named “Ballroom A”, “Ballroom B”, and “Ballroom C”. Each of these zones has eight presets numbered 1 through 8. The channels in each zone are 1>10, 11>20, and 21>30, respectively.

If a user activates any preset in any of the three zones, that preset only operates in that zone. It is assumed that removable walls are separating each of these zones.

Next, the wall that separates Ballroom B and C are removed. The desired effect is to “join” these two rooms. To perform this connection between the two zones, the *Join Group Number* of each of these two zones must match. The following commands will set both zone’s group number to 7:

```
Zone "Ballroom B" Join 7
Zone "Ballroom C" Join 7
```

Once both of these zones have the same *Join Group Number*, any action taken in one zone will be mirrored in the other zones. For instance, if the following command is executed:

```
Zone "Ballroom B" Preset 3 On
```

Preset 3 will become activated in the Ballroom B Zone. Additionally, Ballroom C’s Preset 3 will also be activated.

Join Group “7” was arbitrarily chosen for this example. Any Join Group from 1 through 32 could have been used. It is only important that the *_same** Join Group number be used for each zone that should be joined together.

The concept of the Join Group is flexible enough to create scenarios where very complex and not-necessarily physically connected spaces in the project can be joined together. And because there are 32 possible Join Groups to use, a large number of different logical blocks of zones can be joined together.

When a zone should no longer be joined with any other zones, use the command **Join 0**, or **Join Clear** to set the zone’s Join Group to zero.



Please note that Join Group numbers are *not* related to channel Groups.

Examples

```
Join 3
```

Sets the currently selected zone’s Join Group to 3.

```
Zone "Foyer" Join 5
```

Sets the Join Group for the Foyer zone to 5.

```
Zone "Happy" Join Clear
```

Clears (sets to zero) the Join Group for the Happy zone.

Join ?

Returns the currently selected zone's Join Group.

See Also

- [Preset, Zone](#)

Length

Syntax

Command	Description	Return Value
<code>Length <playback time></code>	Sets the current streaming cue's playback length	The playback length set
<code>Length ?</code>	Returns the current streaming cue's playback length	The current playback length

- `<playback time>`
 - A decimal number of seconds (optionally using decimal digits for fractions of seconds)
 - `0` means not to override the cue's playback length (this is the default)
 - A whole number preceded by a `#` specifies a number of frames instead of seconds

Abbreviation

`LEN`

Description

Setting a Streaming Cue's Playback Length

Use the **Length** command to set the *playback length* for playing back a streaming cue. This time is used to adjust the duration of a streaming cue before it begins playing back. For instance, if a stream is recorded too long, a shorter playback length can be given to have the stream stop playing at the proper place in the cue. The **Length** command must be issued after the cue is selected but before the corresponding **Go** command is issued. For example, **Cue 1 Length 5.5 Go**.

If a playback length is specified that is shorter than the recorded length of the cue, then the cue will stop (or loop, or follow, etc.) at the specified time, which will be before the recorded data in the cue is finished playing back. If a playback length is specified that is longer than the recorded length of the cue, then there will be a delay after the end of the recorded data before the cue will stop (or loop, or follow, etc.).

Playback times are normally expressed in seconds. By using a pound-sign (`#`) before the value allows the playback time to be expressed in *frames*. A frame is 1/40th of a second and is the smallest unit of time that streaming cues are recorded in.



The **Length** command is frequently used in conjunction with the **offset** command, which is used to adjust the starting position of streaming cue playback independently from the actual recorded starting point of the cue.

Determining The Current Playback Time

Use the **Length** command with the question mark (?) to return the current playback time. A playback time such as 5 or 1.025 will be returned.

Examples

```
Length 1
```

Sets the current streaming cue's playback time to 1 second.

```
Cue 4 Length 8.075 Go
```

Loads cue 4, then sets its playback time to 8.075 seconds, and then begins playing the cue.

```
Cue 17 Offset #1 Length 3.5 Go
```

Loads cue 17, then sets its offset time to 1 frame (0.025 seconds) seconds, then sets the playback length to 2.5 seconds, and then begins playing the cue.

See Also

- [Cue](#), [Go](#), [Offset](#)

Link

Syntax

Command	Description	Return Value
<code>Link <cue number></code>	Sets the active playback fader's linked cue number	The linked cue
<code>Link Clear</code>	Clears the active playback fader's linked cue	<i>None</i>
<code>Link ?</code>	Returns the current linked cue of the active playback fader	The current linked cue

- `<cue number>`
 - Any whole number from 0 to 99999
 - May optionally contain decimal numbers from .00 to .99

Abbreviation

L

Description

Setting The Linked Cue

Use the **Link** command to set the *linked cue* for the active playback fader. Whenever a **Go** occurs, this link is used to override the normal sequential execution of cues. If no link is set, cues execute in numerical order. If the link is set to a cue, then this cue will become the next cue after the **Go**.

Clearing The Linked Cue

Use the **Link Clear** command to clear any linked cue in the active playback fader. Without a linked cue, future **Go** commands will execute cues in numerical order.

Determining The Current Linked Cue

Use the **Link** command with the question mark (?) to return the current linked cue. A cue number such as 1 or 100.5 will be returned. If no cue is linked, -1 is returned.

Examples

Link 1

Sets the linked cue to cue 1.

Cue 22 Link 1 Go

Loads cue 22, then overrides it's linked cue to cue 1 before executing it.

Link Clear

Clears any currently linked cue from the active playback fader.

See Also

- [Cue, Go](#)

Lock

Syntax

Command	Description	Return Value
Lock	Locks the selected object(s)	The number of objects locked

Abbreviation

None

Description

The **Lock** command *locks* the currently selected object(s). The **Lock** command can be used with **Buttons**, **Contacts**, and **Stations**. When an object is locked, it does not trigger its events when operated and if it has an indicator, it appears in the “locked” state. **Unlock** has the opposite effect as **Lock**.

The following table shows the various effect of locking or unlocking an object:

Object	When Unlocked	When Locked
Buttons	Responds to presses normally, displays normal indicator states	Does not trigger any actions, displays <i>locked</i> indicator state
Contacts	Responds to closures normally	Does not trigger any actions
Stations	Station operates normally, all controls display normal indicator states	Station is locked, all controls display <i>locked</i> indicator states



Disabling or Locking a Button, Contact or Station both prevent events from being triggered when physically operated, however a Disabled Button, Contact or Station still shows its normal indicator states, where a Locked object shows its Locked indicator state.

Examples

```
Button 1 Lock
```

Locks button 1.

Button 3>5 Lock

Locks buttons 3 through 5.

Station 7 Lock

Locks station 7.

See Also

[Button](#), [Contact](#), [Station](#), [Unlock](#)

Log

Syntax

Command	Description	Return Value
<code>Log <string></code>	Writes a message to the system log	The message is returned
<code>Log Clear</code>	Clears the new message indicator	The number of messages cleared
<code>Log ?</code>	Returns the current new message count	The number of new messages pending

Abbreviation

None

Description

The **Log** command writes a message to the system log. When new messages are written to the system log, the device's power/status LED will blink with a magenta color.

Using **Log Clear** will acknowledge new messages by clearing the new message indicator.

Using **Log ?** will return the number of new messages in the system log.

Examples

```
Log "This is a test"
```

Writes the string "This is a test" to the system log. The power/status indicator will begin to blink, indicating that new messages have been added to the system log.

```
Log Clear
```

Clears the "new message" indicator.

```
Log ?
```

Returns the number of new messages in the system log.

Macro

Syntax

Command	Description	Return Value
<code>Macro <number></code>	Executes the CueScript commands stored in the specified macro	The result of the last command in the macro

Abbreviation

M

Description

The **Macro** command executes the CueScript instructions stored in a macro. A macro is a single command that expands automatically into a set of commands to perform a particular task. Macros are defined within CueServer Studio. When the macro command is executed, all of the commands defined in the macro are executed in it's place.

For instance, if Macro 1 is defined to include the commands `Time 5 At 100`, then from somewhere else the command `Channel 7 Macro 1` were executed, the result would be to select channel 7, then change the fade time to 5 seconds and set channel 7's level to 100%.

Macros can contain an arbitrary number of CueScript commands, and may even call upon other macros. When macros call upon other macros, this is called "nesting".

! Take care to not create infinite loops by having one macro call upon another macro, which in turn calls upon the first macro. This will create an "infinite loop".

! A common mistake is to use the **Go** command in conjunction with the **Macro** command (for example: `Macro 1 Go`). The **Macro** command does not need **Go** in order to function. The CueServer will interpret `Macro 1 Go` as two separate commands, the first will be to execute Macro 1, the second will be to make the active playback fader step to the next cue, which is a valid combination of commands, but usually not intended.

Examples

Macro 3

Executes the CueScript commands stored in Macro 3.

Off

Syntax


Command	Description	Return Value
<code>Off</code>	Turn an object off	0

Abbreviation

None

Description

The `Off` command sets the currently selected object(s) values to the minimum. In other words, it turns the object(s) “off”.

 Note that `Off` is simply an alias for the command `At 0`.

Examples

```
Button 1>5 Off
```

Turns the LED indicators of buttons 1 thru 5 off.

```
Channel 1>3+5>8 Off
```

Sets channels 1 through 3 and 5 through 8 to 0%.

```
Group 5 Off
```

Sets the channels in group 5 to 0%.

See Also

[Button](#), [Contact](#), [Group](#), [Output](#), [Playback](#)

Offset

Syntax

Command	Description	Return Value
<code>Offset <offset time></code>	Sets the current streaming cue's starting offset	The offset time set
<code>Offset ?</code>	Returns the current streaming cue's starting offset	The current offset time

- `<offset time>`
 - A decimal number of seconds (optionally using decimal digits for fractions of seconds)
 - `0` means no offset time (this is the default)
 - A whole number preceded by a `#` specifies a number of frames instead of seconds

Abbreviation

None

Description

Setting a Streaming Cue's Offset Time

Use the **Offset** command to set the *offset time* for playing back a streaming cue. This time is used to adjust the starting point of a streaming cue before it begins playing back. For instance, if a stream is recorded one second too early, an offset time of one second can be given to have the stream start playing at the proper place in the cue. The **Offset** command must be issued after the cue is selected but before the corresponding **Go** command is issued. For example, **Cue 1 Offset 1.3 Go**.

Offset times may be either positive or negative. Positive offset times move the playback pointer into the streaming cue data, starting the stream already somewhat into the recorded data. Negative offset times move the playback pointer to a time *before* the beginning of the streaming cue data, causing a delay before the recorded data plays back.

Offset times are normally expressed in seconds. By using a pound-sign (`#`) before the value allows the offset time to be expressed in *frames*. A frame is 1/40th of a second and is the smallest unit of time that streaming cues are recorded in.

* The **Offset** command is frequently used in conjunction with the **Length** command, which is used to adjust the playback length of a streaming cue independently from the actual recorded length of the cue.

Determining The Current Offset Time

Use the **Offset** command with the question mark (?) to return the current offset time. An offset time such as 5 or 1.025 will be returned.

Examples

`Offset 1`

Sets the current streaming cue's offset time to 1 second.

`Cue 4 Offset 0.075 Go`

Loads cue 4, then sets its offset time to 0.075 seconds, and then begins playing the cue.

`Cue 17 Offset #1 Length 3.5 Go`

Loads cue 17, then sets its offset time to 1 frame (0.025 seconds) seconds, then sets the playback length to 2.5 seconds, and then begins playing the cue.

See Also

- [Cue](#), [Go](#), [Length](#)

On

Syntax

Command	Description	Return Value
<code>On</code>	Turn an object on	255

Abbreviation

None

Description

The `On` command sets the currently selected object(s) values to the maximum. In other words, it turns the object(s) “on”.



Note that `On` is simply an alias for the command `At 100` or `At FL`.

Examples

```
Button 1>5 On
```

Turns the LED indicators of buttons 1 thru 5 on.

```
Channel 1>3+5>8 On
```

Sets channels 1 through 3 and 5 through 8 to 100%.

```
Group 5 On
```

Sets the channels in group 5 to 100%.

See Also

[Button](#), [Contact](#), [Group](#), [Output](#), [Playback](#)

Park

Syntax

Command	Description	Return Value
<code>Park</code>	Parks the selected channel(s)	<i>None</i>

Abbreviation

None

Description

The **Park** command locks the current value of the selected channel(s) in the active playback fader. Parked channels cannot be changed by running cues or using the **At** command. Use the **Park** command to freeze one or more channels so future cues and/or commands have no effect on the channel level.

The **Unpark** command has the opposite effect as the **Park** command.



Channels are parked in individual playback faders (not globally). Beware that if a channel is parked in one playback, it is still possible for another playback to cause the output of that channel to change.

Visual Representation

In CueServer Studio, parked channels appear in the stage view with their channel numbers in red. Note that you can only see parked channels if the Stage View is set to display a specific playback fader. You will not see parked channels in either the input or output stage views.

1	2	3	4	5	6	7	8	9	10
-	-	-	75	75	75	-	-	-	-
19	20	21	22	23	24	25	26	27	28
-	-	-	-	-	-	-	-	-	-

Examples

`Park`

Parks the currently selected channels in the currently active playback fader.

Channel 1>3+5>8 Park

Parks channels 1 through 3 and 5 through 8 in the currently active playback fader.

Playback 3 Channel * Park

Parks all channels in playback 3.

See Also

- [At](#), [Channel](#), [Playback](#), [Unpark](#)

Preset

Syntax

Command	Description	Return Value
<code>Preset <preset number> On</code>	Activates the specified preset in the current zone	The preset number activated
<code>Preset <preset number> Off</code>	Deactivates the specified preset in the current zone	The preset number deactivated
<code>Preset <preset number> At <value></code>	Activates the specified preset in the current zone at the given intensity level	The preset number activated
<code>Preset <preset number> Toggle <value></code>	Activates or deactivates the specified preset in the current zone at the given intensity level	The preset number toggled

- `<preset number>`
 - Any whole number from 1 to 999
- `<value>`
 - A percentage from 0 to 100. When specifying percentages, the value can optionally be followed by the % sign.
 - A decimal number from #0 to #255. When specifying decimal numbers, the value must be preceded with a # sign.
 - A hexadecimal number from \$00 to \$FF. When specifying hexadecimal numbers, the value must be preceded with a \$ sign.
 - FL (Full) or On can be used as a shortcut that means 100%
 - Off can be used as a shortcut that means 0%

Abbreviation

PR

Description

Use the **Preset** command to operate on presets within the current zone. Presets can be activated, deactivated, toggled and/or set to a desired intensity level.

Normally, Presets only operate within their *Zone*. See the [Zone](#) command to see how to manage Zones. In certain applications, it is useful to be able to join multiple zones together (such as in a Ballroom with removable “air walls”). See the [Join](#) command to learn how to join Zones together.

Activating Presets

To activate a preset, use the **Preset *n* On** command. For example, to activate Preset 3, use this command:

```
Preset 3 On
```

The stored look in Preset 3 will appear in the current zone.

Any time a preset is activated in a particular zone, any other preset in that zone is automatically deactivated. The only exception to this rule is when more than one preset contains the same channel values. If more than one preset share the same channel values, they will *all* become activated.

Deactivating a Preset

If a preset needs to be deactivated without activating another, the **Off** action can be used:

```
Preset 3 Off
```

If a zone contains one or more presets that are recorded with all zero channel levels, those presets will become active when other presets are deactivated.

Setting a Preset's Intensity Level

If a Preset needs to be recalled, but at a lower intensity level than recorded, the **Preset *n* At *value*** command can be used:

```
Preset 3 At 33
```

In this example Preset 3 is activated, but the channel levels are scaled down to 33% of the original levels recorded in the Preset.

The value used to scale the preset can be given in percentage, decimal, or hexadecimal, similarly to the [At](#) command.

Toggling Presets

To cause a preset to toggle on and off, use the **Preset *n* Toggle *value*** command. For example:

```
Preset 3 Toggle On
```

Each time the **Toggle** action is used, the preset will turn on or off. The new state will be the opposite of the previous state.

The **Toggle** action can also be used with values other than “On”. For example, to toggle a preset between “Off” and “75%”, use this command:

```
Preset 3 Toggle 75
```

Examples

```
Preset 5 On
```

Activates Preset 5 in the current zone.

```
Zone "Foyer" Preset 3 On
```

Activates Preset 3 in the Zone “Foyer”.

```
Preset 1 Toggle On
```

Activates or deactivates Preset 1 depending on Preset 1’s previous state.

```
Preset 7 Off
```

Deactivates Preset 7 in the current zone.

See Also

- [At](#), [Join](#), [Zone](#)

Press

Syntax

Command	Description	Return Value
<code>Press</code>	Presses the selected object(s)	0

Abbreviation

None

Description

The **Press** command is used to perform the same event that would occur if the user physically *presses* a button (or *closes* a contact). **Release** has the opposite effect as **Press**.

When the **Press** command is executed, the selected Button (or Contact) will receive a “press event”, which causes all of the *press events* to be executed for that Button (or Contact).

It is not necessary to always issue a **Release** command for each **Press** command. If a Button receives multiple “press events”, it will execute its “Whenever this Button is Pressed” actions each time. However, if the Button has a “Whenever this Button is Held” rule, that rule will be executed some number of seconds in the future if a **Release** is not issued for that Button.

Examples

```
Button 1 Press
```

Performs the same actions as if the user has physically pressed Button 1.

```
Button 1+3+5 Press
```

Presses Buttons 1, 3, and 5.

```
Button 2.3 Press; Release
```

Presses and then immediately releases Button 2.3.

See Also

[Release](#)

Random

Syntax

Command	Description	Return Value
<code>Random <value></code>	Generates a random number from 0 to <i>value</i>	A random number
<code>Random {<value1>, <value2>}</code>	Generates a random number from <i>value1</i> to <i>value2</i>	A random number

Abbreviation

`RAND`

The **Random** command to generate a random number. Use the **Random** command in CueScript expressions or commands to introduce randomness.

The **Random** command comes in two forms. If a single number is specified, a random number from 0 through that number (inclusive) will be returned. If an array of two numbers are specified, a random number from the first number through the second number (inclusive) will be returned.

When using the **Random** command as a substitution for a single parameter to another command, it must be enclosed in parenthesis. This is because the random command needs to be evaluated as if it is an expression, so the result of the expression is substituted into the outer command properly. See the examples below for clarification.

Examples

```
Random 5
```

Returns a random number from 0 through 5.

```
Random {10,20}
```

Returns a random number from 10 through 20.

```
Macro (Random{5,8})
```

Executes a random Macro from 5 through 8.

```
Cue (Random{1,4}) Go
```

Executes a random Cue from 1 through 4.

`Channel 1 At (Random{50,100})`

Sets Channel 1 to a random value from 50 through 100.

Reboot

Syntax

Command	Description	Return Value
<code>Reboot</code>	Reboot the CueServer	Always returns <code>1</code>

Abbreviation

None

Description

Causes the CueServer to reboot immediately.

Any show or playback occurring will be interrupted, and the hardware will gracefully shut down and then reboot.

Examples

`Reboot`

Causes the CueServer to reboot.

Record

Use the **Record** command to record/create/store Cues, Streams and Groups.

There are several variants of the **Record** command:

- [Record Cue](#)
- [Record Group](#)
- [Record Stream](#)
- [Record Stop](#)

Record Cue

Syntax

Command	Description	Return Value
<code>Record [options] Cue <cue number></code>	Records cue <i>cue number</i>	The cue number recorded

- `<cue number>`
 - Any whole number from 0 to 99999
 - May optionally contain decimal numbers from .00 to .99
- `[options]`
 - Selection:
 - `All` causes all DMX channels to be recorded; **this is the default**
 - `Empty` causes no DMX channels to be recorded into the cue
 - `Selected` causes only the currently selected DMX channels to be recorded into the cue
 - `Active` causes only active DMX channels to be recorded into the cue
 - Source:
 - `Input` causes the DMX Input to be recorded
 - `Playback n` causes only the channels from Playback *n* to be recorded
 - Other:
 - `Stack s` record into Stack *s*

Abbreviation

`R [ALL, EMPTY, SEL, ACTIVE, IN, P n, STACK s] Q`

Description

Recording Cues

The **Record Cue** command records (or re-records) a normal cue.

By default, all channels being output from the CueServer are captured into the new cue. Use the command *options* to change which channels and what source the channels are recorded from.

If no cue with the cue number exists, a new cue will be created. If a cue with the cue number already exists, it will be deleted first and replaced with an entirely new cue. To re-record just the DMX channels without affecting the other cue parameters, use the **Update Cue** command instead.

Record Cue Options

Several options are available to change how a cue is recorded. More than one option may be used, and they can be listed in any order.

Empty

If this option is used, then the new cue will be created containing no DMX channels. When this cue is played back, it will have no affect on any DMX channels, but it will still behave like a normal cue with follow timing and rules.

Selected

If this option is used, then the new cue will be created containing only the currently selected DMX channels. Using this option, cues can be created that only affect specific channels when being played back.

Active

If this option is used, then the new cue will be created containing only the currently active DMX channels. Using this option, cues can be created that only affect specific channels when being played back. When recording the DMX output or input, active channels are any channels that have a non-zero value. When recording from one of the Playbacks, active channels are any channels active in the playback (including zero value channels).

Input

If this option is used, then the new cue will be recorded by using only values from the DMX Input. The playback faders and DMX output will not be recorded.

Playback *n*

If this option is used, then the new cue will be recorded by using only values from the specified playback fader. The DMX input and output will not be recorded.

Stack *s*

If this option is used, then the new cue will be recorded into Stack *s*. This option overrides the stack chosen in the current playback fader.

Examples

`Record Cue 1`

Records the current output from the CueServer as Cue 1.

`Record Empty Cue 2`

Records Cue 2 with no DMX channels.

`Record Selected Cue 3`

Records the currently selected DMX channels as Cue 3.

`Record Active Input Cue 5.1`

Records only the active channels from the DMX Input as Cue 5.1.

`Record Selected Playback 7 Cue 1.23`

Records the currently selected channels from Playback 7 as Cue 1.23.

`Record Active Playback 1 Stack "Test" Cue 101.5`

Records only the active channels from Playback 1 into Cue 101.5 in the cue stack named "Test".

`Channel 1>10`

`Record Selected Cue 4`

Records Channels 1 through 10 as the only channels in Cue 4.

See Also

- [Update Cue](#)

Record Group

Syntax

Command	Description	Return Value
<code>Record Group <group number></code>	Records group <i>group number</i>	The group number recorded

- `<group number>`
 - Any whole number from 0 to 99999

Abbreviation

`R U` or `R GR`

Description

The **Record Group** command creates a new group from the currently selected channels.

If no group with the group number exists, a new group will be created. If a group with the group number already exists, it will be deleted first and replaced with an entirely new group. To re-record just the selected channels without affecting the other group parameters, use the **Update Group** command instead.

Examples

```
Record Group 1
```

Records the currently selected channels as Group 1.

```
Channel 1>10
```

```
Record Group 2
```

Records Channels 1 through 10 as Group 2.

See Also

- [Update Group](#)

Record Stream

Syntax

Command	Description	Return Value
<code>Record [options] Stream <cue number></code>	Records streaming cue <i>cue number</i>	The cue number recorded

- `<cue number>`
 - Any whole number from 0 to 99999
 - May optionally contain decimal numbers from .00 to .99
- `[options]`
 - `Channel n` causes the stream recording to wait for Channel *n* to become non-zero before recording and return to zero to stop recording
 - `Time t` causes the stream recording to automatically stop after *t* seconds
 - `Stack s` record into Stack *s*

Abbreviation

`R [C n, T t, STA s] STR`

Description

Recording Streaming Cues

The **Record Stream** command begins recording (or re-recording) a streaming cue. As soon as this command is executed, a stream recording of the CueServer's DMX output will begin. Use the **Record Stop** command to stop recording the stream.

If no streaming cue with the cue number exists, a new streaming cue will be created. If a cue with the cue number already exists, it will be deleted first and replaced with an entirely new streaming cue.

Record Streaming Cue Options

Several options are available to change how a streaming cue is recorded. More than one option may be used, and they can be listed in any order.

Channel n

If this option is used, then channel n is used as a stream recording “trigger channel”. Recording will wait until channel n becomes non-zero. Then, recording will continue until channel n returns to a zero value. This option is useful in situations where a particular channel coming from a DMX source has been programmed to signal the precise beginning and end of a DMX clip to record.

Time t

If this option is used, then the stream will automatically stop recording after t seconds have been captured. If a trigger channel is being used, the timer does not start until the actual recording stream has begun.

Stack s

If this option is used, then the new cue will be recorded into Stack s . This option overrides the stack chosen in the current playback fader.

Examples

Record Stream 1

Begins recording the current output from the CueServer into Streaming Cue 1.

Record Channel 512 Stream 2

Begins recording the current output from the CueServer into Streaming Cue 2, while using Channel 512 as the channel that will trigger the automatic start and stop of the recording.

Record Time 3.5 Stream 101.5

Begins recording the current output from the CueServer into Streaming Cue 101.5, and the stream will automatically stop recording after 3.5 seconds.

Record Time 15 Channel 1024 Stream 42

Begins recording the current output from the CueServer into Streaming Cue 42, while using Channel 1024 as the channel that will trigger the automatic start and stop of the recording. If the stream is still recording after 15 seconds, it will automatically stop.

Record Stop

Stops recording the Streaming Cue.

See Also

- [Record Stop](#)
- [Update Stream](#)
- [Update Stop](#)

Record Stop

Syntax

Command	Description	Return Value
<code>Record Stop</code>	Stops recording a streaming cue	The cue number recorded

Abbreviation

`R STO`

Description

The **Record Stop** command stops recording any currently recording streaming cue. Use this command in conjunction with the **Record Stream** command.

The **Update Stop** command is an alias for the same command.

Examples

```
Record Stream 1
```

Begins recording the current output from the CueServer into Streaming Cue 1.

```
Record Stop
```

Stops recording the Streaming Cue.

See Also

- [Record Stream](#)
- [Update Stream](#)
- [Update Stop](#)

Release

Syntax

Command	Description	Return Value
<code>Release</code>	Releases channels from the active playback fader <i>or</i> Releases the selected object(s)	<i>None</i> 0

Abbreviation

`REL`

Description

The **Release** command can be used with Channels, Buttons, or Contacts.

Releasing Channels

Released channels have no effect on the DMX output. One can think of released channels as being “transparent”. Before any channels are set or cues executed, all of the channels of a playback fader are *released*.

Releasing Selected Channels

The **Release** command releases the currently selected channels in the active playback fader. If the **Release** command is executed when channels are selected, those channels are released (they become transparent) immediately. After the channels are released, the selection is cleared.

Releasing All Channels

If the **Release** command is executed when *no* channels are selected, then *all* channels in the active playback fader are released. It is common practice to execute the release command twice (`Release Release`) when one wants to be sure to release all channels in the active playback fader.



The **Release** command does not release parked channels. To release all channels, including parked channels, use the **Clear** command instead.

Releasing Buttons or Contacts

The **Release** command is used to perform the same event that would occur if the user physically *releases* a button (or *opens* a contact). **Press** has the opposite effect as **Release**.

When the **Release** command is executed, the selected Button (or Contact) will receive a “release event”, which causes all of the *release events* to be executed for that Button (or Contact).

It is not necessary to always issue a **Press** command before each **Release** command. If a Button receives multiple “release events”, it will execute its “Whenever this Button is Released” actions each time. However, if the Button has a “Whenever this Button is Held” rule, that rule will be cancelled if the **Release** is received before the timer expires.

Examples

```
Release
```

Releases selected channels in the active playback fader.

```
Channel 1>10 Release
```

Releases channels 1>10 in the active playback fader.

```
Playback 3 Release Release
```

Releases all channels in playback 3.

```
Button 1 Release
```

Performs the same actions as if the user has physically released Button 1.

```
Button 1+3+5 Release
```

Releases Buttons 1, 3, and 5.

```
Button 2.3 Press; Release
```

Presses and then immediately releases Button 2.3.

