

QLab 5

Reference Manual

Last updated on November 25, 2025 for QLab 5.5.8

© 2025 Figure 53, LLC. All Rights Reserved.

Table of Contents

Chapter 1: General

1.1 **New in QLab 5**
1.2 **QLab 5 Change Log**
1.3 **System Recommendations**
1.4 **Preparing Your Mac**
1.5 **Keyboard Shortcuts**
1.6 **Licenses**
1.7 **Features by License**
1.8 **QLab Preferences**

Chapter 2: Fundamentals

2.1 **The Workspace**
2.2 **Workspace Settings**
2.3 **Workspace Templates**
2.4 **Managing Workspaces**
2.5 **Cues**
2.6 **The Inspector**
2.7 **Cue Lists**
2.8 **Cue Carts**
2.9 **Group Cues**
2.10 **Cue Sequences**

Chapter 3: Tools

3.1 **The Tools Menu**
3.2 **Find**
3.3 **Paste Cue Properties**
3.4 **The Launcher Window**
3.5 **Monitor Windows**
3.6 **Auditioning Cues**
3.7 **Override Controls**

Chapter 4: Workflow Tools

4.1 **The Workspace Status Window**
4.2 **The Timecode Status Window**

Chapter 5: Audio

5.1 **Intro to Audio**
5.2 **Object Audio in QLab**
5.3 **Audio Cues**
5.4 **Mic Cues**
5.5 **The Audio Output Patch Editor**
5.6 **The Audio Map Editor**
5.7 **Fading Audio & Audio Effects**

Chapter 6: Video

6.1 **Intro to Video**
6.2 **Video Cues**
6.3 **Camera Cues**
6.4 **Text Cues**
6.5 **Video Output**
6.6 **Using NDI**
6.7 **Fading Video & Video Effects**

Chapter 7: Lighting

7.1 **Intro to Lighting**
7.2 **Light Cues**
7.3 **The Light Dashboard**
7.4 **The Lighting Command Language**
7.5 **The Light Patch Editor**
7.6 **Light Library**

Chapter 8: Networking, MIDI, and Show Control

8.1 **Collaboration**
8.2 **QLab Remote**
8.3 **Stream Deck**
8.4 **Using OSC**
8.5 **Using MIDI & MSC**
8.6 **Using Timecode**
8.7 **Show Control Broadcast**
8.8 **Network Cues**
8.9 **MIDI Cues**
8.10 **MIDI File Cues**
8.11 **Timecode Cues**

Chapter 9: Scripting

9.1 **OSC Dictionary**
9.2 **OSC Queries**
9.3 **Script Cues**
9.4 **AppleScript Dictionary**
9.5 **Parameter Reference**
9.6 **OSC & Scripting Examples**

Chapter 10: Other Cues

10.1 **Transport Cues**
10.2 **Devamp Cues**
10.3 **GoTo Cues**
10.4 **Target Cues**
10.5 **Arm and Disarm Cues**
10.6 **Wait Cues**
10.7 **Memo Cues**

Chapter 11: Tutorials

11.1 **The GO Button**
11.2 **How To Use A Mac**
11.3 **Basic Networking**
11.4 **Understanding USB-C**
11.5 **Sam's Toolbox**

Chapter 1: General

1.1 New in QLab 5
1.2 QLab 5 Change Log
1.3 System Recommendations
1.4 Preparing Your Mac
1.5 Keyboard Shortcuts
1.6 Licenses
1.7 Features by License
1.8 QLab Preferences

New in QLab 5

These are the differences between QLab 5 and QLab 4, including major updates made throughout the lifetime of QLab 5. For a fully inclusive and detailed list of all changes made in each release of QLab 5, check out the change log.

General

Stop, QLaborate, and Listen

QLab 5 allows multiple people on separate Macs to collaborate on a workspace, live and in realtime, on a local network.

Autosave

Autosave prevents accidental loss of work in the event of a crash or power outage. No guarantees are made against data loss due to user error, acts of God, or most hauntings.

I Am Not Throwing Away My Snapshot

As part of Autosave, QLab 5 optionally saves a snapshot of your workspace every time you save, allowing you to easily access earlier versions of your work.

Playlist Mode

Group cues have a new Playlist mode that allows you to create sequential playlists which advance automatically or manually with optional crossfading, shuffling, and looping.

Audition Improvements

The Audition Window has been replaced by a set of audition-related tools:

- Each category of cue output (Audio, Video, MIDI, MTC Timecode, LTC Timecode, Network, and Light) can be individually set to audition to a specific output, the same output as usual, or no output.
- Video output can also be set to audition to an audition window. Video stages (formerly called video surfaces) each get their own audition window.
- Light cues audition to an “Audition” tab in the Light Dashboard.
- Audition GO and Audition Preview actions let you audition individual cues and cue sequences without affecting the rest of the workspace.
- Workspaces can be set to *Always audition* which turns every GO into an Audition GO and every Preview into an Audition Preview, in the style of QLab 4.
- Cues in the midst of auditioning can be interrupted and restarted normally with a single GO or Preview command; no need to stop first, then restart.

Going Dark!

QLab 5 uses macOS Dark Mode so that dialogue boxes, title bars, and other window “chrome” look more consistent with the rest of QLab’s interface.

Cues A La Cart

Cue carts have received a variety of updates:

- You can now manually set the grid size of a cart anywhere from 1 × 1 cells up to 15 × 15 cells.
- Cart cues can now have pre-waits.
- Cart cues display their duration and pre-wait, if applicable, when they are not playing.
- Carts now have three display sizes, just like lists, which can be configured in the General → Display Size section of Workspace Settings.
- Assorted improvements to aesthetics and legibility.

In Living Color

The number of cue colors has been expanded to make it easier to color-code a large variety of cues, and to better support the needs of people with color vision deficiency. The cue color control in the Basics tab has been redesigned to support this larger number of colors, and you can now set both a cue color and an optional second cue color. If enabled, the cue changes to the second cue color after it's started.

Reconnect With Old Friends

A new file search tool makes it easier to reconnect broken cues to their missing file targets, for example after moving a workspace to a new computer or reorganizing media while QLab is not running.

Drag-and-drop now less of a drag

Setting cues' targets by drag and drop now works bi-directionally: you can drag the acting cue onto its target cue, or drag the target cue onto the acting cue. If both cues could potentially be targets of each other, the dragged cue is set as the target of the cue it's dropped onto.

Related but separate, dragging cues onto [ ] Group cues now sets the [ ] Group cue as the target of the dragged cues, instead of potentially re-targeting all the child cues of that Group cue.

Right Click, Right Time

Right-clicking (or clicking with the control key held down) on cues in the cue list or on controls in the inspector now brings up a contextual menu which gives you quick access to relevant actions and extensive in-line help text with links to the relevant section of the QLab manual online.

Hammer Out Danger, Hammer Out Warning

The Warnings tab of the Workspace Status window has been comprehensively updated to give you much more and much better information about broken cues as well as non-breaking warnings such as workspace settings which are in need of attention.

Workspace Settings To Go

Workspace Settings can now be exported to a settings file, imported from a settings file, imported from other open workspaces, and drag-and-dropped between workspaces.

Sharper Fades

The "linear" fade curve type can now be edited, allowing sharp-cornered multi-step fade curves.

Relatively Absolute fading confidence

Fade cues in "absolute" mode will now override and clear out any previously applied changes made by Fade cues in "relative" mode.

Negative Load

Load cues as well as OSC and AppleScript commands that involve loading, can now use negative numbers to load backwards from the end time of their targets, just like the *Load to Time* tool.

More Pasting, More Batching

Many, many more cue parameters are now fancy-pasteable and batch-editable when multiple cues are selected simultaneously.

Hop, Skip, and Jump

Disarmed cues can optionally be skipped, making it easier to pull cues entirely out of a show temporarily or experimentally.

Audio

Objectively Cooler

Object-based audio lets you effortlessly move sounds around your system in a brand new way, routing, fading, and mixing in a fully flexible visual environment.

Patch Upgrades

Audio patches are now called *audio output patches*, and they have been overhauled in QLab 5 with the following improvements:

- QLab now supports an unlimited number of audio output patches in a workspace. Audio output patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the audio settings of other workspaces.
- You can create multiple audio output patches with the same audio device, each with its own routing and audio effects.
- The audio patch editor now supports undo and redo.
- All cues that output audio now use the same set of audio output patches (no more special output patches for Mic cues.)
- If an audio device is disconnected, QLab will still display the name of the disconnected device in the audio inspector, making it easier to work with "offline" audio devices.
- Audio patches can now use the system output, which is the audio device selected in System Preferences → Sound → Output. This makes it easier for a workspace to be used in a context where QLab ought to use the same audio output as the rest of the Mac, even if that output changes.
- Audio patches now support up to 128 cue outputs and 128 device outputs, and you can define exactly how many cue outputs each audio output patch uses, between 1 and 128. Patches which use fewer cue outputs will use less processing power.

Mic Cue Flexibility

Mic cues now use separate *audio input patches* to designate the device they use for input. This lets you use separate audio input and output devices easily, without requiring you to set up an aggregate audio device. Audio input patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the audio settings of other workspaces.

Zero-Count Slices

Slices in Audio cues (and Video cues) can be set to a play count of `0`; zero-count slices are seamlessly skipped during playback.

Batch-edit Waveforms

You can now visually edit the start and end times of multiple selected cues by dragging the start time and end time handles in the Time & Loops tab of the inspector.

Better By a Meter

Audio effect meters now work in all contexts, not just on outputs.

Input Naming

You can now name inputs (rows) in the Levels tab of the inspector for any cue that contains audio. Input names are per-cue, and can be edited in the inspector, by AppleScript, and by OSC.

Mute and Solo

You can now mute and solo cue outputs and audio patch outputs.

Patch Cueing

Fade cues and Reset cues can now target audio output patch routing, allowing you to adjust the overall levels of the entire patch, and easily reset those levels to their initial state.

Video

The Most Metal QLab

The video rendering engine has been completely rewritten using the Metal framework, Apple's modern and fully up-to-date video system. Come for the performance improvements, stay for the longevity.

Will It Blend? Yes.

QLab 5 adds per-cue blend modes, allowing you to composite cues in nearly limitless combinations.

NDI

QLab natively supports NDI 5 for both video input and output. QLab also supports NDI audio input and output.

In-camera Audio

Camera cues contain an embedded Mic cue, allowing you to use live audio alongside live video.

QLab Can You See Me

QLab can display monitor windows for every video input and output so that you can keep tabs on all your visual elements live and in realtime.

Video Effect Upgrades

Video, Camera, and Text cues can now use multiple video effects simultaneously. The list of available video effects has grown, too, thanks to the shift to Metal, and the amount of processing power needed for video effects (especially blurs) has been nicely reduced.

Patch Upgrades

QLab now supports an unlimited number of *video input patches* in a workspace. Video input patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the video settings of other workspaces.

Zero-Count Slices

Slices in Video cues (and Audio cues) can be set to a play count of `0`; zero-count slices are seamlessly skipped during playback.

The Mask of Syphon

Masks and video surface geometry now happen “upstream” of Syphon outputs, allowing you to send more elaborately crafted video feeds to Syphon-receiving clients.

Crop It Like It’s Hot

Video, Camera, and Text cues now have an integrated crop attribute for quick and easy trimming.

Grid Upgrades

Stage grids now show more information and have more visual aids to help you during setup and focus.

Path Fading

Video geometry can now be faded using a 2D path, allowing you to easily fade pairs of settings like X and Y translation or X and Y scale.

Key/Fill Support

QLab supports downstream keying on Blackmagic Designs devices with that capability.

Lighting

The Audition Tab

The Light Dashboard’s new Audition tab lets you audition Light cues, letting you view level changes without actually changing the live levels being output to your lighting system.

More Lights

QLab 5 ships with instrument definitions for over 1700 different fixtures from over 60 manufacturers.

Be What Hue Want To Be

Light instruments with hue, saturation, and intensity (or hue, saturation, and value) parameters can now use a graphical HSI color picker in the Light Dashboard and in the Levels tab of the inspector.

Cut/Copy/Paste

Light commands can now be cut, copied, and pasted when they are selected in slider mode in the Levels tab of the Light cue inspector.

MIDI, Networking, Scripting, and Show Control

Chase Timecode (Not Waterfalls)

QLab 5 has a number of new features related to incoming timecode:

- Cues set to trigger from timecode can now start in the middle of the cue based on incoming timecode, rather than just at the beginning of the cue, and will skip ahead or back in response to timecode skipping ahead or back.
- Lists and Carts set to receive timecode can optionally be set to pause or stop their timecode-triggered cues when incoming timecode stops, with optional freewheeling up to two seconds.
- List and Carts set to receive timecode can respond to jumps in timecode in a number of ways, optionally reaching “back” and starting cues which would have triggered from earlier timecode frames.
- A completely redesigned timecode window shows detailed information about all incoming and outgoing timecode streams.

A More Powerful Network Cue

The Network cue has been substantially revamped and can now send OSC, plain text via UDP, and hexadecimal codes via UDP. It also includes support for directly controlling dozens of network-controllable programs and devices with a minimum of fuss and complexity through a collection of built-in network device descriptions.

The Network cue also supports both TCP and UDP transport, as well as OSC 1.1 argument types `true`, `false`, `impulse`, and `null`.

When live fade preview is switched on, Network cues in 2D fade mode now transmit their message as you drag the control dot around. This should make it easier to experiment with 2D network fades.

Also, 2D fade curve editing has been improved.

Patch Upgrades

Network patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the Network settings of other workspaces.

Network patches can now have an arbitrary number of destinations. This allows a Network cue to send a single message to multiple different network devices. Each destination can have a unique combination of network transport, interface, IP address, port, and/or passcode values.

QLab now supports an unlimited number of MIDI patches in a workspace. MIDI patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the MIDI settings of other workspaces.

MIDI cues, MIDI File cues, and Timecode cues using MTC now all use the same set of patches.

Port Authority

You can now customize the port numbers that workspaces use to receive OSC messages and plain text messages.

Passcode Improvements

Workspaces can have multiple OSC passcodes, each with their own set of access permissions.

Custom OSC remote control commands are now compatible with workspaces that use OSC passcodes.

Expanded Dictionaries

The OSC and AppleScript dictionary for QLab 5 has been greatly expanded.

Show Control Broadcast

QLab can now optionally send show control messages via OSC and MSC which correspond to basic show events such as GO, panic, reset, and cues starting and stopping. OSC messages can be requested by a client using the new `/listen` OSC message which has options for specifying the scope of interest. MSC messages can be sent using MIDI patches.

QLab Remote

- Multiple window support on iPad. This means you can connect to multiple workspaces (on multiple Macs, even) and view them side by side, or you can have multiple simultaneous views into the same workspace.
- QLab Remote 5 does a better job of automatically recovering from a momentary disconnection.
- QLab Remote 5 does a better job of reconnecting to the same Mac after an interruption, even when there are multiple copies of the same workspace open on the network.
- The passcode entry screen is much improved.
- Several bugs are fixed, notably a zoom scale issue when switching between carts and a light instrument check issue pertaining to use of non-ASCII characters.
- When connecting to a QLab 5 workspace, QLab Remote supports encrypted network communication.
- QLab Remote 5 requires iOS 15 or later and is compatible with QLab 3.0 or later.

Specific Changes From QLab 4

- QLab 5 requires a Mac running macOS 11 (Big Sur) or later.
- Group cues now default to timeline mode.
- The overall audio level for cues and devices is now referred to as “main”, not “master.”
- The lighting term “submaster” has been replaced by the term “subcontroller.”
- *Workspace Settings* has moved from the **Window** menu to the **File** menu.
- *Jump to cue…* has moved from the **Tools** menu to the **View** menu, and has been renamed *Select cue…*
- The main audio level of Mic cues and Camera cues now defaults to -INF to help prevent audible accidents.
- The OSC and AppleScript “pitch shift” commands have been replaced with “preserve pitch” commands to directly match the checkbox in the inspector which they control.
- All OSC and AppleScript pertaining to video settings have been substantially revised, since video settings and the video engine have themselves been substantially revised. Many if not most Network cues and Script cues imported from QLab 4 which have anything to do with video will likely need to be revised.
- Copying or importing Network cues from QLab 4 that communicate with the d&b DS100 and use background images will need to have their background images re-connected.
- QLab 5 does not import QCart files. If you have a QCart file in need of updating, you can open it in QLab 4, save it as a QLab 4 workspace, and then open that workspace in QLab 5 and re-save it as a QLab 5 workspace.

QLab 5 Change Log

5.5.8 - November 25, 2025

FIXED: Grids and guides were accidentally broken in 5.5.7, but now they work again.

5.5.7 - November 24, 2025

ADDED: A network device definition for Capture 2025.

ADDED: A network device definition for Holophonix.

ADDED: A network device definition for the ZerOS line of lighting consoles.

ADDED: An option in the VISCA over IP network device definition to include the VISCA over IP header message which is used by many, but not all, devices that support VISCA over IP.

CHANGED: Certain errors reported by audio effects are now logged instead of presenting an alert, to avoid interruptions when opening a workspace.

FIXED: A regression in 5.5 that could cause video stages to lose their connection to video outputs.

FIXED: A regression in 5.5 in which the audio volume level could be incorrect very briefly at the beginning of a cue.

FIXED: A regression in 5.5 in which hot key triggers on Video cues did not persist when re-opening a workspace.

FIXED: A regression in 5.5 in the Time & Loops tab in remote workspaces that prevented clicking in the waveform to load Video cues.

FIXED: Cues that play still images, text, and video inputs now synchronize their start times against cues that play video files.

FIXED: It is no longer necessary to pre-load Camera cues to ensure that you will not see old frames from video inputs.

FIXED: The Basics tab now shows the correct duration for Audio and Video cues on remote workspaces.

FIXED: Several bugs in the Timeline tab of Group cues that caused incorrect playback location cursors in cues with infinitely looping slices or after devamping fixed-count looping slices.

FIXED: The Timeline tab of Group cues now displays slice regions and waveforms for Video cues in remote workspaces.

FIXED: A bug that caused an Audio cue targeting a video file to display an end time of 00:00.000 in the Time & Loops tab of the inspector.

FIXED: The Time & Loops tab now shows the correct loading position while dragging the Load to time tool in cues with infinitely looping slices.

FIXED: Improves the accuracy of the waveform position indicators during playback.

FIXED: An issue in which changes to QLab's preferences were sometimes not saved if the text field that was being edited still had "focus" when closing the preferences window or quitting QLab.

FIXED: A minor omission in the Spikemark 6 network device definition.

FIXED: An error in the Elation Obsidian Onyx network device definition.

FIXED: An error in the Yamaha DM7 network device definition.

FIXED: A bug that would sometimes cause a hang when typing a semicolon into a Light cue.

FIXED: A bug that prevented Start First And Enter Group cues from starting their second child cue when their first child cue was disarmed *and* set to skip if disarmed.

5.5.6 - October 9, 2025

ADDED: Paste Cue Properties now offers presets for just audio levels and just audio objects.

ADDED: Instrument definitions for Arkaos MediaMaster 6.

ADDED: Instrument definitions for the Cameo Zenit W300.

ADDED: Instrument definitions for the Chauvet DJ EVE E-50Z, EVE E-100Z, EVE E-160ZQ, EVE F-50Z, EVE F-160ZQ, EVE P-100 WW, EVE P-140 VW, EVE P-150 UV, EVE P-160 RGBW, Freedom Flex H4 IP, and Freedom Flex H9 IP.

ADDED: Instrument definitions for the Chauvet Professional Ovation F-915FC and F-415FC.

CHANGED: Audio objects within the same cue or map cannot be assigned the same name.

FIXED: A condition in which QLab could freeze when rendering audio through device audio effects.

FIXED: Audio effects on Mic cues will again save their settings properly.

FIXED: A crash that could occur when a lot of audio or video cues stop simultaneously.

FIXED: A bug where cart cues continued to act as if they were selected after switching to Show mode. This could result in surprising behavior such as the workspace GO hotkey triggering a cart cue that did not appear to be selected.

FIXED: Video and Camera cue inspectors now refresh correctly after audio levels are changed by incoming OSC commands.

FIXED: Changing partial screen route layouts will now refresh the live output correctly.

FIXED: Setting the dimensions of a route from its device will now honor the partial output of that device in the route.

FIXED: Several crashes that could occur when setting Network cue parameter values.

FIXED: A crash that could occur when opening a workspace that references a display that is missing when logging at log level 2 or greater.

FIXED: A crash that could occur when editing a stage name on macOS Tahoe.

FIXED: The layout of the GO and Audition buttons on macOS Tahoe.

5.5.5 - September 17, 2025

FIXED: A bug introduced in QLab 5.5.4 that could cause a crash when connecting via collaboration.

5.5.4 - September 16, 2025

ADDED: Instrument definitions for the American DJ UBL6H.

FIXED: Improvements when identifying and reconnecting video displays.

FIXED: A regression in 5.5 which could prevent collaborators from editing audio effects.

FIXED: When cutting and pasting a group cue in a collaborator workspace, child cues of the pasted group will now appear as they should on the primary machine.

FIXED: Moving a project folder no longer requires reselecting your audio map background images.

FIXED: Deleting the last cue object while a cue is playing audio will no longer halt audio from playing to map objects.

FIXED: Certain delay and reverb effects will now be reset correctly when stopping and restarting audio through those effects.

FIXED: Accept certain editing operations on audio effects that were previously ignored.

FIXED: Fading audio on a Group cue now works again if the group contains a Text cue.

FIXED: Looping playlist Group cues are no longer broken when disarmed.

FIXED: Restores v5.4.x behavior of Group cue OSC playhead commands `/next` and `/previous` and AppleScript `movePlayheadUp` and `movePlayheadDown` to fix a bug that broke sending these messages to a non-current cue list.

FIXED: Network cues set to 2D Fade are no longer broken when the fade path has only 1 point.

FIXED: Script cues that are disarmed no longer generate a non-breaking “AppleScript not compiled” warning when re-opening a workspace.

FIXED: The “Compile Script” button is now disabled in the Script inspector tab when a Script cue is disarmed, since only scripts of armed cues can be compiled.

FIXED: Make sure that a workspace settings tab is always selected.

FIXED: A crash when sending an OSC message to an Audio Output Patch or Audio Map that contains an invalid audio channel wildcard pattern.

FIXED: An bug that could cause the Q placeholder to flash briefly in the Geometry tab when changing crop values.

FIXED: An error in the Yamaha Rivage network device definition.

5.5.3 - August 20, 2025

FIXED: A performance issue introduced in v5.5 which reduced the number of simultaneous audio channels that QLab could process.

FIXED: Clicking in Time & Loops tab would sometimes make a very small selection rather than loading.

5.5.2 - August 18, 2025

FIXED: Timecode cues will now generate LTC output again.

FIXED: A bug that caused infinite loop slices to play only once after clicking in the audio waveform in the Time & Loops tab.

CHANGED: Improved the merge pathing algorithm for 2D fades to better handle certain edge cases.

5.5.1 - August 13, 2025

FIXED: A crash when viewing the Time & Loops tab on certain older versions of macOS.

5.5 - August 12, 2025

★ **ADDED:** **Object-based audio** lets you effortlessly move sounds around your system in a brand new way, routing, fading, and mixing in a fully flexible visual environment.

★ **ADDED:** Disarmed cues can optionally be skipped.

★ **ADDED:** OSC commands that include parameters for audio inputs and/or outputs now support OSC wildcard patterns. This allows you to edit multiple channels with a single message. A reply message is sent for each matching input/output channel. For example, `/cue/1/mute/[2-9] 1` will mute cue outputs 2 through 9 in cue “1”. `/cue/1/solo “*L” toggle` will toggle the solo button of all cue outputs with names ending in “L” in cue “1”. `/cue/1/level/*/*/- 1.0` will subtract 1 dB from all input, output, and crosspoint levels in cue “1”.

ADDED: You can now edit the parameters of audio effects on disarmed cues and disconnected audio devices.

ADDED: Native video inputs (webcams, etc.) now support format configuration for choosing things like resolution and frame rate.

ADDED: More information is displayed about video input formats for all video input types.

ADDED: The Levels and Objects inspector tabs show a “dog-ear” indicator if their tab would contribute audio when the cue is run and the main level is above -INF.

ADDED: Clicking on the waveform in the Time & Loops tab now loads to the correct time when looped slices are present, and can now load past infinitely looped slices.

ADDED: The canvas for geometry path fades will infer better default settings when parameters are changed or cues are retargeted, assuming no points have been added.

ADDED: Cue trigger logs (in the Logs tab of the Workspace Status Window) now include additional details about the source of the trigger.

ADDED: Support for logging all OSC output (not just replies) in the Logs tab of the Workspace Status Window.

ADDED: A keyboard shortcut, ⇧⌘**1**, to show the launcher window.

ADDED: An entry in keyboard controls, ⌥**T** by default, to cycle the target type of selected Fade and/or Reset cues.

ADDED: Menu items and keyboard shortcuts to move the selected cue(s) up or down in the list.

ADDED: A preference to prompt before running the workspace open cue.

ADDED: Playlist groups now display an indicator next to the child cue that will play next.

ADDED: Cue OSC command `/isNextInPlaylist` and Group cue OSC commands `/playlist/currentCue` and `/playlist/currentCueID`.

ADDED: Certain OSC commands now "rollover" when using +/- increment syntax, e.g. `/cue/{number}/continueMode/+ 1` will cycle from "Auto-follow" back to "Do not continue", etc.

ADDED: Support for sending the `{delta}` value of +/- increment syntax in the address, e.g. `/cue/{number}/continueMode/+/1`.

ADDED: The `/sliderLevel` OSC message now supports +/- increment syntax.

ADDED: Video, Camera, and Text cue OSC message `/videoEffects/live`.

ADDED: Text cue OSC `/text/format` and AppleScript `text format record` now support drop shadows with added attributes `shadowOffset`, `shadowBlurRadius`, and `shadowRgbaColor`.

ADDED: Timecode cue OSC commands `/outputType`, `/framerate`, `/ltcChannel`, `/startTime`, and `/endTime`.

ADDED: Timecode cue AppleScript properties `timecode start time` and `timecode end time` and timecode SMPTE format enumeration values `fps_23_976`, `fps_24_975`, `fps_29_97_non_drop`, and `fps_29_97_drop`.

ADDED: When handling a deprecated OSC message, the reply payload JSON now includes the key `"deprecated"` with a boolean "true" and the key `"warning"` with a string detailing the deprecation and its replacement.

ADDED: Instrument definitions for the American DJ Vizi Beam CMY.

ADDED: Instrument definitions for the BeamZ Illusion 1 and Illusion 1 Mk II.

ADDED: Instrument definitions for the Blizzard LB Par Hex and Snowblind.

ADDED: Instrument definitions for the Chauvet DJ Abyss 2, COLORband PiX-M USB, EVE P-130 RGB, Festoon 2, Freedom Gobo, Obsession, and Obsession HP.

ADDED: Instrument definitions for the Clay Paky Actoris ParLED, Actoris Profile FC, K-EYE K10 HCR, K-EYE K20 HCR, Klemantis AS line, Orkis Fresnel-Pebble M, Orkis Profile M, Tambora Batten, Tambora Flash, and Tambora Linear 100.

ADDED: Instrument definitions for the CLF Lighting LEDWash CW-WW, LEDWash RGBW, Poseidon Beam Ultra, Poseidon Mirror, Poseidon Wash XS, Poseidon Wash M, Poseidon Wash XL, Sera, Sera Batt, SoftLED 100 VW, and SoftLED 200 VW.