See Also

- [Clear](#), [Playback](#), [Press](#)

Reset

Syntax

Command	Description	Return Value
<code>Reset</code>	Resets the device to it's entirely cleared state	0

Abbreviation

`RESET`

Description

The **Reset** command clears all running show information and returns the device to it's cleared state.

Reset performs the following actions:

- Clears all playback faders (including parked channels)
- Kills any pending Wait commands
- Resets the command context

 The **Reset** command does not honor parked channels. They will be cleared too.

Examples

`Reset`

Entirely resets the device to it's cleared state.

Set

Syntax

Command	Description	Return Value
<code>Set <variable> <value></code>	Sets the value of the variable	The value the variable was set to

- `<variable>`
 - A user variable or system variable name.
- `<value>`
 - A string (a combination of characters enclosed in quotes, such as “Hello World”).
 - A number (a whole number, or a decimal number, such as 123 or 12.7).

Alternate Syntax

See: [Assign](#) command.

Description

Setting Values

The **Set** command sets the value of a variable. The variable can be user defined (such as *xyz*, *LoopCount*, or *IsMyShowEnabled*), or it may be a system variable (such as *button.onColor* or *lcd.backlight*). System variables always contain a “dot” character (`.`). User variables must not contain a “dot” character, otherwise, they will be interpreted as a system variable, and they will not be stored properly.

User variables can be defined on the fly, simply by assigning a value to a variable. There is no need to pre-define variables.

See the [Assign](#) command for an alternate syntax for assigning variable values.

See the [Variables](#) section for how to use variables in the script language.

See the [System Variables](#) section for a complete list of available system variables.

Examples

```
Set x 3
```

Sets the variable *x* to the number 3.

```
Set text "Hello World"
```

Sets the variable *text* to the string `Hello World`.

```
Set lcd.backlight 25
```

Sets the system variable *lcd.backlight* to 25%.

```
Set y ('x' + 1)
```

Sets the variable *y* to the result of the expression `'x' + 1`.

See Also

- [Assign, System Variables](#)

SMPTE

Syntax

Command	Description	Return Value
<code>SMPTE Start</code>	Begins generating internal timecode	1
<code>SMPTE Stop</code>	Stops generating internal timecode	0
<code>SMPTE Clear</code>	Sets the current timecode to <code>00:00:00:00</code>	0
<code>SMPTE "<timecode>"</code>	Sets the current timecode to the specified time	0
<code>SMPTE <frame-number></code>	Sets the current timecode to the specified frame number	0
<code>SMPTE [+/-]<frames></code>	Increments or decrements the current frame	0
<code>SMPTE Input <source></code>	Changes the SMPTE input source	0

- `<timecode>`
 - A string value representing a timecode, such as `"23:59:59:29"`.
 - Fewer places can be specified, for example `"1:23"`, which will be right-justified into a timecode of `"00:00:01:23"`.
- `<frame-number>`
 - A number of frames since "time zero".
- `<source>`
 - The source of the SMPTE time. May be one of the following:
 - `0` or `"Internal"` to set to Internal Generation.
 - `1` or `"Audio"` to set to Audio Input.

Abbreviation

`SMPTE STA`, `SMPTE STO`, `SMPTE CL`, or `SMPTE IN`

Description

The **SMPTE** command can be used to manage timecode, including the internal timecode generator, and external timecode input.

Generating Timecode Internally

The **SMPTE** command can be used to start, stop, reset and set the current timecode using the CueServer's internal timer. When generating timecode internally, events in CueServer's timecode event list will be triggered at the specified times.

Use the **SMPTE Clear** command to reset the current timecode to zero (00:00:00:00).

Use the **SMPTE Start** command to start internal generation of timecode. The current timecode will begin incrementing at 30fps. Any timecode events in the system will trigger when the timecode reaches their marks.

Use the **SMPTE Stop** command to stop internal generation of timecode. The current timecode will freeze at the current time.

Use the **SMPTE** command to set the current timecode to a specific time. A timecode string such as "01:00:04:29" can be given. Additionally, a shorter string can be used, which will be padded with zeros on the left-side. For example, specifying a timecode of "22:11" will set a timecode of "00:00:22:11". Furthermore, a frame number can be given, such as 50, which will be interpreted as the 20th frame of the first second ("00:00:01:20").

Use the **SMPTE ±** command to increment or decrement the current frame by the specified number of frames. For instance, the command **SMPTE +5** will increment the current timecode by 5 frames.



Using any of the **SMPTE** commands that modify the timecode will automatically switch the input source to "Internal".

Using External Timecode

Use the **SMPTE Input "Audio"** command to switch reception of timecode to the audio input port. Timecode will begin tracking the audio input if a valid signal is present. While timecode is being received at the audio input, any timecode events in the system will trigger when the timecode reaches their marks.

Examples

SMPTE Start

Starts generating timecode internally beginning with the current time. The input source will be changed to "Internal".

SMPTE Stop

Timecode will freeze. The input source will be changed to "Internal".

SMPTE Clear

Sets the current timecode to 00:00:00:00. The input source will be changed to "Internal".

SMPTE "5:43:21"

Sets the current timecode to 00:05:43:21. The input source will be changed to "Internal".

SMPTE 32

Sets the current timecode to 00:00:01:02. The input source will be changed to "Internal".

SMPTE -1

Decrements the current timecode by 1 frame. The input source will be changed to "Internal".

SMPTE Input 0 or SMPTE Input "Internal"

Switches the SMPTE input source to "Internal".

SMPTE Input 1 or SMPTE Input "Audio"

Switches the SMPTE input source to "Audio Input".

Stack

Syntax

Command	Description	Return Value
<code>Stack "<stack name>"</code>	Sets the active playback fader's cue stack	The number of the first cue in the stack
<code>Stack Clear</code>	Sets the active playback fader to use the main cue list	The number of the first cue in the main cue list
<code>Stack ?</code>	Queries the current stack name	The name of the current cue stack

- `<stack name>`
 - A name of the desired cue stack

Abbreviation

None

Description

Use the **Stack** command to change what cue stack the active playback fader is using.

By default, all cues are loaded from the main cue list. Additionally, the show file may contain one or more *cue stacks*. The **Stack** command is used to change which cue stack the playback fader is using. Once the stack has been changed on a playback fader, all cues on that playback will be loaded from that cue stack.

Use the **Stack Clear** command to return the active playback fader back to the main cue list. Optionally **Stack ""** can be used to accomplish the same thing.

When switching cue stacks, the first cue in the new stack will automatically become the playback fader's *next cue*. Because the **Stack** command selects the first cue in a cue stack, the [Go](#) command can be used to run the first cue in the stack.

Examples

```
Stack "Surprise"
```

Sets the active playback fader's cue stack to the stack named "Surprise".

Stack "Intro" Go

Sets the active playback fader to use the "Intro" stack and executes the first cue in that stack.

Stack Clear

Sets the active playback fader to use the main cue list.

See Also

- [Cue, Go](#)

Start

Syntax

Command	Description	Return Value
<code>Start</code>	Resumes normal timing operation of the active playback fader	The playback number started

Abbreviation

`STA`

Description

The **Start** command resumes normal timing operation of the active playback fader. **Start** has the opposite effect as the [Stop](#) command.

When a playback fader is stopped, it's timing operation is temporarily suspended in the following ways:

- Using the **Go** command does not crossfade, the channel levels of the cue appear immediately
- Using the **At** command does not crossfade, the levels appear immediately
- The auto-follow timer stops counting down
- Streaming cues are paused

Examples

```
Start
```

Resumes normal timing operation of the active playback fader.

```
Playback 2 Start
```

Resumes normal timing operation of playback 2.

See Also

- [Playback](#), [Stop](#)

Stop

Syntax

Command	Description	Return Value
<code>Stop</code>	Suspends normal timing operation of the active playback fader	The playback number stopped

Abbreviation

`STO`

Description

The **Stop** command suspends normal timing operation of the active playback fader. **Stop** has the opposite effect as the [Start](#) command.

When a playback fader is stopped, it's timing operation is temporarily suspended in the following ways:

- Using the **Go** command does not crossfade, the channel levels of the cue appear immediately
- Using the **At** command does not crossfade, the levels appear immediately
- The auto-follow timer stops counting down
- Streaming cues are paused

Examples

```
Stop
```

Suspends normal timing operation of the active playback fader.

```
Playback 2 Stop
```

Suspends normal timing operation of playback 2.

See Also

- [Playback](#), [Start](#)

Time

Syntax

Command	Description	Return Value
<code>Time <fade time></code>	Sets the global fade time	The global fade time set
<code>Time ?</code>	Returns the current global fade time	The current global fade time

- `<fade time>`
 - A decimal number of seconds (optionally using decimal digits for fractions of seconds)
 - `0` means no fade time (or the channels/values are set immediately without fading)
 - Optionally may include a slash (`/`) which indicates a split (up/down) fade
 - Optionally may include a dash (`-`) which indicates a delayed fade

Abbreviation

T

Description

Setting The Global Fade Time

Use the **Time** command to set the *global fade time*. This time is used to crossfade channels or values whenever the **At** command is executed. The global fade time is used when setting channels, or a playback's submaster value.

Using Split Fade Times

A *split fade time* is used when it is desired to have channels that are fading up occur at a different rate than channels fading down. To specify a split fade time, use a slash character in between two fade times. For instance, the command `Time 3.5/7.5` will cause any channel that is fading up to occur in 3.5 seconds, and any channel that is fading down to occur in 7.5 seconds.

Using Fade Delays

Normally, whenever a channel level is set, the fade begins immediately. A delay can be inserted that would cause the fade to be delayed before starting to change value. To specify a fade delay, use a delay time and

dash character before the fade time. For instance, the command `Time 5.5-10` will cause the fade to delay 5.5 seconds before beginning a 10 second fade.

Using Both Fade Delays And Split Fade Timing

Both fade delays and split fades can be combined. For instance, the command `Time 1-2/3-4` would cause any channels fading up to be delayed 1 second before fading over 2 seconds, while the downward fading channels would be delayed 3 seconds before fading over 4 seconds.

Determining The Current Global Fade Time

Use the **Time** command with the question mark (?) to return the current global fade time. A fade time such as `7.21` or `12/3` will be returned.



Note that the *Global Fade Time* is different from the *Cue Fade Time*. The global fade time effects the **At** command. The cue fade time effects the **Go** command. The cue fade time is set with the **Fade** command.

Examples

```
Time 1
```

Sets the global fade time to 1 second.

```
Time 1.35/7.2
```

Sets the global fade time to 1.35 seconds for upward fading channels and 7.2 seconds for downward fading channels.

```
Channel 1>10 Time 5 At 50
```

Selects channels 1 thru 10, then sets the fade time to 5 seconds, then sets the channels to 50%.

```
Playback 1 Time 3.5 At 25
```

Selects playback 1, then sets the fade time to 3.5 seconds, then sets the playback's submaster to 25%.

See Also

- [Cue](#), [Fade](#), [Go](#)

Toggle

Syntax

Command	Description	Return Value
<code>Toggle <value></code>	Toggles the value of the selected object(s)	The value the object(s) were set to

- `<value>`
 - A percentage from `0` to `100`. When specifying percentages, the value can optionally be followed by the `%` sign.
 - A decimal number from `#0` to `#255`. When specifying decimal numbers, the value must be preceded with a `#` sign.
 - A hexadecimal number from `$00` to `$FF`. When specifying hexadecimal numbers, the value must be preceded with a `$` sign.
 - `FL` (Full) or `On` can be used as a shortcut that means `100%`
 - `Off` can be used as a shortcut that means `0%`

Abbreviation

`TOG`

Description

Toggling Values

The **Toggle** command flip-flops the currently selected object(s) values between a fixed value and zero. In other words, if the selected value is already set to the toggle value, the value is set to zero. But, if the selected value is not equal to the toggle value, then the value is set to the toggle value. This flip-flop behavior creates a situation where each time the **Toggle** command is executed, the selected value(s) alternate between zero and the toggle value.

The **Toggle** command can be used with many types of objects, including **Buttons**, **Channels**, **Groups**, **Outputs**, and **Playbacks**.

Other than the alternating behavior, the **Toggle** command otherwise behaves similarly to the [At](#) command.

Examples

Channel 1 Toggle 100

On each execution, toggles the value of channel 1 between 0% and 100%.

Group 3 Toggle 33

On each execution, toggles the value of the channels in group 3 between 0% and 33%.

See Also

- [Button](#), [Channel](#), [Contact](#), [Group](#), [Output](#), [Playback](#)

Unpark

Syntax

Command	Description	Return Value
<code>Unpark</code>	Unparks the selected channel(s)	<i>None</i>

Abbreviation

None

Description

The **Unpark** command unlocks the current value of the selected channel(s) in the active playback fader. Parked channels cannot be changed by running cues or using the **At** command. Use the **Unpark** command to un-freeze one or more channels so future cues and/or commands will effect the channel level(s).

The **Park** command has the opposite effect as the **Unpark** command.



Channels are parked in individual playback faders (not globally). Beware that if a channel is parked in one playback, it is still possible for another playback to cause the output of that channel to change.

Visual Representation

In CueServer Studio, parked channels appear in the stage view with their channel numbers in red. Note that you can only see parked channels if the Stage View is set to display a specific playback fader. You will not see parked channels in either the input or output stage views.

1	2	3	4	5	6	7	8	9	10
-	-	-	75	75	75	-	-	-	-
19	20	21	22	23	24	25	26	27	28
-	-	-	-	-	-	-	-	-	-

Examples

`Unpark`

Unparks the currently selected channels in the currently active playback fader.

Channel 1>3+5>8 Unpark

Unparks channels 1 through 3 and 5 through 8 in the currently active playback fader.

Playback 3 Channel * Unpark

Unparks all channels in playback 3.

See Also

- [Channel](#), [Playback](#), [Park](#)

Update

Use the **Update** command to change what's stored in Cues or Groups without affecting the other parameters of the object.

There are several variants of the **Update** command:

- [Update Cue](#)
- [Update Group](#)

Update Cue

Syntax

Command	Description	Return Value
<code>Update [options] Cue <cue number></code>	Updates the DMX channels in cue <i>cue number</i>	The cue number updated

- `<cue number>`
 - Any whole number from 0 to 99999
 - May optionally contain decimal numbers from .00 to .99
- `[options]`
 - Selection:
 - `All` causes all DMX channels to be updated
 - `Empty` causes all DMX channels to be removed from the cue
 - `Selected` causes the currently selected DMX channels to become the DMX channels in the cue
 - `Active` causes only active DMX channels to become the DMX channels in the cue
 - Source:
 - `Input` causes the DMX Input to be recorded
 - `Playback n` causes only the channels from Playback *n* to be recorded
 - Other:
 - `Stack s` record into Stack *s*

Abbreviation

`UP [ALL, EMPTY, SEL, ACTIVE, IN, P n, STACK s] Q`

Description

Updating Cues

The **Update Cue** command updates the DMX channels of a normal cue. Use this command to change the DMX channels stored in a cue without changing any of the other properties recorded in the cue (such as the fade and follow time, link, and rules).

By default, any channels currently recorded in the cue are updated.

If no cue with the cue number exists, a new cue will be created. If a cue with the cue number already exists, only the cue's DMX channels will be updated by using this command. All other cue parameters will not be affected.

Update Cue Options

Several options are available to change how a cue is updated. More than one option may be used, and they can be listed in any order.

Empty

If this option is used, then the updated cue will contain no DMX channels. When this cue is played back, it will have no affect on any DMX channels, but it will still behave like a normal cue with follow timing and rules.

Selected

If this option is used, then the updated cue will contain only the currently selected DMX channels. Using this option, cues can be created that only affect specific channels when being played back.

Active

If this option is used, then the updated cue will contain only the currently active DMX channels. Using this option, cues can be created that only affect specific channels when being played back. When recording the DMX output or input, active channels are any channels that have a non-zero value. When recording from one of the Playbacks, active channels are any channels active in the playback (including zero value channels).

Input

If this option is used, then the updated cue will be recorded by using only values from the DMX Input. The playback faders and DMX output will not be recorded.

Playback *n*

If this option is used, then the updated cue will be recorded by using only values from the specified playback fader. The DMX input and output will not be recorded.

Stack *s*

If this option is used, then the updated cue will be recorded into Stack *s*. This option overrides the stack chosen in the current playback fader.

Examples

Update Cue 1

Stores (updates) the current output from the CueServer into Cue 1.

Update Empty Cue 2

Removes the DMX channels from Cue 2.

Update Selected Cue 3

Stores (updates) the currently selected DMX channels into Cue 3.

Update Active Input Cue 5.1

Stores (updates) only the active channels from the DMX Input as Cue 5.1.

Update Selected Playback 7 Cue 1.23

Stores (updates) the currently selected channels from Playback 7 as Cue 1.23.

Update Active Playback 1 Stack "Test" Cue 101.5

Stores (updates) only the active channels from Playback 1 into Cue 101.5 in the cue stack named "Test".

Channel 1>10

Update Selected Cue 4

Stores (updates) Channels 1 through 10 as the only channels into Cue 4.

See Also

- [Record Cue](#)

Update Group

Syntax

Command	Description	Return Value
<code>Update Group <group number></code>	Updates group <i>group number</i>	The group number updated

- `<group number>`
 - Any whole number from 0 to 99999

Abbreviation

`UP U` or `UP GR`

Description

The **Update Group** command updates the channels in a group to the currently selected channels.

If no group with the group number exists, a new group will be created. Use this command to change the DMX channels stored in a group without losing any of the other properties recorded in the group (such as the name).

If no group with the group number exists, a new group will be created. If a group with the group number already exists, only the group's channels will be updated by using this command. All other group parameters will not be affected.

Examples

```
Update Group 1
```

Stores (updates) the currently selected channels into Group 1.

```
Channel 1>10
```

```
Update Group 2
```

Stores (updates) Channels 1 through 10 into Group 2.

See Also

- [Record Group](#)

Update Stream

Syntax

Command	Description	Return Value
<code>Update [options] Stream <cue number></code>	Updates streaming cue <i>cue number</i>	The cue number updated

- `<cue number>`
 - Any whole number from 0 to 99999
 - May optionally contain decimal numbers from .00 to .99
- `[options]`
 - `Channel n` causes the stream updating to wait for Channel *n* to become non-zero before recording and return to zero to stop recording
 - `Time t` causes the stream recording to automatically stop after *t* seconds
 - `Stack s` record into Stack *s*

Abbreviation

`UP [C n, T t, STA s] STR`

Description

Updating Streaming Cues

The **Update Stream** command begins recording (or re-recording) a streaming cue. As soon as this command is executed, a stream recording of the CueServer's DMX output will begin. Use the **Record Stop** command to stop recording the stream. Use this command to change the DMX stream stored in a cue without losing any of the other properties recorded in the cue (such as the follow time, link, and automation rules).

If no streaming cue with the cue number exists, a new streaming cue will be created. If a streaming cue with the cue number already exists, only the cue's DMX channels will be re-recorded by using this command. All other cue parameters will not be affected.

Update Streaming Cue Options

Several options are available to change how a streaming cue is updated. More than one option may be used, and they can be listed in any order.

Channel n

If this option is used, then channel n is used as a stream recording “trigger channel”. Recording will wait until channel n becomes non-zero. Then, recording will continue until channel n returns to a zero value. This option is useful in situations where a particular channel coming from a DMX source has been programmed to signal the precise beginning and end of a DMX clip to record.

Time t

If this option is used, then the stream will automatically stop recording after t seconds have been captured. If a trigger channel is being used, the timer does not start until the actual recording stream has begun.

Stack s

If this option is used, then the new cue will be recorded into Stack s . This option overrides the stack chosen in the current playback fader.

Examples

Update Stream 1

Begins recording the current output from the CueServer as an update to Streaming Cue 1.

Update Channel 512 Stream 2

Begins recording the current output from the CueServer as an update to Streaming Cue 2, while using Channel 512 as the channel that will trigger the automatic start and stop of the recording.

Update Time 3.5 Stream 101.5

Begins recording the current output from the CueServer as an update to Streaming Cue 101.5, and the stream will automatically stop recording after 3.5 seconds.

Update Time 15 Channel 1024 Stream 42

Begins recording the current output from the CueServer as an update to Streaming Cue 42, while using Channel 1024 as the channel that will trigger the automatic start and stop of the recording. If the stream is still recording after 15 seconds, it will automatically stop.

Update Stop

Stops updating the Streaming Cue.

See Also

- [Record Stream](#)
- [Record Stop](#)
- [Update Stop](#)

Update Stop

Syntax

Command	Description	Return Value
<code>Update Stop</code>	Stops updating a streaming cue	The cue number updated

Abbreviation

`UP STO`

Description

The **Update Stop** command stops recording/updating any currently recording streaming cue. Use this command in conjunction with the **Update Stream** command.

The **Record Stop** command is an alias for the same command.

Examples

```
Update Stream 1
```

Begins updating the current output from the CueServer into Streaming Cue 1.

```
Update Stop
```

Stops updating the Streaming Cue.

See Also

- [Record Stream](#)
- [Record Stop](#)
- [Update Stream](#)

Unlock

Syntax

Command	Description	Return Value
<code>Unlock</code>	Unlocks the selected object(s)	The number of objects unlocked

Abbreviation

None

Description

The **Unlock** command *unlocks* the currently selected object(s). The **Unlock** command can be used with **Buttons**, **Contacts**, and **Stations**. When an object is unlocked, it triggers its events when operated and if it has an indicator, it appears in a “normal” state. **Lock** has the opposite effect as **Unlock**.

The following table shows the various effect of locking or unlocking an object:

Object	When Unlocked	When Locked
Buttons	Responds to presses normally, displays normal indicator states	Does not trigger any actions, displays <i>locked</i> indicator state
Contacts	Responds to closures normally	Does not trigger any actions
Stations	Station operates normally, all controls display normal indicator states	Station is locked, all controls display <i>locked</i> indicator states



Disabling or Locking a Button, Contact or Station both prevent events from being triggered when physically operated, however a Disabled Button, Contact or Station still shows its normal indicator states, where a Locked object shows its Locked indicator state.

Examples

```
Button 1 Unlock
```

Unlocks button 1.

Button 3>5 Unlock

Unlocks buttons 3 through 5.

Station 7 Unlock

Unlocks station 7.

See Also

[Button](#), [Contact](#), [Lock](#), [Station](#)

Wait

Syntax

Command	Description	Return Value
<code>Wait <time></code>	Causes the execution of the current script to be suspended for a given number of seconds	An <i>id number</i> to identify the pending commands
<code>Wait Clear</code>	Causes all commands that are currently waiting to be cancelled	The number of cancelled waits
<code>Wait Stop <id></code>	Causes the pending commands with the given <i>id</i> to be cancelled	The number of cancelled waits
<code>Wait ?</code>	Returns the number of currently waiting commands	A number

- `<time>`
 - A decimal number of seconds (optionally using decimal digits for fractions of seconds).
- `<id>`
 - A decimal number internally assigned to the pending commands. Use this number with the **Wait Stop** command.

Abbreviations

`W`, `WCL`, `WSTO`, `W?`

Description

The **Wait** command causes the current command line to be suspended for a given number of seconds. Use **Wait** to cause a delay between script steps.

The **Wait Clear** command cancels **all** currently waiting commands. If more than one command is currently in a waiting state, all of them will be cleared simultaneously.

The **Wait Stop** command cancels only the commands with the given *id*.

Using the Wait Stop command

To use the **Wait Stop** command, the *id* of the pending commands must be stored in a variable.

For instance, if the following command is executed on the command line:

```
Button 1 On; Wait 5; Button 2 On
```

Button 1 will turn on immediately, then 5 seconds later Button 2 will turn on. A more detailed look at what is happening reveals that the first **Button** command executes immediately, then the **Wait** command executes. When CueScript encounters a **Wait** command, the remainder of the command line is placed in a queue and assigned an *id*. Then, CueScript stops processing commands, returning the *id* of the pending commands.

To cancel the queued commands, the **Wait Stop** command can be used. The **Wait Stop** command requires the *id* of the queued commands to stop. In order to save this special *id* for use with the **Wait Stop** command, the *id* can be placed in a variable. See the following example:

```
"myID" = (Button 1 On; Wait 5; Button 2 On)
```

The commands from the first example are placed in parenthesis. The parenthesis cause the enclosed commands to execute as a single expression. The result of that expression is stored in the variable **myID**.

If after a short delay (for example, 3 seconds later) it is desired to cancel the execution of the "Button 2 On" command that is in the queue to be executed, the following command can be executed:

```
Wait Stop 'myID'
```

Examples

```
Channel 1 At FL; Wait 5; At 0
```

Sets Channel 1 to FL, then waits 5 seconds, then sets Channel 1 to 0.

```
Cue 1 Go; Wait 2.5; Clear
```

Executes Cue 1, then waits 2.5 seconds, then clears the playback fader.

```
Wait Clear
```

Clears all currently waiting commands.

```
"stopCue" = (Playback 1 Clear; Wait 10; Cue 1 Go)
```

Clears Playback 1, then waits 10 seconds, then executes Cue 1. Also, the *id* of the pending "Cue 1 Go" is placed in the variable "stopCue".

```
Wait Stop 'stopCue'
```

Stops the pending "Cue 1 Go" from the previous example. Will return "1" if the command was stopped. Will return "0" if the command was not found in the queue.

Write

Syntax

Command	Description	Return Value
<code>Write <port> <string></code>	Writes the given string to the specified port	The number of characters written
<code>Write <ip-address> <string></code>	Sends the given string via UDP to the specified <i>ip-address</i> using the default port of 52737	The number of characters written
<code>Write <ip-address>:<port> <string></code>	Sends the given string via UDP to the specified <i>ip-address</i> and <i>port</i>	The number of characters written

- `<port>`
 - `COM1` refers to the built-in RS-232 port.
 - `COM2` refers to the built-in RS-485 port.
- `<ip-address>`
 - Any valid ip address (such as `192.168.1.234`)

Abbreviation

`WR`

Writes (or sends) the given string to the specified serial port or via a UDP Ethernet packet.

Use the **Write** command to send strings to other devices via one of the serial ports or via Ethernet messages.



Special non-printing characters can be added to strings by using *escape sequences*, such as “\n” for newline, “\r” for carriage return, etc. See the [strings](#) section for more information about escape sequences.

Examples

```
Write COM1 "Hello World"
```

Sends the string "Hello World" to the RS-232 port.

```
Write COM2 "This is a test\r"
```

Sends the string "This is a test" followed by a carriage return character to the RS-485 port.

```
Write "10.0.1.5" "Cue 1 Go"
```

Sends the string "Cue 1 Go" via UDP to the ip address "10.0.1.5" and the default port of 52737.

```
Write "10.0.1.5:1234" "Mission Accepted\x00"
```

Sends the string "Mission Accepted" followed by a NULL byte via UDP to the ip address "10.0.1.5" and the port of 1234.

Zone

Syntax

Command	Description	Return Value
Zone "<zone name>"	Changes the current zone	The zone name
Zone Clear	Cancels the current zone	<i>none</i>
Zone ?	Returns the current zone	The zone name

- <zone name>
 - The alphanumeric name of a zone.
 - Zone names must be enclosed in quotes.

Abbreviation

ZO

Description

Use the **Zone** command to select a zone. Zones are used to separate different areas of a particular project. Zones are also “parent containers” for presets, meaning that any preset must be a member of a zone.

When a zone is selected, only channels within that zone can be changed. For example, a zone named “Meeting Room” only contains channels 10 through 19. If that zone is selected, then any subsequent commands that might change channel values (such as **On**, **Off**, **At**, **Go**, **Release**, **Toggle**, etc.) will *only* operate on channels 10 through 19, regardless of which channels are being targeted by the Cue, Group, Preset, etc.

To remove the restrictions of having a zone selected, use the **Zone Clear** command. Once the zone is cleared *all* channels will again respond to the various channel setting commands.

In certain applications, it is useful to be able to join multiple zones together (such as in a Ballroom with removable “air walls”). See the [Join](#) command to learn how to join Zones together. When zones are joined, the channel setting commands can affect the combined channels of all joined zones.