ADDED: Instrument definitions for the Eurolite LED Bar 6 and LED Bar 12.

ADDED: Instrument definitions for the Equinox Fusion 120 Zoom MkII, Micro Batt, SpectraPix Lithium Batten, and Waterwave XP.

ADDED: Instrument definitions for the GLP Agilio, Fusion EXO Beam 10, Fusion EXO Spot 30, Fusion EXO Wash 30, Fusion MBL 20, Fusion MBL 40, Fusion Stick FS10, Fusion Stick FS16 Z, Fusion Stick FS20, Fusion X-PAR 12Z, Impression FR10 Bar, Impression X5 Bar 1000, JDC Line 500, and JDC Line 1000.

ADDED: Instrument definitions for the Martin MAC 300.

ADDED: Instrument definitions for the Robe CUETE, ColorStrobe, PAINTE Spot, iBOLT, iT12 Profile, iT12 Fresnel, iT12 PC, T15 Fresnel, T15 PC, T32 CYC, T32 CYC Slim, and TetraX.

ADDED: Instrument definitions for the Showtec Compact Par 7/14 Q4 and Dreamwave.

ADDED: Instrument definitions for the Vari*Lite Fresnel FC, Fresnel TCW, Fresnel TWW, IP Image Projecting Profile FC, IP Image Projecting Profile CW, Profile Light Engine FC, Profile Light Engine TCW, Profile Light Engine TWW, VL600 Acclaim Cyc, VL600 Acclaim Fresnel, VL600 Acclaim Fresnel Plus,

VL600 Acclaim Zoomspot, VL1600 Profile, VL3600 Profile IP, and VL3600 LT Profile IP.

ADDED: A network device definition for the Adamson Fletcher Machine.

ADDED: Network device definitions for the Amadeus Acoustics Core and 3D Panner.

ADDED: A network device definition for Audioström LiveProfessor.

ADDED: A network device definition for Creative Connors Spikemark 6.

ADDED: A network device definition for D::Light.

ADDED: A network device definition for the Elation Obsidian Onyx family.

ADDED: A network device definition for the Lawo mc^2 family of consoles.

ADDED: A network device definition for Millumin 4.

ADDED: A network device definition for PJLink.

CHANGED: Copying and pasting cues into an earlier major version of QLab is no longer supported, since major releases are not guaranteed to be backward-compatible. That is to say, copying from 5.5.x and pasting into 5.4.x or earlier is not allowed, but copying from 5.5.3 and pasting into 5.5.1 will be allowed.

CHANGED: When dragging files into a workspace, new cues are now ordered alphabetically according to the way their names sort in the Finder. For example, "10.wav" now imports after "9.wav" instead of between "1.wav" and "2.wav".

CHANGED: Improvements to the performance of audio waveform drawing in the Time & Loops tab of the inspector.

CHANGED: The load position of cues as seen in the Timeline and Time & Loops tabs of the inspector now updates live when you drag the "Load to time" slider.

CHANGED: Non-breaking audio patch warnings (in the Warnings tab of the Workspace Status window) have been improved.

CHANGED: The Workspace Status window now remembers the selected tab and the checkboxes selected in the Logs tab after a workspace is closed and reopened.

CHANGED: When importing a Go Button show, cues and hits that are disabled are imported as disarmed with *Skip if disarmed* enabled.

CHANGED: Typing any value at or below the workspace minimum volume or any non-numeric value into a level control in a Fade cue will "activate" that level at -INF. This makes it easier to edit Fade cues using only the keyboard.

CHANGED: The timeout duration used to filter out duplicate OSC messages from a client is now configurable in QLab Preferences. To disable this filter, you can now set the value to 0.0 seconds.

CHANGED: The method used to create an "undo" step has been improved when receiving certain OSC commands that are likely to be used in continuous changes, e.g. audio levels and object positions, video geometry, etc.

CHANGED: To increase compatibility, more OSC commands that expect numbers will now interpret string arguments as floats or integers. Strings are converted to numbers using the QLab Mac's localization.

CHANGED: OSC update messages now use the OSC-friendly string "__root__" as the ID of the root cue list.

CHANGED: OSC messages that end with `/live` now reply with an "error" status instead of "denied" if the OSC command method does not have a live version.

CHANGED: Non-numeric string arguments passed to OSC +/- relative setters now return an error instead of setting a 0 value.

CHANGED: The Cue List OSC command `/timecodeSMPTEFormat` now accepts only the values 0 - 3, matching the formats that can be set in the Timecode tab of the inspector.

CHANGED: OSC commands `/updates` and `/disconnect` (which became QLab commands in v5.0) now return an error if sent with the legacy `/workspace/{id}` prefix.

CHANGED: Improvements to using 16-bit lighting parameter controls. Holding the ⇧ (shift) key while dragging on a lighting control will now always make fine adjustments, and dragging without any keys held down will always made coarse adjustments.

CHANGED: The labels for audio input channels in the Levels tab of the inspector have been moved to the right in order to keep the first column of the matrix mixer aligned.

CHANGED: The Network cue inspector now supports batch editing cues with different network patches, as long as those patches all have the same type.

CHANGED: In Workspace Settings → Network, network device definitions are now displayed in categories within the Type menu in order to make it easier to look through them.

CHANGED: Updates to the d&b DS100 network device definition.

CHANGED: Updates to the FLUX SPAT Revolution network device definition.

CHANGED: Updates to the Yamaha DM3 and DM7 network device definitions.

CHANGED: The Levels tab for a Fade cue template now shows the maximum number of cue outputs.

FIXED: Changing the configuration of an audio output patch will no longer halt playback of cues that use that patch.

FIXED: A regression in 5.4 that prevented viewing the entire integrated fade curve in very long cues and/or when zoomed in very far.

FIXED: Video stage mask and Network cue 2D fade images are now copied to the workspace project folder immediately when selected, not just when saving the workspace.

FIXED: Improved display matching heuristics for native display outputs.

FIXED: A bug that could cause QLab OSC commands to be denied access when multiple workspaces are open. This bug could result in QLab Remote initially connecting but failing to receive playhead updates due to QLab incorrectly denying the `/updates` message sent by QLab Remote.

FIXED: Group cue post-waits no longer appear to continue running after the Group is stopped by the *Fade & stop* action of another cue.

FIXED: The cue contextual menu item "Jump to cue's target" now works correctly when the target cue is in a different cue list.

FIXED: A bug that prevented using option-drag to duplicate an item in settings, such as a patch or video stage, when the list contained only one item.

FIXED: A bug that could prevent OSC `/update` messages from being sent to connected clients.

FIXED: OSC clients no longer receive multiple playback position `/update` messages after each GO.

FIXED: A bug that allowed OSC and AppleScript to move cues to invalid parent cues.

FIXED: Several issues with the OSC `/next` and `/previous` commands and the AppleScript `movePlayheadUp` and `movePlayheadDown` commands. They now behave the same way as the corresponding controls in the workspace.

FIXED: The duration of a Network cue with no duration can now be edited in the cue list.

FIXED: Setting a duration for a Network cue set to 2D Fade no longer requires having a non-empty fade path.

FIXED: A bug that prevented typing in the Network cue inspector while another workspace was running on the same machine.

FIXED: A bug that prevented being able to clear a value in certain Network cue inspector text fields.

FIXED: The Network cue template inspector now refreshes correctly when changing parameters of a patch using a device description.

FIXED: Setting an audio crosspoint or input level using a `/level` OSC message now properly constrains the value to a maximum volume of 0 dB.

FIXED: A bug that caused Network cues to fail if an OSC query returned a payload containing quotation marks.

FIXED: Passing a non-numeric string argument to an OSC message that sets audio levels now correctly sets the level to -INF instead of 0 dB.

FIXED: Changing the SMPTE timecode format of a cue list using the `/timecodeSMPTEFormat` OSC message now performs the correct conversion between formats, equivalent to selecting the *Keep old timecodes* option from the **Recalculate timecodes?** prompt when changing the format in the Timecode tab of the inspector.

FIXED: Improvements to the `/setLight` OSC message's handling of numeric *setting* arguments.

FIXED: The `/setLight`, `/replaceLightCommand`, and `/removeLightCommandsMatching` OSC messages now return an error when the command fails.

FIXED: If prompted to save while closing a workspace, the workspace will now close after the save is complete.

FIXED: A bug that prevented manually setting the input format on DeckLink devices (though it's really best to use Auto to detect the incoming format unless you know for sure of a reason not to.)

FIXED: Cues created by dragging from the cue bar or toolbox did not auto-number (when auto-numbering is enabled.)

FIXED: A cue that had been assigned a MIDI trigger message that uses two bytes will now correctly clear the second byte when it is re-assigned a new trigger message that uses only one byte.

FIXED: An error in the Behringer/Midas X32/M32 network device definition.

FIXED: An error in the Behringer Wing network device definition.

FIXED: Several errors in the Blackmagic Videohub Protocol network device definition.

FIXED: An error in the d&b DS100 network device definition.

FIXED: Errors in the High End Hog 4 network device definition.

5.4.11 - June 10, 2025

FIXED: Mic and Camera cue Inspector I/O tabs now format the audio input sample rate correctly in remote workspaces.

FIXED: An issue that could cause QLab to crash when connecting to an NDI output on macOS Sequoia.

FIXED: An issue that could cause QLab to very slowly leak memory when using NDI outputs.

FIXED: An issue that could cause QLab to crash when editing the Light Patch.

FIXED: A crash that could occur when viewing the audio waveform in certain circumstances.

5.4.10 - April 29, 2025

FIXED: An issue with displaying the disclosure triangle on a cue list on macOS Sequoia.

FIXED: An issue with relative Z rotation fade cue display.

FIXED: A rare crash that could occur if a fade cue is edited when running.

FIXED: An issue where video effect labels could have their last character cut off in the inspector.

5.4.9 - February 6, 2025

ADDED: OSC `/cropTop`, `/cropBottom`, `/cropLeft`, and `/cropRight` now support +/- relative setter syntax.

CHANGED: Minor tweaks to help with Sequoia network difficulties.

FIXED: Previewed cues are once again shown in the list of active cues.

FIXED: The keyboard shortcut that toggles the workspace Settings window.

FIXED: A rare crash when reading LTC timecode.

FIXED: A bug that caused still image and text cues to display past their duration if they had a post-wait.

FIXED: A layout bug in the save panel when saving to very long paths.

5.4.8 - December 12, 2024

FIXED: A regression (since QLab 5.4.5) which caused dragging video grids to become much slower. Dragging video grids should now be snappy again.

FIXED: Active cues are now synced more reliably to collaborators.

FIXED: The active cue controls are no longer enabled for collaborators who do not have “Control” permission.

FIXED: The duration of a Text cue which was created by a collaborator can now be edited.

FIXED: Collaborators can now select custom parameters for geometry fade paths.

FIXED: Collaborators can now change the loop count for the last slice of a Video cue. Changes to this slice are also now undoable.

FIXED: Video cues with audio-related warnings now show the correct cue icon in the Warnings window and the “Inspect Cue” button now works correctly.

FIXED: Changing the bounds of a video stage region using OSC is now undo-able.

FIXED: A bug that prevented editing other Fade inspector tabs after changing a Fade cue with a geometry 2D path to a different type of cue target.

FIXED: A bug when using OSC or AppleScript to set a patch target of a new Fade or Reset cue.

FIXED: A bug that caused patch targets of Reset cues to appear as “missing” when reopening a workspace.

FIXED: A bug that could cause audio output patch slider levels to fail to update when muting or soloing shortly after changing to a new audio output patch device.

FIXED: An error in the QLab network device definition.

FIXED: An error in the L-ISA network device definition.

5.4.7 - October 28, 2024

ADDED: Instrument definitions for the BeamZ 440Z.

ADDED: Instrument definitions for the Chauvet DJ EZBeam Q3 ILS, EZLink Par Q6BT ILS, Freedom Par Q9, and Freedom Par Tri-6.

ADDED: Instrument definitions for the Chauvet Professional Maverick Silens 1X Profile, Maverick Silens 2X Profile, Maverick Storm 3 BeamWash (single-pixel modes), Maverick Storm 4 SoloWash, Rogue R1E Spot, Rogue R2E Spot, Rogue Outcast 1 BeamWash, Rogue Outcast 1M Beam, STRIKE Bolt 1C, and onAir Flex Drive 2.

ADDED: Instrument definitions for the High End Systems Hyperstar and Ministar.

ADDED: Instrument definitions for the Martin MAC 101, MAC 101 CT, MAC 101 CLD, and MAC 101 WRM.

ADDED: Instrument definitions for the Rockville Stage Panel 864.

ADDED: Instrument definitions for the ShowPro LEDFRE117.

ADDED: Instrument definitions for the Stairville Led Bar 240/8.

ADDED: Instrument definitions for the zactrack system.

ADDED: A network device definitions for the OBSBot Center, Tail Air, and Talent.

ADDED: A network device definition for the zactrack system.

CHANGED: The ADM-OSC network device definition has been updated to support version 1 of the specification.

FIXED: A bug that prevented muting or soloing audio output patch channels in certain cases.

FIXED: A bug that prevented quitting after saving on certain Macs with automatic graphics switching.

FIXED: A crash that could occur when quitting QLab if using certain audio plugins.

FIXED: An error in the DiGiCo SD series network device description.

FIXED: An bug causing stages to display grids at the wrong resolution to Blackmagic outputs in some cases.

FIXED: An bug that could cause problems playing Mic cues to Blackmagic outputs.

5.4.6 - September 12, 2024

FIXED: A regression in 5.4.5 that could, under rare conditions, prevent QLab from saving workspaces.

FIXED: A regression in 5.4.2 that caused the broken state of Network cues on a remote workspace to fail to update.

FIXED: Timecode cues sending LTC are no longer broken if one or more channels in the cue's audio patch are muted or soloed.

FIXED: Several errors in the network device definitions for the Behringer X-Air.

5.4.5 - September 11, 2024

ADDED: We've brought back a progress bar for opening and saving workspaces. The built-in macOS progress bar stopped working, because sometimes Apple just doesn't know when to leave things well enough alone. But we've written a new one to replace it.

FIXED: A bug that caused Camera cues to stop under certain conditions.

FIXED: A bug that prevented live update of all ganged audio levels.

FIXED: A bug that caused Text cues that were loaded to time to start from the beginning.

FIXED: A bug that caused the Integrated Fade mouseover label to show the incorrect decibel value on remote workspaces.

FIXED: A bug that prevented being able to cut or copy cues in a Timeline group using the contextual menu.

FIXED: Clicking in the Timeline group time ruler to seek now preserves the auditioning state of any running cues in the group.

FIXED: A crash that could occur when viewing a Timeline group that contains Fade cues.

FIXED: A bug that could cause the audio waveform image in Audio and Video cues to fail to update.

FIXED: A bug that prevented Fade cues that were edited numerically on a remote collaborator from live previewing.

FIXED: A bug that caused resized columns of additional cue list windows to reset to default widths frequently.

FIXED: A regression that broke moving focus between Network cue inspector text fields using the tab key in certain cases.

FIXED: The constraints on the playback rate parameter of Audio, Video, and Fade cues are now enforced by the cues, not just by the UI. This prevents invalid values when setting the rate parameter via OSC or scripting.

FIXED: A condition during collaboration that could lead to a playhead update infinite loop when a remote workspace has the playhead locked to selection.

FIXED: A bug that could cause an incorrect "License Required" alert to appear at unexpected times.

FIXED: Several errors in the network device definitions for the Yamaha DM3, DM7, and Rivage.

5.4.4 - July 16, 2024

FIXED: When opening workspaces saved prior to QLab version 5.3, the end times of audio and video cues will be preserved correctly.

FIXED: A regression in video rendering that could cause a green flash when starting Blackmagic outputs running in Y'CbCr pixel format in some circumstances.

FIXED: A bug that could prevent being able to set a Fade cue that targets an audio patch to Absolute fade mode.

5.4.3 - July 11, 2024

FIXED: A regression in the Network Cue Inspector in 5.4.2 that prevented being able to set an initial value for certain message parameters.

FIXED: A crash that could occur when crossfading Audio cues in a Playlist group.

5.4.2 - July 8, 2024

ADDED: When opening older workspaces, QLab will now notify you if saving them would require using a newer version.

FIXED: A crash that could occur when clicking on the "Time & Loops" tab in Audio or Video cue template settings.

FIXED: Network cue inspector parameters now refresh correctly when dragging a slider and after undoing changes.

FIXED: A rounding bug when using increment/decrement syntax with OSC command `/fadeAndStopOthersTime`.

FIXED: A bug related to Update Originating Cue when a collated Light cue is currently paused or running.

FIXED: A bug related to zoom levels in the Text Cue editor.

FIXED: Assorted other small bugs.

5.4.1 - June 20, 2024

ADDED: A network device definition for Glypheo.

CHANGED: Workspaces saved in v5.4+ can no longer be opened in QLab 5.3.8 or earlier because of data structure changes that are not backward-compatible.

FIXED: A bug that caused QLab to sometimes crash when using the light dashboard.

FIXED: Displaying the integrated fade curve on the audio waveform will no longer hide the waveform on some macOS 12 systems.

FIXED: Deleting a cue will no longer sometimes send the selection to the top of the cue list.

FIXED: Instantaneous relative fades will now work again.

FIXED: A bug that caused cues in a Playlist group set to auto-shuffle to continue to display in the shuffled order after changing the group mode.

FIXED: A bug that caused playing audio cues to temporarily pause after removing all cue output effects.

FIXED: A bug that caused the Text cue editor to behave erratically at small zoom levels.

FIXED: A drawing bug when using pinned cues in the Timeline.

FIXED: A bug that caused undesirable behavior when multiple instances of the same audio effect were added to a single cue.

FIXED: OSC reply messages customized with the `/replyFormat` command no longer have a double leading slash when the format starts with the `#address#` token.

FIXED: Assorted other small bugs.

5.4 - June 11, 2024

★ **ADDED:** Mute and Solo buttons for cue outputs (found in the Trim tab of the inspector) and audio patch outputs (found in the Patch Routing tab of the audio output patch editor.)

★ **ADDED:** You can now edit audio effects from collaboration remotes. The audio effect must be installed on the remote computer as well as the primary in order to add it or edit it remotely.

★ **ADDED:** Fade cues now support 2D fading of video geometry.

★ **ADDED:** Several new tools for editing 1D fade curves and 2D fade paths, including arrow key navigation, multiple point selection, copy/paste, scaling, aspect lock while dragging, rotation and reverse tools, and a circle drawing tool.

★ **ADDED:** The up-fade and down-fade curves in the Curve tab of Fade cues can now be locked together as mirror images. When locked, changes to either curve affects both curves symmetrically.

ADDED: You can now set a finite duration for Mic cues and Camera cues.

ADDED: A submenu to allow more options when opening a target file in an external editor. You can find this submenu by right-clicking (or control-clicking, or two-finger-clicking) in the waveform view of the Time & Loops tab of the inspector.

ADDED: Support for copying audio effects between cue outputs and between device outputs.

ADDED: Support for batch-editing MIDI File cues.

ADDED: Support for moving cues using the AppleScript “move” command.

ADDED: The “Select cue…” tool now offers an option to quickly select the cue currently at the playhead.

ADDED: The IP address and port number of NDI sources are now shown in the Video Input tab of Workspace Settings → Video.

ADDED: Instrument definitions for the BriTeQ BT-Orbit, BT-Profile HD, BT-Profile WW, BT-Shark, BT-Theatre 100MZ, BT-Theatre 150EZ Mk2, BT-Theatre 250EZ Mk2, BT-Theatre 200TW, BT-Theatre 400TW, BT-Theatre 261FC, BT-Theatre 50WW, BT-Theatre 60FCL, BT-Theatre HD2, BT-Theatre Venuelite4, BT-Tracker, BT-TVPanel TW, BTI-Blizzard BSW2, BTI-Cyclo, BTX-Blizzard Wash1, BTX-Saturn, BTX-Supreme Hybrid, BTX-Titan, Beam Fury-1, Pro Beamer Zoom Indoor Mk3, and Stage Beamer.

ADDED: Instrument definitions for the Chauvet DJ 4Bar Flex Q ILS, 4Bar LTBT ILS, COLORband Q4 IP, Cosmos HP, Freedom H1, GigBar Move ILS, Hurricane Haze 2D, Intimidator Barrel 305 IRC, Intimidator Free Spot 60 ILS, Rotosphere HP, Shocker Panel FX, SlimBANK T18 USB, SlimPAR H6 USB, and SlimPAR T12BT.

ADDED: Instrument definitions for the Chauvet Professional Color STRIKE M, Maverick Force 2 BeamWash, Maverick Force 2 SoloWash, Maverick Force 3 Profile, Maverick Storm 1 Beam, Maverick Storm 1 Hybrid, onAir Panel 3 IP, Ovation E-2 FC, Ovation E-4WW IP, Ovation Reve F-3 IP, and Rogue R3E Spot.

ADDED: Instrument definitions for the Clay Paky Arolla Aqua, Arolla Profile MP, Axcor Profile 600, Midi B FX, Panify, Sharpy X Spot, and Skylos.

ADDED: Instrument definitions for the Epson EB-PU2100 series and EB-PU2200 series.

ADDED: Instrument definitions for the High End Systems Studio Color 575.

ADDED: Instrument definitions for the Martin ELP PAR, ERA 700 Performance IP, MAC One, MAC Viper XIP, RUSH Blinder 1 WW, RUSH CS, and VDO Atomic Bold.

ADDED: Network device descriptions for Behringer WING, Bitfocus Companion version 3.2 and newer, Christie HS series projectors, Flamingo Software Atlas, GarageCube MadMapper 5, Innovate Audio panLab Console, MA Lighting grandMA3, TheatreMix 3, and Yamaha DM7 series.

CHANGED: Carts now support up to 15 rows and 15 columns.

CHANGED: Improved synchronization between audio and video when chasing timecode.

CHANGED: Improved performance for OSC messages referencing cues by `cue_id`.

CHANGED: The minimum height of the inspector has been slightly increased.

CHANGED: QLab’s numeric rounding behavior has been changed from “round half to even” to the more conventional “round half away from zero”.

CHANGED: Network cues will now break if an OSC message contains arguments with incorrect spacing around quotation marks, such as `/my/command “arg1”arg2`.

CHANGED: The “liveFadePreview” AppleScript and OSC messages are now properties of a workspace, rather than of QLab as a whole.

CHANGED: The existing network device description for Bitfocus Companion has been renamed to reflect the fact that it pertains only to Companion version 3.1 and earlier.

CHANGED: The network device description for Vor has been updated to reflect new OSC control options in Vor 1.6 and later.

CHANGED: Slashes and colons are no longer allowed in Light Definition names, as they caused trouble.

CHANGED: Audio Units now use a generic control interface when their custom interface fails to initialize.

FIXED: An issue that led to poor initial synchronization between audio and video when using Blackmagic outputs, possibly creating a brief pitch shift.

FIXED: A bug which caused unattended long-running workspaces using Script cues to stop functioning properly, with symptoms such as: network cues failing to run, script cues failing to run, and document autosaving failing.

FIXED: A bug that could cause the Settings window to remain open after closing a workspace.

FIXED: A bug that prevented being able to swap cart positions when dragging cart cues of certain types.

FIXED: Fade type “Parametric Curve” intensity text fields now support localized number formatting.

FIXED: A bug introduced in 5.2 that resulted in incorrect cue numbers or cue IDs in the reply payload to an OSC message that uses a wildcard pattern to match multiple cues.

FIXED: A bug that prevented addressing cue output 128 by name in OSC messages.

FIXED: Collaborators can now toggle the live fade preview setting of a workspace on the primary Mac.

FIXED: The “Accept Collaborator” dialog box will no longer block the workspace from processing certain incoming messages (such as incoming OSC or timecode.)

FIXED: Several bugs that could occur when deleting cues using AppleScript.

FIXED: Text cues will again be in “custom geometry” mode by default, unless using a computer without a video license, in which case they will still be in full stage mode.

FIXED: A bug that could cause a Text cue to flicker if it was updated while it was running.

FIXED: *Update Originating* after starting a Light cue which is set to collate will now correctly update levels in the originating cues, not the collating cue.

FIXED: A bug that could cause Network cues whose patch contains a combination of local and external destinations to incorrectly display the “overridden” status indicator.

FIXED: A bug that could cause Network cues with very short durations to fail to auto-continue.

FIXED: A bug which would improperly set a value of 0 when dragging in the translation geometry fields of Fade cues which are were untargeted or which targeted Group cues.

FIXED: A bug which prevented Fade cues which targeted Groups from auditioning when children of the Group were already individually auditioning.

FIXED: A bug that could cause the OSC command `/fadeEntries` to return an erroneous error reply after setting new entries.

FIXED: OSC command `/customString` now works when sending to a Network cue that is set to fade.

FIXED: An error in the instrument definition for the Chauvet DJ GigBar 2.

FIXED: An error in the network device description for the Yamaha Rivage.

5.3.8 - May 8, 2024

CHANGED: We’ve (temporarily, we hope) disabled the progress bar that QLab displays when opening workspaces or building lots of new cues. Instead, for large workspaces or lots of cues, you might see a beachball cursor instead. Rest assured QLab is still doing the same work as before, it is just not showing you a progress bar about it. Why change this, you might ask? Because Apple, in their infinite wisdom, changed progress bars from updating “when you told them to update” and instead update them “maybe, eventually, when Apple feels like it”. The workaround we’d used to force progress bars to actually, you know, progress, was causing problems, so we’ve disabled it until we can find a different approach.

FIXED: A crash that could occur when editing a Text Cue with macOS Sonoma’s inline predictive text enabled.

FIXED: A condition in which fading in audio from silence using an integrated fade could sometimes allow the audio to start very briefly at a volume that was not silent.

FIXED: A regression introduced in 5.3.4 that could prevent Timecode cues from running in some circumstances.

FIXED: Opening workspaces with lots of Camera cues is now faster.

FIXED: An issue that prevented Blackmagic outputs from applying changed settings under some circumstances.

FIXED: An issue that caused some machines’ identifier to be inconsistent, thus interfering with QLab’s license status.

5.3.7 - April 2, 2024

FIXED: Still image Video cues will reload their image if the file is edited in another program.

FIXED: A bug that could cause a crash when connecting to a remote workspace which has a Blackmagic device.

FIXED: A bug that could cause a crash when auditioning while an effect editor for an Audio Cue is open.

FIXED: A bug that would cause videos sent to NDI outputs to run very fast if no NDI client was connected.

FIXED: A bug that would cause QLab to crash when changing the audio output to NDI while the cue was running.

FIXED: A bug that would cause NDI audio to fail if the first cue sending to that NDI output used a different audio output.

FIXED: Disconnecting during collaboration will no longer cause the primary workspace to stop.

FIXED: The fade-and-stop trigger option will now properly prevent Group cues from continuing on to a subsequent cue if they should have been stopped.

FIXED: When triggering cues from timecode, the pre-wait of a cue will now be included when calculating if a given cue should be running at the given timecode.

FIXED: Light subcontrollers that have been assigned MIDI triggers will no longer lose their trigger when re-opening a workspace.

FIXED: Changing the “lock playhead to selection” option when collaborating should no longer lead to potential infinite loops where different machines disagree about what the playhead should be.

FIXED: A bug introduced in 5.3.4 that could cause Network cues that contain an OSC query to break with an incorrect “not a legal OSC address” error.

FIXED: A few small errors in the Yamaha Rivage network device description.

FIXED: A memory leak.