Examples

`Zone "Ballroom A"`

Selects the zone named "Ballroom A".

`Zone "Foyer" Preset 3 On`

Activates Preset 3 in the Zone "Foyer".

`Zone "Theater" Channel 1>100 FL@`

Selects the "Theater" zone and then attempts to set channels 1 through 100 to Full. Only the channels defined in the zone will actually be set to Full.

`Zone Clear`

Remove the restrictions of the current zone.

`Zone ?`

Returns the currently selected zone.

See Also

- [At](#), [Join](#), [Preset](#)

Logic Commands

CueScript contains several logic commands that are used to modify the normal execution of commands.

The **If..Then..Else** command is used to conditionally execute commands depending on the result of a conditional expression.

The **Break** command is used to force early termination of execution of a command string.

These commands are described in detail in the following sub-sections.

Break

Syntax

Command	Description	Return Value
<code>Break</code>	Stops executing the current command string	<i>None</i>

Abbreviation

`BR`

Description

The **Break** command stops executing the current command string. Use **Break** in situations where a condition requires that all of the subsequent commands should be ignored.

Examples

```
Cue 1 Go; Break; Button 1 On
```

Executes Cue 1, then stops execution of subsequent commands. The **Button 1 On** phrase will never be executed.

```
Cue 1 Go
If ('myVariable' > 5) Then
    Break
EndIf
Button 1 On
```

Executes Cue 1, then checks to see if *myVariable* is greater than 5. If it is, then none of the remaining commands will execute. If *myVariable* is less than or equal to 5, then Button 1 will be turned on.

```
Cue 1 Go; `continue`; Button 1 On
```

Executes Cue 1, then the contents of the variable *continue* are executed in place. If *continue* contains the value “Break” then execution stops here. If *continue* is empty, then execution continues on to “Button 1 On”. Note that the `accent-quotes` are used around the variable name to *execute* the contents of the variable.

If..Then..Else

Syntax

Command	Description	Return Value
<code>If (<expression>) [Then] <action> [Endif]</code>	Tests <i>expression</i> and performs <i>action</i> if true	The result of <i>action</i>
<code>If (<expression>) [Then] <action1> Else <action2> [Endif]</code>	Tests <i>expression</i> and performs <i>action1</i> if true or <i>action2</i> if false	The result of <i>action1</i> or <i>action2</i>

Abbreviation

None

Description

The **If .. Then .. Else** statements are used to conditionally execute commands based on the value of an expression.

Consider this command:

```
If ('x' == 1) Then Cue 1 Go
```

The above example first checks the value of the variable *x*, and if it is equal to 1, then Cue 1 is executed. On the other hand, if *x* is not equal to 1, then nothing will happen. Since no commands are present after the “Cue 1 Go”, an **Endif** is not necessary.

See the section on [Expressions](#) for more information about the kinds of expressions that can follow an **If** statement.

Using Else

The **Else** keyword can be used to execute commands if the expression is false. Consider this example:

```
If ('mode' > 5) Then Cue 1 Go Else Cue 2 Go
```

In the above example, if the value of *mode* is greater than 5, then Cue 1 will execute, but if *mode* is 4 or less, then Cue 2 will execute. Since no commands are present after the “Cue 2 Go”, an **EndIf** is not necessary.

Using Endif

In the basic form of the **If .. Then** statements, *all* of the commands after the **Then** will be executed if the expression is true. In the case where additional non-conditional commands are needed after the **If .. Then** statement, use the **Endif** keyword to end the conditional part of the **If .. Then** statement.

For example, in the following command, Cue 1 will execute *and* Playback 2 will be cleared if the *showEnabled* variable is 1:

```
If ('showEnabled' == 1) Then Cue 1 Go; Playback 2 Clear
```

But, by inserting the **Endif** keyword, the script can be changed to have Cue 1 execute only if the *showEnabled* variable is 1, but Playback 2 is always cleared:

```
If ('showEnabled' == 1) Then Cue 1 Go Endif Playback 2 Clear
```

Using Multiple Lines

The **If .. Then .. Else** statements can also be used across multiple lines of code, which is particularly useful when the script becomes more complex:

```
Playback 1

If ('testMode' == 1) Then
    Cue 1 Go
Else
    Cue 2 Go
EndIf

Playback 2 Clear

<hr>
```

Nesting Multiple If .. Then Statements

For more complex logic scenarios, you can put **If .. Then** statements *inside* of other **If .. Then** statements.

For example:

```
Playback 1

If ('testMode' == 1) Then
    Cue 1 Go
Else
    If ('eStop' == 1) Then
        Cue 99 Go
    Else
        Cue 2 Go
    EndIf
EndIf

Playback 2 Clear
```

System Variables

CueServer uses System Variables to allow CueScript commands to change properties or behaviors of various system related objects. Setting a system variable has immediate effect, causing the referenced object to change appearance or behavior. For example, to immediately change the brightness of the LCD Backlight, the commands `Set lcd.backlight 25` or `"lcd.backlight"=25` can be used.

The following sections list the available system variables:

Audio

Sets the output volume for the Audio Output jack.

<code>audio.volume</code>	Sets the line level output of the Audio Output jack. Available levels range from 0 to 100. A value of 0 produces no output. The default value is 90.
---------------------------	--

Example:

```
Set audio.volume 50
```

The example above sets the audio volume of the Audio Output jack to 50%.

Buttons

Sets the color and flashing patterns for the built-in user defined function buttons.

Before setting or retrieving one of the button variables, make sure that one or more buttons are selected first. For example, use the **Button** command to specify which button(s) you want to change a property for.

<code>button.flash</code>	Sets the flash pattern for buttons. Available patterns range from 0 to 15. A value of 0 means “no flash”. The remaining 15 values produce various combinations of flashing when the button indicator is turned on.
<code>button.onColor</code> <code>button.offColor</code>	Sets the “on” and “off” colors for buttons. The value can be a single number from 0 to 100, meaning off (black) to full-on (white), or it may be a 3-element array representing an RGB color. For example the array <code>{100, 50, 0}</code> would produce an Orange color.

Example:

```

Button 1
Set button.onColor {100,0,50}
Set button.flash 4
On

```

The example above first selects button 1, then sets it's color to a rose color, then sets it's flash pattern to a fast blink. Then, it turns the button's indicator "on".

Clock

Provides access to the real-time clock.

<code>clock.date</code>	Gets or sets the current date and time. When querying the date variable, a string such as <code>Mon Jan 01 18:30:59 EST 2018</code> will be returned. When setting the date variable, a variety of formats are supported including <code>MM/DD/YY</code> , <code>HH:MM:SS</code> , <code>YYYY-MM-DD HH:MM:SS</code> , <code>YYYYMMDD HH:MM</code> , <code>next year</code> , <code>last friday</code> , and many others. See the Linux documentation on the <code>date</code> command for more information.
<code>clock.second</code>	Gets or sets the current Second. Valid values range from <code>0</code> to <code>59</code> .
<code>clock.minute</code>	Gets or sets the current Minute. Valid values range from <code>0</code> to <code>59</code> .
<code>clock.hour</code>	Gets or sets the current Hour. Valid values range from <code>0</code> to <code>23</code> .
<code>clock.day</code>	Gets or sets the current Day. Valid values range from <code>1</code> to <code>31</code> .
<code>clock.month</code>	Gets or sets the current Month. Valid values range from <code>1</code> to <code>12</code> .
<code>clock.year</code>	Gets or sets the current Year. Valid values range from <code>1900</code> to <code>2999</code> .
<code>clock.weekday</code>	Gets the number of days since Sunday. Valid values range from <code>0</code> to <code>6</code> . <i>This is a read-only property.</i>
<code>clock.yearday</code>	Gets the number of days since January 1. Valid values range from <code>0</code> to <code>365</code> . <i>This is a read-only property.</i>
<code>clock.dst</code>	Returns <code>1</code> if Daylight Saving Time is currently in effect, and <code>0</code> if not. Can return <code>-1</code> in the case where the DST status cannot be determined. <i>This is a read-only property.</i>
<code>clock.zone</code>	Gets or sets the time zone. When querying this variable, a string such as <code>America/New York</code> will be returned. When setting this variable, be sure to provide a time zone in the format of <code><region>/<location></code> from CueServer's available listing of time zones . Zones that have spaces in their names can use either space characters (such as <code>America/New York</code>), or underscore characters (such as <code>America/New_York</code>).

Examples:

```
Set clock.date "1/1/18 12:00:00"
```

The example above sets the time and date to January 1, 2018 at noon.

```
"clock.zone" = "America/Los_Angeles"
```

The example above sets the time zone to “America/Los Angeles”

```
if ('clock.hour' > 12) then
  Cue 1 Go
endif
```

The example above executes Cue 1 only if the hour is greater than 12.



Please note that when setting the time or date, if the CueServer had previously been configured to receive its time via an NTP server, the system will switch to “manual” time mode.

Debug

Enables various debugging functions. Each of the following variables can be set to **1** to enable the function and **0** to disable the function.

<code>debug.all</code>	Enables/disables <i>all</i> of the individual diagnostic functions (see below).
<code>debug.buttons</code>	Enables/disables system logging of button and contact related events (both built-in buttons/contacts and CueStation buttons/contacts).
<code>debug.cue</code>	Enables/disables system logging of all Cue related events processed by the system.
<code>debug.cuescript</code>	Enables/disables system logging of all CueScript commands processed by the system.
<code>debug.show</code>	Enables/disables system logging of show related events.
<code>debug.udp</code>	Enables/disables system logging of all UDP packets received on the CueScript port.
<code>debug.variables</code>	Enables/disables system logging of all changes to variable values.

Example:

```
Set debug.udp 1
Set debug.cuescript 0
```

The example above enables UDP event logging and disables CueScript event logging.



Please note that all debugging functions are reset to “off” when a CueServer is power-cycled. Each desired function must be re-enabled each time a CueServer is rebooted. Also, it is not recommended to leave a debug log function enabled indefinitely, as some of these functions can result in the system log overflowing.

LCD Display

Sets the backlight brightness and various string fields for the LCD display.

<code>lcd.backlight</code>	Sets the brightness of the LCD Backlight. Brightness values range from 0 to 100.
<code>lcd.top</code> <code>lcd.bottom</code> <code>lcd.topLeft</code> <code>lcd.topRight</code> <code>lcd.bottomLeft</code> <code>lcd.bottomRight</code>	Sets a temporary overlay string that replaces the top or bottom lines, or quadrant of the display. Set this value to an empty string ("") to remove the temporary overlay.

Example:

```
Set lcd.backlight 25
Set lcd.top "Hello World"
Set lcd.bottom ""
```

The example above first sets the LCD Backlight brightness to 25%, then writes a temporary string to the top line that says `Hello World`, then removes any temporary string from the bottom line.

Panel

Changes properties of the front-panel of the device.

<code>panel.brightness</code>	Sets the overall brightness of the front-panel function button indicators and the navigation switch backlight. Brightness values range from 0 to 100.
-------------------------------	---

Example:

```
Set panel.brightness 33
```

The example above sets the overall front-panel brightness to 33% of its maximum brightness.

Playbacks

Changes properties of a Playback fader.

<code>playback.mode</code>	Sets the combine mode of a Playback fader. Available modes include "Merge", "Override", "Scale", and "Pin".
----------------------------	---

Example:

```
Playback 1  
Set playback.mode "Override"  
Playback 2  
Set playback.mode "Scale"
```

The example above first sets the combine mode of Playback 1 to Override, then sets the combine mode of Playback 2 to Scale.

Random Numbers

Sets the seed for the random number generator.

<code>random.seed</code>	Sets the random number generator's seed value. The random seed is an unsigned 32-bit value from 0 to 4294967295.
--------------------------	--

Example:

```
Set random.seed 42
```

The example above sets the random seed to 42.

Universes

Sets properties of the DMX Universes.

<code>universe.priority</code>	Sets the priority level of the universe. Available values range from 0 to 200.
--------------------------------	--

Example:

```
Universe 7  
Set universe.priority 150
```

The example above sets the priority of Universe 7 to 150.

Internals

- [Web Server](#)
- [CGI API](#)
- [Show File Format](#)

Web Server

- [Environment Variables](#)

Environment Variables

The following *environment variables* are defined in the built-in Apache 2 web server. These variables are available for use within custom HTML pages and/or CGI type scripts being served from CueServer.

Variable	Description	Example
<code>SHOW_NAME</code>	The name of the current show file	<code>My First Show</code>
<code>SHOW_UNIVERSES</code>	The number of currently configured universes	<code>8</code>
<code>SHOW_CHANNELS</code>	The number of currently configured channels	<code>4096</code>
<code>SHOW_PLAYBACKS</code>	The number of currently configured playback faders	<code>16</code>
<code>DEVICE_MODEL</code>	The model number of the device	<code>CS-940 (Rev. A)</code>
<code>DEVICE_NAME</code>	The assigned name of the device	<code>CueServer 2</code>
<code>DEVICE_SERIAL</code>	The serial number of the device	<code>601234</code>

Using Environment Variables with SSI

Apache environment variables can be used in HTML by utilizing *Server Side Includes* (SSI).

In the HTML code of the page, a special SSI tag can be included that Apache will automatically substitute into the HTML when the page is served from the server. The SSI tag looks like this:

```
<!--#echo var="DEVICE_NAME" -->
```

By default, SSI is not enabled on HTML pages. SSI can be enabled in one of several ways.

- To enable SSI for a single page, the extension of the HTML document can be changed to `.shtml`. The `.shtml` extension causes Apache to process the SSI tags inside the HTML content of the document.
- To enable SSI for a directory, a `.htaccess` file can be created in the directory that includes the `Options +Include` directive.
- To enable SSI site-wide, the Apache configuration can be modified.

CGI API

CueServer includes an embedded web server that responds to HTTP requests. In addition to the standard URLs that a user of the Web Interface would see, a special set of URLs are available in CueServer that can be used to run CueScript commands, fetch real-time information from CueServer, set operating parameters and more.

This section describes the various URLs that are available and their function.

Command Throttling (IMPORTANT)

It is very important to employ a throttling mechanism when sending HTTP requests. **New requests should only be sent after a response to the previous request has been received from CueServer.** This allows CueServer to manage requests and responses gracefully and it prevents your code from sending a large number of commands blindly in a very short period of time (for example, moving a slider in a rapid fashion). Not implementing a throttling mechanism could result in many dropped requests and commands and/or performance issues affecting other functions of CueServer due to high CPU usage.

CueServer CGI URLs:

- [exe.cgi](#)
- [get.cgi](#)
- [pcmd.cgi](#)
- [set.cgi](#)

exe.cgi

The `exe.cgi` URL is used to execute CueScript commands on the CueServer.

The typical format of this URL is:

```
http://<ip-of-CueServer>/exe.cgi?cmd=<command>&<optional-parameters>
```

For example, the following URL will execute the command `Cue 1 Go`:

```
/exe.cgi?cmd=Cue+1+Go
```



Note that when a command is URL-encoded, spaces must be changed to plus (+) characters, and other “special” characters must use standard URL escaping methods, for example a \$ character should be encoded as %24. See [Percent Encoding](#) on Wikipedia for more details.

Parameters

- `cmd=<string>`
 - A CueScript command.
 - Special characters must be [Percent Encoded](#)
- `def=<defaultPlayback>` (*optional*)
 - `1` to `32` specifies the playback that the command will default to.
 - If this parameter is not specified, then no change will be made to the default playback for this context (i.e., the playback will remain the same as it was after a previous command was executed within this context).
- `usr=<contextID>` (*optional*)
 - `-1` specifies that a temporary context should be used.
 - `0` specifies the default context (same as used by CueServer Studio).
 - `1` to `4` specifies User 1 thru User 4 contexts (for multi-user input).
 - `5` specifies the Ethernet context.
 - `6` specifies the Serial context.
 - `7` specifies the Rule Actions context.

- If this parameter is not specified, then a temporary context will be used.
-

Examples

CueScript Command	URL
M1	<code>/exe.cgi?cmd=M1</code>
Cue 73 Go	<code>/exe.cgi?cmd=Cue+73+Go</code>
WRITE "Hello World!"	<code>/exe.cgi?cmd=WRITE+%22Hello+World%21%22</code>
Clear, on Playback 3	<code>/exe.cgi?cmd=Clear&def=3</code>
Button 1.5 Off, in Context 0	<code>/exe.cgi?cmd=Button+1.5+Off&usr=0</code>

get.cgi

The `get.cgi` URL is used to fetch information from the CueServer.

The typical format of this URL is:

```
http://<ip-of-cueserver>/get.cgi?req=<request>&<optional-parameters>
```

Depending on the value of the `<request>` parameter, this URL can fetch many different pieces of information from CueServer.

The following variations of the `get.cgi` URL are available:

- [Button Values \[bv\]](#)
- [Command Context \[cc\]](#)
- [CPU SysInfo \[cpu\]](#)
- [Cue Stack Info \[csi\]](#)
- [DMX Input \[in\]](#)
- [DMX Output \[out\]](#)
- [Extended Command Context \[ecc\]](#)
- [Extended Playback Info \[epi\]](#)
- [Fade Engine Data \[fed\]](#)
- [Group Level \[grp\]](#)
- [Hardwired DMX Input \[hdi\]](#)
- [Network Info \[net\]](#)
- [Ping \[ping\]](#)
- [Playback Info \[pi\]](#)
- [Playback Values \[p*\]](#)
- [Preset Zone Info \[pzi\]](#)
- [Record Stream Info \[rs\]](#)
- [System Log \[log\]](#)
- [System Status \[ss\]](#)
- [Time Info \[ti\]](#)
- [Time Status \[ts\]](#)
- [Variables \[var\]](#)
- [Zone Data \[zones\]](#)

Button Values [bv]

This request returns the current state of of the CueServer's front-panel buttons.

! This request is available in CueServer 2 only for compatibility with the original CueServer 1 API. Use of this request is **deprecated** and is not encouraged.

This request only returns the 8-bit indicator state of the first eight buttons of each of the first 64 stations (as this is what CueServer 1 was limited to).

URL:

```
/get.cgi?req=bv
```

Response:

This request will return 520 bytes. Each byte is an 8-bit color value for each of the 8 buttons on the first 64 button stations (stations 1 thru 64) as defined in CueServer. The final eight bytes returned correspond to the built-in station (station 0).

If there is no "Station 1" defined in the current show file, then the built-in buttons (station 0) will also appear in the first 8 bytes of the returned data.

This structure is provided for compatibility applications that are expecting this format of data as it was previously provided by CueServer 1. Note that CueServer 1 only returned 512 bytes, corresponding to its stations 1 thru 64 (there was no "Station 0") in CueServer 1.

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
0xFF	An internal memory error occurred.

Command Context [cc]

This request returns a *Command Context* data structure for the specified command context. This structure contains detailed information about a command context.

! This request is available in CueServer 2 only for compatibility with the original CueServer 1 API. Use of this request is **deprecated** and is not encouraged. Use [Extended Command Context \[ecc\]](#) instead.

The CC selector was originally designed for CS1 and therefore is restricted to fixed point fade times and only up to 512 bits of the select buffer. This selector is provided for compatibility only and is not recommended for use with CS2.

URL:

```
/get.cgi?req=cc
```

Response:

The following data structure will be returned by this request.

```
typedef struct CCRestult {           // (80 bytes)
    uint8_t      curPlayback;        // Current Playback (1..32)
    uint8_t      reserved1;         // -
    uint8_t      curTarget;         // Current Target
    uint8_t      isDMXTarget;       // Is the target a Channel-based
object?
    uint16_t     fadeDownTime;      // Fade down time in
tenth-second increments
    uint16_t     fadeUpTime;        // Fade up time in tenth-second
increments
    uint16_t     delayDownTime;     // Delay down time in
tenth-second increments
    uint16_t     delayUpTime;      // Delay up time in tenth-second
increments
    uint8_t      reserved2[4];      // -
    uint8_t      selectBuf[64];     // Selection buffer
} CCRestult;
```

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
0xFF	An internal shared memory error occurred.
0xFE	An internal memory error occurred.

CPU SysInfo [cpu]

This request returns the CueServer's internal operating system *SysInfo* structure. This structure contains detailed information about the device's uptime, CPU load, RAM usage, processes and more.

URL:

```
/get.cgi?req=cpu
```

Response:

The following data structure will be returned by this request.

```
typedef struct SysInfo {           // (64 bytes)
    uint32_t      uptime;           // Seconds since boot
    uint32_t      loads[3];        // 1, 5, and 15 minutes load
    averages
    uint32_t      totalRAM;         // Total usable main memory size
    uint32_t      freeRAM;         // Available memory size
    uint32_t      sharedRAM;       // Amount of shared memory
    uint32_t      bufferRAM;       // Memory used by buffers
    uint32_t      totalSwap;       // Total swap space size
    uint32_t      freeSwap;        // Swap space still available
    uint16_t      procs;           // Number of current processes
    uint16_t      reserved1;       // -
    uint32_t      totalHigh;       // Total high memory size
    uint32_t      freeHigh;        // Available high memory size
    uint32_t      memUnit;         // Memory unit size in bytes
    uint8_t       reserved2[8];    // -
} SysInfo;
```

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
0xFF	An unspecified error occurred.

Cue Stack Info [csi]

This request returns a *Cue Stack Info* data structure for the specified cue stack. Use this response to determine which cues are active in a given cue stack.

URL:

```
/get.cgi?req=csi&name=<stackName>
```

Parameters:

- `name=<stackName>` (*optional*)
 - The name of the desired cue stack.
 - If this parameter is not supplied, the default cue stack is assumed.

Response:

The following variable-length structure is returned by this request:

```
#define STACK_NAME_BUF_SIZE      16
#define CSI_TYPE_CUES            0
#define CSI_TYPE_PRESETS        1

typedef struct CueStackInfo {
    uint16_t      signature;           // Signature = 'CS'
    int16_t       version;            // Version = 0x0001 (or negative
error code)
    char          stackName[STACK_NAME_BUF_SIZE]; // Name of stack
    uint8_t       type;               // 0 = Cues, 1 = Presets
    uint8_t       playback;           // The playback number (for
presets only)
    uint16_t      count;              // Number of CueID/Status pairs
    uint32_t      data[64];           // Array of CueID/Status pairs
(32 pairs max)
} CueStackInfo;
```



Please note that the actual number of bytes returned by this request only includes the actual number of pairs of `uint32_t` elements in the `data[]` array.

The CueID/Status pairs are included for any cue in the cue stack that is active in a playback fader. The CueID denotes the cue number and the Status value indicates in which playback fader the cue is active in. The CueID may have the value `0x40000000` added to it to indicate that the cue is active but modified.

Errors:

If an error occurs during the processing of the request, only the first four bytes of the above structure will be returned by this URL. The first two bytes will have the “CS” signature and the next two bytes will contain an error value as described by the following table:

Error Value	Description
-1	An internal shared memory error occurred.

DMX Input [in]

This request returns an array of DMX Input channels. The result contains channel values from *both* Ethernet-based DMX input and the hardwired DMX input ports.

URL:

```
/get.cgi?req=in&index=<0..16383>&count=<0..16384>&pad=<0,1>
```

Parameters:

- `index=<0..16383>` (*optional*)
 - Specifies the starting channel index of the returned array of values.
 - If this parameter is not supplied, the default value is `0`.
- `count=<0..16384>` (*optional*)
 - Specifies the number of channels to return values for.
 - If this parameter is not supplied, all channels up to and including the highest configured channel will be returned.
- `pad=<0,1>` (*optional*)
 - If given as `0`, the returned data will *not* be padded, only the requested channels will be returned.
 - If given as `1`, the returned data will be padded with extra zero bytes (`0x00`) to satisfy the `count` parameter, even if those channels do not exist in the current configuration.

Response:

The data returned is an array of bytes, each one corresponding to the requested channels. Each byte ranges from zero (`0x00`) to 100% (`0xFF`).

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
<code>0xFF</code>	An internal shared memory error occurred.

0xFE	A fade engine error occurred.
0xFD	An internal memory error occurred.

DMX Output [out]

This request returns an array of DMX Output channels. The result contains channel values from the final output stage of the playback stack.

URL:

```
/get.cgi?req=out&index=<0..16383>&count=<0..16384>&pad=<0,1>
```

Parameters:

- `index=<0..16383>` (*optional*)
 - Specifies the starting channel index of the returned array of values.
 - If this parameter is not supplied, the default value is `0`.
- `count=<0..16384>` (*optional*)
 - Specifies the number of channels to return values for.
 - If this parameter is not supplied, all channels up to and including the highest configured channel will be returned.
- `pad=<0,1>` (*optional*)
 - If given as `0`, the returned data will *not* be padded, only the requested channels will be returned.
 - If given as `1`, the returned data will be padded with extra zero bytes (`0x00`) to satisfy the `count` parameter, even if those channels do not exist in the current configuration.

Response:

The data returned is an array of bytes, each one corresponding to the requested channels. Each byte ranges from zero (`0x00`) to 100% (`0xFF`).

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
<code>0xFF</code>	An internal shared memory error occurred.

0xFE	An improper channel range was specified.
------	--

Extended Command Context [ecc]

This request returns an *Extended Command Context* data structure for the specified command context. This structure contains detailed information about a command context.

URL:

```
/get.cgi?req=ecc&id=<contextID>
```

Parameters:

- `id=<contextID>` (optional)
 - 0 specifies the default context (same as used by CueServer Studio) [Default].
 - 1 to 4 specifies User 1 thru User 4 contexts (for multi-user input).
 - 5 specifies the Ethernet context.
 - 6 specifies the Serial context.
 - 7 specifies the Rule Actions context.

Response:

The following variable-length data structure will be returned by this request.

```
typedef struct ECCDataV3 {
    uint16_t      signature;           // Signature = 'CS'
    int16_t       version;            // Version = 0x0003 (or negative
error code)
    uint8_t       curPlayback;        // Current Playback (1..32)
    uint8_t       curTarget;         // Current Target
    uint8_t       isDMXTarget;       // Is curTarget a DMX channel
selection?
    uint8_t       reserved[13];      // -
    FadeTimes     fadeTimes;         // Current Fade Times
    char          stackName[STACK_NAME_BUF_SIZE]; // Name of current
playback's stack
    char          zoneName[STACK_NAME_BUF_SIZE]; // Name of current
zone
    uint8_t       variableData[];    // Selection buffers (see below)
// uint16_t      sizeofSelectData;  // Number of bytes in selectData
// uint8_t       selectData[];     // RLE compressed selection
bitmask (max 2064 bytes)
// uint16_t      sizeofMaskData;   // Number of bytes in maskData
```

```
// uint8_t          maskData[];          // RLE compressed mask bitmask
(max 2064 bytes)
// uint16_t         sizeofStationData;   // Number of bytes in stationData
// uint8_t          stationData[];      // RLE compressed station
bitmask (max 129 bytes)
} ECCDataV3;
```

Errors:

If an error occurs during the processing of the request, only the first four bytes of the above structure will be returned by this URL. The first two bytes will have the “CS” signature and the next two bytes will contain an error value as described by the following table:

Error Value	Description
-1	An internal shared memory error occurred.
-2	An invalid contextID was given.

Extended Playback Info [epi]

This request returns one or more *Extended Playback Info* data structures for specified playback faders. This structure contains detailed information about the current status of each playback fader.

URL:

```
/get.cgi?req=epi&id=<playbackID>
```

Parameters:

- `id=<playbackID>`
 - `1 to 32` specifies an individual playback fader to fetch information for.
 - `0` returns a string of multiple data structures for each of the active playback faders.

Response:

The following data structure will be returned by this request.

```
typedef struct EPIData {           // (160 bytes total)
    uint8_t      version;           // Result Version = 0x02
    uint8_t      playback;         // Playback number (1..32)
    uint8_t      flags;            // Flags (0 = Normal, 1 =
Stopped, -1 = Not Installed)
    uint8_t      mode;             // Mode (0 = Merge, 1 =
Override, 2 = Scale, 3 = Pin)
    uint8_t      reserved1[4];     // -

    int32_t      curCueID;         // Current Cue ID
(0..MAX_CUE_NUMBER, -1=CUE_NONE, -2=CUE_ACTIVE_CHANNELS)
    int32_t      nextCueID;       // Next Cue ID to "Go" to
    int32_t      linkCueID;       // Link Cue ID to link to

    FadeTimes    fadeTimes;       // Current fade/split/delay times
    float        followTime;      // Follow time for next cue go
(0 = Do Not Auto-Follow)
    uint8_t      submaster;       // Submaster
    uint8_t      reserved2[3];    // -

    uint32_t     fadeCurTime;     // Fade progress
    uint32_t     fadeTotalTime;   // Fade total time
```

```

float          followTimeRemain;      // Follow progress
float          followTotalTime;      // Follow total time

uint32_t      streamCurTime;        // Stream playback position
(ticks)
uint32_t      streamTotalTime;      // Stream total time (ticks)

uint8_t       reserved3[12];        // -

char          stackName[STACK_NAME_BUF_SIZE]; // Name of current stack
char          curCueName[32];        // Name of current cue
char          nextCueName[32];      // Name of next cue
} EPIData;

```

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
0xFF	An internal memory error occurred.
0xFE	Invalid playback number was specified.

Fade Engine Data [fed]

This request returns a *Fade Engine Data* data structure. This structure contains detailed information about all playbacks, all universes and all ports simultaneously.

URL:

```
/get.cgi?req=fed
```

Response:

The `FadeEngineData` structure is variable length. The header of 16 bytes is followed by a variable number of `EPIData`, `UniverseData`, and `PortData` records. The total length of a maximum of 32 playbacks, 128 universes, and 4 ports is currently 8,240 bytes. This may grow in future versions.

```
typedef struct FadeEngineData { // (16 bytes)
    uint16_t      signature;          // Signature = 'CS'
    int16_t       version;            // Version = 0x0002 (or negative
error code)
    uint8_t       playbacks;          // Number of EPIData records
(0..32)
    uint8_t       playbackSize;       // Size of EPIData record
(currently 160)
    uint8_t       universes;          // Number of UniverseData
records (0..128)
    uint8_t       universeSize;       // Size of UniverseData record
(currently 24 bytes)
    uint8_t       ports;              // Number of PortData records
(0..4)
    uint8_t       portSize;           // Size of PortData record
(currently 8 bytes)
    uint8_t       reserved[6];        // -
    uint8_t       variableData[];     // EPIData, UniverseData, and
PortData records start here
} FadeEngineData;

typedef struct EPIData {             // (160 bytes total)
    uint8_t       version;            // Result Version = 0x02
    uint8_t       playback;           // Playback number (1..32)
    uint8_t       flags;              // Flags (0 = Normal, 1 =
Stopped, -1 = Not Installed)
    uint8_t       mode;               // Mode (0 = Merge, 1 =
Override, 2 = Scale, 3 = Pin)
    uint8_t       reserved1[4];       // -
```

```

    int32_t          curCueID;                // Current Cue ID
(0..MAX_CUE_NUMBER, -1=CUE_NONE, -2=CUE_ACTIVE_CHANNELS)
    int32_t          nextCueID;              // Next Cue ID to "Go" to
    int32_t          linkCueID;              // Link Cue ID to link to

    FadeTimes        fadeTimes;              // Current fade/split/delay times
    float            followTime;             // Follow time for next cue go
(0 = Do Not Auto-Follow)
    uint8_t          submaster;              // Submaster
    uint8_t          reserved2[3];           // -

    uint32_t         fadeCurTime;           // Fade progress
    uint32_t         fadeTotalTime;         // Fade total time

    float            followTimeRemain;       // Follow progress
    float            followTotalTime;       // Follow total time

    uint32_t         streamCurTime;         // Stream playback position
(ticks)
    uint32_t         streamTotalTime;        // Stream total time (ticks)

    uint8_t          reserved3[12];         // -

    char             stackName[STACK_NAME_BUF_SIZE]; // Name of current stack
    char             curCueName[32];         // Name of current cue
    char             nextCueName[32];        // Name of next cue
} EPIData;

typedef struct UniverseData { // (24 bytes)
    uint8_t          version;                // Result Version = 0x02
    uint8_t          universe;               // Universe number (1..128)
    uint16_t         channelIndex;           // Starting channel index
    uint16_t         channelCount;           // Width of universe
    uint8_t          rxProtocol;             // Rx Protocol
    uint8_t          reserved;               // -
    int16_t          rxChannels;             // Rx Channels (0..512, -1)
    uint16_t         rxUniverse;            // Rx Universe
    uint32_t         rxExtra;                // Rx Extra Data (port number
for KiNET v2)
    uint8_t          txProtocol;             // Tx Protocol
    uint8_t          txEnabled;              // Tx Enabled
    uint16_t         txUniverse;            // Tx Universe
    uint32_t         txExtra;                // Tx Extra Data (port number
for KiNET v2)
} UniverseData;

typedef struct PortData { // (8 bytes)
    uint8_t          version;                // Result Version = 0x01
    uint8_t          port;                   // Port number (1..4)

```

```
uint8_t      universe;           // Universe number (1..32)
uint8_t      direction;         // Data direction
uint8_t      led;               // LED Indicator value
uint8_t      reserved1;        // -
int16_t      channels;          // Tx/Rx Channels (0..512, -1)
} PortData;
```

Errors:

If an error occurs during the processing of the request, only the first four bytes of the above structure will be returned by this URL. The first two bytes will have the “CS” signature and the next two bytes will contain an error value as described by the following table:

Error Value	Description
-1	An internal shared memory error occurred.
-2	An internal memory error occurred.

Group Level [grp]

This request returns the current level of a specified channel group.

URL:

```
/get.cgi?req=grp&id=<groupID>&p=<playback>
```

Parameters:

- `id=<groupID>`
 - `0` to `99999` specifies the group ID (number).
- `p=<playback>` (*optional*)
 - `0` specifies that the CueServer's output should be used to query the group's channels.
 - `1` to `32` specifies that a specific playback should be used to query the group's channels.
 - If this parameter is omitted, the CueServer's output will be used to query the group's channels.

Response:

A 16-bit signed value (in network-byte order) is always returned from this request.

If every channel in the group is set to the same level, that level is returned from zero (`0x0000`) to 100% (`0x00FF`).

If the group's channels have mixed values, then -1 (`0xFFFF`) is returned.

Errors:

If an error occurs during the processing of the request, the meaning of the returned value is explained in the following table:

Error Value	Description
-2	An internal shared memory error occurred.
-3	An invalid parameter was specified.

Hardwired DMX Input [hdi]

This request returns the DMX Input data that is present on *only* the Hardwired DMX input ports.

URL:

```
/get.cgi?req=hdi
```

Response:

The following variable-length data structure will be returned by this request.

```
typedef struct DMXInputUniverse {
    uint8_t      universeIndex;      // Index of universe
    uint16_t     channels;           // Number of channels in this
universe
    uint8_t      values[];          // Variable-length array of
channel values
} DMXInputUniverse;

typedef struct DMXInput {
    uint8_t      universeCount;      // Number of universes received
    DMXInputUniverse universe[];    // Structure repeated for each
universe
} DMXInput;
```

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
0xFF	An internal memory error occurred.

Network Info [net]

This request returns a *Net Info* data structure for the CueServer. This structure contains detailed information about the current operating parameters of CueServer's network interfaces.

URL:

```
/get.cgi?req=net
```

Response:

The following data structure will be returned by this request.

```
typedef struct NetInfo {           // (116 bytes)
    uint16_t      signature;        // Signature = 'CS'
    int16_t       version;          // Version = 0x0002 (or negative
error code)

    char          deviceName[32];   // Device Name (hostname)
    uint16_t      switchModel;      // √ Ethernet switch model number
    uint8_t       switchMode;      // √ Ethernet Switch Mode (0 =
Switch, 1 = VLAN)
    uint8_t       physicalPorts;    // √ Number of Ethernet ports
    uint8_t       reserved3[12];    // -

    uint32_t      primaryIP;        // √ Primary interface IP Address
    uint32_t      primarySubnet;    // √ Primary interface Subnet
Mask
    uint32_t      primaryGateway;   // Primary interface Default
Gateway
    uint8_t       primaryMAC[6];    // Primary interface MAC Address
    uint8_t       primaryDHCP;      // Primary interface DHCP Mode
    uint8_t       primaryLinkStatus; // √ Primary interface Link
Status
    uint8_t       reserved1[12];    // -

    uint32_t      secondaryIP;     // √ Secondary interface IP
Address
    uint32_t      secondarySubnet;  // √ Secondary interface Subnet
Mask
    uint32_t      secondaryGateway; // Secondary interface Default
Gateway
    uint8_t       secondaryMAC[6];  // Secondary interface MAC
Address
```



```
uint8_t      secondaryDHCP;      // Secondary interface DHCP Mode
uint8_t      secondaryLinkStatus; // √ Secondary interface Link
Status
uint8_t      reserved2[12];      // -
} NetInfo;
```

Errors:

If an error occurs during the processing of the request, only the first four bytes of the above structure will be returned by this URL. The first two bytes will have the “CS” signature and the next two bytes will contain an error value as described by the following table:

Error Value	Description
-1	An internal shared memory error occurred.
-2	Could not communicate with Ethernet hardware.
-3	Unknown Ethernet switch model.
-4	TCP/IP stack error.

Ping [ping]

This request returns the device's *Auto-Discovery* data string. Use this request to fetch pertinent device information using a TCP (HTTP) connection.

URL:

```
/get.cgi?req=ping
```

Response:

The returned data is an ASCII string. It begins with `#PING` and contains several fields, each separated by a bar character (`|`).

The following is an example ping string from a CueServer:

```
#PING|600001|10.0.1.5|1.5.5|CueServer 2|255.0.0.0|10.0.1.1|0|239|
000000000000|6/30/2017 12:59:59 PM|N|shows/My Show|1|1|0.0.0.0|0.0.0.0|0
```

The bar-separated fields are explained in the following table:

Field	Example	Description
<i>Fields below are present on both CueServer 1 and CueServer 2 models</i>		
1	600001	Serial number (6 characters, may include <i>only</i> numbers and letters, no special characters)
2	10.0.1.5	Primary interface IP address (standard IPv4 notation)
3	1.5.5	Current firmware version (format is <code><number>.<number>.<number>[<dev-stage><number>[<letter>]]</code> , an extreme example would be <code>2.34.567b99z</code>)
4	CueServer 2	Device name (maximum 15 characters)
5	255.0.0.0	Primary interface subset mask (standard IPv4 notation)
6	10.0.1.1	Gateway address (standard IPv4 notation)
7	0	Primary interface DHCP mode (may be 0 or 1)
8	239	Hardware model (see Hardware Model Identifiers)
9	000000000000	Reserved for future use (only used on CueServer 1)

10	6/30/2017 12:59:59 PM	The current device time (MM/DD/YY HH:MM:SS AP)
11	N	Reserved for future use (only used on CueServer 1)
<i>Fields below are only present on CueServer 2 models</i>		
12	shows/My Show	Current show path
13	1	Number of physical Ethernet ports (may be 1 or 2)
14	1	Number of logical Ethernet interfaces (may be 1 or 2)
15	0.0.0.0	Secondary interface IP address (standard IPv4 notation)
16	0.0.0.0	Secondary interface subnet mask (standard IPv4 notation)
17	0	Secondary interface DHCP mode (may be 0 or 1)

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
0xFF	An internal memory error occurred.
0xFE	Invalid playback number was specified.

Playback Info [pi]

This request returns a *Playback Info* data structure for the specified playback fader.

! This request is available in CueServer 2 only for compatibility with the original CueServer 1 API. Use of this request is **deprecated** and is not encouraged. Use [Extended Playback Info \[epi\]](#) instead.

The PI selector was originally designed for CS1 and therefore is restricted to fixed point fade times, cue IDs with only one digit of precision after the decimal point and it is missing several new playback features available in CS2. This selector is provided for compatibility only and is not recommended for use with CS2.

URL:

```
/get.cgi?req=pi&id=<playbackID>
```

Parameters:

- `id=<playbackID>`
 - Specifies the a playback fader number from 1 to 32.

Response:

The following data structure will be returned by this request.

```
typedef struct PlaybackInfo { // (96 bytes total)
    uint8_t playback; // Playback number (1..32)
    uint8_t runMode; // Run Mode (0 = Normal, 1 =
Stopped)
    uint8_t outputLevel; // Output level (0..255)
    uint8_t combineMode; // Combine Mode (0 = Merge, 1 =
Override, 2 = Scale)
    uint16_t fadeTimer; // Remaining fade time in
progress
    uint16_t followTimer; // Remaining follow time in
progress
    uint32_t streamTimer; // Stream playback position

    uint16_t currentCue; // Cue currently playing (0 =
None, 1 = Cue 0.1, -1 = Active Channels)
    uint16_t nextCue; // Next cue (0 = None, 1 = Cue
```

```
0.1)
uint16_t      fadeUpTime;           // Fade Up Time for next cue
uint16_t      fadeDownTime;        // Fade Down Time for next cue
uint16_t      followTime;          // Follow Time for next cue
uint16_t      linkCue;             // Linked cue for next cue

uint8_t       reserved[8];         // Reserved

char          currentName[32];     // Name of current cue
char          nextName[32];       // Name of next cue
} PlaybackInfo;
```

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
0xFF	An internal shared memory error occurred.
0xFE	An invalid playback number was given.

Playback Values [p*]

This request returns the current channel values of a given playback.

URL:

`/get.cgi?req=p1` – Return Playback 1 Values

...

`/get.cgi?req=p32` – Return Playback 32 Values

`/get.cgi?req=po` – Return Playback Output Values

Response:

This request will return twice the number of configured channels in bytes. For example, if 1,024 channels are configured, this request will return 2,048 bytes.

The first half of the returned bytes (equal to the number of configured channels) will be the individual channel values from zero (`0x00`) to 100% (`0xFF`).

The second half of the returned bytes (also equal to the number of configured channels) will be *channel flags* for each channel. Each channel flag byte is interpreted as a set of eight flag bits. The following table defines the meaning of each bit:

Bit	Description
<code>0x01</code>	The channel is active (i.e. not released)
<code>0x02</code>	–
<code>0x04</code>	–
<code>0x08</code>	The channel is disabled (its value does not propagate beyond playback)
<code>0x10</code>	–
<code>0x20</code>	The channel is parked (its value cannot be changed)
<code>0x40</code>	–
<code>0x80</code>	The channel defaults to a full value when (i.e. in Scale Mode)

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
0xFF	An internal shared memory error occurred.
0xFE	An internal memory error occurred.
0xFD	No show is loaded.

Preset Zone Info [pzi]

This request returns a *Cue Stack Info* data structure for the specified preset zone. Use this response to determine which presets are active in a given zone.

URL:

```
/get.cgi?req=pzi&name=<zoneName>
```

Parameters:

- `name=<zoneName>`
 - The name of the desired zone.

Response:

The following variable-length structure is returned by this request:

```
#define STACK_NAME_BUF_SIZE      16
#define CSI_TYPE_CUES            0
#define CSI_TYPE_PRESETS        1

typedef struct CueStackInfo {
    uint16_t      signature;          // Signature = 'CS'
    int16_t       version;           // Version = 0x0001 (or negative
error code)
    char          stackName[STACK_NAME_BUF_SIZE]; // Name of stack
    uint8_t       type;              // 0 = Cues, 1 = Presets
    uint8_t       playback;          // The playback number (for
presets only)
    uint16_t      count;             // Number of CueID/Status pairs
    uint32_t      data[64];          // Array of CueID/Status pairs
(32 pairs max)
} CueStackInfo;
```



Please note that the actual number of bytes returned by this request only includes the actual number of pairs of `uint32_t` elements in the `data[]` array.

The CueID/Status pairs are included for any preset in the zone that is active. The CueID denotes the preset number and the Status value indicates the preset's active state. The following table shows preset states:

Preset Status	Description
1	This preset is active.
2	This preset is modified.

Errors:

If an error occurs during the processing of the request, only the first four bytes of the above structure will be returned by this URL. The first two bytes will have the "CS" signature and the next two bytes will contain an error value as described by the following table:

Error Value	Description
-1	An internal shared memory error occurred.

Record Stream Info [rs]

This request returns a *Record Stream Info* data structure. Use this request to determine the real-time status of a stream being recorded.

URL:

```
/get.cgi?req=rs
```

Response:

The following data structure will be returned by this request.

```
typedef struct RecordStreamInfo { // (16 bytes)
    uint16_t      signature;        // Signature = 'CS'
    int16_t       version;          // Version = 0x0001 (or negative
error code)

    uint32_t      recordTime;       // Record time (in clicks [1/
40th second])
    uint32_t      recordID;         // ID of cue being recorded
    uint8_t       recordState;      // Stream recording state
    uint8_t       reserved[3];     // -
} RecordStreamInfo;
```

Errors:

This request does not return any error codes.

System Log [log]

This request returns the current system log.

URL:

```
/get.cgi?req=log
```

Response:

The *System Log* is returned as plain ASCII text. The length of this text is variable and might be quite large.

Errors:

If an error occurs during the processing of the request, a single byte is returned. The meaning of this byte is explained in the following table:

Error Value	Description
0xFF	The system log file could not be opened.

System Status [ss]

This request returns a *System Status* data structure for the CueServer. This structure contains detailed information about the current status of the device.

URL:

```
/get.cgi?req=ss
```

Response:

The following data structure will be returned by this request.

```
typedef struct SystemStatus { // (256 bytes)
    uint16_t      signature;           // Signature = 'CS'
    int16_t      version;             // Version = 0x0002 (or negative
error code)

    uint8_t      pcbIndicators[8];    // PCB Indicator Data
    uint8_t      functionButtons[24]; // Function Button Indicator
Data (8xRGB)
    char         lcdData[80];         // LCD Display Buffer
    uint32_t     licenseData;         // License Data

    uint8_t      universeActive[16];  // Universe active bits

    uint16_t     logMessages;         // Number of new CueServer log
messages (rolling count)
    uint16_t     importantLogMessages; // Number of unread "important"
CueServer log messages (cleared by "log clear")

    uint32_t     processRunning;      // Bitmask of running processes
    uint8_t      debugFlags;          // Bitmask of debug logging flags
    uint8_t      reserved1[11];       // -

    uint8_t      resChangeSeed[32];   // Array of resSeed values
(indexed by RES_SEED_XXX constants)

    uint16_t     boardID;             // The board ID of the device
[+v1.4.1]

    uint8_t      dmxInputDisabled;    // Is the DMX Input disabled?
[+v1.5.0]
    uint8_t      lcdBacklight;        // LCD backlight level [+v2.0.0]
```

```
uint8_t reserved2[64]; // -  
} SystemStatus;
```

Errors:

If an error occurs during the processing of the request, only the first four bytes of the above structure will be returned by this URL. The first two bytes will have the “CS” signature and the next two bytes will contain an error value as described by the following table:

Error Value	Description
-1	An internal shared memory error occurred.

Time Info [ti]

This request returns a *Time Info* string from the CueServer. This string contains information about the NTP servers and time zone currently in use.

URL:

```
/get.cgi?req=ti
```

Response:

The *Time Info* string contains two fields separated by a bar character (|).

The first field contains a comma-separated list of NTP Servers (only if Automatic time adjustments are enabled). The second field contains the current POSIX time zone.

The following is an example of a *Time Info* string:

```
0.ntp.pool.org,1.ntp.pool.org,2.ntp.pool.org|America/New_York
```

Errors:

This request does not return any error codes.

Time Status [ts]

This request returns a *Time Status* data structure for the CueServer. This structure contains detailed information about the current time, date, astronomical features, time zone and more.

URL:

```
/get.cgi?req=ts
```

Response:

The following data structure will be returned by this request.

```
typedef struct TimeStatus {
    uint16_t          signature;          // Signature = 'CS'
    int16_t           version;           // Version = 0x0001 (or negative
error code)

    uint32_t          seconds;           // Current time seconds
    uint8_t           day;               // Day (0..30)
    uint8_t           month;             // Month (0..11)
    uint8_t           year;              // Year (0 = 1900, 255 = 2155)
    uint8_t           weekday;          // Weekday (0 = Sunday, 1 =
Monday, etc.)
    uint8_t           dst;              // DST (0 = No, 1 = Yes)
    uint8_t           light;            // Current light (0 = Dark, 1 =
Light)
    uint8_t           reserved[2];      // -

    float             offset;           // Offset from GMT/UTC (Hours)
    float             latitude;         // Latitude
    float             longitude;        // Longitude
    uint32_t          sunriseSeconds;    // Sunrise seconds
    uint32_t          sunsetSeconds;    // Sunset seconds

    char              tzAbbreviation[8]; // Time zone abbreviation
("EST") (null-terminated)
    char              tzName[32];       // Time zone full name
("Americas/New York") (null-terminated)
} TimeStatus;
```

Errors:

This response does not return any error codes.

Variables [var]

This request returns the value of one or more *variables*.

URL:

```
/get.cgi?req=var&id=<variableName>
```

Parameters:

- `id=<variableName>`
 - Given a name of a variable (such as `x` or `MyCue`), the value of that single variable will be returned.
 - If `id=*` (an asterisk), then the entire database of user variables will be returned.

Response:

In the case where a single variable value is being requested, the actual value will be returned.

In the case where the variable database is being requested, an array of null-terminated (C-style) strings is returned. The strings are in pairs of then . If the database contains non-volatile variables, they will be at the end of the database, separated by a `*,*` pair.

Errors:

If an error occurs during the processing of the request, only a single byte will be returned by this URL. The value of the returned byte is explained in the following table:

Error Value	Description
<code>0xFF</code>	An internal memory error occurred.

Zone Data [zones]

This request returns a *Zones Data* data structure. Use this response to determine what zones are defined and their playbacks, join groups, and active presets.

URL:

```
/get.cgi?req=zones
```

Response:

The following variable-length structure is returned by this request:

```
typedef struct ZonesData {
    uint16_t      signature;           // Signature = 'CS'
    int16_t       version;            // Version = 0x0001 (or negative
error code)
    uint8_t       reserved[2];        // -
    uint16_t      zoneCount;          // Number of zones
    ZoneRecord    zones[];           // Variable array of zone records
} ZonesData;

#define STACK_NAME_BUF_SIZE      16

typedef struct ZoneRecord {
    char          name[STACK_NAME_BUF_SIZE]; // Name of zone
    uint8_t       playbackIndex;           // Playback index
    uint8_t       joinGroup;              // Join group
    uint16_t      count;                  // Number of PresetID/Status
pairs
    uint32_t      data[];                 // Array of PresetID/Status
pairs (32 pairs max)
} ZoneRecord;
```



Please note that the actual number of bytes returned by this request only includes the actual number of ZoneRecords, each of which only includes the actual number of pairs of `uint32_t` elements in the `data[]` array.

The PresetID/Status pairs are included for any preset in the zone that is active. The PresetID denotes the preset number and the Status value indicates the preset's active state. The following table shows preset states:

Preset Status	Description
1	This preset is active.
2	This preset is modified.

Errors:

If an error occurs during the processing of the request, only the first four bytes of the above structure will be returned by this URL. The first two bytes will have the "CS" signature and the next two bytes will contain an error value as described by the following table:

Error Value	Description
-1	An internal shared memory error occurred.

pcmd.cgi

The `pcmd.cgi` URL is used to translate (or “parse”) a CueScript string into an English language string.

The typical format of this URL is:

```
http://<ip-of-CueServer>/pcmd.cgi?cmd=<command>
```

For example, the following URL will translate the CueScript `Q1G`:

```
/pcmd.cgi?cmd=Q1G
```

This URL will return an English language string that is the expanded version of the given CueScript.

In the above example, the returned string will be `Cue 1 Go`.



Note that when a command is URL-encoded, spaces must be changed to plus (+) characters, and other “special” characters must use standard URL escaping methods, for example a \$ character should be encoded as %24. See [Percent Encoding](#) on Wikipedia for more details.

Parameters

- `cmd=<string>`
 - A CueScript command.
 - Special characters must be [Percent Encoded](#)

Examples

CueScript Command	URL	Returned Value
<code>M1</code>	<code>/pcmd.cgi?cmd=M1</code>	<code>Macro 1</code>
<code>Q73G</code>	<code>/pcmd.cgi?cmd=Q73G</code>	<code>Cue 73 Go</code>
<code>B1.5OFF</code>	<code>/pcmd.cgi?cmd=B1.5OFF</code>	<code>Button 1.5 Off</code>

set.cgi

The `set.cgi` URL is used to store information into the CueServer.

The typical format of this URL is:

```
http://<ip-of-cueserver>/set.cgi?dst=<destination>&<optional-parameters>
```

Depending on the value of the `<destination>` parameter, this URL can store many different pieces of information to the CueServer.

The following variations of the `set.cgi` URL are available:

- [Audio Properties \[audio\]](#)
- [LCD Properties \[lcd\]](#)
- [Network Properties \[net\]](#)
- [Time Properties \[time\]](#)
- [Station Color Properties \[stcol\]](#)

Audio Properties [audio]

This request sets various audio system properties.

URL:

```
/set.cgi?dst=audio<optional-parameters>
```

Optional Parameters:

- `volume=<0..100>` (*optional*)
 - This parameter (if present) sets the master audio output volume.
 - Valid range is from 0 to 100.

Response:

A single byte is returned. The following table explains the possible return values.

Result	Description
<code>0x00</code>	The operation was successful.
<code>0xFF</code>	The volume failed to be set properly.

Examples:

Function	URL
Set volume to 75%	<code>/set.cgi?dst=audio&volume=75</code>

LCD Properties [lcd]

This request sets various LCD Display properties.

URL:

```
/set.cgi?dst=lcd<optional-parameters>
```

Optional Parameters:

- `backlight=<0..255>` (*optional*)
 - This parameter (if present) sets the LCD Display's backlight brightness.
 - Valid range is from 0 to 255.
- `field=<1..4>&id=<0..16>` (*optional*)
 - This parameter group (if present) sets one of the four quadrants of the LCD Display to one of the built-in display functions.
 - Valid fields are from 1 to 4 (see below).
 - Valid display function ID is from 0 to 16 (see below).

LCD Field	Description
1	Top-Left
2	Top-Right
3	Bottom-Left
4	Bottom-Right

LCD Function	Description
0	Blank
1	Device Name
2	Long Date + 12-Hour Time
3	Short Date + 12-Hour Time
4	Long Date + 24-Hour Time
5	Short Date + 24-Hour Time

6	Long Date
7	Short Date
8	12-Hour Time
9	24-Hour Time
10	User String
11	IP Address
12	Timecode
13	I/O Status
14	CPU Load
15	Show Path
16	Show Name

Response:

A single byte is returned. The following table explains the possible return values.

Result	Description
0x00	The operation was successful.
0xFE	An internal shared memory error occurred.

Examples:

Function	URL
Set the backlight to 75%	<code>/set.cgi?dst=lcd&backlight=191</code>
Set the Top-Left quadrant of the LCD to I/O Status	<code>/set.cgi?dst=lcd&field=1&id=13</code>

Network Properties [net]

This request sets various Network properties.

URL:

```
/set.cgi?dst=net<optional-parameters>
```

Optional Parameters:

- `name=<string>` (*optional*)
 - This parameter (if present) sets the device's name.
 - Maximum length of this string is 15 characters.
- `ipA=<ipAddress>` (*optional*)
 - This parameter (if present) sets the device's primary IP Address.
- `subA=<ipAddress>` (*optional*)
 - This parameter (if present) sets the device's primary subnet mask.
- `dhcpA=<0,1>` (*optional*)
 - This parameter (if present) sets the device's primary DHCP Mode.
 - This parameter may be set to `0` or `1`.
- `ipB=<ipAddress>` (*optional*)
 - This parameter (if present) sets the device's secondary IP Address.
 - This parameter is of no use on a device with a single Ethernet port.
- `subB=<ipAddress>` (*optional*)
 - This parameter (if present) sets the device's secondary subnet mask.
 - This parameter is of no use on a device with a single Ethernet port.
- `dhcpB=<0,1>` (*optional*)
 - This parameter (if present) sets the device's secondary DHCP Mode.
 - This parameter may be set to `0` or `1`.
 - This parameter is of no use on a device with a single Ethernet port.