FIXED: Assorted other small bugs.

5.3.6 - March 2, 2024

FIXED: A regression in 5.3.4 that caused QLab to sometimes crash when running Timecode cues.

5.3.5 - February 28, 2024

FIXED: A regression in 5.3.4 that caused QLab to crash when sending video to an NDI output that had no audio channels.

5.3.4 - February 28, 2024

CHANGED: Blackmagic outputs will now present an error when a configuration is selected that the device reports to be unsupported, such as selecting a high frame rate output with a BGRA pixel format on a device that only supports high frame rates with a Y’CbCr pixel format.

FIXED: NDI outputs which include audio channels will now run more smoothly, and in some cases with significantly improved performance.

FIXED: The size of the masthead will now be restored when opening a workspace.

FIXED: The shutter video effect now works on cues of very large dimensions.

FIXED: Undo/redo now work correctly on the shutter video effect.

FIXED: A bug that could prevent cart buttons from displaying an updated cue number or name.

FIXED: A bug that could cause undo operations to “lose” cues when they are moved in playlist groups.

FIXED: A bug that could cause QLab to freeze when working with Timecode cues of a finite duration.

FIXED: A regression in 5.3 that prevented Network cues from starting if the message contains an OSC query that initially returns an invalid payload.

FIXED: Network cues that send an OSC message containing an OSC query are now broken if the base address of the message is not a valid OSC message.

FIXED: A bug that prevented a show control broadcast “start” message from being sent when a paused cue resumes playing.

FIXED: OSC command `/duckLevel` now supports increment/decrement syntax.

FIXED: Debug logging no longer reports “OSC event message not sent” errors for network clients that are not listening for show control broadcast events.

FIXED: The default port of the Behringer/Midas X-Air series network device description is now 10024 as specified by the manufacturer.

FIXED: Assorted other small bugs.

5.3.3 - December 18, 2023

CHANGED: Improvements to the network device definition for Lightkey.

FIXED: An issue where fading out a cue that is assigned to a patch which has cue output effects could silence all audio going to that patch.

FIXED: A potential playback issue when loading Timeline Groups that contain Video cues.

FIXED: A bug that could cause flashes in video playback when changing the live text of a Text cue.

FIXED: A bug that could cause Camera cues to fail to render.

FIXED: Importing v4 Fade cues with single axis Z-rotations will no longer invert the rotation.

FIXED: Importing v4 Fade cues now supports video effect parameters.

FIXED: A small issue in the d&b DS100 network device description.

FIXED: Assorted other small bugs.

5.3.2 - November 9, 2023

FIXED: The new video shutter effect will no longer crash when running on macOS Big Sur.

FIXED: Timecode cues sending LTC timecode to an audio output will no longer crash if the timecode channel is larger than the available audio patch channels.

FIXED: A bug that caused invalid output when using VISCA over IP network patches.

FIXED: A regression in 5.3 that broke increment/decrement syntax for some OSC commands.

FIXED: A small error in the ADM network device definition.

5.3.1 - November 9, 2023

FIXED: An issue where cue start times could be lost on files that are loaded from cloud storage.

FIXED: A crash when using grayscale colors in the drop shadow video effect.

FIXED: The “live” version of Audio output patch OSC command `/levels`.

FIXED: Handling of cue OSC messages that specify the cue number or unique ID as an argument.

5.3 - November 7, 2023

★ **ADDED:** **A larger, flexible audio matrix.** QLab now supports up to 128 cue outputs and 128 device outputs. You can now define exactly how many cue outputs each audio output patch uses, between 1 and 128. Patches which use fewer cue outputs will use less processing power.

★ **ADDED:** Improved audio playback performance.

★ **ADDED:** The Shutter video effect now supports both feathering and elliptical shutters.

★ **ADDED:** New video effect: drop shadow.

★ **ADDED:** The Workspace Status window has a new tab, Video Metrics, which displays live frame rate and render time information for each video stage and video output.

★ **ADDED:** **Show control broadcast.** QLab can now optionally send show control messages via OSC and MSC which correspond to basic show events such as GO, panic, reset, and cues starting and stopping. OSC messages can be requested by a client using the new `/listen` OSC message which has options for specifying the scope of interest. MSC messages can be sent using MIDI patches.

ADDED: The waveform view in the Time & Loops tab of the inspector can now optionally display a specific channel of audio, rather than the sum of all channels. You can adjust which channel is displayed by using the contextual menu (right-click or option-click) in the waveform view.

ADDED: The Hardware tab in QLab Preferences now displays more information about audio devices.

ADDED: Cues playing to NDI video outputs can now use non-NDI audio outputs.

ADDED: You now have an option to disable edge blending on video grids.

ADDED: Support for RGB-formatted input using Blackmagic Designs devices.

ADDED: Support for the new DMXKing MAX devices for USB DMX communication. These include the ultraDMX MAX, the eDMX1 MAX, the eDMX2 MAX, and the eDMX4 MAX.

ADDED: Instrument definitions for the Beamz BAC503, BAC504, BAC509, and Panther 25.

ADDED: Instrument definition for the Blizzard PixelStorm 240.

ADDED: Instrument definition for the Chauvet DJ 4Bar Quad, Cubix 2.0, EZPar 64 RGBA, Mini Kinta, and Wash FX Hex.

ADDED: Instrument definition for CrowdGlow.

ADDED: Instrument definition for the Elation ACL 360 BAR and ACL 360i.

ADDED: Instrument definition for the Eurolite LED PARty TCL Spot, LED PFE-100 RGBW Profile Spot, LED PLL-480 QCL Panel, LED STP-14 Sunbar, LED THA-100F (MK1, MK2, and MK3), and Stage Panel 16 HCL LED.

ADDED: Instrument definition for the Rockville Rock Par 50, Rock Spot 260w, RockOn-7, RockPAR TRI, Rockstrip 252, and Stage Matrix 36 RGB.

ADDED: Instrument definition for the Stairville CLB2.4 Compact LED Par System, Octagon Theater CW WW 36x1W, xBrick Quad 16x8W.

ADDED: You can now lock the selection to the playhead on collaboration remotes. This allows remote collaborators to operate the workspace as if they were sitting at the primary computer.

ADDED: Network cues set to 1D fade and MIDI cues set to fade can now use the linear fade curve type.

ADDED: Network device descriptions for Dan Murfin ShowPlay, DiGiCo SD series consoles, Lightkey 4, VISCA-over-IP camera control, and Yamaha DM3.

ADDED: QLab will now display a warning message if a workspace open cue or close cue has been set, but that cue does not exist.

ADDED: The Timecode Status window now includes resizable, pop-out windows to display incoming and outgoing timecode streams.

ADDED: Paste Cue Properties and batch edit support for Network cues set to 1D fade have been expanded.

ADDED: Paste Cue Properties of fade curve shape now includes the curve's audio domain where applicable.

ADDED: Paste Cue Properties now supports the crop parameter of Video, Camera, and Text cues.

ADDED: Support for batch editing in the Fade cue's geometry tab.

ADDED: Network cue OSC commands `/message` and `/messageError`.

ADDED: OSC command `/udpReplyPort` for UDP clients to optionally request an alternate port for receiving reply, update, and event messages.

CHANGED: NDI has been updated to version 5.6.0.

CHANGED: The default size for a new NDI output is now 1920 × 1080 instead of 1280 × 720.

CHANGED: NDI and Syphon outputs now remain running once activated, even if the workspace is panicked.

CHANGED: Video grids have been updated to provide more information and visual indicators.

CHANGED: Enabling or disabling a video effect no longer causes the view to scroll to the top.

CHANGED: Light Dashboard can now be used in full screen mode.

CHANGED: The currently-playing cue in a Playlist group no longer restarts when a playlist with second trigger action *do nothing* is re-triggered.

CHANGED: Adding a Load cue to a workspace will now auto-populate that cue with the load time of the selected cue, if any.

CHANGED: Improved error logging of invalid OSC queries.

FIXED: Audio discontinuities when editing audio levels or trim on a running cue.

FIXED: Audio files with incorrect file extensions can now be opened for playback in more cases.

FIXED: A bug that could cause Audio cues to show a length of `0` if the cue's file target is replaced while QLab was not running.

FIXED: Clicking around the audio waveform view of a cue that is playing to the audition audio patch no longer changes output back to the non-audition patch.

FIXED: The audio waveform view did not update when the currently selected track changed in a Video cue with multiple audio tracks.

FIXED: Audio effects on device outputs are now correctly applied in the order that they appear.

FIXED: A bug that could sometimes cause Video cues to continue to count down their post-wait after being panicked.

FIXED: A bug that caused Camera cues using NDI with audio output to immediately stop.

FIXED: A bug that caused Syphon and Blackmagic outputs to double the alpha factor when sending to a receiver which honors alpha information.

FIXED: QLab will now attempt to preserve manually-set start and end times in a cue when the cue's file target is replaced.

FIXED: Auxiliary windows hidden in show mode took up space in Mission Control.

FIXED: A Group cue set to auto-load will now load its grandchildren when it loads itself.

FIXED: Improvements to keyboard navigation and focus when working in the Settings tab of the Network cue inspector.

FIXED: The MIDI cue and Network cue inspector **Send** buttons and the audio waveform **Preview** button all now correctly audition preview when the workspace has *Always Audition* enabled.

FIXED: Outgoing OSC `/update` and `/reply` messages now correctly respect the network output override.

FIXED: OSC clients will no longer continue to receive `/update` messages if the passcode they used to connect to QLab has its *View* permission removed.

FIXED: OSC `/rate` once again supports +/- relative setter syntax.

FIXED: After fixing a broken cue, the list, cart, and/or Group which contains that cue will now immediately refresh its own status.

5.2.4 - October 3, 2023

FIXED: In macOS Sonoma, you can once again close an unsaved workspace without saving it.

FIXED: A regression in 5.2.3 that kept focus in the load-to-time and search text fields after pressing return; it now goes back to the cue list.

FIXED: A regression in 5.2.2 that broke the ability to enter cue input names or gang names.

FIXED: A crash that could happen when importing workspace settings.

FIXED: Assorted other small bugs.

5.2.3 - September 6, 2023

FIXED: A bug that prevented still image Video cues in a Playlist group from continuing if the cue had previously been set to "Infinite Loop".

FIXED: A bug that caused QLab to crash when the Warping tab of the Stage Editor is selected.

5.2.2 - September 5, 2023

FIXED: Issues related to managing the input focus when using audio matrix text fields. Focus is now returned to the cue list when pressing enter, or when selecting a different cue to inspect.

FIXED: An exception that could occur in the Network cue Inspector when an X/Y parameter changes to a checkbox.

FIXED: A bug that allowed mesh-warped regions to become split apart, and caused further control point moves to fail to be applied.

FIXED: A bug that incorrectly showed "mixed" audio matrix levels in the Trim tab.

FIXED: When editing the video anchor point via the geometry text fields, the translation values will be automatically adjusted to try to keep the video cue in its original position.

FIXED: The video geometry view on the primary machine will be updated properly when someone is editing video geometry from a collaboration machine.

FIXED: A bug where undoing pasting in cue notes with rich text formatting did not reset notes formatting to default.

FIXED: An issue that could cause stages with multiple synchronized output displays to show different frames.

FIXED: Opening a workspace will no longer stop the video output of other open workspaces.

FIXED: A bug that would cause QLab to crash when using a mask image with incredibly large dimensions.

FIXED: A bug that would cause the outline in the Geometry tab to not update while dragging.

FIXED: Assorted other small bugs.

5.2.1 - August 1, 2023

ADDED: AppleScript properties that allow getting and setting the video output stage of Video, Text, and Camera cues.

FIXED: Several issues causing distortion or pops in Mic cue output.

FIXED: A bug that prevented fading audio patches if only an Audio license was installed.

FIXED: A bug refreshing the Audio Output Patch list after reordering patches.

FIXED: An issue that could cause a purple or pink tint when layering several ProRes 4444 videos with alpha on top of each other.

FIXED: Closing a workspace will no longer stop the video output of other open workspaces.

FIXED: A bug that could cause a Blackmagic DeckLink video input to fail.

FIXED: QLab can now recover if a video input is disconnected while a video monitor window is viewing that input.

FIXED: Several issues with the Bitfocus Companion and Behringer/Midas X32, M32, XR-18 network device descriptions.

FIXED: The `/live` version of audio output patch OSC command `/level` ... now returns the correct volume values.

FIXED: Assorted other small bugs.

5.2 - June 20, 2023

★ **ADDED:** **Patch cueing.** Fade cues and Reset cues can now target audio output patch routing, allowing you to adjust the overall levels of the entire patch, and easily reset those levels to their initial state.

★ **ADDED:** **Extended cue colors.** The number of cue colors has been expanded to make it easier to color-code a large variety of cues, and to better support the needs of people with color vision deficiency. The cue color control in the Basics tab has been redesigned to support this larger number of colors, and you can now set both a cue color and an optional second cue color. If enabled, the cue changes to the second cue color after it's started.

★ **ADDED:** **Multi-destination Network patches.** Each Network patch can now have an arbitrary number of destinations. This allows a Network cue to send a single message to multiple different network devices. Each destination can have a unique combination of network transport, interface, IP address, port, and/or passcode values.

★ **ADDED:** **Audio input names.** You can now name inputs (rows) in the Levels tab of the inspector for any cue that contains audio. Input names are per-cue, and can be edited in the inspector, by AppleScript, and by OSC.

ADDED: QLab can now optionally delete old backups of your workspace in order to keep from using too much disk space. This option keeps the 20 most recent backups from the last hour, the latest backup per hour from the last day, and the latest backup per day beyond that.

ADDED: Region names and the selected control point are now included in the grid display to assist with warping and focusing.

ADDED: Video, Camera, and Text cues in "Fill Stage" mode can now be set to one of three fill styles: fit to the stage, fill the stage, or stretch to fill the stage.

ADDED: The Active Cues display can now optionally be set to display the most recently started cue first, rather than last.

ADDED: You can now use arrow keys to move and resize video regions. A new "?" help button in the video stage editor makes it easier to keep track of available keyboard shortcuts and modifiers.

ADDED: You can now drag to re-order video stages and video routes.

ADDED: For all OSC commands that set a boolean value, you can now use the string argument " `toggle` " or append " `/toggle` " to the address to toggle the current value.

ADDED: Workspace MIDI and OSC controls have been added to move the playhead to the next and previous cue sequence.

ADDED: Audio output patch OSC commands `/level` , `/reset` , `/routing` , and `/routing/reset` .

ADDED: Group cue OSC commands `/collapse` and `/expand` .

ADDED: Cues that output audio can use `/audioOutputPatch/…` to send OSC messages to their currently selected audio output patch.

ADDED: The Light Dashboard OSC command to set light values, `/dashboard/setLight` , now has a "live" variation, which allows you to modify light values without editing the dashboard, and thus also avoiding the undo stack.

ADDED: A network device definition for Reaper (using the default OSC mapping.)

ADDED: Instrument definitions for the American DJ COB Cannon LP200ST, Eliminator Stinger Spot 30, Eliminator Stryker Beam, Eliminator Stryker Wash, Encore FR Pro Color, Encore FR50Z, Encore FR 150Z, Encore LP5IP, Encore LP7IP, Encore LP12IP, Encore LP18IP, Encore Profile 1000 Color, Encore Profile 1000 WW, Encore Profile Pro Color, Encore Profile Mini Color, Encore Profile Pro WW, Encore Profile Mini WW, Encore Z7LP, Focus Flex, Focus Flex L7, Focus Hybrid, Focus Profile, Focus Wash 400, Hydro Beam X12, Hydro Profile, Hydro Spot 1, and Hydro Spot 2.

ADDED: Instrument definitions for the Chauvet COLORdash Par H7X, H12X, H18X, Ovation E-2, Rogue Outcast 2 Beam, Rogue Outcast 2 Hybrid, and Rogue Outcast 3 Spot.

ADDED: Instrument definitions for the Clay Paky Arolla Profile HP, Arolla Spot MP, Mini Xtylos, Mini Xtylos HPE, Mini-B Aqua, Mini-B Spot, Sharpy X Frame, Sharpy X Frame FD, Sinfonya Profile 600, Volero Batten Aqua, and Volero Wave.

ADDED: Instrument definitions for the ETC ColorSource Fresnel V, Desire Fresnel, and Source 4WRD Color II.

ADDED: Instrument definitions for the German Light Products Impression X5 and Impression X5 Compact.

ADDED: Instrument definitions for the Robe Footsie 1 WW, Footsie 1 TW, Footsie 1 RGBW, Footsie 2 WW, Footsie 2 TW, Footsie 2 RGBW, iBar 15, iBeam 350, iForte, iPainte, iSpiider, iTetra 2, Painte, T11 Profile, T11 PC, T11 Fresnel, and TX1 PosiProfile.

CHANGED: Instead of keeping one autosave copy that is periodically updated, QLab creates timestamped backup copies so that you can easily see a version of your workspace from a specific point in time.

CHANGED: Instead of having separate “autosave” and “snapshot” files, all backup copies of your workspace are now simply called “backup” copies.

CHANGED: The audition tab of the light dashboard is now more visually distinct.

CHANGED: Timecode cues now have a default duration.

CHANGED: Entering show mode now clears the find results.

CHANGED: The MIDI File cue has received some visual improvements.

CHANGED: Media logging now logs timestamps down to the second, rather than just minute. It also includes Video cues which target still images.

CHANGED: The waveform view in the Time & Loops tab of the inspector now supports trackpad pinch gestures for zooming in and out. To zoom using the vertical scroll gesture on a trackpad, hold the ⌥ (option) key while scrolling. Zooming with the scroll wheel on a mouse or trackball has not been changed.

FIXED: Batch editing very large numbers of cue properties is now much faster.

FIXED: Updating the warping of a video region from a collaborator machine will no longer flicker as you drag the region.

FIXED: A bug affecting how audio effects on Cue Outputs are loaded that could cause errors with some plugins when opening a workspace.

FIXED: A bug that caused a still-image Video cue in a crossfading Playlist group to flash briefly when the next cue started fading in.

FIXED: Cues whose second trigger action is set to “second trigger on release” will now respond to a MIDI Note Off message at any velocity.

FIXED: Several fixes to Network cue OSC `/fadeEntries` and AppleScript “fade entries”.

FIXED: Using AppleScript to collapse / expand cue lists and carts now works in the workspace window Lists / Carts pane.

FIXED: A bug in the Chamsys MagicQ network device description that caused Playback “10” to output as “20”.

5.1.4 - May 22, 2023

CHANGED: Video monitor and audition windows now have more visible edges on dark backgrounds.

FIXED: A bug that prevented Video cues that target still images from refreshing when their target file was updated.

FIXED: A bug which could cause strange problems in the final slice of a sliced cue if the duration of its target file changed.

FIXED: OSC clients are now correctly notified of changes to the `/liveColorName` property of cues.

FIXED: Assorted other small bugs.

5.1.3 - May 8, 2023

ADDED: A network device definition for the Behringer X-Air series of mixers.

FIXED: Several very annoying bugs with undo, particularly a case where invoking undo from a collaboration remote could undo many changes at once.

FIXED: Syphon and NDI outputs are now left running when physical output devices are reconfigured.

FIXED: Edge blends where more than two regions overlap are now calculated correctly.

FIXED: Deleting a video region as a collaborator will now correctly hide that region’s grid image if it was previously visible.

FIXED: Command-left arrow and command-right arrow now moves between tabs in the audio output patch editor.

FIXED: An error in the AtemOSC network device definition.

FIXED: An error in the X32/M32 network device definition.

5.1.2 - April 26, 2023

ADDED: You can now tab between slice loop counts in the audio waveform view.

ADDED: Video cue OSC command `/opacity` now supports +/- increment syntax.

CHANGED: When trying to collaborate with an incompatible version of QLab, a better message is displayed about the incompatibility. The explanation also includes a button to take you to the download page.

FIXED: Improved synchronization between the video and audio of Video cues, both when cues start and during playback.

FIXED: Improvements to how Video cues start playback.

FIXED: A regression introduced in v5.1 where certain operations from a collaboration machine would stop playback of cues on the primary.

FIXED: A bug where pausing a DeckLink camera input with a disconnected camera could lead to a crash.

FIXED: Loading or saving certain audio effect presets should now work correctly.

FIXED: A condition in which a video stage with multiple output routes could send video to the wrong routes if they were assigned in a particular order.

FIXED: The hotkey for opening workspace settings (command-comma) did not always work the first time it was used.

FIXED: Bugs that prevented Group cue OSC `/currentTimecode` and Light cue OSC `/subcontroller` from working correctly.

FIXED: A bug that prevented being able to edit Network cues after pasting cue properties.

FIXED: Network cue OSC command `/customString` now works when used in an OSC query.

FIXED: Overall behavior of the light definition editor has been improved.

FIXED: Pasting both start time and end time now works properly all the time.

FIXED: Group cues' drag-and-drop targeting behavior now matches other cues' behavior. This will make it easier to drag a Fade cue inside a group cue without accidentally changing the target of the Fade cue.

FIXED: You can now specify the default video clock in the Video cue template.

FIXED: Pasting levels onto Fade cues will no longer deactivate the previously active fade channels.

5.1.1 – March 21, 2023

ADDED: Instrument definitions for the American DJ Eliminator Stealth Beam, Stealth Craze, Stealth Spot, and Stealth Wash Zoom.

ADDED: Instrument definitions for the Chauvet Rogue R3 Beam.

ADDED: Video & Text cue OSC commands `/cueSize/width` and `/cueSize/height`.

ADDED: Text cue OSC commands `/text/outputSize/width` and `/text/outputSize/height`.

FIXED: A regression that caused Text cues to stop when you edited their text while running.

FIXED: A bug that could cause QLab to crash when loading a playlist under some circumstances.

FIXED: Setting a route to the pixel dimensions of a rotated device will now work correctly.

FIXED: Video effects with a color parameter can now be used in Video cue templates.

FIXED: Assorted other small bugs.

5.1 – February 27, 2023

★ **ADDED:** **Media logging.** QLab can now optionally create a csv-formatted log of all audio and video file playback in a workspace, using information from QLab and metadata in the target media files. This is primarily intended to make it easy to create reports for rights-holding organizations, like ASCAP or BMI, but could be useful for other purposes too. This feature can be enabled in the QLab application preferences.

★ **ADDED: HSI color picker for light instruments.** Light instruments with hue, saturation, and intensity (or hue, saturation, and value) parameters can now use a graphical HSI color picker in the Light Dashboard and in the Levels tab of the inspector.

ADDED: New options to control how QLab behaves when it starts receiving a new timecode stream. Options include: do not start cues whose timecode triggers precede the incoming timecode (similar to QLab 4), start cues whose timecode triggers are within the most recent minute of the incoming timecode, start cues whose timecode triggers are within the most recent hour of the incoming timecode, start cues whose timecode triggers are within a user-defined "lookback" window of time, or start all cues whose timecode triggers precede the incoming timecode.

ADDED: A completely redesigned timecode window, showing detailed information about all incoming and outgoing timecode streams.

ADDED: Option to display video grids with a light background via the "Grid options..." button in the video stage editor.

ADDED: Export video stage grid images via the "Grid options..." button in the video stage editor.

ADDED: Show all / hide all video grids in the stage editor, via the "Grid options..." button or by option-clicking a grid checkbox.

ADDED: You can now specify the pixel format of NDI and Blackmagic video outputs, which offers improved compatibility with some receivers.

ADDED: "Paste audio levels" and "Paste integrated fade shape" commands now appear in the Tools menu when appropriate.

ADDED: Instrument definitions for the Astera FP6 HydraPanel and PB15 PixelBrick.

ADDED: Instrument definitions for the Blizzard Colorise Zoom RGBAW, Kryo Mix CMY, Stiletto GLO19, and Stiletto Ray-Z.

ADDED: Instrument definitions for the Cameo Instant Fog 1700 Pro.

ADDED: Instrument definitions for the Chauvet Maverick Storm 2 Profile, Maverick Storm 4 Profile, onAir IP Panel Min, onAir IP Panel 1, and onAir IP Panel 2.

ADDED: Instrument definitions for the Chauvet DJ COLORband H9 USB, COREpar 80 USB, COREpar Q60 ILS, COREpar Q120 ILS, COREpar UV USB, COREpar UV 120 ILS, and SlimPAR 64 RGBA.

ADDED: Instrument definitions for the Elation DLED Strip RGBAW, Opti Tri Par, and Platinum Beam 5R.

ADDED: Instrument definitions for the ETC ColorSource Spot V.

ADDED: Instrument definitions for the ETC/High End Systems Halcyon Gold, Titanium, and Platinum.

ADDED: Instrument definitions for the Martin ERA 150 Wash, ERA 600 Profile, ERA 800 Profile, MAC Aura XIP, MAC Ultra Performance, and MAC Ultra Wash.

ADDED: Instrument definitions for the Robe ProMotion.

ADDED: Instrument definitions for the Showtec (Highlite) Compact Par 7 Tri MKII, Spectral PC 600Z IP, and Spectral PC 1800Z IP.

ADDED: Network device description for the Mixeffect iOS app.

ADDED: Network device description for Bitfocus Companion.

ADDED: Sending an empty string argument to OSC command `/replyFormat` now resets to the default reply format.

ADDED: Additional support for importing show files from Go Button.

CHANGED: NDI has been updated to version 5.5.2.

CHANGED: The expansion state of Group cues and cue lists is now synced to collaborators when they first connect.

CHANGED: Improvements to the search filter in the file target picker on remote collaborator workspaces.

CHANGED: The minimum volume level for a workspace is now limited to -180 dB.

CHANGED: All instrument definitions for Astera instruments have been updated to reflect the latest firmware from Astera.

CHANGED: All instrument definitions for Chauvet instruments have been appropriately categorized as either Chauvet DJ or Chauvet Professional.

FIXED: A white flash that could show on some video hardware when stopping all video output.

FIXED: A rare issue in which QLab could crash when stopping all video output.

FIXED: A few cases where hiding video grid images did not work correctly.

FIXED: An issue where QLab would sometimes complain and report an exception if an attached display was disconnected.

FIXED: Video inputs such as webcams that have the same name but different internal identifiers are now managed better.

FIXED: Audio inputs will no longer accumulate additional latency when restarted.

FIXED: Timecode cues sending LTC output could previously emit invalid data when panicked.

FIXED: Blackmagic DeckLink video inputs can now be paused correctly.

FIXED: Video routes rotated 90° or 270° relative to their output device will now be scaled correctly.

FIXED: A regression in 5.0.12 that made it difficult to construct a cue sequence that pauses a Video cue on the first frame of video.

FIXED: A regression in 5.0.11 that prevented Text cues from respecting their duration.

FIXED: Several bugs related to loading Video cues, where it was possible to construct a cue sequence where Video cues would not be loaded properly.

FIXED: Video cues triggered via timecode and following the video clock will now have smoother audio playback.

FIXED: Paste Cue Properties now includes drop shadows when pasting Text cue formatting.

FIXED: A bug that could cause collaborator workspaces to crash when setting file targets on the primary.

FIXED: Cues inside a group that was deleted and brought back via undo will now run.

FIXED: Adding or removing audio and video effects will now ensure that Fade cues which target those Audio or Video cues will also get updated effects.

FIXED: Changing certain attributes of a Network cue fade while the cue is running now properly resets the cue.

FIXED: An error in the atemOSC network device description.

FIXED: An error in the Behringer/Midas X32, M32, XR-18 network device description.

FIXED: A bug that could cause Script cues to break with an error after sending an AppleScript command to the Light Dashboard.