- `gateway=<ipAddress>` (*optional*)
 - This parameter (if present) sets the device's gateway address.
- `interfaces=<1,2>` (*optional*)
 - This parameter (if present) sets the device's number of interfaces.
 - A setting of `1` puts the device into a mode where each physical port is connected to a built-in switch connected to a single interface.
 - A setting of `2` puts the device into a mode where each of the two physical ports become their own separate interfaces, each with their own IP addresses.
 - This parameter has no effect on a device with a single Ethernet port.

Response:

A single byte is returned. The following table explains the possible return values.

Result	Description
<code>0x00</code>	The operation was successful.
<code>0xFF</code>	The network interfaces file could not be read.

Examples:

Function	URL
Set the device name to "My CueServer"	<code>/set.cgi?dst=net&name=My+CueServer</code>
Set the primary IP parameters	<code>/set.cgi?dst=net&ipA=10.0.1.5&subA=255.0.0.0&dhcpA=0</code>

Time Properties [time]

This request sets various Network properties.

URL:

```
/set.cgi?dst=time&<optional-parameters>
```

Optional Parameters:

- `ntpList=<string>` (*optional*)
 - This parameter (if present) sets the device's list of NTP servers.
 - This parameter should not be used if the time is being set manually.
- `year=<1900..2199>` (*optional*)
 - This parameter (if present) sets the device's clock's year.
- `month=<1..12>` (*optional*)
 - This parameter (if present) sets the device's clock's month.
- `day=<1..31>` (*optional*)
 - This parameter (if present) sets the device's clock's day.
- `hour=<0..23>` (*optional*)
 - This parameter (if present) sets the device's clock's hour.
- `minute=<0..59>` (*optional*)
 - This parameter (if present) sets the device's clock's minute.
- `second=<0..59>` (*optional*)
 - This parameter (if present) sets the device's clock's second.
- `timeZone=<timeZone>` (*optional*)
 - This parameter (if present) sets the device's time zone.
 - The time zone string must be from the [tzdatabase](#) (i.e.: "America/New_York").

Response:

A single byte is returned. The following table explains the possible return values.

Result	Description
0x00	The operation was successful.
0xFF	A required parameter was missing (i.e.: month was given but day was not)

Examples:

Function	URL
Set the time manually to 6/30/2017 1:00:42 PM	<code>/set.cgi?dst=time&year=2017&month=6&day=30&hour=13&minute=0&second=42</code>
Set the time via list of NTP servers	<code>/set.cgi?dst=time&ntpList=pool0.ntp.org%0Dpool1.ntp.org%0Dpool2.ntp.org</code>
Set the time zone	<code>/set.cgi?dst=time&timeZone=America%2FNew_York</code>

Station Color Properties [stcol]

This request sets various Station Indicator Color properties.

URL:

```
/set.cgi?dst=stcol<optional-parameters>
```

Optional Parameters:

- `station=<-1,0..1000>` *(optional)*
 - This parameter (if present) chooses which station to operate on.
 - If this parameter is not preset or `-1` is specified, then this function will operate on the “global” station settings.
- `button=<-1,0..1000>` *(optional)*
 - This parameter (if present) chooses which button to operate on.
 - If this parameter is not preset or `-1` is specified, then this function will operate on all buttons of the specified station.
- `<colorName>=<rgbColor>` *(optional)*
 - The `colorName` parameter may be any of `user1`, `user2`, `user3`, `user4`, `on`, `off`, `mixed`, `locked`.
 - The `rgbColor` parameter may be a 6-digit or 8-digit hexadecimal color. For example, Red would be expressed as `FF0000`, and a Dark Blue would be `000033`.
 - One or more color parameters may be specified in the same URL.

Response:

A single byte is returned. The following table explains the possible return values.

Result	Description
<code>0x00</code>	The operation was successful.
<code>0xFF</code>	An internal shared memory error occurred.

Examples:






















Function	URL
Set the "On" color of Button 2 of Station 3 to Green	<code>/set.cgi?dst=stcol&station=3&button=2&on=00FF00</code>
Set the "Off" color of all buttons on Station 4 to Dark Yellow	<code>/set.cgi?dst=stcol&station=4&on=222200</code>
Set the "User 1" and "User 2" colors of all buttons	<code>/set.cgi?dst=stcol&user1=FF8800&user2=00FF44</code>




























Show File Format

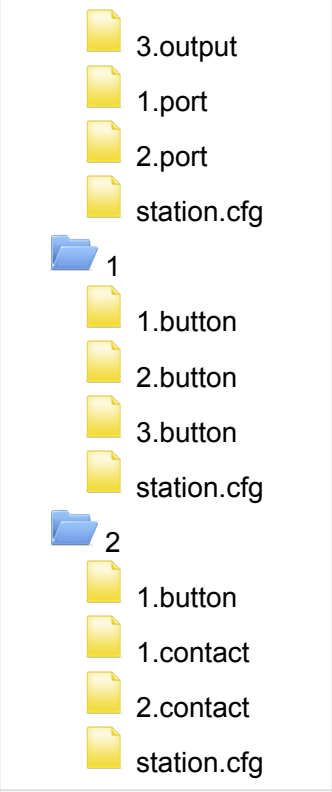
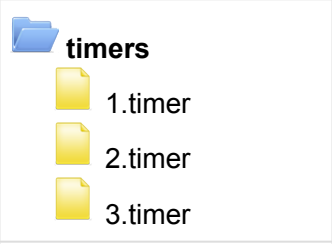
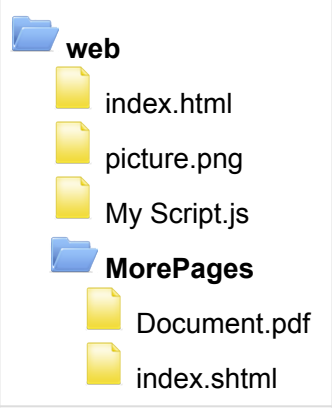
Directory Structure

The show file for CueServer is arranged in a directory structure.

The following table illustrates the directory structure of a typical show file:

File	Description
 audio  chime.wav  MySound.ogg  My Music.mp3	<p>The audio directory contains audio files. Audio files can be any type supported by the system.</p>
 cues  1.00.cue  101.50.cue  123456.78.cue  MyStack  1.00.cue  2.10.cue	<p>The cues directory contains both cue resources and/or cue stack directories.</p> <p>Cue file names use the same number as the cue with two decimal places of precision. Cue files end with the .cue extension. Valid cue numbers range from 0.00 to 999999.99.</p> <p>Each cue stack is its own directory in the cues directory. Cue stack names are limited to 15 characters any may not contain spaces. Cues in cue stack directories follow the same rules as cues in the root cues directory.</p>
 dmxtriggers  1.dmxtrigger  2.dmxtrigger  3.dmxtrigger	<p>The dmxtriggers directory contains DMX Input Trigger resources. DMX Trigger file names are based on the resource ID followed by the .dmxtrigger extension.</p> <p>See DMXTrigger Resource for file format details.</p>
 groups  1.group  2.group  3.group	<p>The groups directory contains Group resources. Group file names are based on the resource ID followed by the .group extension.</p> <p>See Group Resource for file format details.</p>
 macros  1.macro	<p>The macros directory contains Macro resources. Macro file names are based on the resource ID followed by the .macro extension.</p> <p>See Macro Resource for file format details.</p>

 2.macro  3.macro	
 presets  Conference  0.zone  1.preset  2.preset  3.preset  Lobby  0.zone  1.preset  2.preset	<p>The presets directory contains zone directories.</p> <p>Each zone includes a 0.zone zone configuration file.</p> <p>Within each zone, Preset file names use the same number as the preset. Preset files end with the .preset extension. Valid preset numbers range from 1 to 999.</p> <p>Each additional zone is its own directory in the root directory. Zone names are limited to 15 characters any may not contain spaces. Each zone directory also contains a 0.zone configuration file and its own set of Preset resources.</p>
 rules  1.rule  2.rule  3.rule	<p>The rules directory contains Rule resources. Rule file names are based on the resource ID followed by the .rule extension.</p>
 show.cfg	<p>This is the main show configuration file. It is always included in the root level off the show directory. The show.cfg file contains settings for DMX patching, physical location, LCD display, playbacks, global show preferences and more.</p> <p>See show.cfg for details on this file format.</p>
 stations  0  1.button  2.button  3.button  1.contact  2.contact  3.contact  1.output  2.output	<p>The stations directory contains sub-directories for each station. Station directories are named with the ID number of the station. Station IDs always start at 0.</p> <p>Individual resources for Buttons, Contacts, Outputs, and Serial Ports are included in each station directory. The ID numbers for each resource and number of resources for each type is dictated by the type and configuration of station. These files end with the .button, .contact, .output, and .port extensions.</p> <p>Each station directory includes a station.cfg file that defines the station properties.</p> <p>A station with ID 0 is always present and represents the “built-in” station that corresponds to the front-panel buttons and included contact closures, outputs and serial ports.</p>

	
	<p>The timers directory contains Timer resources.</p> <p>Timer file names are based on the resource ID followed by the <code>.timer</code> extension.</p>
	<p>The web directory is served by the embedded Apache web server. This directory may contain any combination of files and folders as needed.</p>

Configuration Files

- [show.cfg](#)

show.cfg

The `show.cfg` file is a simple text file in [LibConfig](#) format. This file is located in the root of the show's file system directory structure.

The following elements appear in the `show.cfg` file:

<code>astro</code>	Dictionary (astro)	Contains a dictionary of Astronomical Time configuration elements.
<code>audio</code>	Dictionary (audio)	Contains a dictionary of Audio configuration elements.
<code>description</code>	String	A textual description for the show file that appears in the Settings/Description panel of CueServer Studio.
<code>dmx</code>	Dictionary (dmx)	Contains a dictionary of DMX configuration elements.
<code>lcd</code>	Dictionary (lcd)	Contains a dictionary of LCD Display configuration elements.
<code>name</code>	String	<i>DEPRECATED: This element used to contain the name of the show. Now, the actual name of the show file directory is used as the show's name.</i>
<code>preferredModel</code>	Integer	Contains the Model ID number of the "preferred" CueServer model for this show file. CueServer Studio uses this ID to display the correct UI for specific configuration elements that are different for each model. <ul style="list-style-type: none"> • 0 = Any • 160 = CS-900 • 176 = CS-920 • 224 = CS-940
<code>station</code>	Dictionary (station)	Contains a dictionary of Station related configuration elements.

'astro' Dictionary

The following elements appear in the `astro` dictionary:

<code>latitude</code>	Float	The device's latitude coordinate (in degrees, for example 34.24963).
<code>longitude</code>	Float	The device's longitude coordinate (in degrees, for example -84.05723).

<code>offset</code>	Float	<i>DEPRECATED: This element had previously been used to specify the device's offset from UTC/GMT. Now the time zone offset is retrieved from the system's global clock settings.</i>
---------------------	-------	--

'audio' Dictionary

The following elements appear in the `audio` dictionary:

<code>volume</code>	Integer	The audio output volume (from 0 to 100).
---------------------	---------	--

'dmx' Dictionary

The following elements appear in the `dmx` dictionary:

<code>channelCount</code>	Integer	The number of channels being output (must be equal to <code>universeCount * 512</code>).
<code>playbackCount</code>	Integer	The number of playbacks configured.
<code>playbacks</code>	Dictionary Array (playbacks)	An array containing the configuration dictionaries for each playback.
<code>ports</code>	Dictionary Array (ports)	An array containing the configuration dictionaries for each port.
<code>universeCount</code>	Integer	The number of CueServer universes configured (must be equal to <code>channelCount / 512</code>).
<code>universes</code>	Dictionary Array (universes)	An array containing the configuration dictionaries for each universe.

'dmx/playbacks' Dictionary

The following elements appear in the `dmx/playbacks` dictionary:

<code>name</code>	String	The descriptive name for the playback (may be blank).
<code>mode</code>	Integer	An ID for the default mode of the playback. <ul style="list-style-type: none"> • 0 = Merge • 1 = Override • 2 = Scale

- | | | |
|--|--|-----------|
| | | • 3 = Pin |
|--|--|-----------|

'dmx/ports' Dictionary

The following elements appear in the `dmx/ports` dictionary:

<code>direction</code>	Integer	The input/output "direction" for the port. [OPTIONAL] <ul style="list-style-type: none"> • 0 = Off • 1 = Input • 2 = Output
<code>universe</code>	Integer	The CueServer universe number (1..32) corresponding to this port. <i>Previous versions of CueServer used universe 0 to mean that the port was disabled, which is deprecated. Instead use the direction element to indicate that a port is disabled.</i>

'dmx/universes' Dictionary

The following elements appear in the `dmx/universes` dictionary:

<code>channels</code>	Integer	The number of channels in this universe (1-512). If this element is missing, the default is 512.
<code>name</code>	String	Descriptive name for the universe (may be blank).
<code>rx_port</code>	Integer	The port number for KiNET v2 protocol. <i>Only used for the KiNET v2 protocol.</i>
<code>rx_priority_high</code>	Integer	High limit of the priority range of received packets (0-200). <i>Only used for the sACN protocol.</i>
<code>rx_priority_low</code>	Integer	Low limit of the priority range of received packets (0-200). <i>Only used for the sACN protocol.</i>
<code>rx_protocol</code>	Integer	Number of protocol to use for receiving DMX over Ethernet for this universe. <ul style="list-style-type: none"> • 0 = None • 1 = sACN • 2 = KiNet v1 • 3 = KiNet v2 • 4 = Art-Net
<code>rx_universe</code>	Integer	External universe number to receive channels from. <ul style="list-style-type: none"> • sACN = Universe 1..63999

		<ul style="list-style-type: none"> • KiNET = Universe 0..2147483647, -1 = All • Art-Net = Port-Address 0..32767
<code>tx_flags</code>	Integer	<p>The transmit flags for KiNET v2.</p> <ul style="list-style-type: none"> • 0x01 = Chromatic • 0x02 = Sync Packets
<code>tx_ip</code>	String	IP Address to transmit packets to. <i>Only used for KiNet and Art-Net protocols.</i>
<code>tx_mode</code>	Integer	<i>DEPRECATED: This element had previously been used to specify the Art-Net broadcast mode. Now the broadcast mode is implied by special values in the <code>tp_ip</code> element.</i>
<code>tx_port</code>	Integer	The port number for KiNET v2 protocol. <i>Only used for the KiNET v2 protocol.</i>
<code>tx_portout</code>	String	<i>DEPRECATED: This element had previously been used to list KiNET v2 "portout" parameters.</i>
<code>tx_priority</code>	Integer	Priority level for transmitted packets (0-200). <i>Only used for the sACN protocol.</i>
<code>tx_protocol</code>	Integer	<p>Number of protocol to use for transmitting DMX over Ethernet for this universe.</p> <ul style="list-style-type: none"> • 0 = None • 1 = sACN • 2 = KiNet v1 • 3 = KiNet v2 • 4 = Art-Net
<code>tx_universe</code>	Integer	<p>External universe number to transmit channels to.</p> <ul style="list-style-type: none"> • sACN = Universe 1..63999 • KiNET = Universe 0..2147483647, -1 = All • Art-Net = Port-Address 0..32767

'lcd' Dictionary

The following elements appear in the `lcd` dictionary:

<code>top-left</code>	Integer	<p>The ID of the display element that will appear in the top-left of the LCD Display. Possible values include:</p> <ul style="list-style-type: none"> • 0 = None • 1 = Device Name • 2 = Long Time/Date 12 Hour • 3 = Short Time/Date 12 Hour • 4 = Long Time/Date 24 Hour
-----------------------	---------	---

		<ul style="list-style-type: none"> • 5 = Short Time/Date 24 Hour • 6 = Long Date • 7 = Short Date • 8 = 12 Hour Time • 9 = 24 Hour Time • 10 = User String • 11 = IP Address • 12 = Timecode • 13 = I/O Status • 14 = CPU Load • 15 = Show Path • 16 = Show Name
<code>top-right</code>	Integer	The ID of the display element that will appear in the top-right of the LCD Display (uses same constants as above).
<code>bottom-left</code>	Integer	The ID of the display element that will appear in the bottom-left of the LCD Display (uses same constants as above).
<code>bottom-right</code>	Integer	The ID of the display element that will appear in the bottom-right of the LCD Display (uses same constants as above).
<code>backlight</code>	Integer	The brightness level of the LCD backlight (from 0 to 255).

'station' Dictionary

The following elements appear in the `station` dictionary:

<code>onColor</code>	String	The default onColor for stations, represented as a hexadecimal string. For example, an Orange color would be represented as <code>FF8800</code> . Colors may optionally have an Alpha component, which is used to denote the flash pattern. For example, Green with a flash pattern of 3 would be represented as <code>00FF0003</code> .
<code>offColor</code>	String	The default offColor for stations, represented as a hexadecimal string.
<code>mixedColor</code>	String	The default mixedColor for stations, represented as a hexadecimal string.
<code>lockedColor</code>	String	The default lockedColor for stations, represented as a hexadecimal string.
<code>user1Color</code>	String	The default user1Color for stations, represented as a hexadecimal string.
<code>user2Color</code>	String	The default user2Color for stations, represented as a hexadecimal string.
<code>user3Color</code>	String	The default user3Color for stations, represented as a hexadecimal string.
<code>user4Color</code>	String	The default user4Color for stations, represented as a hexadecimal string.

Resource Structures

- [Cue Resource](#)
- [DMXTrigger Resource](#)
- [Group Resource](#)
- [Marco Resource](#)

Cue Resource

A Cue (or Preset) Resource is a binary file with a format described by the following C structures and constants:

```
//
// -----
//      Cue Resource (Public)
// -----

// Constants
#define CUEID_MIN          0           // Minimum cueID corresponds to
Cue 0.00
#define CUEID_MAX          999999999 // Maximum cueID corresponds to
Cue 999,999.99
#define CUEID_MULTIPLIER   100        // CueID is 100x the natural cue
number
#define PRESETID_MAX        999        // presetIDs do not use decimal
numbers
#define CUE_OFFSET_STREAM_DATA 0x1000 // This is a fixed file position
for the start of streaming data blocks

// Resource Identifiers
#define CUE_RESTYPE        'C'        // Resource type identifier
#define CUE_RESVERS        '1'        // Version 1 identifier

typedef struct Cue {
    // -----
    uint8_t      resType;           // (0x00) Resource type (Cue = 'C')
    uint8_t      resVers;          // (0x01) Resource version (Cue = '1')
    uint8_t      cueType;          // (0x02) Cue Type
    uint8_t      cueFlags;         // (0x03) Cue flags
    FadeTimes    fadeTimes;        // (0x04) Fade times (up/down/delay)
    float        followTime;       // (0x14) Follow time (0 = none)
    int32_t      linkCueID;        // (0x18) Link Cue ID (-1 = none)
    uint32_t     reserved1;        // (0x1C)
    // -----
    uint32_t     streamDuration;    // (0x20) Total time of stream (clicks
[40Hz])
    uint32_t     streamTrimStart;  // (0x24) Number of clicks into stream
to start playback
    uint32_t     streamTrimEnd;    // (0x28) Number of clicks from end of
stream to end playback
    uint8_t      streamMode;       // (0x2C) Auto-Follow/Loop/Hold/Release
    uint8_t      reserved2[3];     // (0x2D) -
}
```

```

// -----
uint8_t      reserved3[13];      // (0x30) -
uint8_t      ruleCount;         // (0x3D) Number of rules in rules[]
uint16_t     channels;         // (0x3E) Channel count (must be
multiple of 8)
uint8_t      mask[];           // (0x40) Bitmask (size is channels/8)
// uint8_t     levels[*];       // (????) Channel values (size is
channel count) [This field has a size of zero for streaming cues!]
// char       name[*];         // (????) Cue Name (c-string)
// char       action[*];       // (????) CueScript action (c-string)
[deprecated; must include termination byte]
// char       rules[*];        // (????) Rules (c-string)
// ----- NULL PADDING FOR STREAMING CUES ONLY -----
// char       streamData[*];    // (0x1000) Streaming Data Starts at
0x1000 (CUE_OFFSET_STREAM_DATA)
// -----
} Cue;

// cueType
#define CUE_TYPE_NORMAL          0x00      // This cue is a normal cue
#define CUE_TYPE_STREAMING      0x01      // This cue is a streaming cue
#define CUE_TYPE_PRESET         0x02      // This cue is a preset

// streamMode
#define STREAM_MODE_FOLLOW      0x00      // At the end of this stream,
follow to the next cue
#define STREAM_MODE_LOOP        0x01      // At the end of this stream,
loop back to the beginning of the stream
#define STREAM_MODE_HOLD        0x02      // At the end of this stream,
hold the final channel values
#define STREAM_MODE_RELEASE     0x03      // At the end of this stream,
release all channel values

//
-----
//      Fade Times (Public)
//
-----

typedef struct FadeTimes {          // 16 bytes
    float      upTime;             // (0x00) Fade Up Time (in
seconds)
    float      upDelay;           // (0x04) Fade Up Delay (in
seconds)
    float      downTime;         // (0x08) Fade Down Time (in
seconds)
    float      downDelay;        // (0x0C) Fade Down Delay (in
seconds)
} FadeTimes;

```

Additionally, if the Cue is a *streaming cue*, then a series of “stream blocks” will be written to the file starting at file offset `0x1000`. Each stream block has the format as described by the following C structures and constants:

```
//
-----
//      Streaming Cues (Public)
//
-----

typedef struct StreamBlockHeader0 {      // (4 bytes)
    uint16_t      universeIndex;        // Index of universe (0..127)
    uint16_t      reserved;              // -
} StreamBlockHeader0;

typedef struct StreamBlockHeader1 {      // (4 bytes)
    uint32_t      endToken;              // 'END!'
} StreamBlockHeader1;

typedef struct StreamBlockHeader2 {      // (4 bytes)
    uint16_t      channelIndex;          // Index of first channel of
changes (0..511)
    uint16_t      channelCount;          // Channels in update
(1..512)
} StreamBlockHeader2;

typedef struct StreamBlockHeader {       // (16 bytes)
    uint16_t      identifier;             // Constant = "SB"
    uint8_t       blockType;              // 0 = One universe of data
    uint8_t       reserved1;              // -
    uint16_t      reserved2;              // -
    uint16_t      blockSize;              // Size of data after header
    uint32_t      time;                   // Timestamp for block
(expressed in 1/100 second units)

    union {
        StreamBlockHeader0 type0;         // StreamBlockHeader0
        StreamBlockHeader1 type1;         // StreamBlockHeader1
        StreamBlockHeader2 type2;         // StreamBlockHeader2
    } info;
} StreamBlockHeader;

// Constants
#define STREAM_BLOCK_ID          0x4253    // 'SB'
#define STREAM_END_TOKEN        0x21444E45 // 'END!'

// blockType
```

```
#define STREAM_BLOCK_UNIVERSE    0           // Single universe
#define STREAM_BLOCK_END        1           // End Block
#define STREAM_BLOCK_RANGE      2           // Range of channels
```

DMXTrigger Resource

A DMXTrigger Resource is a binary file with a format described by the following C structure and constants:

```
#define DMXTRIG_RESTYPE      'D'
#define DMXTRIG_RESVERS     '1'

#define MAX_DMXTRIG_COUNT   100      // Maximum number of DMX Triggers loaded at
once

typedef struct DMXTrigRange {          // 6 bytes
    uint16_t      rangeLow;           // Low end of range
    uint16_t      rangeHigh;         // High end of range
    uint8_t       reserved[2];       // -
} DMXTrigRange;

typedef struct DMXTrigSubmaster {     // 6 bytes
    uint16_t      playback;           // Playback index
    uint16_t      reserved1;         // -
    bool          invert;             // Invert input
    uint8_t       reserved2;         // -
} DMXTrigSubmaster;

typedef struct DMXTriggerResource {
    // -----
    uint8_t       resType;            // (0x00) Resource type (DMXTrig =
'T')
    uint8_t       resVers;           // (0x01) Resource version (DMXTrig
= '1')
    uint8_t       mode;               // (0x02) Mode (0=Range,
1=Submaster, 2=Continuous, etc.)
    uint8_t       reserved1;         // (0x03) -
    uint16_t      channel;           // (0x04) DMX Channel (0..16383)

    union {
        struct DMXTrigRange      range;           // (0x06) Data for Range
        struct DMXTrigSubmaster  submaster;       // (0x06) Data for Submaster
    } info;

    uint8_t       reserved2[3];       // (0x0C) -
    uint8_t       ruleCount;          // (0x0F) Number of rules in
variableParams
    // -----
    char          variableParams[];   // (0x10) Beginning of variable
"C-String" parameters
    // char          name[];           // (0) Name (c-string)
    // char          rules[][];        // (1+) Rules list (c-string list)

```

```
    // -----  
} DMXTriggerResource;  
  
// Modes  
#define DMXTRIG_MODE_RANGE          0    // Trigger occurs within a range of  
channel values  
#define DMXTRIG_MODE_SUBMASTER    1    // Trigger directly controls a  
submaster value  
  
// Variable Strings  
#define DMXTRIG_STR_NAME            0  
#define DMXTRIG_STR_RULES          1
```

Group Resource

A Group Resource is a binary file with a format described by the following C structure and constants:

```
#define GROUP_RESTYPE          'G'
#define GROUP_RESVERS         '1'

typedef struct GroupResource {
    // -----
    uint8_t      resType;          // (0x00) Resource type (Group = 'G')
    uint8_t      resVers;         // (0x01) Resource version (Group = '1')
    uint16_t     maskBytes;       // (0x02) Number of bytes in mask
    uint8_t      reserved1[12];   // (0x04) -
    // -----
    uint8_t      variableParams[]; // (0x10) Beginning of variable
parameters
// uint8_t      mask[*];          // (????) Bitmask (size is maskBytes)
// char         name[*];         // (????) Group Name (c-string)
    // -----
} GroupResource;
```


Marco Resource

A Macro Resource is a binary file with a format described by the following C structure and constants:

```
typedef struct MacroResource {
    uint8_t      resType;
    uint8_t      resVers;
    uint8_t      showInMenu;
    uint8_t      reserved[13];
    // -----
    char         variableParams[];      // (0x10) Beginning of variable
"C-String" parameters
    // char      name[];                // (0) Name (c-string)
    // char      script[];              // (1) Macro script (c-string)
    // -----
} MacroResource;

// Variable Strings
#define MACRO_STR_NAME          0
#define MACRO_STR_SCRIPT       1
```

Hardware Model Identifiers

One of the fields in the Ping response string from a CueServer is the *Hardware Model Identifier*.

This number is a 16-bit value divided into several fields. When looking at this value in hexadecimal, its digits are broken into the following meanings:

Value	Description
0xWXYZ	<p>W = Reserved for future use, should be 0</p> <p>X = Hardware Revision, see below (0 to F)</p> <p>Y = Hardware Platform, see below (0 to F)</p> <p>Z = Hardware Variant, see below (0 to F)</p>

Hardware Revision

A single hexadecimal digit from 0 to F corresponds to Hardware Revision "A" through "P".

Note that CueServer 1 products do not report their hardware revision, and therefore they always return 0 in this field.

Hardware Platform

Hex Digit	Description
0	CS-8xx Series (model indicated by Hardware Variant field)
A	CS-900 CueServer 2 Pro
B	CS-920 CueServer 2 Mini
E	CS-940 CueServer 2 DIN
All others	Reserved for future use

Hardware Variant

For the CueServer 1 series, this field indicates the specific model of CueServer:

Hex Digit	Description
0	Unknown Model
1	CS-800 CueServer Pro
2	CS-810 CueServer Mini
3	CS-820 CueServer Mini DIN
4	CS-830 CueServer Mini DIN with Buttons
5	CS-PCB CueServer PCB
6	CS-815 CueServer BTO
7	CS-816 CueServer Express
8	CS-840 CueServer DIN
9	CS-811 CueServer Mini II
All others	Reserved for future use

For the CueServer 2 series, this field is reserved for future use and typically returns **F**.

Examples

Identifier (Hex)	Value (Decimal)	Decoded Meaning
0x0001	1	CS-800 CueServer Pro
0x0007	7	CS-816 CueServer Express
0x00EF	239	CS-940 CueServer 2 DIN, Rev. A
0x01AF	431	CS-900 CueServer 2 Pro, Rev. B
0x03BF	959	CS-920 CueServer 2 Mini, Rev. D
0x07EB	2027	CS-940 CueServer 2 DIN, Rev. H, Special Variant



Please note that since all CueServer 1 series report 0 as the hardware revision and 0 as the hardware platform, it is safe to assume that if the Hardware Identifier is 15 (0x000F) or lower, then the device is a CueServer 1. If the Hardware Identifier is 16 (0x0010) or higher, then the device is a CueServer 2.

Autodiscovery

CueServers on the network can be discovered by using an *auto-discovery* technique.

All CueServers (both the original CueServer and the CueServer 2 series) are listening on the CueServer Multicast Group Address (239.255.204.2) on port 52737. This socket is typically used to send CueScript commands to the CueServer, but it also used for auto-discovery.

To ask CueServers to report themselves, simply send the 6 character string `#PING#` via UDP to this group address and port number.

Every CueServer that receives this message will reply to the sender with a UDP packet. The returned data is an ASCII string. It begins with `#PING` and contains several fields, each separated by a bar character (`|`).

The following is an example ping reply packet from a CueServer:

```
#PING|600001|10.0.1.5|1.5.5|CueServer 2|255.0.0.0|10.0.1.1|0|239|
000000000000|6/30/2017 12:59:59 PM|N|shows/My Show|1|1|0.0.0.0|0.0.0.0|0
```

The bar-separated fields are explained in the following table:

Field	Example	Description
<i>Fields below are present on both CueServer 1 and CueServer 2 models</i>		
1	600001	Serial number (6 characters, may include <i>only</i> numbers and letters, no special characters)
2	10.0.1.5	Primary interface IP address (standard IPv4 notation)
3	1.5.5	Current firmware version (format is <code><number>.<number>.<number>[<dev-stage><number>[<letter>]]</code> , an extreme example would be <code>2.34.567b99z</code>)
4	CueServer 2	Device name (maximum 15 characters)
5	255.0.0.0	Primary interface subset mask (standard IPv4 notation)
6	10.0.1.1	Gateway address (standard IPv4 notation)
7	0	Primary interface DHCP mode (may be 0 or 1)
8	239	Hardware model (see Hardware Model Identifiers)
9	000000000000	Reserved for future use (only used on CueServer 1)

10	6/30/2017 12:59:59 PM	The current device time (MM/DD/YY HH:MM:SS AP)
11	N	Reserved for future use (only used on CueServer 1)
<i>Fields below are only present on CueServer 2 models</i>		
12	shows/My Show	Current show path
13	1	Number of physical Ethernet ports (may be 1 or 2)
14	1	Number of logical Ethernet interfaces (may be 1 or 2)
15	0.0.0.0	Secondary interface IP address (standard IPv4 notation)
16	0.0.0.0	Secondary interface subnet mask (standard IPv4 notation)
17	0	Secondary interface DHCP mode (may be 0 or 1)

Release Notes

The following list shows a revision history of the software releases for CueServer 2.

The current version may be downloaded from the main [CueServer Downloads](#) page.

Pre-release and archived versions may be downloaded from the [CueServer Software Site](#).

- [Release v2.1.1 \[May 18, 2018\]](#)
- [Release v2.1.0 \[May 16, 2018\]](#)
- [Release v2.0.4 \[March 14, 2018\]](#)
- [Release v2.0.3 \[February 14, 2018\]](#)
- [Release v2.0.2 \[January 22, 2018\]](#)
- [Release v2.0.1 \[November 3, 2017\]](#)
- [Release v2.0.0 \[October 24, 2017\]](#)
- [Release v1.5.5 \[October 28, 2016\]](#)
- [Release v1.5.4 \[September 8, 2016\]](#)
- [Release v1.5.3 \[August 9, 2016\]](#)
- [Release v1.5.2 \[July 25, 2016\]](#)
- [Release v1.5.1 \[July 19, 2016\]](#)
- [Release v1.5.0 \[June 3, 2016\]](#)
- [Release v1.4.3 \[April 18, 2016\]](#)
- [Release v1.4.2 \[March 17, 2016\]](#)
- [Release v1.4.1 \[February 24, 2016\]](#)
- [Release v1.4.0 \[January 21, 2016\]](#)
- [Release v1.3.0 \[November 11, 2015\]](#)
- [Release v1.2.0 \[July 24, 2015\]](#)
- [Release v1.1.0 \[May 22, 2015\]](#)
- [Release v1.0.8 \[April 27, 2015\]](#)
- [Release v1.0.7 \[April 7, 2015\]](#)
- [Release v1.0.6 \[March 13, 2015\]](#)
- [Release v1.0.5 \[March 11, 2015\]](#)
- [Release v1.0.4 \[February 9, 2015\]](#)
- [Release v1.0.3 \[January 22, 2015\]](#)
- [Release v1.0.2 \[January 9, 2015\]](#)
- [Release v1.0.1 \[December 23, 2014\]](#)
- [Release v1.0.0 \[December 18, 2014\]](#)

Release v3.0.0 [Pre-Release]

Version 3.0.0b1

Version 3.0.0b1 is a near-final release of the upcoming v3.0.0 software that adds many exciting new functionality to CueServer including fully interactive web-based CueStations, a built-in graphical layout editor, entirely new station management tools, support for the new Insite 7" Touchscreen, and station pages.

Please Note: Beta software is not likely to be *bug-free*. Please don't use it on a critical project without first testing to make sure it will function properly for your application.

Thank you for your interest in helping us test this exciting new software. If you have any issues, comments, suggestions, or would like to report any bugs, please send an email to support@interactive-online.com.

- **Major Features**

- **New** Fully Interactive Web-Based CueStations.
- **New** Built-in Graphical Layout Editor.
- **New** Entirely new Station management tools.
- **New** Support for new 7" Insite Touchscreen.
- **New** Station Pages.

- **Special Notes for v3.0.0 Beta 1**

- **Special Note** CueServers updated to firmware v3.x can **not** currently be downgraded to v2.x. This is a known issue and will be addressed shortly.
- **Issue** PIN Numbers on station pages are non-operative.
- **Issue** Page automation rules are not triggered by the station.

- **Editor Window**

- **Feature** The command line's status bar now shows the current station's active page and the target page.
- **Bug** An issue that could cause a crash when opening an offline show has been fixed.
- **Bug** The Editor Window no longer crashes if it is closed while the stack/zone lists are loading.

- **Playbacks View**

- **Bug** Fixed a problem that could cause Playbacks 26+ to appear improperly in the Playbacks view.

• Stations

- **Feature** The *Stations* panel has entirely been redesigned.
- **Feature** Stations are now viewed and selected by changing the popup menu in the top-left of the panel.
- **Feature** The Station navigation panel shows choices for a station's settings, optional page settings, layout, and the button/contact/output/port resources applicable for that station.
- **Feature** *Web Access* for any station may be enabled, providing a CueServer-hosted URL that displays the station's web representation.
- **Feature** Web-based stations may be protected by a user name and password.
- **Feature** Stations now have *pages* of buttons.
- **Feature** Legacy stations (Mystique/Ultra) can be optionally upgraded to use pages.
- **Feature** Pages have various automation rules, including Page Open, Page Close, Page Idle, etc.
- **Feature** Pages may be protected by PIN number.
- **Feature** The *layout* of each station's page(s) may be edited with a new graphical layout editor.

• CueScript

- **Feature** The STATION command can now be used with the AT command to change the active page of a Station. For example, the command `STATION 1 AT 3` changes Station 1's active page to 3.
- **Feature** A new PAGE command can now be used to change the station page for which subsequent commands operate. For example, the command `PAGE 5 BUTTON 1 ON` turns on Button 1 on Page 5 of the current station.
- **Bug** Non-volatile variables are now preserved properly after power loss or reboot.
- **Bug** The inline CueScript parser would improperly show that a SET command with a numeric literal, such as `Set "x" = 3` was an error.

• DMX

- **Feature** Added Floor and Mask modes to Playback faders.
- **Bug** Fixed an issue that prevented more than 32 sACN input universes to be able to be received. The new limit is 128 sACN input universes.

• Serial

- **Bug** Fixed an issue that could cause a crash if incoming serial is being processed as CueScript commands while replies are being sent back to the serial output, and the hardwired DMX output ports are in use.

Release v2.1.2 [Pre-Release]

Version 2.1.2d6

Version 2.1.2d6 contains several bug fixes.

- **Editor Window**

- **Bug** An issue that could cause a crash when opening an offline show has been fixed.
- **Bug** The Editor Window no longer crashes if it is closed while the stack/zone lists are loading.

- **Playbacks View**

- **Bug** Fixed a problem that could cause Playbacks 26+ to appear improperly in the Playbacks view.

- **CueScript**

- **Bug** Non-volatile variables are now preserved properly after power loss or reboot.
- **Bug** The inline CueScript parser work improperly show that a command like [Set "x" = 3] was an error.

- **DMX**

- **Feature** Added Floor and Mask modes to Playback faders.
- **Bug** Fixed an issue that prevented more than 32 sACN input universes to be able to be received. The new limit is 128 sACN input universes.

- **Serial**

- **Bug** Fixed an issue that could cause a crash if incoming serial is being processed as CueScript commands while replies are being sent back to the serial output, and the hardwired DMX output ports are in use.

Release v2.1.1 [May 18, 2018]

Version 2.1.1

Version 2.1.1 is a quick patch that resolves a few minor issues found in v2.1.0.

- **CueScript**
 - **Bug** Minor CueScript Helper syntax improvements for the Enable, Record, and Wait commands.
- **Settings**
 - **Bug** Removed an unnecessary checkbox from the *Settings > General > SMPTE* panel.

Release v2.1.0 [May 16, 2018]

Version 2.1.0

Version 2.1.0 adds a suite of SMPTE Timecode features, a Live CueScript Helper, high-resolution graphics, plus dozens of other important enhancements to CueServer 2.

- **Major Features**

- **New** SMPTE Timecode
- **New** Live CueScript Helper
- **New** Retina/Hi-DPI Graphics
- **New** New Windows framework for improved flicker-free drawing
- **New** CueServer Studio is now 64-bit for macOS

- **Editor Window**

- **Feature** The command line and other CueScript entry windows now include a live “CueScript Helper”.
- **Feature** A new *Timecode Event Editor* has been added to the Triggers section.
- **Feature** The CueScript editor popup window is now resizable.
- **Bug** Fixed a crash that could occur if the Stage View palette window is opened before the embedded Stage View panel.
- **Bug** The *Station Status* field of the command line could temporarily display incorrect information.
- **Bug** Typing the *Forward-Delete* key on the command line did not delete the proper character.

- **CueScript**

- **Feature** A new SMPTE command has been added that allows the current timecode to be set, cleared, started, and stopped. Additionally the SMPTE command can also be used to enable or disable the external SMPTE Timecode Audio input and control internal generation of timecode.
- **Feature** CueScript buttons are now formatted with better multi-line support.
- **Bug** Updating a Preset now properly updates the same preset in joined zones.
- **Bug** Fixed a possible crash that could occur if a WAIT CLEAR is executed in the same command string after a WAIT.
- **Bug** The BREAK command could crash if it was in an IF/THEN/ELSE clause.
- **Bug** The CUE command was improperly limiting the highest Cue Number below the actual maximum of 999,999.99.

- **Groups**

- **Feature** Hovering over the Channels column in the Group List now reveals a popup window that lists all of the channels in the group.
- **Bug** Deleting all of the text from the Channels field is now possible.

- **Zones**

- **Bug** Fixed a crash that could occur if the Zones panel is closed while it is still fetching the zone list from the show file.

- **Rules**

- **Bug** Fixed a firmware crash that would occur if a SET rule is created and then saved without specifying an action.
- **Bug** Some *Preset Actions* were failing to execute if assigned to the highest numbered button on a station.
- **Bug** Popup buttons in rules are now visible on lines with long CueScript.
- **Bug** Rule actions within Cues and Presets did not have the proper default Playback Fader.

- **Settings**

- **Feature** A new *Timecode* panel in General settings allows external SMPTE Timecode audio input processing to be enabled or disabled, and also to set a threshold for jumping over or fast-forwarding through discontinuous time periods.
- **Bug** Fixed a problem that could cause show file corruption after increasing the number of Playbacks in the *Settings > DMX > Resources* panel.

- **Status**

- **Feature** A new *Timecode* panel in the System Status area displays the live timecode along with internal generation and external input indicators. The current dBFS scale of the audio input is displayed as well.

- **Audio**

- **Feature** A new Audio Input processor is included that can receive and decode incoming Linear Timecode (LTC) signals.

- **CueServer Studio**

- **Bug** Restore the ability to update the firmware of a remote CueServer that was broken in v2.0.4.

Release v2.0.4 [March 14, 2018]

Version 2.0.4

Version 2.0.4 is an incremental update to CueServer 2 including 10 general feature enhancements, and 11 bug fixes. Focus areas of this release include working with show files, CueScript additions, system log improvements and general usability.

• Navigator Window

- **Feature** Downloading a show now warns the user if the operation will overwrite an existing file.
- **Feature** Creating a show with an existing name now warns the user.
- **Feature** Firmware updates are now disallowed on CueServers that are not on the same network subnet as CueServer Studio.
- **Feature** The contextual menu for CueServer devices now includes the *Apply License Code* and *Update Firmware* items.
- **Bug** The *Upload Show* file chooser no longer allows non-show directories to be uploaded.
- **Bug** Addressed an issue that could cause a crash if a show upload/download is cancelled.
- **Bug** If a show upload is cancelled while in progress, the partially uploaded show is now properly removed from the device.
- **Bug** Fixed a crash that would occur if an offline show is deleted or moved before editing.
- **Bug** Fixed a crash that would occur if an offline show is deleted or moved while its editor window is open.

• Editor Window

- **Feature** Switching away from the *Status* panel and then back now remembers the last used sub-panel.

• CueScript

- **Feature** A new LOG RESET command has been added that removes all entries from the System Log.
- **Bug** Unterminated strings now fail properly.
- **Bug** The PLAYBACK command now provides the proper error message if an incorrect parameter is given.
- **Bug** Addressed a problem that prevented decimal numbers without a leading zero in expressions (such as "5 * .3") to be evaluated properly.

• Rules

- **Feature** A confirmation dialog is presented before a rule is deleted.
 - **Feature** The “port letters” for DMX Ports can now be used in DMX Port event rules.
 - **Bug** Global button/contact rules would fail when specified without a station number has been fixed.
- **System Log**
 - **Feature** A new *Clear Log* button has been added to the System Log panel.
- **DMX Triggers**
 - **Bug** The 16-bit checksum opcode (“\S”) had previously been outputting the value in the wrong byte order.
- **LCD Display**
 - **Bug** Fixed a problem that could cause Macros to be listed out of numerical order.
- **Installer**
 - **Feature** The macOS .dmg image now includes Retina artwork.

Release v2.0.3 [February 14, 2018]

Version 2.0.3

Version 2.0.3 is an incremental update that concentrates on bug fixes. This update includes the following 13 improvements:

- **Editor Window**

- **Feature** Renamed “Rules” to “Global Rules” for clarity.
- **Bug** Editing an offline show now properly hides the live controls.
- **Bug** Chevron buttons no longer disappear after a warning icon displays on that row.

- **CueScript**

- **Feature** Variables can now be used in arrays.
- **Bug** The PLAYBACK command no longer reports an error if used with the wildcard (*) to select all Playbacks.
- **Bug** Nested CueScript can now return string values to the calling context.

- **Presets**

- **Bug** Rules in Presets no longer improperly show “Whenever This Cue...”.
- **Bug** The “Whenever This Preset” event is now triggered properly.
- **Bug** The Preset Toggle rule action now operates as expected.

- **Timers**

- **Bug** Addressed a problem that could prevent astronomical time events with negative offsets to be switched to positive offsets.
- **Bug** Attempting to set the offset of an astronomical event beyond 360 minutes no longer created a UI inconsistency.

- **LCD Display**

- **Bug** Fixed a problem introduced in v2.0.2 that prevented the LCD Menu to be able to change shows and/or switch DHCP mode.

- **Serial Ports**

- **Bug** Setting the serial output format to “8-O-1” or “8-E-1” now properly outputs the parity bit.

Release v2.0.2 [January 22, 2018]

Version 2.0.2

Version 2.0.2 is an important update to CueServer 2 including 11 general feature enhancements, and 17 bug fixes. Focus areas of this release include the WAIT command, Macro behavior, and CueScript bug fixes.

• Navigator Window

- **Feature** The *Network Settings* window now displays the device's MAC Address(es).
- **Feature** A show can now be deleted by pressing the Delete or Backspace keys.
- **Bug** Addressed a problem that sometimes caused shows to not be removed from the show listing after being deleted.
- **Bug** Renaming a show no longer produces an error message when the show contains non-volatile variables.
- **Bug** No longer allow slash characters in show file names.
- **Bug** Addressed a problem that could allow the window's toolbar items to be improperly enabled after deleting a show file.

• CueScript

- **Feature** Several new escape codes have been added to the handling of strings to enable the automatic substitution of channels, levels and checksums into strings.
- **Feature** Added several new system variables to get and set the date, time, and timezone in several ways.
- **Feature** CueScript editors now allow tab characters.
- **Feature** Added the WAIT STOP command that allows a single pending Wait to be stopped before it fires.
- **Feature** Added new `accent quotes` syntax for allowing the value of a variable to be executed on the command line.
- **Bug** The remainder of a command that executes after a WAIT now properly executes in the same command context as the beginning of the command.
- **Bug** Commands that operate on triggers (such as PRESS and RELEASE) now properly work after a WAIT command.
- **Bug** Calling a macro that contains WAIT commands no longer causes an out-of-order command execution sequence.
- **Bug** Macros now properly execute in the same command context as the calling script.
- **Bug** Multiplication and Division operators now work as expected.

- **Bug** Concatenation now works properly when combining numeric values to strings.
 - **Bug** Inline variable substitutions of string values now work as expected.
 - **Bug** Addressed a problem with nested CueScript returning floating point numerical results.
 - **Bug** Pressing Enter or Return on an empty command line no longer sends an empty command to the CueServer.
 - **Bug** Addressed a problem that could occasionally cause the CueScript parser to crash when an inline variable substitution occurs within an IF/THEN/ELSE/ENDIF block.
- **Editor Window**
 - **Bug** Adding or removing a rule from a Cue, Station, or DMX Trigger now no longer scrolls the rule list back to the top.
- **DMX Triggers**
 - **Feature** A new “Act on Changes” function has been added to DMX Triggers.
- **Status**
 - **Feature** Status panels can now be opened into their own separate palette windows.
 - **Bug** The *RAM Used* bar graph in the *CPU Info* panel was incorrect and misleading. More accurate RAM Usage numbers are now shown.
- **LCD Display**
 - **Feature** Time, Date, and Time Zone can now be changed from the LCD Display.
- **Show Database**
 - **Feature** Creating a new show file now pre-configures the show based on the hardware device it is created on.
- **API**
 - **Bug** Setting the Timezone via set.cgi or UDP is now much faster.

Release v2.0.1 [November 3, 2017]

Version 2.0.1

- **Live Stage**

- **Bug** The *Input* layer in the *Stage View* no longer shows incorrect channel values when the DMX patch contains universes with less than 512 channels.
- **Bug** The *Stage View* no longer crashes when viewing more channels that are licensed on the device.
- **Bug** The *Stage View* no longer crashes in certain circumstances when displayed universes have less than 512 channels.

- **Stations**

- **Bug** Stations that are “locked” no longer allow button presses to execute.

- **Settings**

- **Bug** The *Settings > LCD Display* panel now works properly when editing an offline show file.
- **Bug** The *Settings > LCD Display* panel no longer “beeps” when opening.

- **LCD and Front-Panel Display**

- **Feature** The Power LED indicator now flashes Red/Blue when the device is in Identify Mode.
- **Bug** Fixed a bug introduced in v2.0.0 that prevented the *Self Test* to start properly in certain circumstances.

Release v2.0.0 [October 24, 2017]

Version 2.0.0

Version 2.0.0 is a significant update to CueServer 2 including 14 major new features, 67 general feature enhancements, and 54 bug fixes.

• Major Features

- **New** Zones and Presets have been added.
- **New** Rules have 18 new built-in action templates.
- **New** Object lists now have a live column for easy monitoring and control.
- **New** New Stage View options show channels grouped by universe.
- **New** Automatically updating indicators.
- **New** Button Indicators are now fully configurable via popup palettes.
- **New** Improved universe patching system that's more flexible and powerful.
- **New** The licensing model now enables channels instead of universes.
- **New** Variables can now be predefined and/or designated as non-volatile.
- **New** Added additional rule conditions.
- **New** New Settings panels, including settings for Hardware, Stations and Audio.
- **New** Redesigned command line status provides more contextual details.
- **New** Expanded KiNET v2 support.
- **New** Application Preferences have been added.

• Live Stage

- **Feature** The Stage View now shows channels that are not accessible because of the currently selected zone.
- **Feature** The Stage View now shows selected channels in the current playback color.
- **Bug** Addressed a problem that caused the input view of the *Stage* window to not show channel values for incoming Ethernet protocols.

• Live Playbacks

- **Bug** Addressed a problem where the active playback indicator was not being drawn properly in the *Playbacks* panel [Windows only].

• Live Zones

- **Feature** Added new Zones panel to the Live section that shows the current presets and join status of each zone.

- **Feature** When in a zone, commands that set channel values are now prevented from modifying channels outside of the zone.
- **Live Status**
 - **Feature** The Front Panel Status panel now reflects the LCD backlight brightness.
 - **Feature** The *Status > Variables* panel now sorts the variables alphabetically.
- **Cues**
 - **Feature** Added additional cue details to the Extras column.
 - **Feature** When adding Cue Stacks, the main *Cues* navigation item auto-expands if it had been closed.
 - **Bug** Addressed a problem that caused the capture channels popup to default to *No Channels* when a Cue was first created.
 - **Bug** If a Cue or Preset is new or edited, the capture panel now requires the user to save or apply changes before new channel levels can be captured.
 - **Bug** Addressed a problem that caused streaming cues to not be duplicated properly with the Duplicate Cue command.
 - **Bug** Addressed a problem that caused a crash if a Cue was duplicated on an international system that uses the comma character as numerical decimal separator.
 - **Bug** Addressed a problem that could cause a cue to lose its name when re-recording its streaming content.
 - **Bug** Addressed a problem that could cause the Capture and Record buttons in the Capture panel to be enabled when no cue is selected.
 - **Bug** Addressed a problem that could cause capturing channels in a cue to fail if the cue is in the default stack but the current playback fader has a different stack selected.
- **Zones**
 - **Feature** Added new Zones list view that shows an overview of each zone defined in the system.
 - **Feature** Added Zones sub-views that allow Presets to be added to each zone.
 - **Feature** Added a zone configuration panel that configures each zone.
- **Variables**
 - **Feature** Added new Variables panel that allows a project to predefine variables and/or designate them as “non-volatile”.
 - **Feature** Non-volatile variables retain their values when the power is lost or shows are switched.
- **Timers**
 - **Feature** Added additional information to the details column of the Timers list.
 - **Bug** Addressed a problem with the Timer List panel that would draw the list controls improperly.

- **Bug** Addressed a problem where the checkbox labels may be truncated in the year-picker window.

• Rules

- **Feature** The THEN clause in rules now include 18 new canned actions, including operation on cues, playbacks, channels, groups, presets, indicators, outputs, etc.
- **Feature** Added new Date, Month, Day, Year, Hour, Minute and Second conditionals to rules.
- **Feature** Added new Was Held/Was Not Held conditions for Button and Contact rules.
- **Feature** When choosing a time-based date conditional, the current time and/or date appear as defaults.

• Stations

- **Feature** A new cascading button indicator architecture has been added. Each indicator in the system now has four levels of scope: live, button, station and global. The global scope has the lowest priority and the live scope (as set by the SET command) has the highest priority.
- **Feature** Button indicators now have eight customizable states named: On, Off, Mixed, Locked, User 1, User 2, User 3, and User 4.
- **Feature** A new button indicator color picker has been added to general preferences, station and button editor panels.
- **Feature** Stations now have a Zone popup menu that allows each station to be assigned to a zone.
- **Feature** The *Station* and *Button* editor panels now allow their indicator colors to be set.
- **Feature** Added a Test button to the button/contact closure panel to allow for live testing of press/release events.
- **Feature** Added the ability to communicate with CueStation Hubs via RS-232 and/or RS-485.
- **Feature** Added the ability to set serial ports to use CueStation Hub protocol.
- **Bug** Switching shows now automatically refreshes all button indicators in the system.
- **Bug** The serial port panel no longer allows the user to choose to reply with CueScript result if one of the CueScript protocols are not selected.
- **Bug** Improved station handling performance.

• Timers

- **Bug** Addressed a problem that could cause the Months or Years popup controls to show "Unknown" if the corresponding popup window was dismissed while no months or years were selected by clicking outside of the window.
- **Bug** Addressed a problem that caused Sunrise and Sunset offset times to be limited to a maximum of +/- 60 minutes.

• Settings

- **Feature** Added a new *Hardware* settings panel to choose which hardware type is being used.
- **Feature** Moved the old *General* settings panel to a new *Notes* settings panel.
- **Feature** Added a new *General* settings panel that contains several subcategories of simple preferences.
- **Feature** Added a new *General > Indicators* settings panel to set the global button indicator colors.
- **Feature** Added a new *General > Audio* settings panel to set the audio output volume.
- **Feature** The LCD Display settings panel now updates the device “live” when adjusting settings.
- **Feature** Added backlight brightness control to the LCD Display settings panel.
- **Feature** The “Show on Maps” button in the Location panel now defaults to using the Apple Maps service instead of Google Maps (although pressing the Option key switches back to Google Maps).
- **Feature** Added a “Search timeanddate.com” button.
- **Bug** Addressed a problem that caused Latitude/Longitude changes to require a reboot or show reload to take effect in certain circumstances.

• Editor Window

- **Feature** Added an expanded target description and stack, zone and station context to the command line status bar.
- **Feature** The command line status bar is now drawn in the currently selected playback color.
- **Bug** Addressed an issue in the Macros, Timers, Rules, and DMX Triggers editor panels that could display the “Cancel”/“Save” buttons when making changes to an existing resource instead of the “Revert”/“Apply” buttons.
- **Bug** Addressed a problem that could cause a crash if the stand-alone Playbacks window is opened during switching of active shows.
- **Bug** Addressed a problem that could cause a crash if a show file’s resources are not saved properly (due to card removal or power outage).
- **Bug** Addressed a problem that could cause a rare crash during show switching.
- **Bug** Addressed a problem that could sometimes cause CueScript statements to draw outside the bounds of a CueScript button.
- **Bug** Addressed a problem that sometimes caused command status changes to appear to lag after a new command was submitted on the command line.
- **Bug** Addressed a problem that could case a rare crash while entering text into a new rule.

• Navigator Window

- **Feature** Time Zone changes now take effect more quickly.

- **Bug** Addressed a problem that could cause a navigation list to have no visibly selected item even though its editor panel is shown if the item is clicked quickly while the mouse is moving off the side of the list.
- **Bug** The *New Show* window now prevents show names from containing double-quote characters.
- **Bug** Addressed a problem where CueServer Studio could become unresponsive after a CueServer device goes offline and then later returns.
- **Bug** Addressed a problem that caused shows from being deleted if they included 'single-quote' characters.