FIXED: Several issues when importing from a Go Button show file: cart cues now match the order of hits, Audio and Wait cue continue modes are now correct in a group created from a cue that has a go action but no start actions, and empty cues now import with the correct display name.

FIXED: Assorted other small bugs.

5.0.13 - January 9, 2023

ADDED: Instrument definitions for the American DJ 5PX Hex.

ADDED: Instrument definitions for the Chauvet DJ Cumulus, Geyser P7, Intimidator Spot 355Z IRC, Intimidator Spot 375Z IRC, and SlimPAR Q12 USB.

FIXED: A regression in 5.0.12 that prevented Video cues from displaying video when following the audio clock.

FIXED: A situation where dragging files from the Finder onto a cue could stop working even though the cue should accept the file.

FIXED: An exception or crash that could occur when refreshing the Video cue inspector.

FIXED: Assorted other small bugs.

5.0.12 - December 20, 2022

ADDED: Instrument definitions for the Cameo Otos H5.

ADDED: Instrument definitions for the Chauvet Maverick Storm 2 BeamWash, Ovation CYC 3 FC, Rogue Outcast 1L Beam, and Rogue Outcast 2X Wash.

ADDED: Instrument definitions for the Chauvet DJ Freedom Q1N.

FIXED: Avoid showing the first frame of a video file more than one time when starting a Video cue.

FIXED: File targets assigned by remote collaborators will now be copied correctly during a Save-As operation on the primary machine.

FIXED: A crash that could happen if an OSC message contained an invalid wildcard pattern.

FIXED: Avoid loading Video, Camera, and Text cues that would no longer be playing when chasing timecode. This can significantly reduce the amount of processing required to begin chasing timecode in the middle of a heavy sequence when using these cue types.

FIXED: One omission and one error in the d&b DS100 network device description.

FIXED: Assorted other small bugs.

5.0.11 - November 29, 2022

ADDED: Instrument definitions for the Blizzard Colorise Zoom RGBAW, Kryo Mix CMY, Stiletto GLO19, and Stiletto Ray-Z.

ADDED: Instrument definitions for the Cameo Instant Fog 1700 Pro.

ADDED: Instrument definitions for the Chauvet Maverick Storm 2 Profile, Maverick Storm 4 Profile, onAir IP Panel Min, onAir IP Panel 1, and onAir IP Panel 2.

ADDED: Instrument definitions for the Elation DLED Strip RGBAW, Opti Tri Par, and Platinum Beam 5R.

ADDED: Instrument definitions for the ETC ColorSource Spot V.

ADDED: Instrument definitions for the ETC/High End Systems Halcyon Gold, Titanium, and Platinum.

ADDED: Instrument definitions for the Robe ProMotion.

ADDED: Instrument definitions for the Showtec (Highlite) Compact Par 7 Tri MKII, Spectral PC 600Z IP, and Spectral PC 1800Z IP.

5.0.10 - November 16, 2022

FIXED: A crash when importing certain v4 workspaces.

5.0.9 - November 15, 2022

ADDED: Network device description for Shure ULX-D wireless receivers and P10T transmitter.

ADDED: Support for batch editing waveform start, end, and slice times if all the selected files are the same duration.

CHANGED: The network device description for the ETC Eos family now supports optionally specifying a user number.

FIXED: An issue that prevented integrated audio fades from being applied after a separate Fade cue adjusted the main volume.

FIXED: QLab has an improved ability to decode and chase LTC timecode in cases where the incoming signal is slightly degraded.

FIXED: Fade cue video effect parameters can now be imported from v4 workspaces.

FIXED: Fade cues on collaborator machines will now show the correct number of level input channels.

FIXED: An issue that could prevent MIDI patches from finding MIDI hardware.

FIXED: Improvements and fixes to DeckLink video outputs.

FIXED: Compositing video files from different color spaces will now produce more accurate and consistent results.

FIXED: Currently visible guides and grids are now managed properly when assigning or editing output routes.

FIXED: Video cues that are stopped and then rapidly restarted by a cue sequence should end up actually playing now.

FIXED: Stage masks now update live when the underlying image file is modified.

FIXED: Worked around bug in Vega 56 on iMac Pro that caused video output to be tinted blue when the stage had a mask.

FIXED: Assorted other small bugs.

5.0.8 - October 14, 2022

FIXED: A regression in 5.0.7 that caused QLab to crash when starting multiple video, text, or camera cues simultaneously.

5.0.7 - October 13, 2022

ADDED: Video stage region OSC commands `/uniqueID` and `/name`.

CHANGED: QLab will now more pro-actively turn off audio inputs (and thus the terrible orange dot) if no cues and/or audio input patches use audio inputs.

FIXED: A regression that caused QLab to crash if a Start cue targeted a Group that contained it.

FIXED: An issue that could cause QLab to crash if cues were added, deleted, or moved during autosave.

FIXED: An issue that could cause cue playback to stall if a cue was started and within the same cue sequence immediately stopped again.

FIXED: An issue that could cause a brief flash of gray or white at the beginning of video playback on some machines.

FIXED: An exception in the Network cue inspector Settings tab when the cue batch had mixed values for a boolean parameter.

FIXED: Popup menus for Network cue inspector parameters that are in a "(mixed)" state now work correctly when selecting a new value.

FIXED: Collaboration workspaces will now correctly show the cue order of auto-shuffled Playlist groups.

FIXED: Collaboration workspaces will now show the correct number of audio file channels.

FIXED: Assorted other small bugs.

5.0.6 - October 7, 2022

ADDED: Network device description for the Shure Axient Digital line of wireless receivers.

CHANGED: QLab's color rendering approach has been improved.

FIXED: Restored the ability to set the cue list playhead by clicking on the row to the left of the cue's icon.

FIXED: An issue where deleting an audio effect from an audio patch would not immediately remove the effect.

FIXED: The checkbox for enabling or disabling an audio effect in the Fade cue inspector now works.

FIXED: You can now always set Fade cues to absolute mode if they have no target cue. This also fixes the ability to set absolute mode on the template Fade cue.

FIXED: Fade cues imported from a v4 workspace that adjust the parameters of an audio effect should now work correctly.

FIXED: MIDI patches assigned to virtual devices will now be reconnected when reopening a workspace.

FIXED: Support sending Network messages that require extra whitespace (such as newlines) at the end of a message.

FIXED: Corrections to the Network device description for the MusicTribe X32/XR18/M32

FIXED: Sending audio to an NDI output will no longer have a very brief ramp-up of playback rate at the beginning of playback.

FIXED: Assorted other small bugs.

5.0.5 - September 26, 2022

FIXED: The infinite loop property of Video cues will again be restored properly when opening a workspace.

FIXED: A bug that broke functions related to the video warp control points. This fixes things like “dragging a whole region in the warp editor.”

FIXED: Under certain conditions, a Timecode cue sending MTC triggered in a heavy cue sequence of other cues could send out a small number of MTC messages simultaneously. Fixing this prevents an issue where where some receivers would interpret this as time moving into the future very fast for a very short period of time.

FIXED: Anything (such as OSC) that addresses audio patch channel names is no longer case-sensitive.

FIXED: The *Move playhead to this cue* contextual menu item will now work on collaborator machines.

FIXED: Assorted other small bugs.

5.0.4 - September 22, 2022

ADDED: Network device descriptions for Synthe FX’s Luminair 4, Blackmagic Design’s Videohub series, and MOTU’s AVB audio interface series.

ADDED: OSC message `/cartPosition` now accepts optional `/row` and `/column` selectors, e.g. for using in OSC queries.

ADDED: AppleScript read-only property “cart position” to get the row and column of the position of a cart cue.

ADDED: Group cue OSC commands `/playlist/next` and `/playlist/previous`.

CHANGED: Group cue OSC commands `/playlistShuffle`, `/playlistLoop`, `/playlistCrossfade`, and `/playlistCrossfadeDuration` are deprecated. Use new commands `/playlist/doShuffle`, `/playlist/doLoop`, `/playlist/doCrossfade`, and `/playlist/crossfade/duration` instead.

FIXED: Playlist groups set to “Auto-shuffle” behave more predictably if loaded or triggered while running.

FIXED: Playlist groups can now be paused and resumed.

FIXED: Start cues can now target playlist group cues.

FIXED: When importing v4 workspaces, v5 will now find media files that it previously could not locate.

FIXED: A bug that caused still image Video cues with a duration to sometimes behave as if no duration was set.

FIXED: OSC messages sent via the special internal localhost path will now be logged in Workspace Status > Logs.

FIXED: A bug that prevented Network cues sending to a Plain Text destination from having line breaks.

FIXED: The Network device description for SPAT Revolution has been updated with a few minor corrections.

FIXED: The OSC Access settings pane now disables the “Edit” and “Control” checkboxes when “View” permission is not enabled.

FIXED: A bug that prevented collaborators from editing audio patches.

FIXED: A crash that could occur when managing Audio Effect presets.

FIXED: Several issues that could pop up when showing or hiding video grids. Now, just the video grids will pop up.

FIXED: A crash that could occur when showing a video monitor window.

FIXED: It is now possible to edit the settings of Blackmagic DeckLink devices.

FIXED: When assigning a region to a route, the region’s “size on stage” is constrained so it does not land entirely outside of the route raster.

FIXED: Editing large mesh splits will no longer slow QLab down.

FIXED: A bug that could prevent being able to edit the text color of a Text cue after pasting text into the Inspector.

FIXED: A bug where Timecode cues set to MTC mode were not able to send MTC messages for a period of time after rebooting your computer. If you’re stuck using MTC, you now have both our sympathies and also a functional MTC Timecode cue.

FIXED: The valid cue output audio channels will now be shown on remote collaborator workspaces.

FIXED: Conditional cue colors will now be shown on remote collaborator workspaces.

FIXED: An issue that could cause the selection of multiple cues to be lost when shift- or command-clicking on the cue list.

FIXED: Assorted other small bugs.

5.0.3 - September 12, 2022

FIXED: An error preventing automatic updates.

5.0.2 - September 12, 2022

CHANGED: Network cue inspector parameter tooltips now show both the underlying parameter keys and values to help with scripting.

CHANGED: The OSC messages for interacting with a Video, Camera, or Text cue's anchor point have been updated to `/anchor` in order to match the language used in the inspector. The older `/origin` messages remain for backwards compatibility, but are deprecated.

CHANGED: Playlist Group cues set to loop now require at least one of their children to have a duration greater than zero. This prevents a recursion crash which caused a recursion crash.

FIXED: It is now possible for a Fade cue to fade a Group if it contains Network or MIDI cues.

FIXED: Video cues loaded to the very end of the file will no longer be stuck "playing" when they have nothing left to play.

FIXED: Video cues that use a file encoded with different lengths between the audio and video tracks will no longer risk being stuck "playing" when they have nothing left to play.

FIXED: Video cues with looping slices now correctly show their duration before they're loaded.

FIXED: Changes in video inputs will now only reset the geometry of Camera cues that use those inputs, and not reset the geometry of unrelated Video cues.

FIXED: An audio license is no longer required for Camera cues.

FIXED: A bug that could prevent creating a new cue if the cue template had an audio effect enabled in it.

FIXED: A crash that could occur when connecting to some NDI input sources.

FIXED: A crash that could occur when editing the output route associated with an NDI output device.

FIXED: A bug that could cause Syphon video output to fail when using a computer with multiple discrete GPUs.

FIXED: A bug that prevented getting "point" values using the Network cue AppleScript "parameter values" property.

FIXED: The workspace setting which restricts collaborators to view-only permission when the workspace is in show mode now works correctly.

FIXED: Assorted other small bugs.

5.0.1 - September 2, 2022

CHANGED: The minimum duration allowed for the Playlist group crossfade view is now 0.05 seconds.

CHANGED: The Network device description for ZoomOSC has been updated to reflect new features in ZoomOSC.

FIXED: MIDI virtual outputs can now be used as MIDI destinations.

FIXED: A crash that could occur when importing network patches from a QLab 4 workspace.

FIXED: A crash that could occur when deleting a stage whose monitor window is open.

FIXED: NDI video output devices with certain non-standard pixel dimension widths now output correctly.

FIXED: NDI and Syphon output devices will now be updated when their parameters are edited.

FIXED: Syphon will now work on certain older hardware, where previously it would crash.

FIXED: Clicking in the audio waveform or in the active cues panel will no longer stop a cue that is currently being auditioned.

FIXED: When setting new audio or video targets as a remote collaborator, the new audio waveform (and other properties defined by the new file) will be now be synced back to the remote workspace.

FIXED: A bug that prevented creating Text cues if the cue template text formatting has a shadow applied.

FIXED: A bug that prevented displaying the Fade shape view for some Network cue parameters.

FIXED: Assorted other small bugs.

5.0 - August 30, 2022

General

★ **ADDED:** **Collaboration.** QLab 5 allows multiple people on separate Macs to collaborate on a workspace, live and in realtime, over a local network.

★ **ADDED:** **Autosave and Snapshot.** Autosave prevents accidental loss of work in the event of a crash or power outage. Snapshots let you easily revisit an earlier version of your workspace to compare against or recover deleted work.

★ **ADDED:** **Playlist mode.** Group cues have a new Playlist mode that allows you to create sequential playlists which advance automatically or manually with optional crossfading, shuffling, and looping.

★ **ADDED:** **Audition improvements.** Cues can now be auditioned on an individual basis, and each type of cue output can be given its own audition behavior:

- Audio, Video, MIDI, MTC, and LTC output can be individually set to audition to a specific output, the same output as usual, or no output.
- Video output can also be set to audition to an audition window. Each video stage (formerly called surface) gets its own audition window.
- Light cues audition to an “Audition” tab in the Light Dashboard.
- Workspaces can be set to *Always audition* which turns every GO into an Audition GO and every Preview into an Audition Preview, in the style of QLab 4.
- Cues in the midst of auditioning can be interrupted and restarted normally with a single GO or Preview command; no need to stop first, then restart.

★ **ADDED:** **Cart refinements.**

- You can now manually set the grid size of a cart anywhere from 1 × 1 cells up to 10 × 10 cells.
- Cart cues can now have pre-waits.
- Cart cues display their duration and pre-wait, if applicable, when they are not playing.
- Carts now have three display sizes, just like lists, which can be configured in the General → Display Size section of Workspace Settings.
- Assorted improvements to aesthetics and legibility.

★ **ADDED:** A new **I/O** tab in the inspector for Audio, Mic, Video, Camera, and Text cues.

★ **ADDED:** The new **file search tool** makes it dramatically easier to reconnect broken cues to their missing file targets, for example after moving a workspace to a new computer or reorganizing media while QLab is not running.

★ **ADDED:** Workspace Settings can now be exported to a settings file, imported from a settings file, imported from other open workspaces, and drag-and-dropped between workspaces.

★ **ADDED:** The Warnings tab of the Workspace Status window has been comprehensively updated to give you much more and much better information about broken cues as well as non-breaking warnings such as workspace settings which are in need of attention.

★ **ADDED:** QLab 5 runs natively on Apple Silicon.

ADDED: A **contextual menu** is now available when right-clicking (or control-clicking) on cues in the cue list and controls in the inspector. The contextual menu gives you quick access to relevant actions and extensive in-line help text with links to the relevant section of the QLab manual online.

ADDED: Cue colors can be set to appear all the time, only before the cue has played, or only after the cue has played, making it easier for you to use color to mark cues as “unplayed” or “played.”

ADDED: The “linear” fade curve type can now be edited, allowing sharp-cornered multi-step fade curves.

ADDED: You can now use the arrow keys to work with control points in fade curves. Use ← and → to select the previous or next point, ⌥ (option) plus arrow keys to move the selected control point, and ⌥⇧ (option shift) plus arrow keys for finer-grained movement.

CHANGED: Setting cues' targets by drag and drop now uses only the target column in the cue list, making it harder to change a cue's target accidentally.

CHANGED: Setting cues' targets by drag and drop now works bi-directionally; you can drag the acting cue onto its target cue, or drag the target cue onto the acting cue. If both cues could potentially be targets of each other, the dragged cue is set as the target of the cue it's dropped onto.

CHANGED: Fade cues in "absolute" mode will now override and clear out any previously applied changes made by Fade cues in "relative" mode.

CHANGED: Load cues can now use negative numbers to load backwards from the end time of their targets, just like the Load to Time tool.

CHANGED: QLab 5 uses macOS Dark Mode so that dialogue boxes, title bars, and other window "chrome" look more consistent with the rest of QLab's interface.

Audio

★ **ADDED:** QLab 5 has a **new audio patch system** which comes with a number of improvements:

- Workspaces support an unlimited number of audio output patches. Audio output patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the audio settings of other workspaces.
- You can create multiple audio output patches with the same audio device, each with its own routing and audio effects.
- The audio patch editor now supports undo and redo.
- All cues that output audio now use the same set of audio output patches (no more special output patches for Mic cues.)
- If an audio device is disconnected, QLab will still display the name of the disconnected device in the audio inspector, making it easier to work with "offline" audio devices.
- Audio output patches can now use the system output, which is the audio device selected in System Preferences → Sound → Output. This makes it easier for a workspace to be used in a context where QLab ought to use the same audio output as the rest of the Mac, even if that output changes.

★ **ADDED:** Mic cues now use separate audio input patches to designate the device they use for input. This lets you use separate audio input and output devices easily, without requiring you to set up an aggregate audio device. Audio input patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the audio settings of other workspaces.

★ **ADDED:** Slices in Audio cues (and Video cues) can be set to a play count of 0; zero-count slices are seamlessly skipped during playback.

ADDED: The integrated fade curve in the Time & Loops can now use a linear fade shape; no obligatory Bézier curves.

ADDED: Mic cues (and Camera cues) are now able to use input channels numbered above 64. You can't use more channels than before, but you can, for example, use channels 70 and 71 of your high-channel count device for a two-channel Mic cue.

ADDED: Audio effect meters now work in all contexts, not just on outputs.

ADDED: You can now visually edit the start and end times of multiple selected cues by dragging the start time and end time handles in the Time & Loops tab of the inspector.

Video

★ **ADDED:** QLab's video rendering engine has been completely rewritten using the Metal framework, Apple's modern and fully up-to-date video system. Come for the performance improvements, stay for the longevity.

★ **ADDED:** QLab 5 has an entirely re-designed output system. Instead of Surfaces, video outputs are called Stages and they have considerably more power and flexibility.

★ **ADDED:** QLab 5 adds per-cue blend modes, allowing you to composite cues in nearly limitless combinations.

★ **ADDED:** QLab natively supports NDI 5 for both video input and output. QLab also supports NDI audio input and output.

★ **ADDED:** Camera cues contain an embedded Mic cue, allowing you to use live audio alongside live video.

★ **ADDED:** QLab can display monitor windows for every video input and output so that you can keep tabs on all your visual elements live and in realtime.

★ **ADDED:** Video, Camera, and Text cues can now use multiple video effects simultaneously. The list of available video effects has grown, too, thanks to the shift to Metal, and the amount of processing power needed for video effects (especially blurs) has been nicely reduced.

★ **ADDED:** Video cues which target video files that contain multiple audio tracks now allow you to choose which audio track to use. This is done in the I/O tab.

★ **ADDED:** When a Video cue's target file contains metadata describing the audio channel layout, that information is used to label the rows in the Levels tab.

★ **ADDED:** QLab now supports an unlimited number of video input patches in a workspace. Video input patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the video settings of other workspaces.

★ **ADDED:** Slices in Video cues (and Audio cues) can be set to a play count of 0; zero-count slices are seamlessly skipped during playback.

ADDED: Masks and video surface geometry now happen "upstream" of Syphon outputs, allowing you to send more elaborately crafted video feeds to Syphon-receiving clients.

ADDED: Video, Camera, and Text cues now have an integrated crop attribute for quick and easy trimming.

ADDED: Video effects are previewed live in the Geometry tab of the inspector.

ADDED: When using QLab without a video license installed, Workspace Settings → Video → Video Outputs provides a single, simple control to set all video output for the workspace to a single attached display. All cues will automatically play to this display.

Lighting

★ **ADDED:** The Light Dashboard has gained an Audition tab which displays the results of auditioned Light cues.

ADDED: QLab 5 ships with instrument definitions for over 1400 types of lighting fixtures from over 60 manufacturers.

ADDED: Light commands can now be cut, copied, and pasted when they are selected in slider mode in the Levels tab of the Light cue inspector.

Networking, MIDI, and Show Control

★ **ADDED:** **Timecode chasing.** Cues set to trigger from timecode can now start in the middle of the cue based on incoming timecode, rather than just at the beginning of the cue, and will skip ahead or back in response to timecode skipping ahead or back. Additionally, Lists and Carts set to receive timecode can optionally be set to pause or stop their timecode-triggered cues when incoming timecode stops, with optional freewheeling up to two seconds.

★ **ADDED:** Network patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the Network settings of other workspaces.

★ **ADDED:** QLab now supports an unlimited number of MIDI patches in a workspace. MIDI patches can be reordered, copied and pasted within a workspace and between workspaces, drag-and-dropped into the Finder to create a settings document, and drag-and-dropped to copy them into the MIDI settings of other workspaces.

★ **ADDED:** The Network cue has been substantially revamped and now includes support for directly controlling a number of OSC-controllable programs and devices with a minimum of fuss and complexity. The currently available modes for the Network cue are:

- OSC Message
- Plain Text (ASCII strings)
- Hex Codes (hexadecimal values)
- QLab 5
- Go Button 3
- atemOSC
- Audio Definition Model (ADM)
- Borealis Vor
- Chamsys MagicQ family
- Creative Connors Spikemark 5
- d&b Soundscape DS100 (which improves upon and replaces the QLab 4 Soundscape/DS100 feature)
- Disguise D3
- ETC ColorSource AV
- ETC EOS family (Element, Ion, Eos, Geo, etc.)
- Flux Spat Revolution
- High End Hog 4
- Innovate Audio panLab 2
- L'Acoustics L-ISA
- Meyer GALAXY (Normal mode and Spacemap mode)
- MusicTribe Behringer X32 and Midas M32
- Yamaha Rivage family (PM3, PM5, PM7, and PM10)
- ZoomOSC

ADDED: You can now customize the port numbers that workspaces use to receive OSC messages and plain text messages.

ADDED: Workspaces can have multiple OSC passcodes, each with their own set of access permissions. Custom OSC remote control commands are now compatible with workspaces that use OSC passcodes.

ADDED: Network cues now support both TCP and UDP transport, as well as OSC 1.1 argument types true, false, impulse, and null.

ADDED: When live fade preview is switched on, Network cues in 2D fade mode now transmit their message as you drag the control dot around. This should make it easier to experiment with 2D network fades.

ADDED: MIDI cues, MIDI File cues, and Timecode cues using MTC now all use the same set of patches.

ADDED: The Triggers tab of the inspector now has a capture button for capturing incoming timecode. For this button to work, timecode must be enabled on the cue list or cart that contains the cue and timecode must be incoming. Capturing timecode can also be done via AppleScript and OSC.

Scripting and Automation

★ **ADDED:** QLab 5's OSC dictionary has been substantially overhauled and expanded upon. Many new OSC commands have been added, many existing commands have new powers or options, and the whole set of commands has been reorganized and refined for clarity.

★ **ADDED:** There are now separate overrides for local network messages versus network messages that go to other devices. For outgoing messages, "local" is defined as any network patch whose address is localhost or 127.0.0.1. For incoming messages, "local" is defined as any message originating from the same computer that QLab is running on.

System Recommendations

This page was last updated on October 17, 2025.

QLab 5 is a Mac-only program. It requires macOS Big Sur (macOS 11) or higher, and can run on any Mac that can run macOS Big Sur or higher. QLab 5 runs natively on both Apple Silicon processors and Intel processors. While it is technically possible to run QLab in a virtual machine or on a home-built "hackintosh," these configurations have unpredictable problems and are neither recommended nor supported.

Apple tends to refer to technical specs in non-technical language, such as "processor" instead of "CPU" and "memory" instead of "RAM." While reasonable people may differ on whether this decreases or increases confusion, this manual will attempt to match Apple's terminology where possible in hopes of making it easier to compare Apple's published specifications against the experience of QLab users.

Mindset

QLab is designed to be as flexible as possible so that it can be useful in a wide range of settings. As a consequence, it can be difficult to lay down strict rules about how much computing power QLab needs to run well. This section of the manual aims to discuss general concepts surrounding processor, graphics, memory, and storage as they pertain to QLab, and should be read not as directions about what to do, but as recommendations about what to consider.

Processor (CPU)

Processor power has the most straightforward relationship to QLab performance; the more work QLab needs to do, the more processor power it will use. These things all have a substantial impact on the amount of processor power needed by QLab:

- The number of simultaneously running cues.
- The resolution, frame rate, and bitrate of video files played by Video cues.
- The sample rate, bit depth, and number of channels of audio files played by Audio cues.
- The number of lighting instruments used in Light cues.
- The use of live audio and video effects.
- The use of blend modes in Video cues.
- The complexity of AppleScripts run by Script cues.

Conversely, here are some things which generally do *not* have much of an impact on the amount of processor power needed by QLab:

- The total number of cues in a workspace.
- The file size of media targeted by Audio and Video cues. (The file size per se is not important, although higher resolution, sample rate, and bitrate often go hand in hand with larger file sizes.)
- Using a single media file as the target of multiple cues.
- Playing whole files for cues versus setting custom start times and end times.
- The number of slice markers in a cue.

The upshot

New Macs are available today with Apple Silicon processors. They can be categorized as follows:

- The base processors (**M** with a number and no other name) offer terrific performance. Their main limitation is related to the number of displays that they can drive (see below.)
- The **Pro** and **Max** processors offer superb performance with higher limits on displays.
- The **Ultra** processors offer the best possible performance.
- The speed increase of each generation of Apple Silicon processor over the previous is difficult to explain in brief, but generally speaking each generation is a step-wise improvement over the one previous, and more of a leap when compared to the generation before that.

You can also find refurbished or older but still new-in-box Macs which use Intel processors.

- **i3** and **i5** processors can handle simple shows,
- **i7** and **i9** processors are much better at running multiple simultaneous cues and live effects,
- **Xeon** processors, found in the Mac Pro and iMac Pro only, offer the best possible performance for an Intel Mac. Despite this, most Apple Silicon Macs outperform even the most powerful Mac Pro with a Xeon processor.

If you are spending money on a Mac that's new to you, we cannot overstate how much better the Apple Silicon Macs are. They truly are better in every single way. A mid-range Apple Silicon Mac will easily outperform a very, very high end Intel Mac.

Memory (RAM)

Loading and playing cues uses memory, so the more audio or video that needs to be loaded at any given moment, the higher the memory requirement will be. 4 GB should be considered the minimum for the simplest of shows, and 8 GB should be considered the minimum for shows of middling complexity or higher. As with processing power, more complex shows can benefit from (and may require) more memory. QLab is able to address as much memory as your Mac provides.

Storage (disk space)

The most important aspect of storage is thankfully the simplest; you need to have a sufficient amount of storage installed in your Mac to hold the data needed for your show. QLab workspaces are generally very small, so the real determining factor is the total size of all the media files used by your workspace.

The second most important aspect of storage is the read speed, which is the speed at which the Mac is able to gather data from storage. Nearly all Macs today use solid state drives (SSD), the very slowest of which is still awfully fast.

For best performance, we recommend using a Mac with an SSD, and keeping your workspace and its media stored internally on that Mac. If you must store your data on an external drive, it should be an SSD or a very fast, enterprise-class hard disk, and should be connected to your Mac via Thunderbolt, USB4, or USB 3.2.