• CueScript

- **Feature** The CHANNEL command and associated selection commands have been extended to support channel numbers in the *universe.channel* format.
- **Feature** A new DO command was added to allow manual triggering of a button or contact's events based on the current physical state of the button or contact.
- **Feature** The LOG command can now log scalar values and arrays in addition to strings.
- **Feature** Escape characters (i.e.: \n, \r, \x00, etc.) are now processed in all strings.
- **Feature** Added the AT INPUT syntax to the AT command.
- **Feature** Added the AT OUTPUT syntax to the AT command.
- **Feature** The AT command can be used to set button indicators to User colors by using the constants 1, 2, 3, and 4.
- **Feature** The AT command can now set the submasters of multiple playbacks simultaneously using a command such as [Playback 1>5 At 50].
- **Feature** The CLEAR command can now clear multiple playbacks simultaneously using a command such as [Playback 6>10 Clear].
- **Feature** Playback faders can now be enabled and/or disabled using [Playback *n* Enable/Disable].
- **Feature** Added the JOIN command. Zones can become members of a join pool with a command such as [Zone "Lobby" Join 3].
- **Bug** Nested CueScript statements now properly inherit the context of the parent.
- **Bug** Addressed a problem that caused the WRITE command to not properly process escape characters in the output string if the destination was a UDP message.
- **Bug** Addressed a problem with the MINUS command that would sometimes deselect the wrong object.
- **Bug** Addressed a problem that caused SET and WRITE commands nested within an IF/THEN/ELSE statement to cause the entire script to fail to execute.
- **Bug** Addressed a race-condition that could allow an auto-follow to occur while the RESET command is executing which could cause Cue 0 (zero) to attempt to execute.

- **Bug** Addresses a problem that could cause the RECORD or UPDATE commands to record a cue into the wrong cue stack in certain circumstances.
- **Bug** Addressed a problem where the assignment operator would not accept negative numbers.
- **Bug** Addressed a problem where the RELEASE command would not entirely release a streaming cue.

• DMX

- **Feature** CueServer Universes can now be scaled down to any number of channels for more efficient usage of resources.
- **Feature** Each CueServer Universe can now have an arbitrary number of “extra outputs” added to allow the same data to be retransmitted with a different protocol or to a different IP address, different KiNET port, etc.
- **Feature** A range of incoming priorities can now be specified in the sACN input configuration of a universe.
- **Feature** New system variables ‘universe.rxpriority’, ‘universe.rxprioritylow’ and ‘universe.rxpriorityhigh’ have been added to dynamically control the range of incoming sACN priorities that will be received for a particular universe.
- **Feature** The ‘universe.priority’ system variable has been renamed to ‘universe.txpriority’ to avoid confusion with the new receive priority variables.
- **Feature** DMX Output ports can now independently transmit DMX in one of 5 speeds (40Hz, 38Hz, 35Hz, 30Hz, or 20Hz), each of which has increasingly exaggerated DMX timing to allow receivers with poorly implemented DMX protocols to (hopefully) work properly.
- **Feature** The built-in DMX output ports now only transmit as many channels as their corresponding universe are configured for.
- **Feature** KiNET v1 and v2 protocols can now be received, captured, output and/or converted to other protocols.
- **Feature** KiNET v2 protocol now supports Chromasic fixtures and synchronization of multiple universes.
- **Bug** Addressed a problem that could cause incoming sACN priorities to not properly block lower priority input.
- **Bug** Addressed a problem that allowed broadcast Art-Net output to be received by the same CueServer, possibly creating a channel value loop.
- **Bug** Addressed a problem that would cause CueServer to drop out of real-time while recording a high-bandwidth streaming cue.
- **Bug** Addressed a problem with the *Settings > DMX > DMX Ports* panel that could show a warning about the wrong universe chosen for ports that are turned off.
- **Bug** Addressed a problem that caused the RELEASE command to not properly terminate streaming cues in certain circumstances.

- **Bug** Addressed a problem that caused DMX signal loss events to occur just after the DMX input buffer was cleared instead of just before.
- **LCD and Front-Panel Display**
 - **Feature** Added optional CPU Load, IO Status, and Timecode displays to the LCD idle screen.
 - **Feature** The brightness curves for the built-in function button indicators LEDs have been adjusted to produce a more linear visual response.
 - **Bug** Improved performance.
- **Show Database**
 - **Bug** Addressed a problem that could occur if a copy of the show directory was manipulated by the Finder on Mac OS and then re-uploaded into the CueServer.
- **Web API**
 - **Feature** Added additional variables to the 'in' and 'out' get.cgi API functions to limit the scope of the returned result.
 - **Bug** The correct indicator colors are now properly sent to the CuePad app and CueTouch panels.
 - **Bug** Addressed a problem with the 'in' and 'out' selectors of the get.cgi API not returning the proper data.
 - **Bug** Improved performance of the API for multiple clients.
- **Auto-Discovery**
 - **Bug** Improved the Internet reachability detection algorithm.
 - **Bug** Addressed a problem that could cause device discovery to not function after the host computer sleeps and then re-awakes.
- **Diagnostic Tools**
 - **Feature** The *CueStation Network Monitor* now automatically scrolls the event window as events are added to the bottom of the list.
- **Windows Build**
 - **Bug** Addressed several problem areas that were creating excessive flickering on Microsoft Windows.
 - **Bug** Addressed the problem of pushbutton control height being too small on Windows builds.
 - **Bug** Several popup windows are no longer resizable nor do they have close/minimize buttons.
 - **Bug** Addressed a problem that caused the Helvetica Neue font to be used in several places instead of the Windows theme font.

- **CueServer Studio**

- **Feature** The first time the application is run, the user is asked to register their copy of the software.
- **Feature** The splash screen now shows a banner in the top-right corner if the build is a “pre-release” version.
- **Feature** The user can choose to receive application updates only for public releases, or receive notices about pre-release versions.
- **Feature** Added new time zones for Chile and generic “Etc” zones (including GMT, UTC, Zulu, etc.).
- **Bug** Addressed a problem that could cause a crash when network interfaces are changed while the app is open.

- **Firmware**

- **Special Note** Devices upgraded from firmware version 1.5.5 or earlier that have show configurations set to output KiNET v2 protocol will need to manually update the KiNET v2 settings in *Settings > DMX > Universe Patch*.
- **Feature** Firmware updates now show their progress on the LCD Display in addition to within CueServer Studio.

Release v1.5.5 [October 28, 2016]

Version 1.5.5 (10/28/16)

- **CueServer Studio 2**

- **Bug** Auto-discovery now works properly if network interfaces are enabled and/or disabled while the app is open.
- **Bug** The CueServer device name now appears in the stand-alone Stage and Playbacks windows.
- **Bug** Addressed a problem that could cause a crash if the active show is switched while a Playbacks view is visible.
- **Bug** Addressed a problem introduced in v1.5.4 that could cause a crash if a newly created cue's number is changed from the default assigned number before saving the cue for the first time.
- **Bug** Addressed a problem that could cause corruption of a station's configuration when editing a show offline and the station's name is reduced in size.
- **Bug** Addressed a problem that could cause a UI inconsistency and/or a crash when expanding a station's contents when the station was not previously selected.
- **Bug** Addressed a problem introduced in v1.5.3 that could cause undefined variables to substitute unexpected values into a CueScript statement.
- **Feature** Application crashes are now handled by a built-in error reporting mechanism.

- **Firmware**

- **Feature** Added the AT ? syntax to the AT command to query the value of various selectable objects.

Release v1.5.4 [September 8, 2016]

Version 1.5.4 (9/8/16)

- **CueServer Studio 2**

- **Bug** Addressed a problem that could cause a streaming cue to become corrupted if its cue number is changed.
- **Bug** Addressed a problem that caused offline shows to not show cue stacks in the hierarchical menu below the *Cues* item.
- **Bug** Addressed a problem with the *sACN Network Monitor* that could cause it to show an incorrect RGB color in the last channel position of the preview area.

- **Firmware**

- **Bug** Streaming cue playback now freezes properly when a playback fader is stopped.
- **Bug** Front-panel brightness is now properly reset when switching shows.
- **Bug** Addressed a problem that caused the show file to not receive a System Power-On event.
- **Bug** Addressed an issue that made it not possible for a show to set the LCD brightness when a show was loaded.
- **Bug** Addressed a problem that could cause CueServer to become unresponsive when a CS-940 is improperly configured as a CS-900 and one of the DMX Input ports is set to be a DMX Output.

Release v1.5.3 [August 9, 2016]

Version 1.5.3 (8/9/16)

- **New Features**

- **New** Added a new *Debug Mode* feature to the System Log. Now various system functions such as button presses, CueScript commands, UDP messages, variable assignments, etc., can be logged to the System Log for general project troubleshooting.

- **Firmware**

- **Feature** Variable substitution is now handled by direct textual replacement inline as they are encountered in CueScript statements. This allows variables to be expanded anywhere in a script, and variables may be numbers, strings, or even additional CueScript commands.
- **Feature** Added new “Stream Recording Monitor” to the DMX Utilities menu on the LCD Display.
- **Feature** Added new “panel.brightness” system variable that adjusts the overall brightness of the function buttons and navigation switch backlighting.
- **Feature** Added new “debug.buttons”, “debug.cue”, “debug.cuescript”, “debug.show”, “debug.udp”, and “debug.variables” system variables to enable system logging of various internal events.
- **Bug** Fixed a bug that could cause a crash in the *Network Settings* LCD Menu if a displayed IP Address had 14 or more characters.
- **Bug** Addressed a problem that prevented some selection commands from accepting nested script statements. For example, “CHANNEL (RANDOM{1,10}) @ FL” previously did not work.
- **Bug** Addressed a problem that could cause timers to not fire properly if other timers in the list were disabled.
- **Bug** Addressed a problem with the assignment operator that would mistakenly attempt to set a variable value instead of perform an equality operation if the left-hand value was an undeclared variable.

- **Windows Installer**

- **Feature** The windows installer now includes both 32-bit and 64-bit Microsoft Visual C++ packages.

Release v1.5.2 [July 25, 2016]

Version 1.5.2 (7/25/16)

- **CueServer Studio 2**
 - **Bug** Addressed a problem that could cause KiNET v2 parameters to not be saved properly to the show file.
- **Firmware**
 - **Feature** Now supports Revision D of the CS-940 hardware.

Release v1.5.1 [July 19, 2016]

Version 1.5.1 (7/19/16)

- **New Features**

- **New** Added a *System Clock* panel to allow the device's time, date and various astronomical parameters to be viewed live.

- **CueServer Studio 2**

- **Feature** When changing protocols in *Settings > DMX > Universes*, the appropriate next field is automatically focused for convenience.
- **Bug** Fixed a bug in the *Network Settings* window that displayed the wrong mode when changing settings for a device with only a single LAN port.
- **Bug** Fixed a bug in the *Sounds* panel that would improperly place an imported file in the web directory if the "+" button was used to add a file.
- **Bug** Addressed a problem in the various resource editors (Cues, Groups, Macros, Rules, etc.) that could prevent a new resource from being created if the current resource has unsaved changes.
- **Bug** Fixed several bugs in the file browser panel related to the selected item caption, delete button, and directory refreshing.
- **Feature** Added additional legal notices as required.

- **Firmware**

- **Feature** Added OFFSET and LENGTH commands that are used to manually set the starting point and playback length of a streaming cue.
- **Bug** Fixed a bug that caused KiNET v2 IP addresses to have their bytes reversed.
- **Bug** Fixed a bug that caused the *Apply License Code* window to improperly indicate that the code had not been accepted.
- **Bug** No longer display an error message for having multiple CueServer universes set to receive the same sACN universe.
- **Bug** The factory reset function now properly resets the NTP Server parameters.
- **Feature** Improved build optimization to reduce resource usage and increase performance.

Release v1.5.0 [June 3, 2016]

Version 1.5.0 (6/3/2016)

- **New Features**

- **New** Added DMX Input Triggers.
- **New** Added Art-Net Protocol.
- **New** New *Playbacks* view now shows DMX Input/Output.
- **New** New *Cues* editor panel now has separate tabs for Properties, Contents and Capture.

- **CueServer Studio 2**

- **Feature** Improved the *Playbacks* view to show blocks for DMX Input and DMX Output that feature what sources and destinations are flowing into and out of the CueServer via various Ethernet protocols and/or the hardwired DMX ports.
- **Feature** Added flowchart-style arrows between the blocks in the *Playbacks* view to visually indicate the direction of data movement through the fade engine.
- **Feature** Layer blocks in the *Playbacks* view now dynamically resize as needed.
- **Feature** Added an explicit “Add Rule” button to resources that have rules (Buttons, Contacts, Cues, DMX Input Triggers, etc.) instead of using the small, circular “+” button.
- **Feature** Improved the Details column of the Triggers listings, including Timers, Rules and DMX Input Triggers.
- **Feature** Added flowchart style arrows to the Playbacks view showing the direction of data movement.
- **Feature** Added a new *Contents* tab to the Cue Editor panel that shows all of the recorded channels in a cue.
- **Feature** Added a new *Capture* tab to the Cue Editor panel that is used to take snapshots and/or record streaming cue data.
- **Feature** Added an indicator to the Playbacks view that shows when DMX Input is disabled.
- **Feature** Added a visible “Captured” acknowledgement to the *Cues > Capture* panel that appears after a snapshot is captured.
- **Feature** Streaming cues are now only partially loaded when editing the cue’s properties, vastly improving load and save speeds.
- **Bug** Addressed a problem that caused the *CueStation Network Monitor* to fail to capture packets in certain circumstances.
- **Bug** Missing show directories are now automatically created if needed.
- **Bug** Switched the order of the Input and Output sections of the *Settings > DMX > Universes* editor panel to follow the conventions used elsewhere in the application.

- **Bug** Scroll bars in several windows (such as *Stage*, *Playbacks*, *Contents*, etc.) now have more reasonable paging and step sizes.
 - **Bug** Fixed a bug that caused the sACN Universe field to lose focus when entering a new value.
 - **Bug** Fixed a bug that caused the *Settings > DMX* scrolling list to return to the top each time a universe configuration was saved.
 - **Bug** Fixed a bug that caused the default values that would appear in *Settings > DMX > Universes* to be set to unexpected values.
 - **Bug** Fixed a bug that caused the hour field for on/off timers to not be able to be set to zero after it was set to a non-zero value.
 - **Bug** Fixed a bug with the *Cue > Properties* panel that was not calculating the scroll bar height correctly when a cue had rules.
 - **Bug** Fixed a bug that caused the hierarchical control to not show as expanded in the Navigator Window when a new show is created on a CueServer device when its show listing is collapsed.
 - **Bug** Fixed a bug that caused a crash if the *Network Info* window was closed shortly after opening it, while it was still checking for a connection to the Internet.
 - **Bug** Addressed a problem with the Playback view that sometimes caused the bottom part of the display to disappear when the window was resized smaller than its original size.
 - **Bug** Addressed a problem that could cause the *Groups* editor panel to be very slow when complex groups are shown in the list.
 - **Bug** Addressed a problem with the mini text field used in rule conditions that allowed non-printing characters to be entered into the field (such as page up/down, arrow keys, forward delete, etc.).
 - **Bug** Addressed a problem introduced in v1.4.2 that caused device auto-discovery to fail from computers that have multiple active network interfaces in certain circumstances.
 - **Bug** Addressed a problem introduced in v1.4.3 that prevented remote CueServers from displaying their live status.
 - **Bug** Addressed a problem that was causing excessive UI flickering in the *Settings > DMX > Universes* editor panels [Windows only].
 - **Bug** Renamed the Resources sub-directory of the Windows version of the application to include the application's name [Windows only].
 - **Bug** Fixed a bug that would cause the installed application to work fine for the user that installed the application but when another user was logged into the same computer the application icon, version information, and other important application data would be missing [OS X only].
- **Firmware**
 - **Feature** Restructured the show database server for faster show switching.
 - **Feature** Added better LCD display messages for when the device is rebooting or shutting down.

- **Feature** Implemented a fail-safe mechanism during streaming cue recording that prevents runaway stream recording if the CueServer Studio client disappeared because of a network error or program crash.
- **Feature** The built-in Hardware Self Test function can now be accessed via the LCD Display menu.
- **Feature** Optimized the DMX fade engine code for increased performance in several key areas, especially during stream recording.
- **Feature** Optimized the CGI code for increased performance.
- **Feature** Switching shows now clears all cached user variables and command contexts.
- **Feature** The built-in front-panel function buttons' default colors are now reset to defaults each time a show is loaded for consistency.
- **Bug** Addressed a problem that was preventing Groups from being joined using the THRU command.
- **Bug** Fixed a bug that could cause a crash of the fade engine if a streaming cue is updated (instead of recorded) before the streaming cue existed yet.
- **Bug** Fixed a bug that caused universes to not be able to be disabled if they had no output protocol selected.
- **Bug** Fixed a bug that caused DMX Input Restore/Fail events for the "DMX 2 Input" port on the CS-940 to be reported as Port 3.
- **Bug** Addressed a problem that could cause show switching to fire a WAIT command or perform a playback auto-follow in the middle of the switch when the old show is partially unloaded.
- **Bug** Addressed a problem that was causing a show to not be able to properly set the built-in function button colors in its Show Loaded event during system boot time.
- **Bug** Addressed a problem that caused a playback fader to show that Cue 0 was active after the RELEASE command was used.
- **Bug** Addressed several problems with show switching that would cause the show to be restarted whenever DMX settings were changed.
- **Bug** Addressed a problem that caused newly created shows to have improper default values for some universe settings.
- **Bug** Addressed a problem that caused disabled buttons, contacts or DMX input to become re-enabled whenever any show configuration settings were changed.
- **Bug** The Apache web server's CGI interface is shut down during firmware updates to prevent a race condition that occasionally caused firmware updates to fail.

Release v1.4.3 [April 18, 2016]

[April 18, 2016]h3. Version 1.4.3 (4/18/2016)

- **CueServer Studio 2**

- **Bug** Addressed a problem that could cause the discovery of local CueServers to be intermittent if the found devices are not on the same subnet as the host computer.
- **Bug** Addressed a problem that could cause a crash when editing certain objects within offline shows.
- **Bug** Addressed a problem that could cause CueServer Studio to experience a long delay on startup if the computer is on a network, but the internet is not reachable [OS X only].
- **Bug** Addressed a problem that caused the sACN Network Monitor window to flicker while updating and redraw improperly when the window is resized [Windows only].
- **Feature** The Windows version of CueServer Studio now uses the Microsoft Universal C Runtime libraries for better compatibility and stability [Windows only].

- **Firmware**

- **Feature** In the Network Settings LCD menu, a “null icon” will be displayed with the gateway address if the chosen gateway is not accessible on the local network.
- **Bug** Addressed a problem that would cause sACN data to not be transmitted if the chosen gateway address is not accessible on the local network.
- **Bug** Addressed a problem that sometimes caused the Network Settings LCD menu to display a gateway address of 0.0.0.0 even though a non-zero gateway address was actually in use.
- **Feature** The gateway address can now be explicitly set to 0.0.0.0 to configure the network to have no default gateway.
- **Bug** Addressed a problem that could cause the “Contacting DHCP Server” message to get stuck on the LCD Display.
- **Feature** The ‘ssh’ and ‘rpcbind’ services are no longer running by default in CueServer for increased security and performance.
- **Bug** Addressed a problem that caused network time updates to fail in certain circumstances.
- **Bug** Addressed a problem that caused show file modification dates to be written in UTC time instead of local time.
- **Bug** Addressed a problem that caused the AT+ or AT- commands from working properly when changing the submaster level of a playback fader.

Release v1.4.2 [March 17, 2016]

Version 1.4.2 (3/17/2016)

• CueServer Studio 2

- **Feature** Added a new Diagnostic Tools section to the Help menu. Two network analysis tools have been added:
 - **sACN Network Monitor** – Scans the network for active universes of sACN data. Shows the sources of sACN data on a particular network. Visualizes a chosen sACN source by showing the channel values as bar graphs or RGB pixels.
 - **CueStation Network Monitor** – Monitors and captures any CueStation button and/or indicator commands found on the network. Displays (or filters) each event based on event type, source, station or button.
- **Feature** When switching to the Stage or Playback views, the command line is now always active.
- **Feature** KiNET v2 now allows each port to be assigned it's own separate IP Address.
- **Bug** Addressed a problem that caused a crash when changing the number of buttons on a station while editing an offline show file.
- **Bug** Addressed a problem that occurred when renaming the active show file, or a show with an open editor window.
- **Feature** Changed the behavior of the Network Settings window to automatically adjust the gateway field when changing the IP Address and/or Subnet Mask to guarantee that the chosen gateway would always be reachable on the network.
- **Bug** Addressed a problem with the System Log view on Retina displays that caused the text to appear very small.
- **Bug** Addressed a problem that caused the Navigator Window to not become frontmost when right-clicking on the Device or Project lists [OS X Only].

• Firmware

- **Feature** Added an update routine that periodically refreshes indicator values in case a CueStation Hub misses update messages or is power cycled.
- **Bug** Addressed a problem with CueStation Hub communications that occurred when setting many indicator values at once that could cause some indicators to not properly change value.
- **Feature** The WRITE command now processes embedded C-style escape characters in the string.
- **Feature** Changed the KiNET v2 driver to support separate IP Addresses for each port.

- **Feature** Changed the behavior of the LCD Display's Network Settings to automatically adjust the gateway field when changing the IP Address and/or Subnet Mask to guarantee that the chosen gateway would always be reachable on the network.
- **Bug** Addressed a problem on the CS-900 that caused ports set to DMX Outputs to not be configured properly in certain circumstances.
- **Bug** Addressed a problem that caused hardwired DMX and sACN inputs on the same universe to not be merged properly.
- **Bug** Addressed a small memory leak that could occur when transmitting strings out one of the serial ports.
- **Bug** Addressed a problem that caused the network time daemon to not start properly in certain circumstances.
- **Bug** Adjusted the network time daemon parameters to generate less network traffic.
- **Feature** Increased compiler optimization to improve overall system performance.

Release v1.4.1 [February 24, 2016]

Version 1.4.1b (2/24/2016)

- **Windows Installer**

- **Bug** Addressed a problem with the Windows installer that corrupted the application in a way that would cause it to not be able to properly create new offline show files [Windows Only].

Version 1.4.1 (2/9/2016)

- **CueServer Studio 2**

- **Feature** Editor Windows now darken and display a “CueServer Offline” message when the device becomes unreachable.
- **Bug** Addressed a problem that would cause cue display problems and a crash in the stream capture window on host computers that are localized to use a comma as the decimal value separator.
- **Bug** Addressed a problem that could cause some show files to not appear in the Navigator window when multiple CueServers’ show listings were being displayed at the same time.
- **Bug** Addressed a problem that could cause the “Input Status” indicator in the *Settings > DMX > Universes* panel to show an improper status.
- **Bug** Addressed a problem introduced in v1.4.0 that caused a warning icon to appear for the *Settings > DMX* navigation item for offline shows.
- **Bug** Addressed a problem that caused Blue LED Indicators on the CueServer device to appear Magenta in CueServer Studio.
- **Bug** Addressed a problem that caused the Indicators in the *Status > Front Panel* to not be labeled properly.

- **Firmware**

- **Feature** Temporary internal files are no longer stored in non-volatile memory to reduce wear on flash memory.
- **Feature** Changed the default subnet mask after factory reset to 255.0.0.0.
- **Bug** Fixed a problem that caused the IP Address to show “No Address” when the Ethernet cable was unplugged, instead of properly showing the IP Address.
- **Special Note** Devices running firmware version 1.4.1 (or higher) can not be downgraded to 1.4.0 (or lower) without assistance from Technical Support. Downgrading firmware is not typically necessary. Should a situation arise that requires a firmware downgrade, please contact Technical Support first.

Release v1.4.0 [January 21, 2016]

Version 1.4.0 (1/21/2016)

• CueServer Studio 2

- **Feature** Added ability to set the Input/Output direction of DMX Ports in the DMX Port Settings panel.
- **Feature** Added a new Network Settings window that allows the network mode to be switched between Single and Dual LAN configurations (on hardware that supports this).
- **Feature** Added preferred CueServer model selection to DMX Port Settings panel (used to show the appropriate DMX port options).
- **Feature** Added a warning to the DMX Resources Settings panel when the allocated number of universes exceeds the licensed number of universes.
- **Feature** Added warnings to the DMX Port Settings panel when the port direction or universe number is questionable.
- **Feature** Added ability to duplicate Cues, Groups, Macros, Timers, and Rules using the list's settings menu or contextual menu.
- **Feature** Added a warning icon that shows when a local CueServer is discovered but is not accessible on the network because it is on a different subnet.
- **Feature** Added the ability to specify what ranges of DMX channels are broadcast to each port of a KiNET v2 device.
- **Feature** CueServer devices now show "A" or "B" after their IP Address if the device has Dual-LANs enabled to indicate which LAN the device is connected via.
- **Feature** The Macro editor now fills the available window space with the script editor.
- **Feature** The "create new show" dialog window now opens with the text field selected.
- **Feature** Added buttons to the System Log panel that will add marks or text to the log file.
- **Feature** Show files in the Navigator Window are now listed in alphabetical order.
- **Feature** Devices that go offline in the Navigator Window now collapse their show listing, lose their hierarchical control and cannot be edited.
- **Feature** CueServer 1 models now appear in the device list in gray text.
- **Feature** Opening a CueServer 1 model now provides a warning that CueServer Studio 2 cannot edit older CueServer models.
- **Feature** The Device and Project lists are now sorted by name by default.
- **Feature** Improved the Show Network Info window to display if the Internet is reachable from the local machine.
- **Feature** New shows files now default to not transmitting or receiving DMX over Ethernet.

- **Feature** The DMX Settings sub-panels are now scrollable when the content is larger than the parent window.
- **Feature** The Universe Settings sub-panel now shows only the properties that apply to the selected input/output protocol.
- **Bug** Addressed a problem that could cause the renumbering of resources to fail.
- **Bug** Addressed a problem that caused a crash when offline show files contained more than about 375 of a single resource (such as Groups, Macros, Cues, etc.).
- **Bug** Fixed a bug that caused active show changes to remote CueServers to not be updated properly in the Navigator Window.
- **Bug** Fixed a bug that could cause the DMX Resource Settings bar graphs to draw improperly after user changes were reverted.
- **Bug** Addressed a problem that could cause unwanted settings to appear in the universe settings after increasing the number of universes in a project.
- **Bug** Fixed a bug that would cause the Done button in the License Code Window to redraw incorrectly in certain circumstances.
- **Bug** Addressed a problem that caused remote CueServers to not show their online status properly.
- **Bug** Addressed a problem that could cause the time to not be able to be updated on a CueServer if the host computer had more than one active network interface.
- **Bug** Fixed a bug that caused the application to experience a long (30 second) delay when launching on a machine that was not connected to the Internet.
- **Firmware**
 - **Feature** Added support for the CS-900 bi-directional DMX port hardware. Now, each DMX port can be switched to either an input or an output (or disabled).
 - **Feature** Added support for the CS-900 Dual-LAN hardware. Now, CueServer networking can run in one of two modes, (1) a single LAN that carries both lighting and management data combined with a built-in Ethernet switch between the two physical ports, or (2) dual separate LANs with management data on LAN A and lighting data on LAN B.
 - **Feature** Improved DMX port LED indicators to show additional port status information.
 - **Feature** Improved the scroll speed of adjusting LCD menu items.
 - **Feature** Whenever the system contacts a DHCP server, it is shown on the LCD display.
 - **Feature** Added the ON and OFF keywords to the TOGGLE command.
 - **Feature** Added ability for CueServer's Ethernet ports to be "hot plugged" while the device is running.
 - **Feature** The LCD display now shows a small "null" icon next to the device's IP address if there is no active link on that port.
 - **Feature** Added additional power-on hardware test for FPGA bitstream.