For those who are pushing the limits of what is possible, the best possible performance comes from the built-in SSDs in Mac Pros, Mac Studios, and high-end MacBook Pros, as well as PCIe-based storage cards in modern Mac Pros.

Graphics (GPU)

If you use QLab for anything besides video, you almost definitely do not need to worry about the graphics performance of your Mac. QLab does make use of graphics processing for non-video related things, notably drawing the waveform view in the Time & Loops tab of the inspector and updating the Light Dashboard, but any reasonably modern Mac has sufficient graphics processing power for these tasks.

If you use QLab for video, however, the graphics processing capability of your Mac is very important and in keeping with the theme of this section of the manual, what you need depends entirely upon what you're trying to accomplish.

Intel-based Macs

We do not recommend using most Intel-based Macs for any video-related project. They may suffice for exceedingly simple shows, but thorough testing is recommended early enough in the process to allow for the possibility of securing a replacement if necessary.

There are two notable exceptions:

- The **2017-2019 15" or 16" MacBook Pro** came in various models, the best of which all supported multiple external displays at fairly high resolutions. These Macs work well for moderately complex video needs.
- The **2019-2022 "return of the cheese grater" Mac Pro** allows you to install multiple video cards. Dedicating one modest video card for your operator display and one or more fancier cards for your audience-facing displays is a good strategy. These Macs work well for complex video needs, and the best of them (while *terrifically* expensive) work well for very complex video needs.

Intel-based Macs can use eGPUs, which are GPUs connected via Thunderbolt, and QLab can "see" and make use of them. While these are useful, any eGPU setup will cost more than a new Mac Mini which can offer better utility in almost all cases. For that reason, we do not recommend using eGPUs unless you are given one to use for free.

Apple Silicon-based Macs

All Apple Silicon Macs have integrated graphics processors, which means that the graphics processor is built into the same physical package as the regular processor and is therefore not removable or replaceable. Historically, integrated graphics processors offered fairly lackluster performance and were not recommended for serious video use. The Apple Silicon integrated graphics system, on the other hand, uses a novel design which is profoundly more powerful than older integrated graphics processors. Additionally, Apple Silicon processors include dedicated circuitry for video decoding, making them the best possible choice for video performance.

The overall, general performance of Apple Silicon graphics scales with price; the more expensive the Mac, the better its graphics performance will be.

Apple Silicon-based Macs do not support eGPUs.

Audio Output

All Macs have a ⅛" (3.5 mm) stereo output jack, and if all you need is one- or two-channel output, this can be fine. This connection is unbalanced, so it is recommended that the length of the cable that you plug into this jack be no longer than six feet or 1.8 meters (metres) in order to minimize noise.

If you need more than two channels, balanced connections, or outputs in other physical formats (such as analogue XLR3, AES, MADI, etc.) you can use any Core Audio-compliant audio device, which is nearly every audio device that works with a Mac. Through Core Audio, QLab supports output at a resolution of 16 or 24 bits at sample rates up to 192 kHz, although sample rates above 48 kHz require substantially more processing power and are not recommended.

QLab also supports audio output over a network via Dante Virtual Soundcard and macOS's built-in implementation of AVB. For Video and Camera cues which output to NDI, QLab can use NDI's built in audio channels as well.

The use of more than two channels of audio output requires an audio license.

Video Output

QLab supports video output through the built-in connections on your Mac, including the connections on graphics cards installed Mac Pros with PCI slots, and connections on eGPUs used with Macs that support eGPUs (Intel-based Macs with Thunderbolt 3 connections.)

Using a video license, QLab can also output video via Blackmagic Design's UltraStudio, DeckLink, and Intensity capture and playback devices; via Syphon; and over a network using NDI.

All current Macs utilize USB-C connectors to deliver video, usually via an adapter which provides a DisplayPort or HDMI connection. Understanding USB-C and the various ways that it deals with video can be challenging. You can learn more about this topic in the section on understanding USB-C in this manual. Many, but not all, current Macs also have HDMI connectors built in.

The number of displays supported by Macs is highly variable and dependent on both the kind of Mac and its configuration. You can learn about the video output capabilities of any Mac from its page on Apple's website.

This table shows the maximum number of total displays supported by a given Apple Silicon processor. For laptops and the iMac, the built-in display almost always counts as one of these. In most cases, displays can run at up to 6K resolution at 60 fps, though sometimes one display is limited to 4K resolution. Generally, displays connected to the built-in HDMI connector on the Mac can achieve higher frame rates than displays connected through Thunderbolt adapters, although in a theatrical context that is rarely important.

Processor	Display support

Processor	Display support
M1 M2	**Laptops and iMac:** the internal screen plus one external display. **Mac Mini:** two displays.
M1 Pro M3	Two displays total. Laptops with M3 and newer can run with the lid closed and use an external display in place of the built-in display.
M2 Pro M3 Pro M4 M4 Pro M5	Three displays total.
M1 Max M2 Max M3 Max	Four displays total.
M1 Ultra M4 Max	Five displays total.
M2 Ultra	Eight displays at 4K *or* six displays at 6K *or* three displays at 8K.
M3 Ultra	Eight displays at 6K *or* four displays at 8K.

Using Blackmagic Design devices, all Macs can support additional displays at various resolutions and frame rates, although the exact details vary widely depending on the model of Mac, the choice of Blackmagic Designs device, and the other hardware connected to the Mac.

Using NDI for video output, you are more likely to be limited by your network capacity than by your Mac's capacity.

We do not recommend, nor do we support, video output via USB DisplayLink monitors or graphics adapters. While these devices can work, they are not compatible with hardware graphics acceleration and they have a history of spotty performance. Your mileage may vary, but our advice is to avoid them entirely. If you are not sure whether a device uses DisplayLink, the litmus test is this: if you need to install a driver or other special software to make the display work, then it's probably DisplayLink.

Lighting Output

QLab can connect to DMX-controlled devices via Art-net and via a small list of specific USB-DMX adapters. QLab uses Art-net version 3, which means it's compatible with any devices that use Art-net 3 or Art-net 4.

Ask Us

If you have specific questions about hardware choices or requirements, please email the QLab support team, and tell us about your show. We will be happy to help you.

Preparing Your Mac

What follows here is a list of the programs or processes which we recommend disabling and instructions for doing so, as well as some other recommended practices. This section presupposes a basic understanding of macOS and at least a passing familiarity with the Terminal, which can be found in Applications → Utilities.

The directions here have been updated in December 2024 to reflect language changes and new features available in macOS Sequoia (15), but they are almost entirely applicable for Macs using macOS Big Sur (11)or newer. In Big Sur and Monterey, *System Settings* is instead called *System Preferences*.

Click here to download our "Prep and Restore" tutorial workspace (current version: December 2024). The workspace contains two Script cues. The first one executes all the prep steps below, not including the video-specific ones or the ones which have to be done using the mouse. The other Script cue reverses these steps. Because it is a tutorial workspace, it does not require a QLab license to work despite using Script cues.

A note about `sudo`

`sudo` is a Unix command which is short for "superuser do." It allows you to run a command in the Terminal with "superuser" privilege. In normal human terms, this means that using `sudo` allows you to run a command that you ordinarily could not, usually because that command could have serious consequences, and it would be a security risk to *not* require some kind of extra confirmation. There have also been reports of `sudo` being successfully used to obtain a sandwich.

`sudo` can only be used by an administrator account. If you only have one account on your Mac, it's almost certainly an administrator account. If someone else set up your Mac, ask them about it. It is really very useful to run QLab using an administrator account and we recommend it if at all possible.

The Terminal commands below all use `sudo`, which means that the first time you use them in a new Terminal window, you'll be asked to enter your password. This is not nefarious; this is macOS making sure that you are indeed proactively choosing to do this thing, and also that you are a person and not a sneaky script or bot trying to pull a fast one. None of the commands below transmit anything to or receive anything from outside your Mac.

The Basics

First, make sure that your copy of QLab resides in the Applications folder on your Mac. Most of the time, Macs are very permissive about where things can be stored, but you can occasionally encounter problems which you would not otherwise encounter when running a program that is not kept inside the Applications folder.

Next, make sure that there are not multiple copies of QLab on your Mac, or if there are (for example if you deliberately want to use QLab 5 sometimes and QLab 4 sometimes,) ensure that only one of them is named "QLab" and the other is named something else like "QLab 4" or "Old QLab" or "QLab Just For The Holiday Show."

System Software

There are a number of programs, processes, and tasks that your Mac runs either periodically or all the time in the background. Many of these programs are essential, but many are not and disabling them will increase the total percentage of your computer's resources which are available to QLab.

Disable Spotlight

Spotlight is Apple's name for the macOS search tool. What makes the tool work so well, and so quickly, is a clever program which periodically updates an index of all the files on every disk or drive connected to your Mac. This periodic updating can be fairly intensive, and can temporarily prevent QLab from getting access to data, which can cause late cues or stuttering playback.

Using System Preferences

Open System Settings → Siri & Spotlight, an click on the button labeled **Spotlight Privacy...**. In the box which opens, and add every disk or drive that's connected to your Mac to the list. You can do that by dragging and dropping the disk or drive's icon from the Desktop into the list, or by using the +

button at the bottom.

To restore Spotlight's default behavior, remove all disks or drives from the Privacy tab list.

Using the Terminal

Open a Terminal window, type or paste in this command, and then press enter:

```
sudo mdutil -ai off
```

`mdutil`, short for "metadata utility", is the program that handles Spotlight configuration. The `a` in `-ai` means you want this command to apply to all disks connected to your Mac. The `i` in `-ai` means you want to turn indexing on or off, and `off` means turn it off.

If you prefer to apply this command to a specific disk, you can use this form of the command instead:

```
sudo mdutil -i off /Volumes/diskname
```

Replace "diskname" with the actual name of the disk.

To restore Spotlight's default behavior, use this command:

```
sudo mdutil -ai on
```

Prevent Sleep and disable Screen Saver

Obviously we don't want the computer to go to sleep during the show, nor do we want the screen to be turned off.

Using System Settings

Open System Settings → Lock Screen, and set the time for all four settings at the top to "never."

If you're using a portable Mac such as a MacBook, MacBook Air, or MacBook Pro, open System Settings → Battery, click the **Options...** button, set "Put hard disks to sleep when possible"[1] to "never."

If you're using a desktop Mac such as a Mac Mini, iMac, Mac Studio, or Mac Pro, open System Settings → Energy Saver and then:

- Turn on "Prevent computer from sleeping automatically when the display is off"
- Turn off "Put hard disks to sleep when possible."[1]

Using the Terminal

Open a Terminal window, type or paste in this command, and then press enter:

```
sudo pmset -a displaysleep 0 disksleep 0 sleep 0
```

`pmset`, short for "power management settings", is the program that handles power management configuration. `-a` means you want to apply this command to all power supply situations (i.e. both while on battery and while plugged in, for laptops). The `0` following each keyword disables that form of sleeping.

This command often displays a little informational warning after it's run, which you can ignore.

To enable sleep using the Terminal, use this command:

```
sudo pmset -a displaysleep {x} disksleep {y} sleep {z}
```

Replace `{x}`, `{y}`, and `{z}` with times measured in seconds. For example, `sleep 600` will set your Mac to go to sleep after ten minutes (six hundred seconds) of inactivity.

Unfortunately, the ability to adjust screen saver settings using the Terminal does not work as of macOS Big Sur. It's not clear if that was a choice by Apple, or a mistake. It can still be done via AppleScript, though:

```
tell application "System Events" to tell screen saver preferences
    set delay interval to 0
end tell
```

Disable Automatic Time Machine Backups

Backups are wonderful. You should back up everything as often as possible. But on a computer used for your show, backups should only be done manually. Time Machine, much like Spotlight, uses indexing and background processes which can take hold of the disk at inopportune moments.

Using System Settings

Open System Settings → General → Time Machine, click the button labeled **Options...** and set "Back up frequency" to "Manually."

To restore Time Machine's default behavior, set the frequency to "Automatically Every Hour."

Using the Terminal

Open a Terminal window, type or paste in this command, and then press enter:

```
sudo tmutil disable
```

`tmutil`, short for "time machine utility", is the program which handles Time Machine configuration. Unlike its friends, it has very straightforward syntax.

To restore automatic backups, use this command:

```
sudo tmutil enable
```

Disable Software Update

You don't want your computer trying to update software in the middle of a run; if everything works and an automatic update introduces a bug or problem or even variation, it can cause a lot of last-minute stress or even a cancelled performance.

To prevent your Mac from automatically installing software updates, open System Preferences → Software Update and uncheck the box labeled "Automatically keep my Mac up to date". You'll be asked to confirm this choice.

Unfortunately, the ability to adjust Software Update settings using the Terminal does not work as of macOS Big Sur. It's not clear if that was a choice by Apple, or a mistake.

Prevent Notifications

The usefulness of the macOS notifications system can be debated, but it fortunately makes it very easy to disable all notifications at once, leaving your QLab operator in relative peace.

To prevent notifications, open System Settings → Focus, click *Do Not Disturb,* and look at the section labeled **Set a Schedule.** There, either add a schedule or edit an existing schedule to cover your whole working day, or even the whole day. You could, for example, turn on *Do Not Disturb* from 4:00 AM until 3:59 AM (all day) or from 5:00 PM until 11:00 PM (from before pre-show call until after post-show shutdown.)

Disable Photos Processing and Cloud Services

Apple's Photos app does a huge amount of background processing, include face and object detection. The only way to make sure that Photos doesn't keep your Mac busy whenever it thinks the system is idle (that is, during standby right before a major cue), make sure the Mac isn't connected to your iCloud photo library.

1. Open Photos;
2. Choose Preferences from the Photos menu;
3. Click iCloud;
4. Uncheck the box marked "iCloud Photos".

Log Out of iCloud

Even when your Mac is offline, iCloud is surprisingly assertive about checking in with the iCloud servers in Apple's data centers. Logging out of iCloud ensures that this check-in process doesn't claim processor power when you need it.

Open System Preferences → iCloud and click "Sign Out" to sign out of iCloud.

Minimize Internet Accounts

Similarly, any accounts used to sync Mail, Contacts, and Calendars can potentially try to access the Internet and take up processing power while doing so, even while network access is disabled.

Open System Settings → Internet Accounts, choose an account, and uncheck each service type that you can do without. Repeat for each account.

Limit Internet Access

Many individual applications, including QLab, have their own internal scheme to check for updates, but as discussed above we recommend not updating anything during the run of a show.

One of the simplest ways to guarantee that automatic software updates or any other network traffic won't bother your show is to disconnect the show computer from the Internet. We absolutely do not require this, but it unless you need internet access on your show Mac for some specific reason, this is a simple and fairly foolproof way to be sure that things like this don't cause trouble.

If you use a network to connect your QLab computer to other hardware, and your show doesn't require Internet access, make sure that network is a closed LAN (local area network) and has no path to the Internet.

Show Mode and Task Switching

By default, QLab does the following when your switch your workspace into Show Mode:

- Prevents use of **⌘-Tab**
- Disables Mission Control and Exposé
- Disables automatic Dock hiding
- Prevents your computer from sleeping
- Activates "latency critical" mode, which prevents QLab from getting deprioritized access to system resources.

If you want QLab to *not* do this when in Show Mode, choose *QLab Preferences...* from the **QLab** menu and uncheck the box marked *Disable disruptive OS features in Show Mode*.

If You're Doing Video

If you're using QLab for video, the following settings are also very important:

Disable *Mirror Displays*

Whenever you have more than one display connected to a Mac (including the built-in display on a laptop or iMac), you can either have the displays mirroring each other, showing the same thing, or turn off mirroring, which lets each display show its own image. That's how you want it set for QLab, so that you can see QLab on your display, and the audience sees your cues on the other display or displays. Display mirroring is configured differently in different versions of macOS, but in any case it's done in System Settings → Displays or System Preferences → Displays.

Disable *Displays have separate Spaces*

Spaces is Apple's name for virtual desktops. If you don't know what this means, don't worry about it; the main purpose of Spaces is irrelevant to QLab, but they have a side effect that is important for video users: if your displays are set to have separate spaces, the Menu bar also appears on all displays, which means it will be visible to your audience when no cues are playing through QLab.

To fix this, open System Settings → Desktop & Dock, then look under the **Mission Control** heading. Set *Displays have separate Spaces* to off.

Disable separate Spaces using the Terminal

Open a Terminal window, type or paste in this command, and then press enter:

```
defaults write com.apple.spaces spans-displays -bool TRUE
```

To restore separate Spaces using the Terminal

```
defaults write com.apple.spaces spans-displays -bool FALSE
```

Important: you'll need to log out, then back in again for this to take effect. This is true whether you use System Preference or the Terminal command to make the change.

Blackout the Desktop

When QLab is playing a Video cue, it places a black "backdrop" over any screen that is being used by that Video cue. When no video is playing, however, QLab does not display this backdrop. Therefore, in order to prevent your audience from seeing anything when no Video cue is playing, you'll need to set the desktop background on your projector (or other audience-visible display) to black. You can do that in two ways. Either:

1. Open System Settings;
2. Choose Desktop & Screen Saver;
3. Choose Desktop;
4. On your projector (or other display), choose "Solid Colors";
5. Click "Custom Color…";
6. Set the color to black.

Alternately, QLab provides a quick and easy way to do the same thing. Simply choose *Black out desktop backgrounds* from the **Tools** menu, and all desktop backgrounds will be set to black. You can later choose *Restore saved desktop backgrounds*, also from the **Tools** menu, to restore the desktop backgrounds you had previously.

Disable the Orange Dot

Recent versions of macOS have what Apple calls a "privacy indicator" in the menu bar, which is an orange or green dot which appears in the menu bar to indicate when a program on your Mac is using the microphone (orange) or camera (green.) This is a great security feature since it makes it very difficult for a program to spy on you using your microphone or camera without you knowing, but one aspect of the feature is very troublesome to QLab users: the

dot appears on *all* displays not just the main display. Fortunately, as of macOS Sonoma version 14.4, you can limit the privacy indicator to only show on the main screen, which is presumably the screen your QLab operator is looking at.

To disable the privacy indicator on secondary screens:

1. Start up your Mac in recovery mode (not as scary as it sounds! Don't worry!) Here are Apple's directions for Apple Silicon-based Macs and here are the directions for Intel-based Macs.
2. Once in recovery mode, choose *Terminal* from the **Utilities** menu.
3. Enter this command: `system-override suppress-sw-camera-indication-on-external-displays=on`
4. Quit Terminal.
5. Chose *Restart* from the **Apple** menu. Your Mac will restart in normal mode.
6. Once your Mac has restarted, go to System Settings → Privacy & Security → Microphone. There will be an option to turn off the privacy indicator. Turn it off.
7. Go to System Settings → Privacy & Security → Camera. There will be an option to turn off the privacy indicator. Turn it off.

To be clear: there is no meaningful security risk introduced by this change, since the privacy indicators will still appear on your main display.

Other Software

Other programs running in the background can interfere with QLab's performance in two basic ways: first, they can use system resources like processing power and memory capacity which would otherwise be available to QLab. Second, they can engage in activity that conflicts with QLab's activity, such as altering a file which QLab is expecting to remain unaltered.

The single biggest and simplest thing you can do to ensure good QLab performance is to avoid running other programs on the same Mac as QLab during your show, besides programs which are necessary for your show.

Some programs are more... shall we say *assertive* about their background behavior than others. For example, Ableton Live is a very processor-intensive program, but only when it's running. Simply having Live installed on the same Mac as QLab isn't a problem at all; if you're not using Live, it has no real effect on your computer. Adobe Creative Cloud, on the other hand, has not one but *several* helper programs which run all the time to keep the Adobe apps updated, sync data, facilitate logging into the Adobe CC website, and so forth. These processes are designed to have minimal impact, but minimal is not the same as zero and when you're trying to get the best possible performance out of a computer, every little bit helps.

The Upshot

Quit programs that aren't necessary during show time, disable or uninstall background processes that run all the time, and learn about the background behavior of the programs you do use so that you're aware of what they're doing when you're not actively using them.

Avoid installing these programs on any audience-facing QLab system:

- Any Adobe Creative Cloud program
- Google Chrome, Google Earth, and Google SketchUp
- LogMeIn, AnyDesk, TeamViewer, or other remote screen sharing software
- Backblaze or other whole-disk automatic backup software
- Dropbox, Box, Google Drive, or other automatic cloud file sharing software
- Vectorworks Cloud Services
- Antivirus software of any kind

To be clear, none of this should be read as an indictment of any of these programs, per se[2]. Mac computers are used for many different things, and different configurations of software are appropriate for different environments. A setup that works perfectly for one environment might be abysmal for another environment. That doesn't mean there's a problem with that setup.

Squeezing Every Last Drop of Performance

Sometimes we don't get the budget we hoped for. Sometimes we are stuck using a Mac that was donated by a board member's uncle's friend's former roommate. Sometimes the scope of the design outgrows the Mac that was fast enough for the original plan. These things happen! When they do, sometimes we need to dig as deep as we can and look for every 0.01 percent of processing power and memory availability to get our shows running smoothly.

The following settings are not recommended in general because they have a truly minimal effect on the overall processing load of your Mac. But when you're really down to it, these are things you can try when you really are looking for that 0.01 percent.

Avoid MP3 and MP4/AAC files

Playing MP3 and MP4/AAC files takes more processing power than playing AIFF, WAV, or CAF audio files.

Match sample rate and bit depth

Technically, all audio needs to match the sample rate and bit depth of the output device that it's playing through. Core Audio does an amazing job of resampling audio on the fly with essentially no processing overhead, but *not* resampling on the fly takes even *less* processing overhead.

Match resolution

Scaling video cues takes processing power. Render your video files at the exact size you need them wherever possible.

Use native video outputs

Outputting video via Blackmagic devices, Syphon, and NDI takes more processing power than using native video outputs. A native video output is one which appears in System Settings → Displays. The difference is smaller on Apple Silicon-based Macs than Intel-based Macs, but there's still a difference.

Clear the Desktop

Each icon on the Mac's Desktop requires processing power to display, believe it or not. Get as many items off of the Desktop as you can.

Reduce Motion and Transparency

macOS uses lovely motion and transparency effects for aesthetic effect, both of which takes processing power. Open System Settings → Accessibility, select *Display*, and turn on *Reduce motion* and *Reduce transparency*.

---

1. If your Mac only has SSDs attached to it, this checkbox has no effect.↩
2. A notable exception: all antivirus software for the Mac is bad, and you should never use it. macOS has built-in security processes which do a better job, cost nothing, don't slow down your Mac, and don't try to scare you about invisible and unlikely threats.↩

Keyboard Shortcuts

QLab and its manual are written in American English using a US-standard QWERTY keyboard layout. Apple sells Macs with a wide variety of keyboard layouts for many different languages, alphabets, and cultures. There is, unfortunately, no simple way to plan for keyboard shortcuts which transcend these boundaries. If you are using a keyboard layout other than US QWERTY, and/or a language other than US English on your Mac, you may find that some of the keyboard shortcuts do not work as listed. You may find that the key in the corresponding position as the listed key on a US keyboard will work, but will simply have the wrong label. We did not cause this problem, and we cannot solve it, but we are nevertheless sorry if it causes you trouble.

Key Symbols

Symbol	Meaning
⌘	command
⇧	shift
^	control
⌥	option
⌫	delete

Default Keyboard Controls

Most of these keyboard shortcuts can be edited in the Controls → Keyboard section of Workspace Settings.

Command	Key	Note
GO	space	When the workspace is set to always audition, this keyboard shortcut invokes Audition GO instead of GO.
Preview Selected	V	When the workspace is set to always audition, this keyboard shortcut invokes Audition Preview Selected instead of Preview Selected.
Audition GO	⌥space	
Audition Preview Selected	⌥V	
Load Last Selected	L	If multiple cues are selected, only the most recently selected cue will be loaded. The playhead will also jump to that cue.
Panic Selected	S	Double-tap this keyboard shortcut to hard stop the selected cues.
Pause/Resume Selected	P	
Pause All	[	
Resume All	]	
Panic All	escape	Double-tap this keyboard shortcut to hard stop all cues. This shortcut cannot be changed.
Edit cue number	N	
Edit cue name	Q	
Edit cue notes	O	
Edit cue target	T	
Cycle cue target type	⌥T	Only relevant to Fade cues and Reset cues.
Edit cue pre-wait duration	E	
Edit cue action duration	D	
Edit cue post-wait duration	W	
Cycle cue continue mode	C	

Command	Key	Note
Flag/unflag	F	

Menu Keyboard Shortcuts

QLab

Command	Key
Hide QLab	⌘H
Hide Others	⌥⌘H
Quit QLab	⌘Q

File

Command	Key	Note
New Workspace	⌘N	*
New From Template...	⇧⌘N	*
Open Workspace...	⌘O	
Connect to Workspace...	⇧⌘K	
Close	⌘W	
Save	⌘S	
Save As...	⇧⌘S	
Workspace Settings Window	⌘,	

* These keyboard shortcuts can be co-opted by the Light Dashboard and by some sections of the Workspace Settings window. Co-opted shortcuts are always noted visibly on screen when this is the case.

Edit

Command	Key	Note
Undo	⌘Z	
Redo	⇧⌘Z	
Cut	⌘X	
Copy	⌘C	
Paste	⌘V	
Paste Cue Properties...	⇧⌘V	
Paste and Match Style	⌥⇧⌘V	
Delete	⌘⌫	*
Select All	⌘A	
Find...	⌘F	
Find Next	⌘G	*
Find Previous	⇧⌘G	
Show Fonts	⌥⇧⌘T	
Bigger	⌘+	

Command	Key	Note
Smaller	⌘-	
Show Colors	⌥⇧⌘C	
Copy Style	⌥⌘C	
Paste Style	⌥⌘V	

* These keyboard shortcuts can be co-opted by the Workspace Settings window. Co-opted shortcuts are always noted visibly on screen when this is the case.

Cues

Note: You can reassign keyboard shortcuts for cues by re-arranging them in the toolbox.

Command	Key
Group	⌘0
Audio	⌘1
Mic	⌘2
Video	⌘3
Camera	⌘4
Text	⌘5
Light	⌘6
Fade	⌘7
Network	⌘8
MIDI	⌘9

Tools

Command	Key	Note
Load to time...	⌘T	
Renumber selected cues...	⌘R	*
Delete numbers of selected cues...	⌘D	
Jump to selected cues' targets	⇧⌘J	
Turn on/off always audition	⇧⌘A	
Turn on/off live fade preview	⇧⌘P	

* This keyboard shortcut can be co-opted by the Light Dashboard window. Co-opted shortcuts are always noted visibly on screen when this is the case.

When a Fade cue is selected, the following Tools are also available:

Command	Key

Command	Key
Set Audio Levels From Target	⇧⌘T
Set Video Geometry From Target	^⌥⌘V
Revert Fade Action	⇧⌘R

When the Light Dashboard is the front-most window, the following Tools are available instead of the standard ones:

Command	Key
New Cue with Changes	⌘N
New Cue with All	⇧⌘N
Update Latest Cue	⌘U
Update Selected Cue(s)	⇧⌘U
Update Originating Cue(s)	⌥⌘U
Revert Changes	⌘R

View

Command	Key	Note
Enter/Exit Full Screen	⇧⌘F	
Inspector	⌘I	*
Inspector for selected cue	⇧⌘I	
Toolbox	⌘K	
Lists / Carts & Active Cues	⌘L	
Toggle Between Lists / Carts & Active Cues	⇧⌘L	
Warnings	⌘B	
Select cue...	⌘J	Within the Select cue panel, ⌘P selects the cue currently at the playhead.
Select next	⌘↓	
Select previous	⌘↑	
Select next tab	⌘→	
Select previous tab	⌘←	
Move playhead to cue...	⇧⌘J	
Move playhead to next cue	⇧⌘↓	
Move playhead to previous cue	⇧⌘↑	
Move playhead to next sequence	=	
Move playhead to previous sequence	-	
Move selected cues down	⌥⌘↓	
Move selected cues up	⌥⌘↑	
Enter Edit Mode	⇧⌘[	
Enter Show Mode	⇧⌘]	

* This keyboard shortcut can be co-opted by the Workspace Status window. Co-opted shortcuts are always noted visibly on screen when this is the case.

Window

Command	Key
Minimize	⌘M
Workspace Status	⇧⌘W
Override Controls	⇧⌘O
Light Dashboard	⇧⌘D

While in the cue list...

Command	Key
Expand all Group cues	>
Collapse all Group cues	<

While focused on the waveform view of the Time & Loops tab...

Command	Key
Set start time to current time	⇧I
Set end time to current time	⇧O
Add slice at current time	M
Zoom in	⌘+ / ⌘=
Zoom out	⌘-

While editing the integrated fade curve in the Time & Loops tab...

Command	Key
Select previous control point	←
Select next control point	→
Move selected control point up	⌥↑
Move selected control point down	⌥↓
Move selected control point left	⌥←
Move selected control point right	⌥→
Move selected control point up a small amount	⇧⌥↑
Move selected control point down a small amount	⇧⌥↓
Move selected control point left a small amount	⇧⌥←
Move selected control point right a small amount	⇧⌥→

While focused on the Timeline tab of Timeline Group cues...

Command	Key
Zoom in horizontally	⌘=
Zoom out horizontally	⌘-
Zoom in vertically	⇧⌘=
Zoom out vertically	⇧⌘-
Nudge selected cues +0.1 second	⌥→
Nudge selected cues -0.1 second	⌥←
Nudge selected cues +0.01 second	⇧⌥→
Nudge selected cues -0.01 second	⇧⌥←

While working in a 2D canvas...

These keyboard shortcuts and modifiers are relevant when working with ❋ Network cues and Fade cues with 2D paths, editing audio object maps, and working in the Objects inspector tab.

Command	Key
Hold to toggle the active tool.	^
Hold to disable snapping.	⌘
Hold to reverse the direction of the rotate tool.	⌥
Zoom in and out.	⌥ + scroll
Select adjacent point.	← / →
Nudge single selected point.	⌥ + ← / ↑ / ↓ / →
Nudge multiple selected points.	← / ↑ / ↓ / →
Coarse adjustments.	⇧ + nudge
Lock to single axis in 90 degree increments.	⇧ + drag
Lock to aspect ratio.	⇧ + resize
Scale from center.	⌥ + resize
Select all points.	⌘A
Insert a point on a path.	⌥ + click
Draw a circle from the center.	⌥ + circle tool

Licenses

QLab 5 uses an account-based licensing system which allows you to install and remove licenses quickly and easily without having to keep track of license files. Licenses can be installed directly on a Mac that's connected to the internet, or they can be installed to an offline Mac, or installed onto a USB drive for portable use.

No matter how a license is installed, QLab does not need internet access to use it. A Mac running a licensed copy of QLab does not need to be connected to the internet.

Types Of Licenses

There are several types of licenses available for QLab, although many of those types are more or less the same from a functional perspective and only differ in terms of their pricing structure and the number of computers they can be used on at once.

Functional types of licenses

- An **audio** license unlocks pro-level audio features,
- A **video** license unlocks pro-level video features,
- A **lighting** license unlocks the ability to patch more than 16 DMX addresses.
- **Each of the three license types** unlocks MIDI, networking, show control, scripting, and workflow features.

Licenses are sold individually, in pairs, or as a bundle of all three types. Pairs and bundles give you a discount, but otherwise don't differ from licenses purchased individually.

You can read more about the features unlocked by each license type in the Features by License section of this manual.

Structural types of licenses

- **Standard** licenses are one-time purchases for a fixed price that can be installed on up to two computers at once. These licenses can be used in perpetuity, as long as you have a Mac capable of running QLab 5.
- **Rent-to-own** licenses are one-time purchases priced *per day,* with a start and end date of your choosing. Within that date range, they operate exactly the same way as standard licenses and can be installed on up to two computers at once. After the end date passes, they self-deactivate.
- **Site** licenses are like standard licenses, except they are priced *per activation* with a ten-activation minimum. They are intended to make it easier to manage large numbers of QLab activations, for example in a large production studio, rental shop, or multi-venue facility. Site licenses are only available upon request; contact support@figure53.com for more information.

We also offer discounts and special license types to educational users. You can find policy and technical details about educational licenses at https://qlab.app/educational-pricing.

Understanding activations

Most types of QLab 5 licenses can be activated on two computers at once, more or less without limitation. You can use your two activations for a main and a backup, a main stage and second stage, a theater and a rehearsal hall, or however else you like.

If you bought a pair or bundle of licenses, each type of license within your purchase functions individually, so you can put the audio, video, and lighting portion of a bundle license onto two computers *each,* individually.

Purchasing Licenses

You can purchase QLab licenses on our secure online store using a credit card, debit card, or PayPal. Once you've chosen the license or licenses that you wish to purchase, click *Checkout*. You can then log in to your QLab account if you already have one, or just enter your billing details if you don't. It's important to double-check your email address before completing your purchase.

If you do not already have an account, completing the purchase will create an account for you, log you in, and prompt you to create a password for the account. You will use this password and your email address in QLab to install your licenses.

Exchanging Licenses for Trade-in Discount During Purchase

Any standard QLab 5 license can be used in exchange for a trade-in discount. Upon checkout, trading in a license deactivates that license wherever it is currently installed and removes it from your account. The trade-in value is equal to the amount that you paid for the license, and may only be used during that purchase. For example:

- *I bought an audio license, then discovered I need a video license too. I wish I had bought them together to get the reduced price!* You can trade in the audio license, then use the trade-in discount to help pay for a new audio+video license. You'll pay only the difference in price, which means in the end you'll have paid the same amount as if you had bought the audio+video license at the beginning.
- *I bought a video license, but my needs have changed and I no longer need it. Now I need an audio license!* You can trade in the video license and use the trade-in discount to buy a new audio license. Since those two licenses cost the same thing, you end up paying nothing extra.

Store credit and trade-in discounts can only be used to purchase standard licenses, not rent-to-own licenses.

The relationship between store credit and site licenses is a bit more complex; please contact support@figure53.com if you'd like to discuss trading in site licenses or buying site licenses using store credit. Both can be done, but there's no automatic process for it.

Rent-To-Own licenses

Rent-to-own licenses are purchased on a day by day basis. When you purchase one, you choose the start date and the end date, and the license will be valid for that period of time. It's important to note that a one-day rental isn't just 24 hours starting from the time of purchase; the actual calendar date matters. If you purchase a one-day rental license with a start date of today, it will remain valid until midnight tonight.

The Figure 53 shop uses your computer's clock to determine the timezone for your rental license.

You can purchase and install rental licenses ahead of time, choosing a start date in the future. On the appropriate day, the license will automatically "wake up" and unlock the appropriate features. The first time that QLab launches after the rental period is over, the license will simply stop working on its own. You do not need to stop or cancel a rental. Rentals are one-time purchases, not subscriptions.

The reason we call this type of license "rent-to-own" is that every dollar that you spend on a rental license becomes store credit. Each day during the span of a rental license, at midnight, you will receive store credit equal to the value of rental for the day that just ended. You can use this credit at any point; if you rent a license for two days and then decide to buy it, you'll get those two days' worth of credit to use. If you rent a license for so many days that you spent the same amount on rentals as you would have on a standard license, you can get that standard license for no extra cost. If you'd like to end an ongoing rental license early and exchange its value for store credit even before the time has elapsed, you can do that on your Account page.

Upgrade Pricing

QLab 4 licenses can also be exchanged to purchase QLab 5.

Trading in a QLab 4 license is just like trading in a QLab 5 license. Trading in a license deactivates that license wherever it is currently installed, removes it from your account, and adds the appropriate amount of store credit to your account.

Go to the Upgrade to v5 Page to get started.

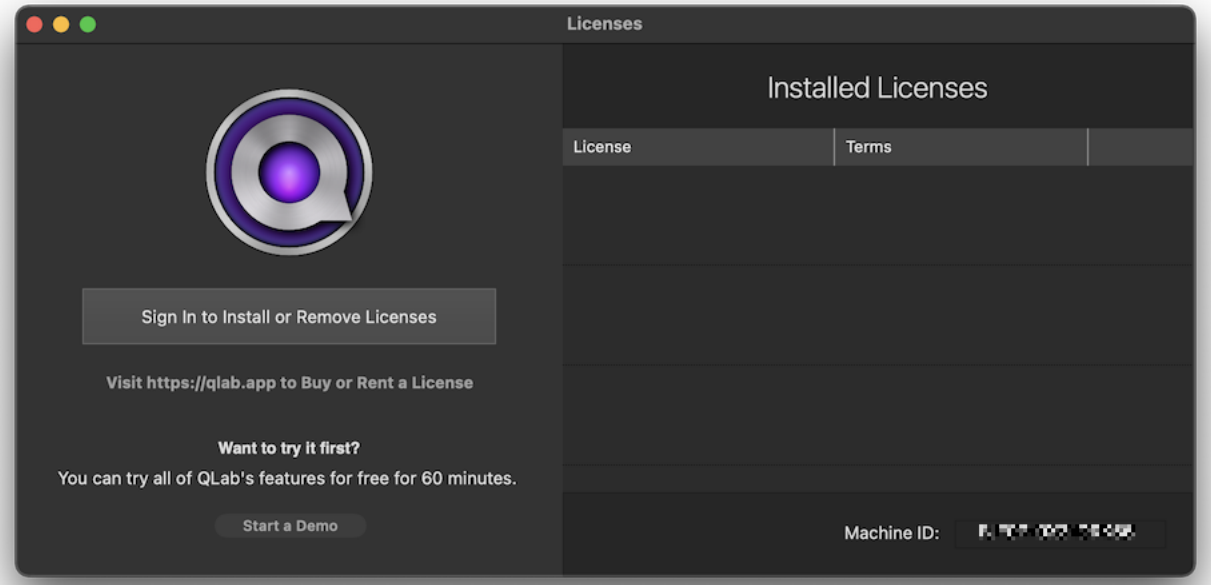
License	Trade-in value
QLab 4 audio, video, or lighting	$299
QLab 4 "pick two"	$599
QLab 4 pro bundle	$749

For more details on pricing, see the FAQ on QLab 4 upgrades.

And of course you can always email support@figure53.com to ask questions or get help.

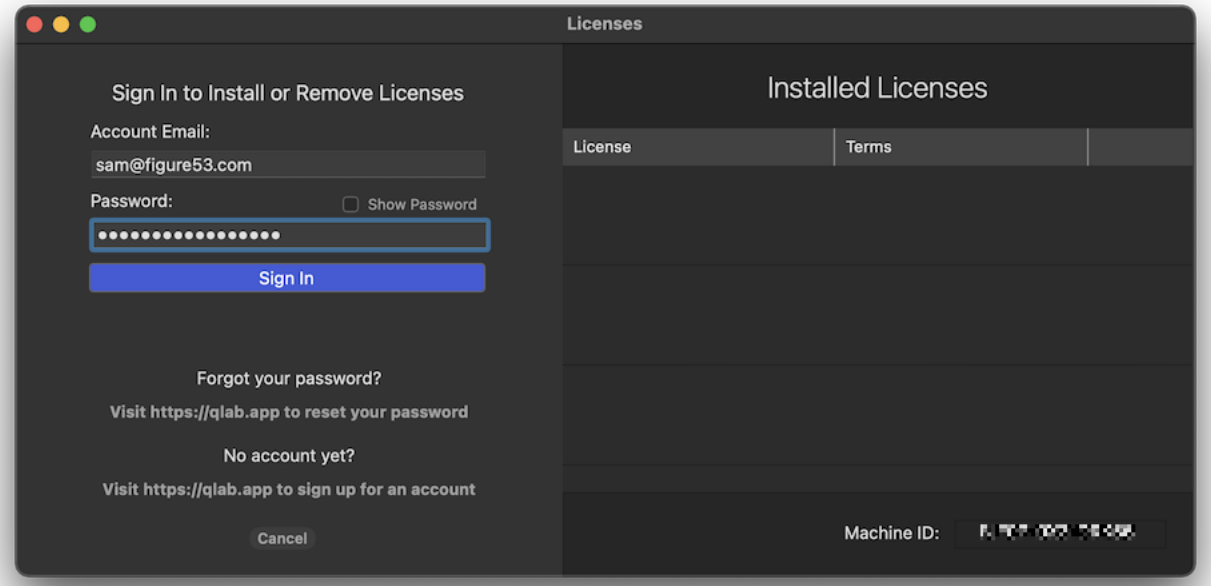
Installing Licenses

To install a license, open QLab and choose *Manage Your Licenses...* from the **QLab menu** (or click on the **Licenses** button in the Launcher Window) to open the License Manager.



Without signing in, you'll be able to see what license or licenses are installed, see the Machine ID of the Mac (used for remote activation, discussed below), and start demo mode.

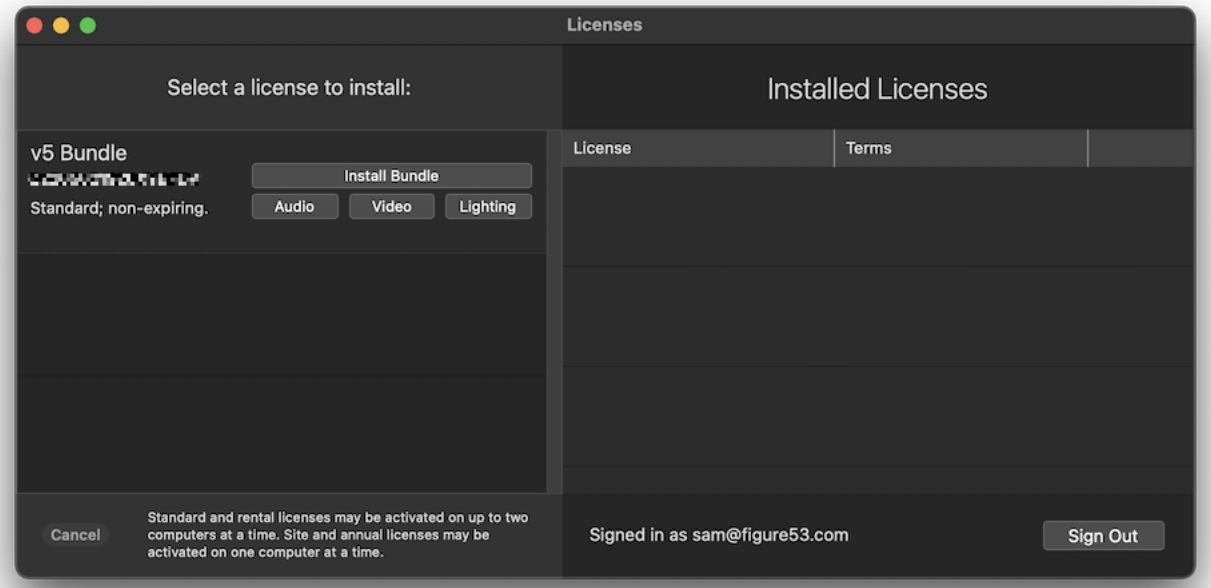
After clicking **Sign In to Install or Remove Licenses** you will be able to enter the email address and password for your QLab account.



In case you've forgotten your password, or not set a password for your account, or not yet created an account, there are clickable links for you to follow to address these needs.

Select a license

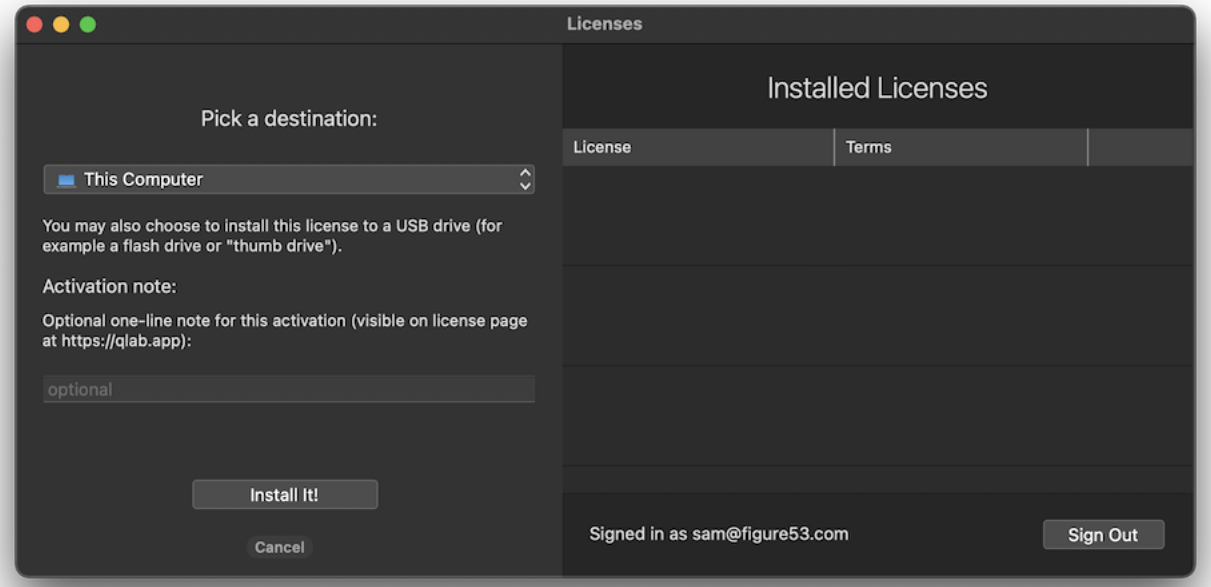
Once you're signed in, the left side of the window shows you a list of the licenses in your account.



To install a license, click its button. Greyed-out buttons represent licenses that cannot be installed, usually because they are already installed on two other Macs.

Select a destination

Once you've selected a license, you'll be presented with a choice of destination:

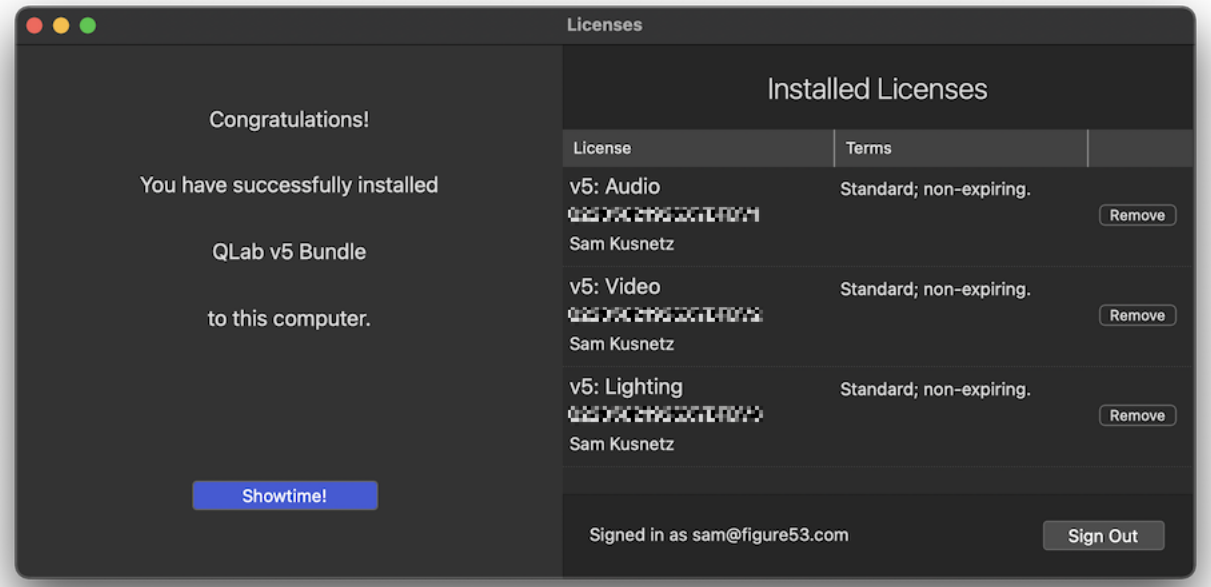


If you choose *This Computer*, the license will be installed onto your Mac, and will be accessible to all user accounts on that Mac. This is the most common way to install a license.

Alternately, you can install the license seat onto a connected USB drive. To learn more about this, skip down to the section entitled USB Licenses.

If you choose *This Computer*, you can optionally enter a single-line note to go along with the installation. This note can be seen when viewing the details of your licenses on your account page at https://qlab.app/account.

Click **Install It!** to complete the installation process. The License Manager will then show you the completed installation.



Once the license is installed, you can disconnect your Mac from the internet if you like. Licenses do not need to "phone home" except during installation and removal.

USB Licenses

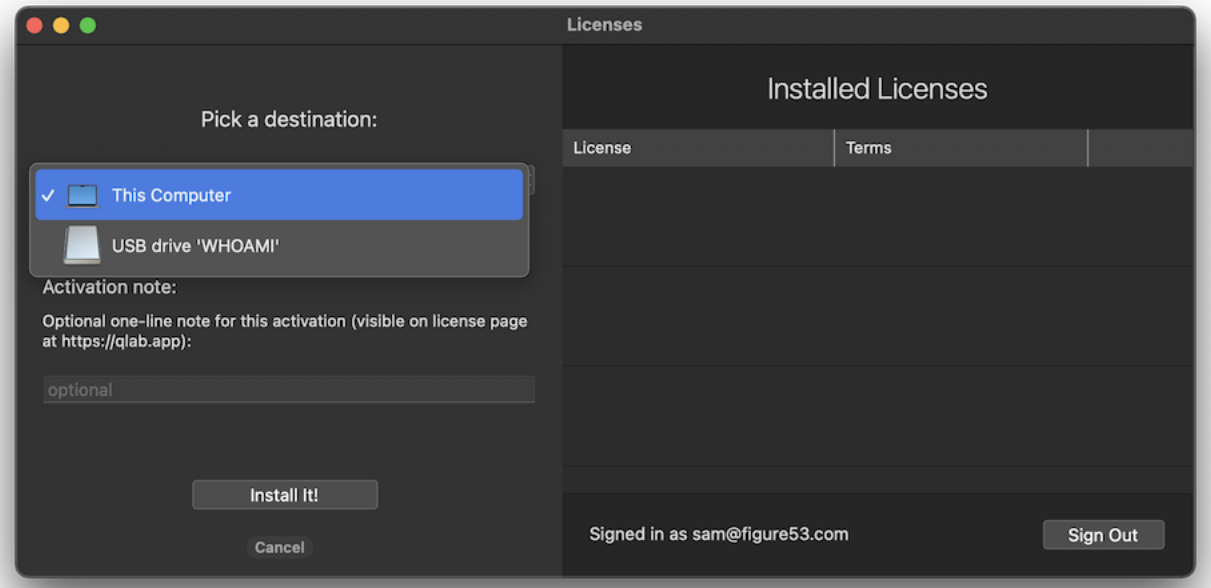
Installing a license on to a USB drive makes it easy to loan a QLab license activation to a short-term collaborator, use licenses with Macs that aren't connected to the internet, or be ready with a license when you're not sure what Mac you'll be using.

Because not every USB device manufacturer adheres to the same standards, only USB drives made by the following manufacturers can be used for QLab licenses. This restriction makes it possible for us to provide the highest possible guarantee of stability for USB licenses without resorting to using something like an iLok or Sentinel key.

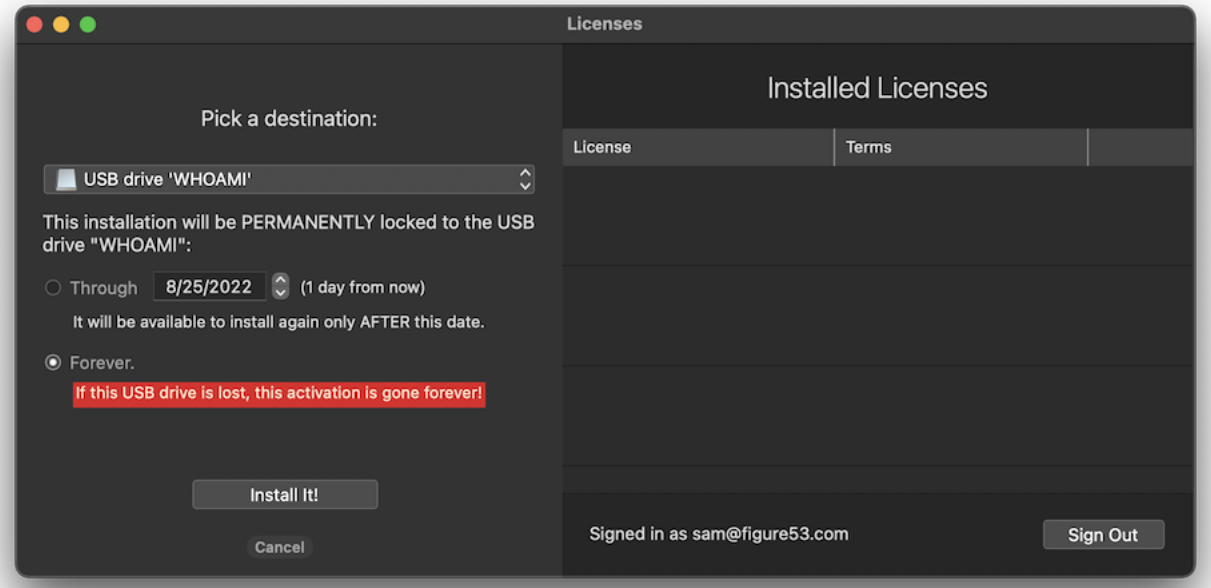
- Corsair
- HGST
- HP
- Kingston
- LaCie
- Lexar
- OCZ
- SanDisk

We have received some reports of USB drives not working even when they appear to be made by one of these manufacturers. In some cases, this may be due to the relatively rampant counterfeiting that exists in the world of USB drives. In other cases it may be a resolvable minor issue. In all cases, as always, contact support@figure53.com if you have any concerns.

To install a license to a USB drive, your Mac must be connected to the internet and have QLab installed. Follow the first several steps above for installing a license, but when you get to the step of selecting a destination, choose an attached, compatible USB device from the drop-down menu.



Once you choose a USB device, you'll be given a choice between installing the seat temporarily or permanently. Choosing a temporary installation lets you select an expiration date after which the license seat will automatically deactivate itself on the USB drive and become available for you to install once more. Please note that this is entirely separate from the expiration date of a rental license; you cannot install a rental license onto a USB drive and set an expiration date that's after the expiration date of the license itself.



It is not possible to remove a license from a USB key before the expiration date. Once the license is on the key, it will remain there until the expiration date. Please be ready to wait!