- **Feature** A new DHCP fallback address of 10.0.1.234 has been set for the primary interface, and the secondary interface (if present) will fallback to 192.168.1.234.
- **Feature** Changed the DHCP function to timeout more quickly if a DHCP server could not be reached.
- **Bug** Fixed a bug that caused streaming cues inside of cue stacks to not record properly.
- **Bug** Fixed a bug that caused KiNET v2 to not be received properly by some CK power supplies.
- **Bug** Addressed a problem that could cause the timer and trigger daemons to crash if no show file is loaded.
- **Bug** Fixed a problem that caused the csportd and csfpuid daemons to report that they were improperly killed during a firmware update.
- **Bug** Addressed a problem that could cause cue numbers with decimal points to fail to record properly in certain circumstances.
- **Bug** Fixed the warning during the update of apache configuration during a firmware update.
- **Feature** Improved the factory initialization routine.
- **Bug** Addressed a problem that caused a failed assertion when a web client was requesting channel values and no show was loaded.
- **Bug** Addressed a problem that could cause a memory leak during an NTP lookup.
- **Bug** Addressed a problem that could cause networking to fail if the device is connected directly to a laptop without a router.
- **Feature** The LCD display now shows additional information about the boot process.
- **Kernel**
 - **Feature** Updated to CueServer Kernel v1.3.
 - **Feature** Supports booting of CS-900 Hardware.
 - **Feature** Supports auto-update of FPGA bitstream.

Release v1.3.0 [November 11, 2015]

Version 1.3.0 (11/3/2015)

• CueServer Studio 2

- **Feature** Macros editor now has an inline script editor, instead of requiring the user to click into a separate script editor window.
- **Feature** The “New...” menu item in the File menu now creates a new resource or trigger of the currently selected type.
- **Feature** Enabled Cut/Copy/Paste/Clear menu commands for the command line.
- **Feature** Added “Command Line” (Command-L) menu option to move keyboard focus to command line.
- **Feature** The command line now has focus when opening a new Editor Window.
- **Feature** Changing navigation pages or clicking on editor panels no longer removes keyboard focus from the command line.
- **Feature** Added user-selectable serial port protocols.
- **Feature** Added local echo option to serial ports.
- **Bug** Fixed a bug that prevented the Capture window of a cue to not operate properly when a Cue Stack is selected.
- **Bug** Fixed a bug that prevented decimal cue number from being entered, introduced in v1.2.0.
- **Bug** Fixed a bug that caused the Output and Port panels to not clear properly when an output or port component of a station was deselected.
- **Bug** Fixed a bug that caused the Open Web button to do nothing if the active show is selected in the Navigator Window.
- **Feature** Windows: Add ability for Delete key to perform same actions as Backspace key.

• Firmware

- **Feature** Added initial hardware support for new CS-900 model.
- **Feature** Added “Macros” LCD menu item to manually trigger macros.
- **Feature** Added “Shows” LCD menu item to manually switch between shows.
- **Feature** Added parsing module for incoming serial data as CueScript commands in either CueServer 1 compatible format or new CR/LF format.
- **Feature** Added syntax to AT command to handle +/- delta values.
- **Feature** Added comparison operators !=, >= and <=.
- **Bug** Playing audio clips are now halted when switching shows.
- **Bug** LCD Display overrides are now cleared when switching shows.
- **Bug** Addressed a problem with certain sACN alternate start codes being improperly recognized as dimmer levels.

- **Bug** Addressed a problem with AT command that prevented it from properly handling array values.
- **Bug** Addressed a problem that caused the live fade countdown indicator from appearing properly during some fades.
- **Bug** Addressed a problem with serial port baud rate and character format not changing when switching shows.
- **Bug** Addressed a problem that could cause the front-panel display to become unresponsive.
- **Bug** Addressed a problem that caused improperly formed IF/THEN statements to cause the system to become unresponsive.

Release v1.2.0 [July 24, 2015]

Version 1.2.0 (7/24/2015)

• CueServer Studio 2

- **Feature** Added entirely new way to create cues and to capture scenes and/or streams.
- **Feature** Added stream recording trigger channel and recording duration parameters.
- **Feature** Added new record modes for capturing scenes.
- **Feature** Added ability to create and edit cues offline.
- **Feature** Added new “Button Held for n Seconds” rule.
- **Feature** Added the ability to test scripts from within rule definitions.
- **Feature** Improved the end-of-stream action choices (None, Loop, Follow, Release).
- **Feature** Improved fields and controls available for various editors when the show is online/active/offline.
- **Feature** The delete key now removes the selected Cue, Macro, Group, Station, Timer or Rule; holding down the Option/Alt key avoids the warning dialog.
- **Bug** Fixed a bug that caused remote devices appear offline if their connection was interrupted and restored.
- **Bug** Fixed a bug that could cause the drop target of a show upload to remain highlighted even if the drop did not occur.
- **Bug** Fixed a bug that could cause the selected button/contact/output to change selection when the apply button was pressed.
- **Bug** Fixed a bug that displayed a cue’s rule list in the editor panel after a cue was deselected.
- **Bug** Fixed a bug that caused a crash if viewing a streaming cue with a length of zero.
- **Bug** Fixed a bug with trimers set to operate between two dates that would cause the timer to not fire in certain circumstances.
- **Bug** Windows: Addressed a problem that caused some displays to flicker and/or display incorrect information in certain circumstances.
- **Bug** Windows: Addressed a problem that could cause show file uploads/downloads to fail.

• Firmware

- **Feature** Added ACTIVE, ALL, CHANNEL n, INPUT, PLAYBACK n, and TIME n options to the RECORD and UPDATE commands.
- **Feature** Added UPDATE STREAM syntax.
- **Feature** Added ability to convert normal cues to streaming cues and vice versa.
- **Feature** Added ability to send UDP messages using the WRITE command.
- **Feature** Adjusted the value returned by the Indicator command for external button stations.
- **Bug** Fixed a bug that caused the DIO-588 interface to not trigger it’s contact rules.

- **Bug** Fixed a bug that caused the DIO-588 to attempt to use default RGB indicator colors to turn on/off its outputs.
- **Bug** Fixed a bug that caused external button station events to trigger their events twice.
- **Bug** Fixed a bug with the TOGGLE command when operating on indicators of external button stations.
- **Bug** Fixed a bug that prevented streams from being recorded into cue stacks.
- **Bug** Fixed a bug that would cause a single frame of stale DMX Input data to pass through the fade engine after DMX Input was disabled and then re-enabled.
- **Bug** Addressed a problem that caused jitter in the DMX Input stream coming from the built-in DMX ports.

Release v1.1.0 [May 22, 2015]

Version 1.1.0 (5/22/2015)

• CueServer Studio 2

- **Feature** Added KiNET v1 and v2 protocols.
- **Feature** Added the ability to rename shows.
- **Feature** Removed ambiguous “Name” field from General Settings.
- **Bug** Fixed a problem with the “Set Time Now” button in the Clock dialog.
- **Bug** Fixed a bug introduced in 1.0.8 that prevented editing of remote devices that have custom port numbers.
- **Bug** Fixed a bug that would cause the stand-alone Stage and Playbacks windows to not be able to be reopened after closing them.
- **Bug** Adjusted the rendering of LED Indicators on OS X to eliminate color bleeding.
- **Feature** Improved contextual menus in the Navigator window.

• Firmware

- **Feature** Added support for newly updated CuePad app (v2.2).
- **Feature** Added support for interactive web content.
- **Feature** Added KiNET v1 and v2 protocol output for Philips/Color Kinetics fixtures.
- **Feature** Added audio.volume system variable for adjusting the audio output level.
- **Bug** Adjusted the default stereo audio line-out level.
- **Feature** Improved the switching behavior between manual and automatic time adjustment.
- **Bug** Fixed a bug that could cause the real-time clock from being properly updated.
- **Bug** Improved the DMX fade engine’s timebase to be immune from accumulated drift.
- **Bug** Addressed a problem that could cause the front-panel user-interface to crash when the NTP daemon receives a time update.
- **Bug** Addressed a problem that could cause Show Loaded/Unloaded events to not be triggered.
- **Feature** Implemented group query in the get.cgi API.
- **Bug** Fixed a bug that could cause the CueScript parser to crash when the length of the result string was certain multiples of 8 bytes long.
- **Bug** Addressed a problem that could cause overlapping audio clips to not properly terminate.
- **Feature** Rolled the LCD display driver into the UI server for better performance.

Release v1.0.8 [April 27, 2015]

Version 1.0.8 (4/27/2015)

• CueServer Studio 2

- **Feature** Added colored icons to the Stage View's "view" menu to make it easier to identify which playback is being selected.
- **Feature** Added a new "View" menu to the Stage View that allows all universes, or only a specific universe to appear in the display.
- **Feature** Added user-assigned names to the playback faders in the Playbacks view.
- **Feature** The command line and live views are now only available from the active show editor window.
- **Feature** The Editor Window now shows "[ACTIVE]" in its title bar when viewing an active show.
- **Feature** Resource and Trigger editor panels now remember what state they were in when switching between panels.
- **Feature** Resource editor panels now refresh automatically when an object is recorded or updated by CueScript commands.
- **Bug** Fixed a bug introduced in 1.0.7 that crashed when opening offline shows.
- **Bug** Fixed a problem with the Stage View that would only allow the first eight playback faders to be selected in the View menu.
- **Bug** Fixed an issue that would cause the editor for Stations or Buttons to disappear when changes were applied.
- **Bug** Fixed a problem where the entire device list in the Navigator window could get a green background when dragging a project into the list.
- **Feature** Renamed the previous View menu to Layer in the Stage View for consistency.
- **Bug** Fixed a problem with the Editor Window that would reload the active editor if the currently selected editor was clicked on.
- **Feature** Updated the default index.shtml file in the new show template.
- **Bug** Addressed a problem that could cause the reported uptime to be blank.
- **Bug** Added missing category icons for several editor panels.
- **Feature** Added "Refresh" menu item to pop-up gear menu for Cues, Groups, Macros, Sounds and Web Pages.
- **Feature** Updated compilers resulting in more compact Windows builds.
- **Bug** Fixed a Drag & Drop highlighting problem with the Web and Sound file browsers.
- **Bug** Fixed a problem that caused folders to not be able to be dragged into the Web and Sound browsers.
- **Bug** Addressed problems with creating/deleting stations when editing offline show files.

- **Bug** Addressed a problem that could cause the station panel to crash when changing the selected station.
- **Bug** Addressed a problem with text fields that would not properly select the entire field on mouse click entry.
- **Bug** Improved the text entry interaction with hours/minutes/seconds entered into timers.
- **Bug** Fixed an issue with improperly formatted data being stored for the “only specific days” type of timer scheduling.
- **Bug** Windows: Fixed a problem with offline show paths appearing with slashes instead of backslashes.
- **Bug** Windows: Fixed a problem that prevented Drag & Drop to the Web and Sound file browsers from the Desktop.
- **Bug** Windows: Fixed a problem with field validation that sometimes caused the insertion point to move unexpectedly.
- **Bug** Windows: Fixed a problem in the Specific Month and Year dialogs that would cause the checkboxes to not draw properly in certain circumstances.
- **Bug** Windows: Fixed a problem where the System Log may not display the entire log file.
- **Firmware**
 - **Feature** Added the INPUT ENABLE/DISABLE syntax to enable or disable the DMX Input layer of the playback stack.
 - **Feature** Added automatic updating of playback fader user preferences for combine modes when loading or switching shows.
 - **Feature** Added ability to specify a wider range of weeks of the month when picking date ranges for timers (i.e.: 5th Friday, or 2nd from Last Wednesday, etc.).
 - **Feature** Added the ability to query variable values to the get.cgi API.
 - **Bug** Addressed a problem with CueScript parsing timeouts being raised too quickly.
 - **Bug** Fixed an issue that would cause the CueScript parser to erroneously report itself as shut down after executing a WAIT command.
 - **Bug** Fixed a problem introduced in 1.0.7 that caused the WAIT CLEAR command to raise an exception.
 - **Bug** Fixed a problem where incomplete CueScript strings would silently fail without reporting an error.
 - **Bug** Fixed a problem that could cause CueStations to not respond after switching active shows.
 - **Bug** Addressed a problem that could cause timers to fail to trigger that are set for “nth weekday of the month” in certain circumstances.
 - **Feature** Improved error descriptions for unrecognized commands.

Release v1.0.7 [April 7, 2015]

Version 1.0.7 (4/7/2015)

• CueServer Studio 2

- **Feature** Added a display of the current Stack Name to the Playback view.
- **Feature** Added an indicator to the Universe Settings panel to show if a universe is receiving input data.
- **Feature** Added Variables sub-view to the Status panel that shows any currently defined user variables in the system.
- **Feature** Added CPU Info sub-view to the Status panel that shows the running status of the various CueServer processes, average CPU load and memory usage.
- **Feature** Added System Log sub-view to the Status panel that shows the system log file and allows the system message indicator to be cleared.
- **Feature** Added a warning indicator to the navigator panel in the Editor Window that shows when an important message is available in one of the sub-panels.
- **Bug** Addressed a problem with the format of the query string when CueServer Studio attempts to fetch the current version of software.
- **Bug** Addressed a problem with the progress indicators in the Stations, Timers and Rules panels not moving properly when the window is resized.
- **Bug** Changed the global fade time label in the command field to "Time".
- **Bug** Fixed a spelling mistake in the Clock Settings window.
- **Bug** Adjusted the minimum allowable size for the Navigator Window.

• Firmware

- **Feature** Added the AT CUE syntax for selectively recalling specific channels from a cue.
- **Feature** Added the AT PLAYBACK syntax for selectively recalling specific channels from a playback fader.
- **Feature** Added syntax for adding or subtracting groups to the current group selection using GROUP x + y - z.
- **Feature** Changed the behavior of testing rule condition variables to interpret a null-string ("") as being equal to zero (0).
- **Bug** Addressed a problem that caused CueServer to not communicate properly with sACN or CueStation nodes if there was no router on the network.
- **Bug** Addressed a problem that could cause only one universe of sACN data to be received as input into the system.
- **Bug** Addressed a problem that would leak socket resources when sending CueStation Hub indicator changes.

- **Bug** Addressed a problem with the Stack command that caused empty stacks to produce an error.
- **Bug** Addressed a problem that prevented a Playback Fader to have its stack name cleared by setting the stack name to the empty string.
- **Bug** Addressed a problem that caused the Clear command to not clear a Playback Fader's stack property.
- **Bug** Addressed a problem that could prevent setting of static IP address parameters via the LCD Menu.
- **Bug** Addressed a problem that would cause the device to not be discoverable when booted on a network without a router.
- **Bug** Addressed several issues with the get.cgi API for compatibility with the CuePad iOS app. CueServer 2 requires CuePad v2.2 or greater.
- **Bug** Addressed a problem that prevented CueScript commands to be able to be unicast to the CueServer's IP Address.
- **Feature** Added additional error checking and reporting to the various daemon processes.

Release v1.0.6 [March 13, 2015]

Version 1.0.6 (3/13/2015)

- **CueServer Studio 2**
 - **Bug** Addressed a problem introduced in 1.0.5 that caused the Stations editor panel to not appear properly if an external station was edited immediately after editing the built-in station.
- **Firmware**
 - **Bug** Addressed a problem introduced in 1.0.5 that caused buttons and contacts on external stations to not trigger properly.

Release v1.0.5 [March 11, 2015]

Version 1.0.5 (3/11/2015)

• CueServer Studio 2

- **Feature** Show project files can now be downloaded/uploaded to/from your computer.
- **Feature** Offline project files can now be edited without the CueServer hardware.
- **Feature** Added a second list view to the main Navigator window to make is easier to work with offline project files.
- **Feature** Added the ability to create new projects from within CueServer Studio.
- **Feature** Drag and Drop has been added to move show project files between a CueServer and your computer.
- **Feature** Added Network Settings window to remotely change a CueServer's network settings.
- **Feature** Added Time Settings window to set manual or automatic time settings, including definitions for over 400 time zones.
- **Feature** A warning dialog now appears when you try to edit a CueServer that has outdated firmware.
- **Feature** Changed CueScript buttons to show newlines as semicolons (;) to be more consistent with syntax rules.
- **Bug** Addressed a problem that caused device discovery to only work on the host's default Ethernet interface.
- **Feature** Windows: Enabled the main window's close box.
- **Bug** Windows: Reduced the flickering of the Playback and Status panels.
- **Bug** Windows: Addressed a problem that would cause CueScript buttons to not display multiline text properly.
- **Bug** Windows: Addressed a problem that caused the Control-C shortcut for "Copy" to not work properly in the CueScript popup editor window.
- **Bug** Windows: Addressed a problem that caused the popup menu controls in the Stage view to display incorrect labels.
- **Bug** Windows: Addressed a problem that would cause the main window to not open properly if reopened after the app was closed while minimized.

• Firmware

- **Feature** Added the ability for Buttons and Contacts to be enabled/disabled.
- **Feature** Implemented the Update Cue and Update Group commands.
- **Feature** Pressing and holding the Up/Down navigation buttons while editing values on the LCD Display now continuously adjusts the value.

- **Feature** Changed the behavior of the “=” command to dynamically act as either Assign or Equals, depending on the context of the parameters.
- **Bug** Addressed a problem that caused sACN to not receive data properly from certain consoles.
- **Feature** Improved sACN receive logic to deal with transmitters that do not properly terminate
- **Bug** Addressed problems with setting the Network Settings using the LCD Display that would cause unexpected results.
- **Bug** Addressed a problem with the Fade and Time commands that caused them to not be able to receive their values from variables.
- **Bug** Addressed a problem with the LCD Display that could cause it to freeze if the system time was adjusted in certain circumstances.
- **Bug** Addressed a problem with the sACN protocol not properly supplying a valid CID field for transmit packets.
- **Feature** Added additional network diagnostics support to csctl.

Release v1.0.4 [February 9, 2015]

Version 1.0.4 (2/9/2015)

• CueServer Studio 2

- **Feature** Added the ability to open local offline show files.
- **Feature** Added an Cue Fade Times popup window that provides direct access to extended fade time attributes.
- **Feature** Added cursor and history control to the command line using the up/down/right/left arrow keys.
- **Feature** Added contextual menu to Web Pages that includes option to open files in the user's web browser.
- **Feature** Added new rule conditions for testing indicators and outputs.
- **Feature** Added a watermark that appears when no editor panel is selected.
- **Feature** Added support for Rev. B hardware.
- **Feature** The navigator window now remembers it's preferred size and column widths.
- **Bug** Improved the description and input validation of the Add Remote CueServer window.
- **Bug** Addressed a problem that caused the Fade/Follow/Link fields in the Cue panel that could make it difficult to remove unwanted values or enter decimal numbers.
- **Bug** Addressed a cosmetic problem with the Month/Day/Year popup menus in the Active Days section of the Timers panel.
- **Bug** Addressed a problem with Timers set to trigger between two dates that would cause the Weekdays field to have an invalid default value.
- **Bug** Addressed a problem that caused the default value of the Sun Brightness Rule Condition to be undefined.
- **Bug** Addressed a problem that could cause a crash if a CueScript popup editor button was double-clicked.
- **Bug** Windows: Fixed the titles of several dialog box windows.
- **Bug** Windows: Select entire text field when entering the Rename Cue Stack window.
- **Bug** Windows: Change the Sounds and Web Pages panels to display file paths with the correct path delimiters.

• Firmware

- **Feature** Changed the rule execution behavior to execute rules in two passes (check eligibility first, then perform actions), which solves a multi-rule race condition problem.
- **Feature** Added a new default index.shtml file to the Web Resources of new shows.
- **Feature** Exported several show and device related environment variables in the Apache virtualhost for use with web pages' CGI and SSI scripting.

- **Bug** Addressed a problem with switching shows not causing Apache to properly switch the served web pages.
- **Bug** Addressed a problem that caused the Fade command to fail to modify the next cue's execution time in certain circumstances.
- **Bug** Addressed a problem that caused the Toggle command to not work with button indicators or digital outputs.
- **Bug** Addressed a problem that caused shows with spaces or other special characters in their name to not be able to be deleted.
- **Bug** Addressed a problem that caused Telnet sessions to hang in certain situations.

Release v1.0.3 [January 22, 2015]

Version 1.0.3 (1/22/2015)

- **CueServer Studio 2**

- **Feature** First version available as both OS X and Windows builds.
- **Feature** Added the ability to add and remove remote devices to the CueServer Navigator window.
- **Feature** Added firmware version column to the Navigator window.
- **Feature** Added a built-in firmware image that matches the release version of Studio.
- **Bug** Addressed a problem that caused the command line text to appear very small on Retina displays.
- **Bug** Addressed a problem that could allow automatic text substitutions to occur in the script editor popup window.
- **Bug** Addressed a problem that would cause the app to not launch properly if the splash screen was clicked.

- **Firmware**

- **Bug** Addressed a problem that caused remote devices to not return the correct ping data via HTTP.

Release v1.0.2 [January 9, 2015]

Version 1.0.2 (1/9/2015)

• CueServer Studio 2

- **Feature** Added rules to cues. Previously, cues only had a single action field. Now each cue can have an arbitrary number of rules associated with them.
- **Feature** Added a cue “contents preview” to the Cue editor panel. This panel shows the first channels and/or information about the stream.
- **Feature** Added resizable divider between panels in the Cues, Groups, Macros, Timers and Rules panels.
- **Feature** Added an automatic software version check when the application is launched.
- **Feature** The Editor Window now remembers it's preferred size.
- **Feature** The License Code Details window no longer displays window resize controls.
- **Feature** Added drop-down menu arrows to buttons in Stage View.
- **Feature** Enabled the File>Close menu item for separate Stage and Playback view windows.
- **Feature** The New Cue window now remembers the last used capture mode.
- **Bug** Addressed a problem that caused the Capture Selected Channels cue recording mode to fail.
- **Bug** Addressed a problem that could cause a crash when exiting from full-screen mode on OS X.

• Firmware

- **Feature** Added the FOLLOW CLEAR command variant.
- **Bug** Addressed a problem that could cause cues with more than about 4000 channels to not play back correctly.
- **Bug** Addressed a problem that caused the Universe Loss event to be sent more often than expected to the global rules.
- **Bug** Addressed a problem that could cause cue execution to improperly return an error if the cue contained an action.
- **Bug** Addressed a problem with the CLEAR command not clearing parked channels.
- **Bug** Addressed a problem with the CLEAR command not restoring a playback's submaster to 100%.
- **Bug** Addressed a problem that could cause the RESET command to crash the CueScript parser.
- **Bug** Addressed a problem with the RESET command not clearing commands that were in the wait queue.

Release v1.0.1 [December 23, 2014]

Version 1.0.1 (12/23/2014)

- **CueServer Studio 2**

- **Feature** Added “User’s Manual...”, “Support Website...”, and “Release Notes...” to the Help Menu.
- **Feature** Changed the font used for displaying CueScript commands.
- **Bug** Addressed a problem on OS X that made it possible to insert “smart quotes” into CueScript fields, which would cause the execution of the script to fail.
- **Bug** Addressed a problem with the *Open Web* command that could cause the web page to not open properly.

- **Firmware**

- **Feature** Changed the LCD Menu display for System Information.
- **Bug** Addressed a problem that could cause show data to not be synchronized with the memory card.
- **Feature** Improved error reporting for I2C Bus Daemon.

Release v1.0.0 [December 18, 2014]

Version 1.0.0 (12/18/2014)

- First public release version.
- All versions prior to v1.0.0 were private.

Legal Notices

Copyright © 2006-2018 Interactive Technologies, Inc. All rights reserved.

Interactive Technologies, the Interactive Technologies logo, CueServer, CueServer 2, CueStation, and CueTouch are trademarks of Interactive Technologies, Inc.

Acknowledgements:

Portions of CueServer Studio and/or CueServer 2 Firmware may utilize the following copyrighted material, the use of which is hereby acknowledged.

- **alsa-lib**

Parts of this software include the alsa-lib libraries (alsa-project.org). Under terms of the GNU Lesser General Public License (LGPLv2.1), this package's corresponding source code is included alongside the compiled binary within the filesystem of this product. Complete instructions for accessing and re-compiling this software will be provided upon written request to Interactive. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details; a copy of the LGPLv2.1 is included below.

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

Copyright © 1991, 1999 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages—typically libraries—of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be

allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables. The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the

Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General

Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a “work that uses the Library”. Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License. However, linking a “work that uses the Library” with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a “work that uses the library”. The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a “work that uses the Library” uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a “work that uses the Library” with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the

Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable “work that uses the Library”, as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user’s computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the “work that uses the Library” must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice. This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and

conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the library's name and an idea of what it does.

Copyright © year name of author

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the

License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.

signature of Ty Coon, 1 April 1990

Ty Coon, President of Vice

That's all there is to it!

- **Apache**

Apache License Version 2.0, January 2004

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

“Derivative Works” shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

“Contribution” shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, “submitted” means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as “Not a Contribution.”

“Contributor” shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

You must give any other recipients of the Work or Derivative Works a copy of this License; and
You must cause any modified files to carry prominent notices stating that You changed the files; and
You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

If the Work includes a “NOTICE” text file as part of its distribution, then any Derivative Works that You

distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any

other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: HOW TO APPLY THE APACHE LICENSE TO YOUR WORK

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets “[]” replaced with your own identifying information. (Don’t include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same “printed page” as the copyright notice for easier identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the “License”);
you may not use this file except in compliance with the License.

You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

- **Art-Net™**

Art-Net™ Designed by and Copyright Artistic Licence Holdings Ltd. Information about Art-Net can be found at art-net.org.uk.

- **CGIC**

CGIC, copyright 1996-2011 by Thomas Boutell and Boutell.Com, Inc (boutell.com). Permission is granted to use CGIC in any application, commercial or noncommercial, at no cost. HOWEVER, this copyright paragraph must appear on a “credits” page accessible in the public online and offline documentation of the program. Modified versions of the CGIC library should not be distributed without the attachment of a clear statement regarding the author of the modifications, and this notice may in no case be removed. Modifications may also be submitted to the author for inclusion in the main CGIC distribution.

- **CZMQ**

Parts of this software include the CZMQ libraries (czmq.zeromq.org). This software used under license of the GNU Lesser General Public License (LGPLv3) with Static Link Exception. A copy of the LGPLv3 is included below.

STATIC LINK EXCEPTION

As a special exception, the Authors give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you must extend this exception to your version of the library.

Note: this exception relieves you of any obligations under sections 4 and 5 of this license, and section 6 of the GNU General Public License.

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to

be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.
- d) Do one of the following:

- 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

- 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

- e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application

Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy’s public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.

- **Fat Icons Pack**

Fat Icons icon pack made by [Designerz Base](#) from [flaticon.com](#) is licensed by [CC 3.0 BY](#).

- **Freepik Icons**

Some icons included in this software are made by [Freepik](#) from [flaticon.com](#) is licensed by [CC 3.0 BY](#).

- **Justicons Icon Pack**

Justicons icon pack made by [Rami McMin](#) from [flaticon.com](#) is licensed by [CC 3.0 BY](#).

- **LibConfig**

Parts of this software include the [LibConfig](#) libraries owned by Hyperrealm. Under terms of the GNU Lesser General Public License (LGPLv2.1), this package’s corresponding source code is included alongside the compiled binary within the filesystem of this product. Complete instructions for accessing and re-compiling this software will be provided upon written request to Interactive. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even

the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

[Interactive Note: For a copy of the GNU Lesser General Public License Version 2.1, please refer to the alas-lib attribution, where a copy of the license is included.]

- **ZeroMQ**

Parts of this software include the ZeroMQ libraries (zeromq.org). This software used under license of the GNU Lesser General Public License (LGPLv3) with Static Link Exception. [Interactive Note: For a copy of the GNU Lesser General Public License Version 3, please refer to the czmq attribution, where a copy of the license is included.]

SPECIAL EXCEPTION GRANTED BY COPYRIGHT HOLDERS

As a special exception, copyright holders give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you must extend this exception to your version of the library.