Lost, Stolen, Erased, and Broken USB Devices

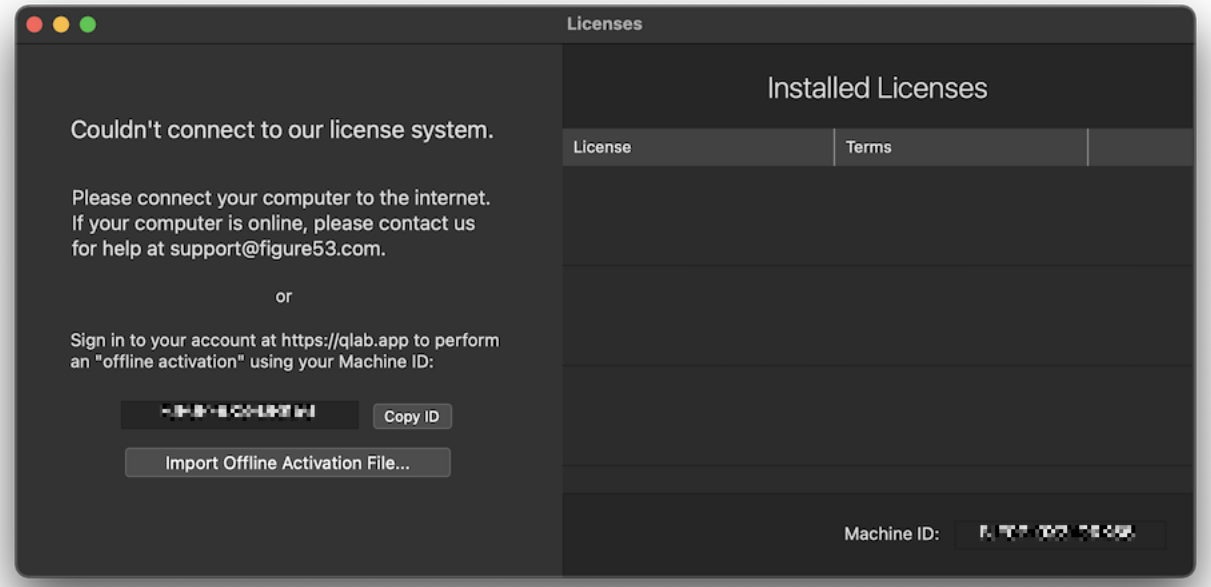
If the USB drive is lost, stolen, erased, or broken, please contact support@figure53.com for more information.

Remote Activation

If your Mac cannot be connected to the internet, maybe because you're on a boat or your IT department is grouchy, and you cannot or do not wish to use USB licenses, you can install a QLab license by using remote activation. You'll need to have QLab installed on the offline Mac in order to begin.

You could also use this process to install a license onto a Mac that *is* connected to the internet as well, if that's useful to you in any way.

To perform a remote activation, you'll need the Machine ID for the Mac that you want to use. You can find the Machine ID in the License Manager window, which looks a bit different when your Mac is offline, or online but unable to contact the license server:



The alphanumeric code near the bottom of the window is the Machine ID, which is a QLab-only identification code unique to each Mac. You can also find your Mac's Machine ID by opening a workspace, choosing *Workspace Status* from the **Window menu**, and clicking on the Info tab.

You can click **Copy ID** to copy that code for easy pasting to other places.

Do It Yourself

If you have access to another computer that's online, you can do a remote activation by yourself. Once you've got the Machine ID for the QLab Mac, log into your QLab account at https://qlab.app/account and find the license you want to install listed under the heading **Licenses**.

When you've found the license you want to install, click *View Details* next to that license, and you'll be taken to a page showing the details of that license.

Find the license you want to install, and click *Remotely Activate* in the **Actions** column. You'll be asked to enter the Machine ID of the Mac, then click *Activate*. The web page will refresh, and then you'll see a link in the **Actions** column that says *Download Activation File*. Click to download the file, and then copy that file over to the Mac. You can then double-click on it or drag it onto QLab's icon to install it.

Phone A Friend

If you cannot or do not want to go through that process yourself, we are happy to do it for you. Copy the Machine ID of the Mac you want to use, send us an email at support@figure53.com. Paste in the Machine ID and tell us which license you want to install on this Mac. For example, you might say "I want to install the audio license of my bundle" or you might say "I want to install all three licenses of my bundle."

When we receive your message, we'll use the code to generate an activation file, which we will email back to you. When you receive that email, copy the file to the Mac using a local network or a USB drive or some similar method. You can then double-click on it or drag it onto QLab's icon to install it.

Removing Licenses

To remove a license installed on your Mac, first make sure the Mac is connected to the internet and then open QLab, choose *Manage Your Licenses...* from the **QLab menu**, and click the **Remove** button next to the license you wish to remove. The **Remove** button is only available when you're logged into the account that owns the license. If the button is not visible, log in to the account that owns the license to make it appear.

If you need to remove a license from a Mac which has been broken, lost, stolen, or erased, or if you need to remove a license belonging to an account that you cannot access, please write to support@figure53.com and we will help you out, no problem.

Remote Deactivation

If you are not able to physically access a Mac that has your license installed, you can remotely deactivate the license which makes it immediately available to install on a different Mac.

To do this, log into your QLab account at https://qlab.app/account and find the license you want to remotely deactivate listed under the heading **License Overview**.

When you've found the license you want to remotely deactivate, click **View Details** next to that license, and you'll be taken to a page showing the details of that license.

In the **Actions** column, click *Deactivate*. You'll asked to confirm, and once you do the license seat will be deactivated and ready to install on another Mac.

The next time QLab launches on the Mac that had the license installed, the now-deactivated license seat will be automatically uninstalled.

Using QLab 5 licenses with QLab 4

You can use any QLab 5 license with QLab 4.7 or newer. License activation happens on a per-computer basis, so using both QLab 4.7 and QLab 5 on the same Mac only counts as a single activation. Please note that using a QLab 5 license with QLab 4.7 does not enable any of the 5.x features... you'll need to use QLab 5 to get access to those.

QLab 5 licenses cannot be used with QLab 3.

Features by License

QLab 5 is a free program, with additional optional features that can be unlocked by purchasing and installing a license. This chart shows which of the major features are enabled by each type of license.

Demo Mode

You can experiment with the advanced features of QLab 5 for free by choosing *Start Demo Mode...* from the **QLab** menu. This will create a new workspace in demo mode. While demo mode is active, all of QLab's licensed features are available, but copying cues and settings into the demo workspace is disabled.

Demo mode will expire after 60 minutes. After demo mode expires, or after the workspace is saved, closed, and reopened, licensed features will once again require a license.

Features

Feature	Free	Audio	Video	Lighting
Audio features				
Channels of audio output	2	128	2	2
Channels of audio per file	2	24	2	2
Audio waveform view	✓	✓	✓	✓
Unlimited slices per Audio or Video cue	✓	✓	✓	✓
Sample-accurate playback sync	✓	✓	✓	✓
Audio fades	✓	✓	✓	✓
Audio playback rate fading		✓		
Edit audio output patch routing		✓		
Edit audio output patch channel names		✓		
Edit audio input channel names per cue	✓	✓	✓	✓
Output mute and solo buttons	✓	✓	✓	✓
Audio effects on cues		✓		
Audio effects on cue outputs		✓		
Audio effects on device outputs		✓		
Audio effects fading		✓		
Audio output patch fading		✓		
Mic cues		✓		
Object audio		✓		
Video features				
Single-output video stages	1	1	unlimited	1
Multi-output video stages	0	0	unlimited	0
1000 video layers	✓	✓	✓	✓
Full-stage Video cues	✓	✓	✓	✓

Feature	Free	Audio	Video	Lighting
Custom geometry Video cues			✓	
Video fades			✓	
Video playback rate fading			✓	
Masking & edge blending			✓	
Video output warping & keystone correction			✓	
Video effects			✓	
Per-cue blend modes			✓	
Syphon input & output			✓	
NDI input & output			✓	
Blackmagic device input & output			✓	
Camera cues			✓	
Text cues			✓	
Lighting features				
Patchable DMX addresses	16	16	16	unlimited[1]
Addressable universes	unlimited	unlimited	unlimited	unlimited
RGB+, CMY, and HSI graphical color pickers	✓	✓	✓	✓
Build your own instrument definitions	✓	✓	✓	✓
Networking & Show Control features				
Remote control via OSC, MIDI, MSC, iOS app	✓	✓	✓	✓
Act as Collaboration Primary	✓[2]	✓	✓	✓
Act as Collaboration Remote	✓[3]	✓	✓	✓
Network cues		✓	✓	✓
MIDI cues		✓	✓	✓
MIDI File cues		✓	✓	✓
Timecode cues		✓	✓	✓
Timecode triggers		✓	✓	✓
Show control broadcast via MSC and OSC	✓	✓	✓	✓
Workflow & programming features				
Automatic workspace backups	✓	✓	✓	✓
Media logging	✓	✓	✓	✓
Unlimited cue lists	✓	✓	✓	✓
Unlimited cue carts	✓	✓	✓	✓
Live fade previews	✓	✓	✓	✓
Pause cues		✓	✓	✓
Devamp cues		✓	✓	✓
Target cues		✓	✓	✓
Arm & Disarm cues		✓	✓	✓
Script cues		✓	✓	✓
Workspace settings import & export		✓	✓	✓

Feature	Free	Audio	Video	Lighting
Fade patch parameters		✓	✓	✓

1. Primary workspaces on Mac with QLab licenses can accept connections from multiple Remotes, but only one Remote without a license can connect with full access. Additional Remotes without licenses can connect, but in view-only mode. You can learn more about this from the Collaboration section of this manual.↩
2. The behavior of Remotes depends upon the license(s) installed on the Primary. You can learn more about this from the Collaboration section of this manual.↩
3. Theoretical. The actual limits are based on the capabilities of your Mac and the rest of your hardware setup.↩

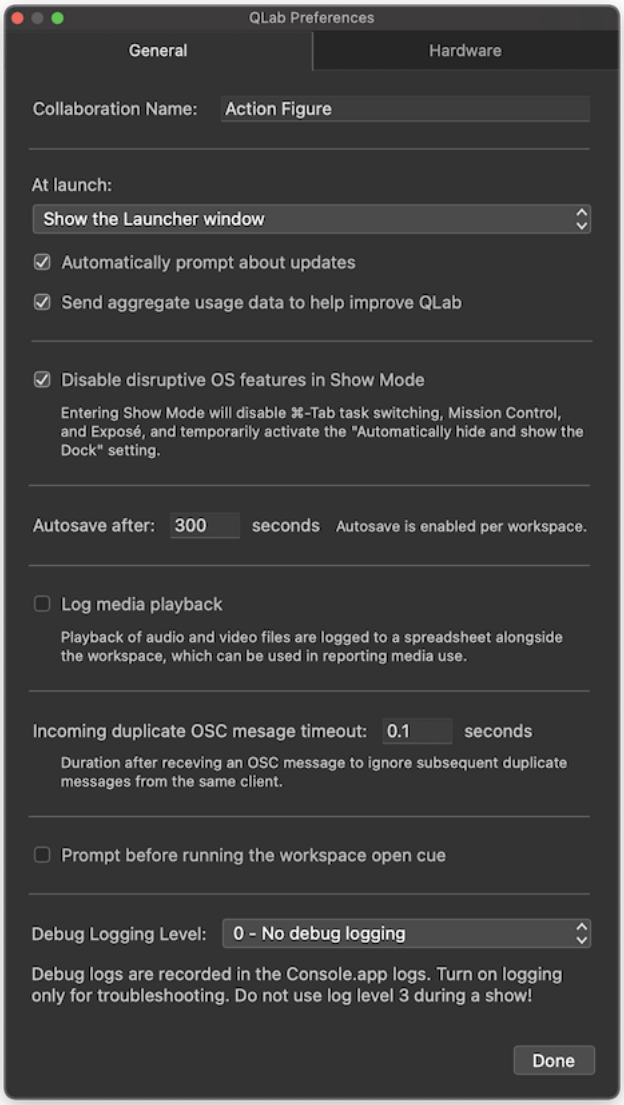
QLab Preferences

You can set application-wide preferences for QLab by choosing *QLab Preferences...* from the **QLab menu**. This menu is really called the **Application menu** and its name changes based on which program you're using. Some items in this menu are the same for all programs, such as *Show*, *Hide*, and *Quit*, but some are specific to each program.

QLab Preferences apply to all workspaces on your Mac, and they do not transfer with workspaces which are copied or moved to other Macs.

QLab preferences are divided into the General tab and the Hardware tab.

The General Tab



Collaboration Name

This is the name by which this Mac will be identified when it connects as a Remote to a Collaboration Primary. By default, this is the user name of the active account on your Mac, but you can change it to any text you like.

At launch

You can choose one of five behaviors for QLab to perform when it's launched:

- **Show the Launcher Window.** Display the Launcher window, which gives you quick access to several actions that typify the beginning of a working session with QLab. You can learn more from the Launcher Window section of this manual.
- **Restore most recent workspaces.** Re-open the workspace or workspaces that were open last time QLab was running.
- **Create a new blank workspace,** using QLab's built-in blank workspace template.
- **Create a new workspace from default template.** You can learn about workspace templates and how to set a default template from the Workspace Template section of this manual.
- **Do nothing.** This is exactly what it sounds like.

Automatically prompt about updates

When this box is checked, QLab will attempt to connect to the internet and check for available updates. If an update is available, QLab will ask you if you'd like to download and install it. Updates are never automatically downloaded. If no internet connection is available, QLab simply tries again on the next launch. No error message appears, and no further attempt to check will be made until QLab is relaunched. We recommend leaving this box checked except on a show Mac during the time between dress rehearsal and closing, during which time we recommend leaving it unchecked.

Send aggregate usage data to help improve QLab

At present, this checkbox does absolutely nothing. Over time, we plan to add various measures of QLab's behavior which will help us track and solve problems. As we add these measures, we will list exactly what data is sent right here. We will never include personally identifiable data, media files, or other information proprietary to your designs.

Disable disruptive OS features in Show Mode

When this box is checked, entering Show Mode adjusts the behavior of the following OS-level features to reduce the chances of trouble during a performance:

- **App Nap.** QLab reports itself as "latency critical" so as to prevent App Nap, which is a power-saving feature that can limit how much processing power is used by a program.
- **Sleep.** QLab will prevent your computer from sleeping, even if sleeping is enabled in System Settings.
- **⌘Tab Task Switching.** Typically, typing **⌘Tab** allows you to quickly move between applications on your Mac. QLab will block access to this keyboard shortcut in order to reduce the chance of accidentally switching away from QLab, although you can still use the mouse to switch to another application.
- **Mission Control.** QLab will block hot corners, gestures, and keyboard shortcuts for invoking Spaces, Missing Control, Show Desktop, Show Dashboard, and Application Windows. These features are sometimes also referred to as **Exposé**.

Autosave interval

Autosave is enabled on a per-workspace basis, but the interval between autosaves is globally set here. You can set the interval from 5 seconds to 600 seconds (ten minutes.)

Log media playback

When this box is checked, QLab will create a CSV-formatted log of all audio and video file playback in a workspace, using information from the cue in QLab as well as metadata in the target media files. This is primarily intended to make it easy to create reports for rights-holding organizations, like ASCAP or BMI, but could be useful for any sort of logging.

Duplicate OSC message interval

When QLab receives two or more identical OSC messages within this interval, it will discard the second and subsequent messages. This helps prevent QLab from being overwhelmed by network traffic from an extremely talkative device. Sometimes, however, you may want QLab to be more responsive to repeated identical messages. Shortening this interval increases responsiveness. On fast Macs, you may not see any adverse effects. On slower Macs, it's best to keep this interval at or above the default of `0.1` seconds unless you're sure you need it to be lower.

Prompt before running the workspace open cue

When this box is checked, if a workspace is opened which is set to run a cue automatically upon opening, QLab will ask for confirmation before running that cue. This setting can help protect against unexpected events if you routinely open unfamiliar workspaces.

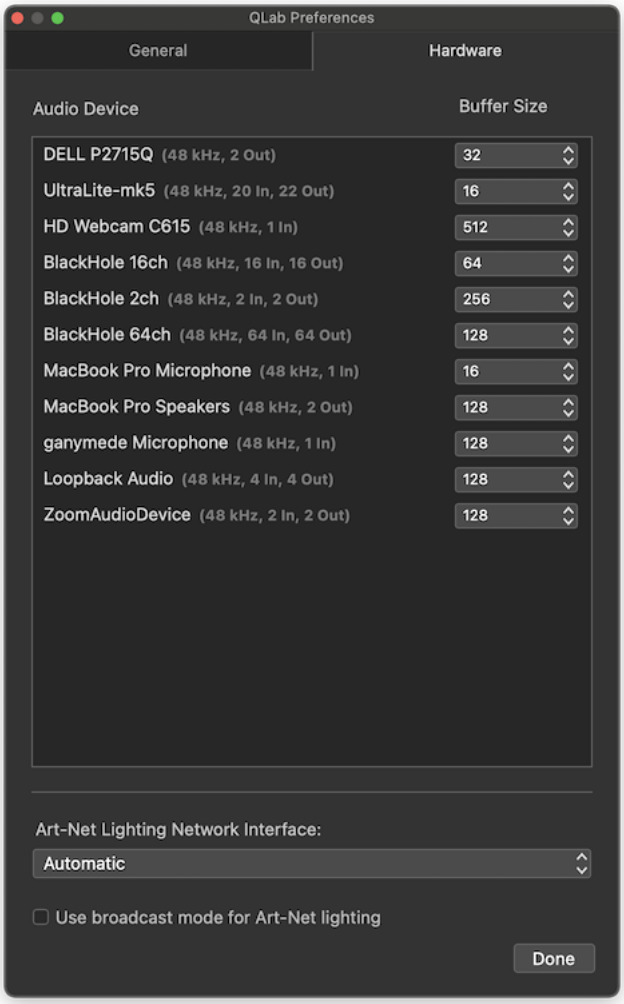
To temporarily prevent a workspace open cue from starting, hold down the control and option keys (^⌥) while opening the workspace.

Debug logging level

QLab records messages to the Mac's system log at four possible levels:

- **Level 0 - No debug logging** is the default setting. QLab logs the fewest types of events, focusing on starting up QLab, opening workspaces, making initial contact with external hardware, and reporting the most important basic errors. This is the recommended log level for most situations.
- **Level 1 - Minimal debug logging** is recommended if you need more information from QLab but still need QLab to perform as though under live show conditions. At level 1, QLab logs additional details pertaining to hardware events, OSC and MIDI messages, converting QLab 4 workspaces, opening and saving workspaces, drag-and-drop actions, and Audio cue slices.
- **Level 2 - Some debug logging** is recommended only once you've definitively identified that you're having a problem, and you're actively engaged in assessing the problem. Using log level 2 may have an impact on QLab's overall performance. At level 2, QLab logs additional details pertaining to hardware events, cue playback, overrides, sending and receiving OSC messages, opening and closing workspaces, saving workspaces, wall clock triggers, light command parsing, DMX communication, licenses and communication with the license server, using audio across multiple devices, and video playback metrics.
- **Level 3 - Verbose debug logging** is recommended only in consultation with the QLab support team. Using log level 3 will almost definitely have an impact on QLab's performance; you should avoid using level 3 in a live show situation. At level 3, QLab logs additional details pertaining to logging (meta-logging? log-ception?), settings housekeeping, licenses and communication with the license server, file management, handling timecode, handling MIDI Show Control messages, cue playback, cue triggers, and communicating with hardware.

The Hardware Tab



Audio devices

All audio devices available to QLab are listed here, showing their sample rate and channel counts for both input and output.

Each device has a pop-up menu which allows you to set the buffer size to each audio device. QLab defaults to a buffer size of 512 samples. You can manually adjust buffer sizes here. Smaller buffers reduce latency but may cause choppy playback, particularly with audio effects such as reverb and

echo. Larger buffers increase latency but can smooth out choppiness, particularly on less powerful computers.

Art-Net Lighting Network interface

You can choose a specific network interface through which QLab will send Art-Net messages, or you can leave the default *Automatic* setting to have QLab negotiate the correct interface on its own.

Automatic is the best choice most of the time. The main reason you might need to change this is if you are using QLab in a complex networking situation, in which two or more network interfaces on your Mac are connected to different networks, but the networks use the same IP addressing scheme. If none of this makes sense to you, leave it on *Automatic* and everything will be fine.

If you are not using Art-Net for lighting, leave this set to *Automatic* and forget about it.

Use broadcast mode for Art-Net lighting

Check this box to use Art-Net's broadcast mode which allows QLab to communicate with Art-Net nodes that do not support polling.

Also check this box to allow QLab to properly communicate with other software which also uses Art-Net running on the same computer.

Un-checking this box will reduce the amount of network traffic generated by QLab while using Light cues. We recommend leaving this box unchecked if possible.

Chapter 2: Fundamentals

2.1 The Workspace
2.2 Workspace Settings
2.3 Workspace Templates
2.4 Managing Workspaces
2.5 Cues
2.6 The Inspector
2.7 Cue Lists
2.8 Cue Carts
2.9 Group Cues
2.10 Cue Sequences

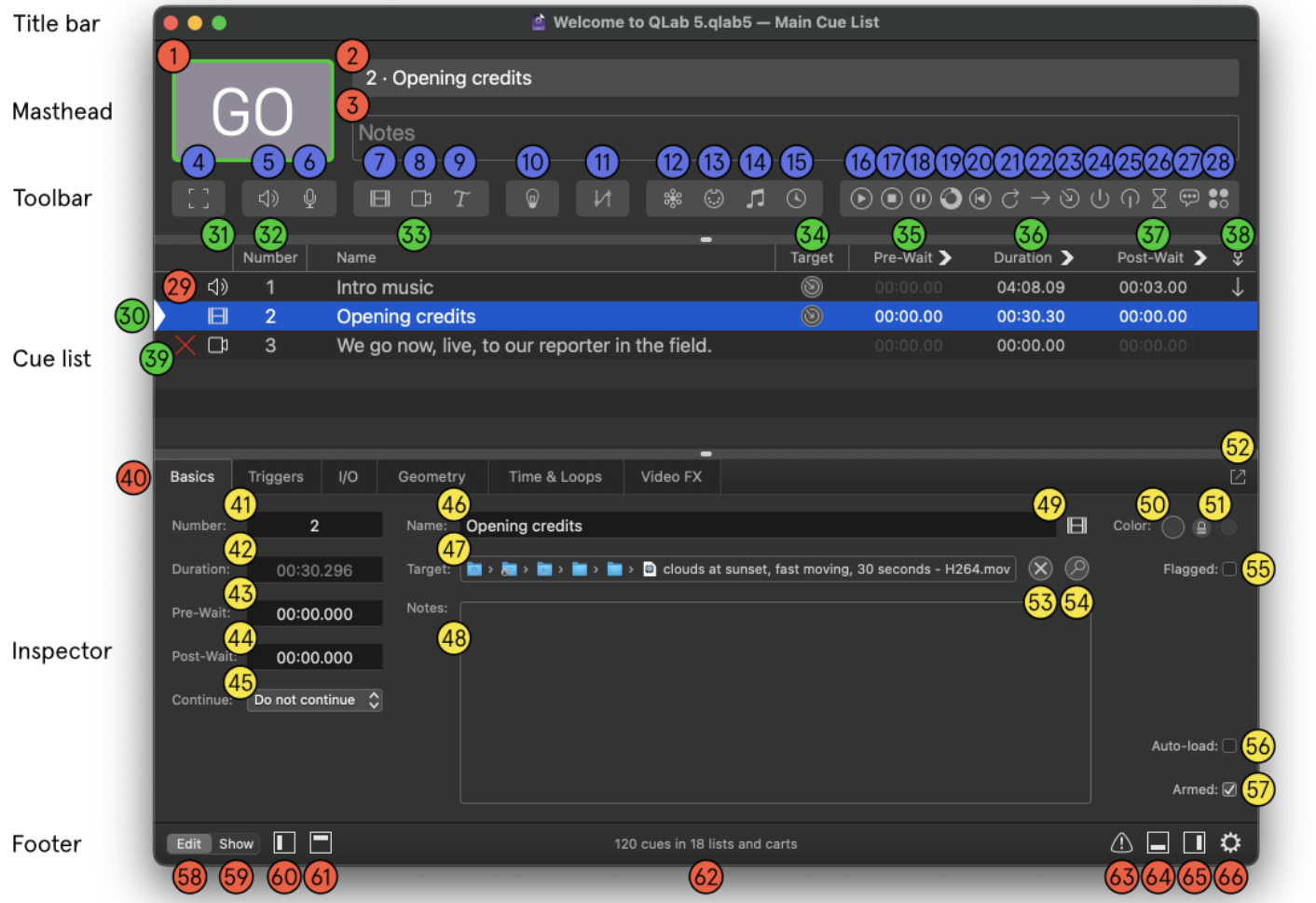
The Workspace

In this section, every time a new tool, interface item, or concept that we feel is particularly essential is mentioned, it will appear in bold text. This is meant to help you notice that you're being introduced to a new idea. Thereafter, and throughout the rest of this documentation, bold text will be used for emphasis, to highlight keyboard shortcuts (like **⌘S**), and for indicating a menu name (such as the **File** menu.)

A QLab document is called a **workspace**. Workspaces contain one or more **cue lists** and/or **cue carts**, which each contain *cues*. Workspaces also contain settings and configuration details which govern how QLab behaves when that workspace is in use.

A QLab workspace can have multiple windows that belong to it. Most of those are optional, or only necessary to use from time to time, and you'll learn about them throughout this manual.

The main workspace window is what you'll see when you create a new QLab workspace or when you first open an existing one.



◯ – General
◯ – Cue types
◯ – Cue list elements
◯ – Inspector elements

1 – The GO Button
2 – The Standby display
3 – The Notes field

4 – Group cue
5 – Audio cue
6 – Mic cue
7 – Video cue
8 – Camera cue
9 – Text cue
10 – Light cue
11 – Fade cue
12 – Network cue
13 – MIDI cue
14 – MIDI File cue
15 – Timecode cue
16 – Start cue
17 – Stop cue
18 – Pause cue
19 – Load cue
20 – Reset cue
21 – Devamp cue
22 – GoTo cue
23 – Target cue
24 – Arm cue
25 – Disarm cue
26 – Wait cue
27 – Memo cue
28 – Script cue

29 – Cue
30 – Playhead

31 – Cue type icon
32 – Cue number
33 – Cue name
34 – Target
35 – Pre-wait
36 – Duration
37 – Post-wait
38 – Continue mode
39 – Status icon

40 – Inspector tabs
41 – Selected cue number
42 – Selected cue duration
43 – Selected cue pre-wait
44 – Selected cue post-wait
45 – Selected cue continue mode
46 – Selected cue name

47 – Selected cue target
48 – Selected cue notes
49 – Selected cue type icon
50 – Selected cue color
51 – Selected cue second color
52 – Pop out inspector button
53 – Clear selected cue target
54 – Reveal selected cue target
55 – Selected cue flagged
56 – Selected cue auto-load
57 – Selected cue armed

58 – Edit mode
59 – Show mode
60 – Show/hide toolbox
61 – Show/hide masthead
62 – Cue count
63 – Show warnings
64 – Show/hide inspector
65 – Show/hide sidebar
66 – Workspace settings

The Title Bar

Most windows have a title bar, and QLab workspace windows are no exception.



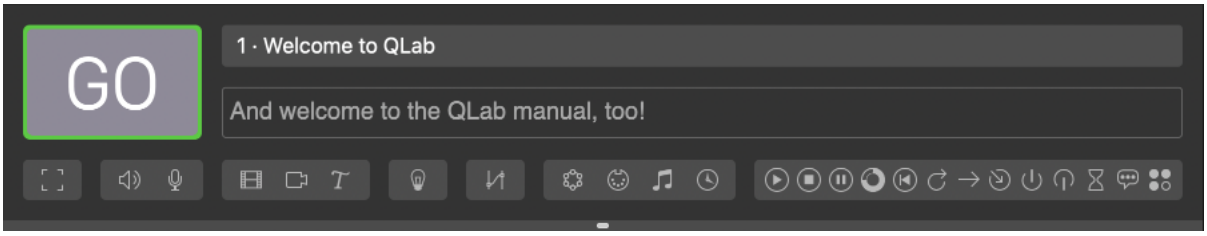
The “traffic light” buttons on the left control the visibility of the window: clicking the red button closes the window, clicking yellow minimizes the window into the Dock, and clicking green zooms the window into fullscreen mode.

If the workspace has unsaved changes, a dark dot will appear inside the red button. If you try to close a workspace with unsaved changes, you’ll be asked if you want to save those changes or not before closing.

The title bar also displays the name of the workspace followed by the name of the currently visible list or cart. In the screen shot above, the workspace is named `Welcome to QLab 5.qlab5` and the current cue list is called `Main Cue List`.

The Masthead

The top section of the workspace window is called the Masthead, borrowing a journalistic term for the top section of the front page of a newspaper.



The four elements of the masthead are the GO **button**, the **standby display**, the **notes field** and the **toolbar.**

The GO button

> The GO Button tutorial is a hands-on exploration of the topics discussed in this section.

Clicking the GO **button** starts the cue which is listed in the standby display. QLab then advances the **playhead** to the next cue, which will then be standing by and ready to GO.

The appearance of the GO button changes based on context:

- A green border appears around the GO button when a cue is standing by and QLab is ready to play that cue. In essence, the green border means that GOing will do something.
- A red border appears around the GO button to indicate that GOing is temporarily disabled due to double-GO protection.
- When a Cart is active, the GO button becomes a "Preview" button. You can learn more about what that means and why it happens in the Cue Carts section of this manual.
- When the workspace is set to always audition, the GO button becomes an "Audition" button. You can learn more about auditioning in the Auditioning Cues section of this manual.

The default keyboard shortcut for the GO button is **space**. You can change this shortcut in Workspace Settings. This keyboard shortcut always operates the GO button, whether it's currently labeled GO, Preview, or Audition.

The Standby display

The cue that is standing by is listed here, to the right of the GO button. If the cue has a cue number, the cue number is shown first, separated from the cue name with a bullet (•). The cue that is standing by is also identified in the cue list below using the **playhead**, which is a light-colored, right-pointing triangular indicator.

Each cue list in a workspace has its own playhead, and therefore its own idea of which cue is standing by. The GO button and its keyboard shortcut only apply to the active list.

The Notes field

Beneath the standby display, also to the right of the GO button, is the **Notes** field. Text entered here is connected to the cue that is standing by, and is visible whenever that cue is standing by. This makes it the perfect place for any notes or special instructions to your operator which pertain to that cue.

Text in the Notes field is searchable using the Find tool. It can be styled and formatted, and can include emoji and images.

You can edit a cue's notes here in this field, or in the Basics tab of the inspector. You can also select a cue and use the keyboard shortcut to edit notes, which is **O** by default.

The Toolbar

The **toolbar**, found underneath the Notes field, is a row of buttons for each of the different cue types available in QLab. Clicking any of these icons will create a new cue of that type in the current cue list. Explanations of the different cue types can be found in the Cues section of this Manual.

This space can also be used by the Find tool and the Load To Time tool.

The masthead can be resized by clicking and dragging on the horizontal divider that separates it from the cue list. Dragging all the way up will collapse and hide the masthead completely, and dragging back down from the top will restore it. When the cursor is hovering over the divider, the pointer will turn into a horizontal line with arrows.

The masthead can also be shown and hidden by choosing *GO Button / Standby Display / Toolbar* from the **View** menu, or by clicking the ▬ button in the footer.

The Cue List

The middle section of the workspace window shows the current active cue list or cart. Newly created workspaces generally default to showing a cue list, although you can make workspaces that contain only carts and no lists. When a cue list is active, it looks like this:



The cue list is a table with one row for each cue, listed in the order that they will play from top to bottom. Each column shows a different essential piece of information about each cue.

Status

The status column shows icons indicating the current status of the cue.

The Playhead

If the cue is standing by, the **playhead** appears on the absolute left edge. The playhead is a very light grey triangular pointer. If a cue is not standing by, clicking in the status column moves the playhead to that cue, thus making that cue stand by.

Cues might also display any of the following icons:

- A green triangle means the cue is **active.**
- A pair of yellow bars means the cue is **paused.**
- A yellow disc means the cue is **loaded.** Loaded cues are ready to be started with a minimum of latency, although loading cues before starting them is usually unnecessary.
- A yellow slope means the cue has been stopped, but has an effect that is **tailing out.** This icon remains visible as long as the cue's effects are still producing audio, such as an echo, a reverberation, or a distortion effect. The length of this "tail" can be adjusted within the audio effect itself, although QLab also leaves the audio pathway open for some additional time afterwards, just to be completely sure that no effect gets cut off prematurely.
- A red X means the cue is **broken** and cannot be played. Hovering the mouse over the red X will show a tool tip with a brief explanation of the problem. You can also see a list all the broken cues in the workspace in the Warnings tab of the Workspace Status window.
- A yellow triangle with an ellipsis inside is a **non-breaking warning**, meaning there's something wrong with cue, but not in a way that is likely to interfere with a show. Cues with non-breaking warnings are also listed in the Warnings tab of the Workspace Status window.
- A red circle with a slash through it means that an **override** is suppressing the cue's output. You can learn more about overrides in the Override Controls section of this manual.
- A yellow flag means the cue has been **flagged.**
- A yellow double-slope icon means the cue is currently **crossfading** with another cue. It's only used in conjunction with a Group cue in playlist mode.
- A hollow green play triangle in parentheses means the cue is being **auditioned.**

Cue Type

To the right of the status icon is an icon depicting the type of the cue. For Group cues and Fade cues, which each have more than one mode, the icon varies depending upon the mode of the cue. These icons match the icons used in the Toolbar and the Toolbox.

Cue Number

A cue number may be any text, or may be empty. All cue numbers in a workspace must be unique. Cue numbers do not need to be consecutive, nor do they need to be digits. Some examples of acceptable cue numbers are: `1`, `1.5`, `A`, `AA`, `A.5`, `Steve`, or □.

While QLab allows you to use digits, letters, punctuation, and emoji in cue numbers, you need to stick to only digits if you plan to use MIDI or MSC to remote-control cues, and only ASCII characters[1] if you plan to use OSC.

You can change the number of a cue by double-clicking in the cue's number column, or by selecting the cue and using the keyboard shortcut for editing cue numbers which is **N** by default. Changing a cue's number will not affect any cues that target it, or really anything else in fact.

It's important to understand that since cue numbers are text strings, `1`, `1.0`, and `1.00` are technically three different unique cue numbers in QLab. If you're using MIDI, MSC, or OSC for show-control and relying on matching cue numbers across different systems, be sure to match numbers character-by-character, not just semantically.

Reordering cues in the list does not automatically renumber them, and QLab does not care about "out of order" cue numbers.

Renumbering Selected Cues

You can automatically assign new cue number to each currently selected cue by selecting *Renumber Selected Cues* from the **Tools** menu, or using the keyboard shortcut ⌘**R**.

The cue renumbering tool allows you to select a starting number, an increment value, an optional prefix, and an optional suffix. The tool will use the starting number for the first selected cue and then add the increment value to the starting number for each subsequent cue. If you provide a prefix and/or suffix, those will be prepended or appended to every renumbered cue accordingly.

Since cue numbers must be unique within the workspace, the renumber tool will automatically skip over numbers that already exist within the workspace.

Cue Name

A cue name may be any text, or may be empty. For many cue types, QLab fills in a default name which reflects an essential attribute about the cue.

Cue type	Default name
Audio	The file name of the target audio file.
Video	The file name of the target video or image file.
Text	The text entered in the Text tab of the inspector.
Light	If the Light cue contains a single light command, the default name will be that full command. If the Light cue contains more than one command, the default name will be "fade `x` lights" where `x` is the number of instruments that have commands in the cue.
Fade	"fade" plus the name of the cue being faded. If the fade is set to stop target when done, the default name is "fade and stop" plus the name of the cue being faded.
Network	The network command sent by the cue.
MIDI	The type of MIDI message sent by the cue.
MIDI File	The file name of the target MIDI file.
Timecode	The starting frame of timecode sent by the cue.
Start, Stop, Pause, Load, Reset, Go To, Arm, Disarm	The type of cue plus the name of the target cue.
Wait	"wait" plus the duration of the cue.

You can change the name of a cue by double clicking in the cue's name column, or by selecting the cue and using the keyboard shortcut for editing cue names which is **Q** by default. Unlike cue numbers, cue names do not have to be unique.

Changing the name of a cue will also change the name of any cues that target it, if those cues are using their default names.

Deleting the name of a cue will reset it to its default name, if that type of cue has one, or to empty if not.

Target

Some types of cues require a target which is the recipient of the cue's action.

Cue type	Target type	Acceptable targets
Group, Mic, Camera, Text, Light, Network, MIDI, Timecode, Wait, Memo, Script	None	–

Cue type	Target type	Acceptable targets
Audio	File target	Any core-audio-compatible audio file (AIFF, WAV, CAF, MP3, MP4, M4A, AAC, etc.), except for files which are subject to digital rights management.
Video	File target	Any AVFoundation-compatible video file (MOV, MP4, M4V, etc.) containing video in an AVFoundation-compatible format (ProRes, PhotoJPG, H.264, H.265, Hap, and others.)
Fade	Cue target	Group, Audio, Mic, Video, Camera, and Text cues.
Start, Stop, Pause, Load, Reset, Devamp, Go To, Target, Arm, Disarm	Cue target	Any cue. Some cue targets may not have much of an effect (e.g. a Pause cue can target a Memo cue, but since a Memo cue doesn't have a duration, there's nothing to pause.)

Since different types of cues have different types of targets, the target column can show different types of information for different cues.

For cues which accept file targets, namely Audio, Video, and MIDI file cues, the Target column displays a round button with this icon: . Clicking this button will display a window in which you can select the file you wish to target. You can also set a cue's file target by dragging and dropping an appropriate file onto the cue's Target column from the Finder.

Cues which accept cue targets, such as Fade cues and Stop cues, will display the cue number of their target cue. If the target cue has no number, the target cue's name will be displayed instead. If the cue lacks a target, the target column will display a question mark. You can assign a target to these sorts of cues by typing a cue number into the Target column, by dragging and dropping the cue onto the Target column of its intended target, or by dragging and dropping the intended target cue onto the Target column of the cue.

The default keyboard shortcut for changing the selected cue's target is **T**.

Cues which do not require a target will show nothing in this column.

Pre-Wait

Pre-wait is an optional delay that transpires before the cue starts. For example, when you start an Audio cue with a pre-wait of 3 , that cue will allow three seconds to elapse, counting down its pre-wait, and then the audio will begin playing.



The pre-wait of a cue can be edited by double clicking and typing in the pre-wait column, or by using the keyboard shortcut for editing pre-wait which is **E** by default.

Duration

The **duration** of a cue is the length of time it takes for the cue to complete, not counting its pre-wait. The duration of some cues cannot be edited directly, but for those that can, they can be edited by double clicking and typing in the duration column, or by using the keyboard shortcut for editing durations which is **D** by default.

Cues which have an instantaneous action, such as a Pause cue or a MIDI cue that sends a single Note On message, have no listed duration.

Post-Wait

The **post-wait** of a cue is meaningful only in combination with an as-yet-un-discussed feature of cues: **auto-continue**. When a cue is set to auto-continue, it starts the next cue in the cue list when it is itself started. If the first cue has a post-wait time, QLab waits for the post-wait to elapse and then starts the second cue.

The post-wait of a cue can be edited by double clicking and typing in the post-wait column, or by using the keyboard shortcut for editing post-wait which is **W** by default.

Continue Mode

The final column, labeled with this icon: ⇳, displays the **continue mode** of the cue. Cues have one of three continue modes:

If a cue is set to **auto-continue** (↓), starting that cue will start the following cue as well. If the first cue has a post-wait, the post-wait will be honored before the next cue is started.



If a cue is set to **auto-follow** (⇊), then the next cue will be started as soon as the first cue completes. When you set a cue to auto-follow, QLab will automatically show a post-wait time equal to the duration of the cue. This cannot be edited, and serves as a visual reminder of the auto-follow.

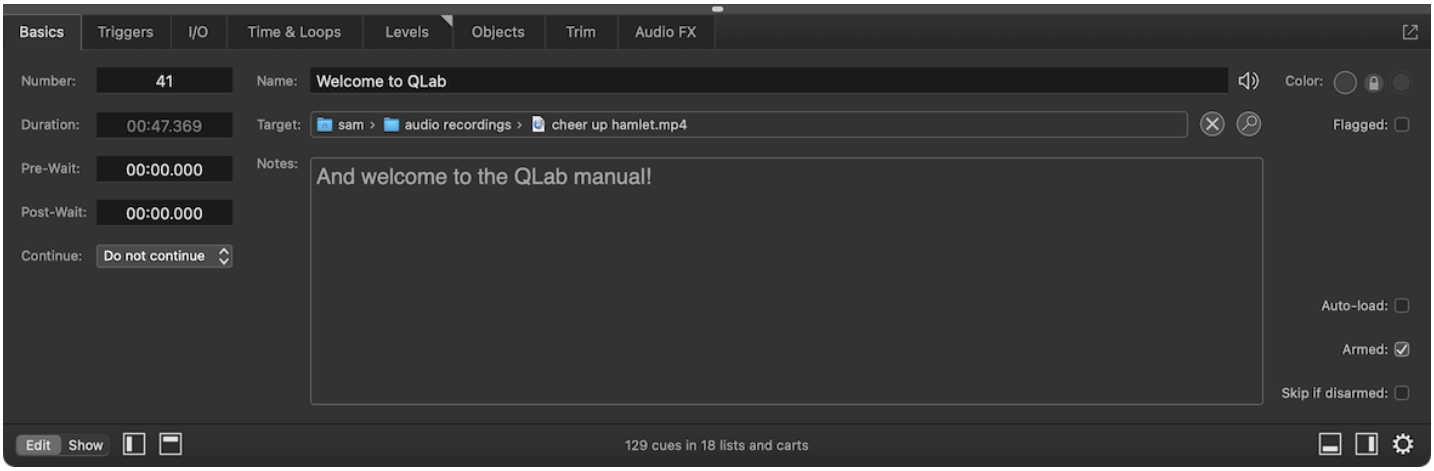


When you connect two or more cues with auto-follows and/or auto-continues, that's called a **cue sequence,** which you can learn more about from the cue sequences section of this manual.

The continue mode of a cue can be editing by clicking in the cue's continue mode column and using the pop-up menu which appears, or by using the keyboard shortcut for editing continue mode which is **C** by default.

The Inspector

Below the cue list is the inspector, which lets you view and edit the parameters of the selected cue or cues.



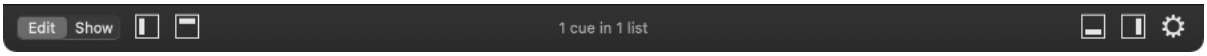
The tabs that appear in the inspector, and the contents of those tabs, will vary based upon the type and number of cues that are selected. The inspector always shows at least the Basics tab, shown in the screen shot above, and the Triggers tab.

The inspector can be resized by clicking and dragging on the horizontal divider that separates it from the cue list. Dragging all the way down will collapse and hide the inspector completely, and dragging back up from the bottom will restore it. The inspector can also be shown and hidden by choosing *Inspector* from the **View** menu, by using the keyboard shortcut ⌘**I**, or by clicking on the ▬ button in the footer.

You can learn more about the inspector and how to use it in the Inspector section of this manual.

The Footer

At the very bottom of the workspace window is the window footer.



The **Edit/Show** mode buttons let you switch the workspace between edit mode, which is the default mode, and show mode. When a workspace is in show mode, the following tools and functions of QLab are disabled:

- The inspector
- The toolbar
- The toolbox
- Load to Time
- Find
- Adding, deleting, and moving cues
- Editing, copying, and pasting any cue properties using the mouse or keyboard

Equally important are the things which are *not* disabled by when in Show mode:

- Quitting QLab
- Opening, closing, and saving workspaces
- Using the **escape** key
- Editing cue properties using AppleScript and OSC
- Viewing and changing workspace settings

Show mode is a safety mechanism designed to prevent accidental changes to a workspace. It is *not* a security mechanism for preventing deliberate changes.

When a workspace is in show mode, QLab will ask for confirmation before closing the workspace or quitting.

Next to the Edit/Show buttons are two icons:

- Click here to open or close the Toolbox.
- Click here to open or close the Masthead.

The **cue count** in the center of the footer displays the number of cues in your workspace and the number of cue lists and carts in which those cues reside. This is useful for confirming which copy of a workspace has the most recent work, and it's also pretty useful for bragging about your show.

The center of the footer also sometimes displays other informational text about the overall status of the workspace. This text can pertain to workspaces in Demo mode, workspaces set to always audition, and Override Controls.

The right side of the footer shows three or four icons:

- Always visible. Click here to open the Workspace Settings window.
- Always visible. Click here to open or close the Sidebar.
- Always visible. Click here to open or close the Inspector.
- Visible only when the workspace contains a broken cue or a warning. Click here to open the Warnings tab of the Workspace Status window and see details about the problem.

The Sidebar

The sidebar is an optional display on the right side of the workspace window. It has a set of controls at the top, and a two-tabbed section below. You can open and close the sidebar by clicking on the button in the footer, by choosing *Lists / Carts & Active Cues* from the **View** menu, or by using the keyboard shortcut ⌘**L**.

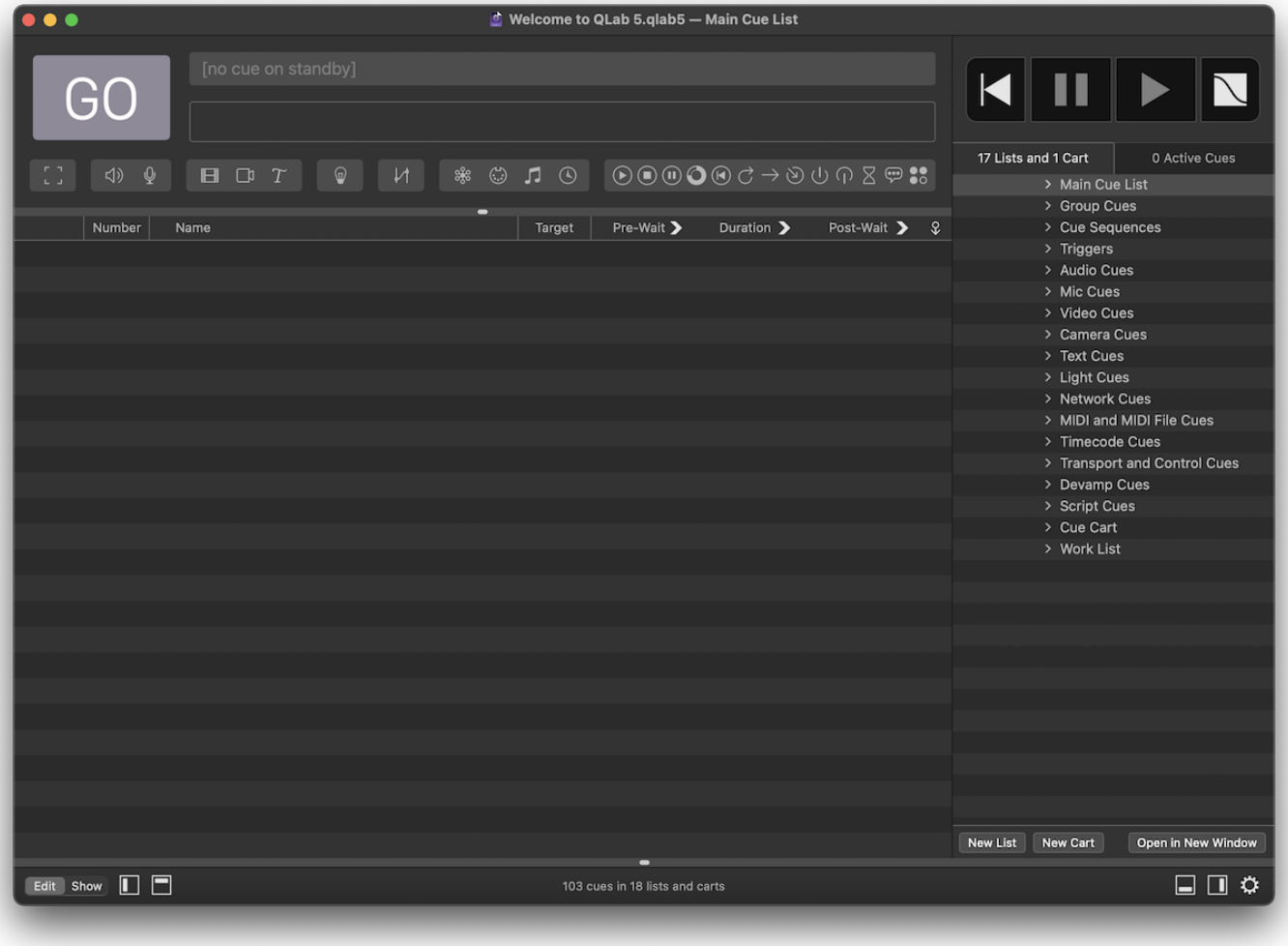
Sidebar Controls

The four buttons at the top of the sidebar have an effect on all cues in all lists and carts in the workspace.

- The **reset** button hard stops all running cues, resets all temporary properties[2] of all cues, returns the playhead for each cue list to the first cue in that list, and loads the first cue in every list. In effect, clicking the reset button makes the workspace behave as though it was just opened. If the workspace has a workspace open cue, however, that cue will not automatically run when the workspace is reset.
- The **pause all** button pauses all currently running cues in the workspace. This button is yellow when cues are running, and thus pause-able.
- The **resume all** button resumes all currently paused cues in the workspace. This button is green when cues are paused, and thus resume-able.
- The **panic** button panics all running cues. In QLab, *panic* means to fade and stop all cues using the panic duration, which is set in Workspace Settings → General.

Lists and Carts

The left tab in the sidebar displays the lists and carts in the workspace. The label of the tab will change to reflect the number of list and/or carts in the workspace.



By default, a workspaces will contain one list named “Main Cue List.” You can change this name as you prefer. You can add lists and carts using the **New List** and **New Cart** buttons at the bottom of the sidebar. To delete a cue list or cart (and all the cues in it!), select the list or cart and choose *Delete* from the Edit menu or use the keyboard shortcut ⌘⌫ (delete).

To open a list or cart in a new window, select that list or cart and click **Open in New Window**, or by right-clicking (or control-clicking, or two-finger-clicking) on the list or cart and using the contextual menu. Windows opened in this way are called **secondary windows**, and they have some behavior that is different from the main workspace window:

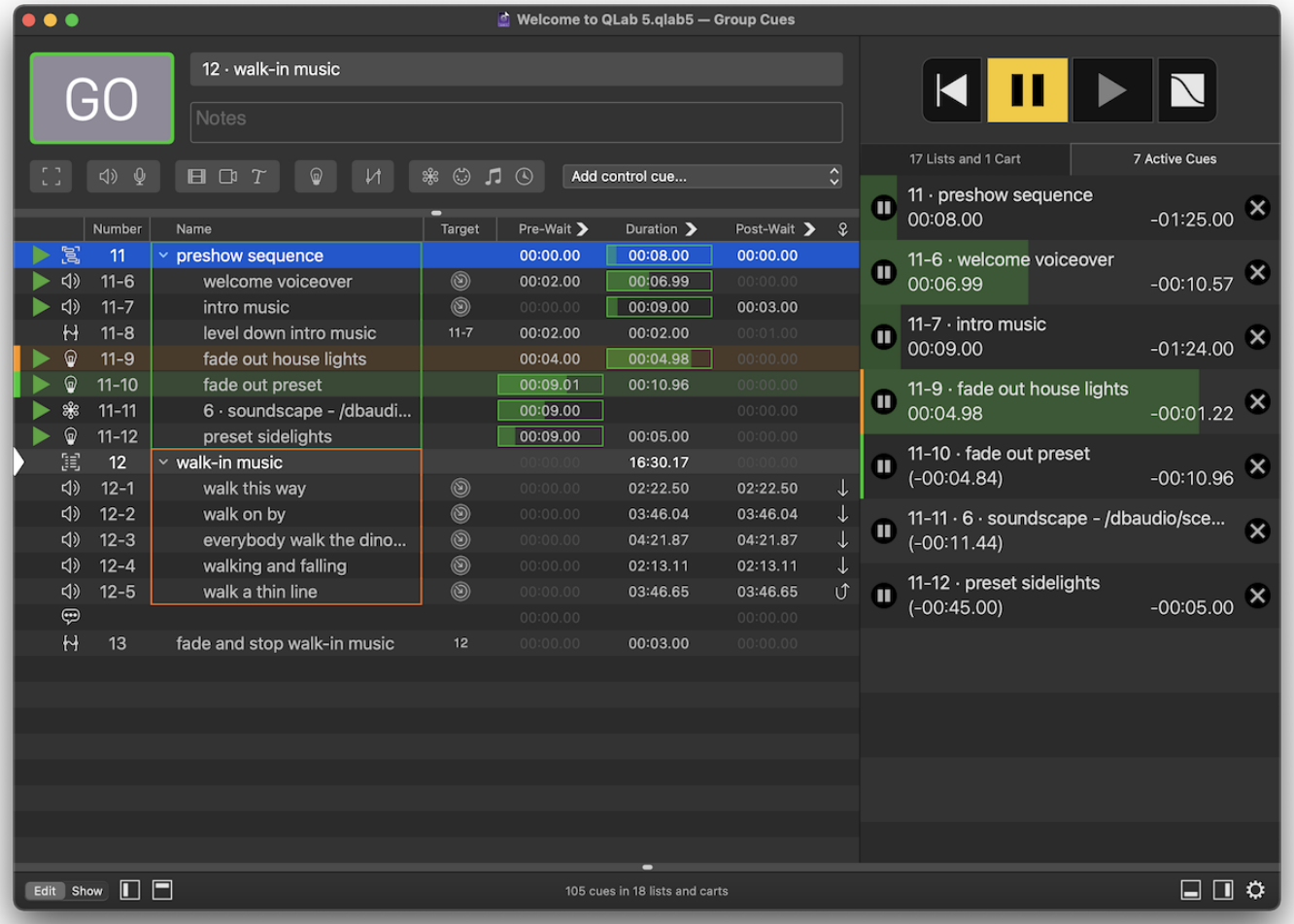
- They have no toolbar
- They have no inspector
- They have a different footer which contains only a menu to choose which list or cart they show
- When the workspace is in show mode, they display the words **SHOW MODE** in their footer.

Although secondary windows appear to always be in show mode, as long as the workspace is in edit mode you can edit cues in secondary windows using the basic editing keyboard shortcuts found in Workspace Settings → Controls → Keyboard or by right-clicking (or control-clicking, or two-finger-clicking) and using the contextual menu.

For more information about cue lists and cue carts, check out the cue list section of this manual and the cue carts section of this manual.

Active Cues

The right tab in the sidebar shows all cues in the workspace which are currently playing or paused.



Each active cue is displayed in its own row showing the cue's number and name with the elapsed and remaining time just below. If the cue has a cue color, that color is shown on the left edge of the row.

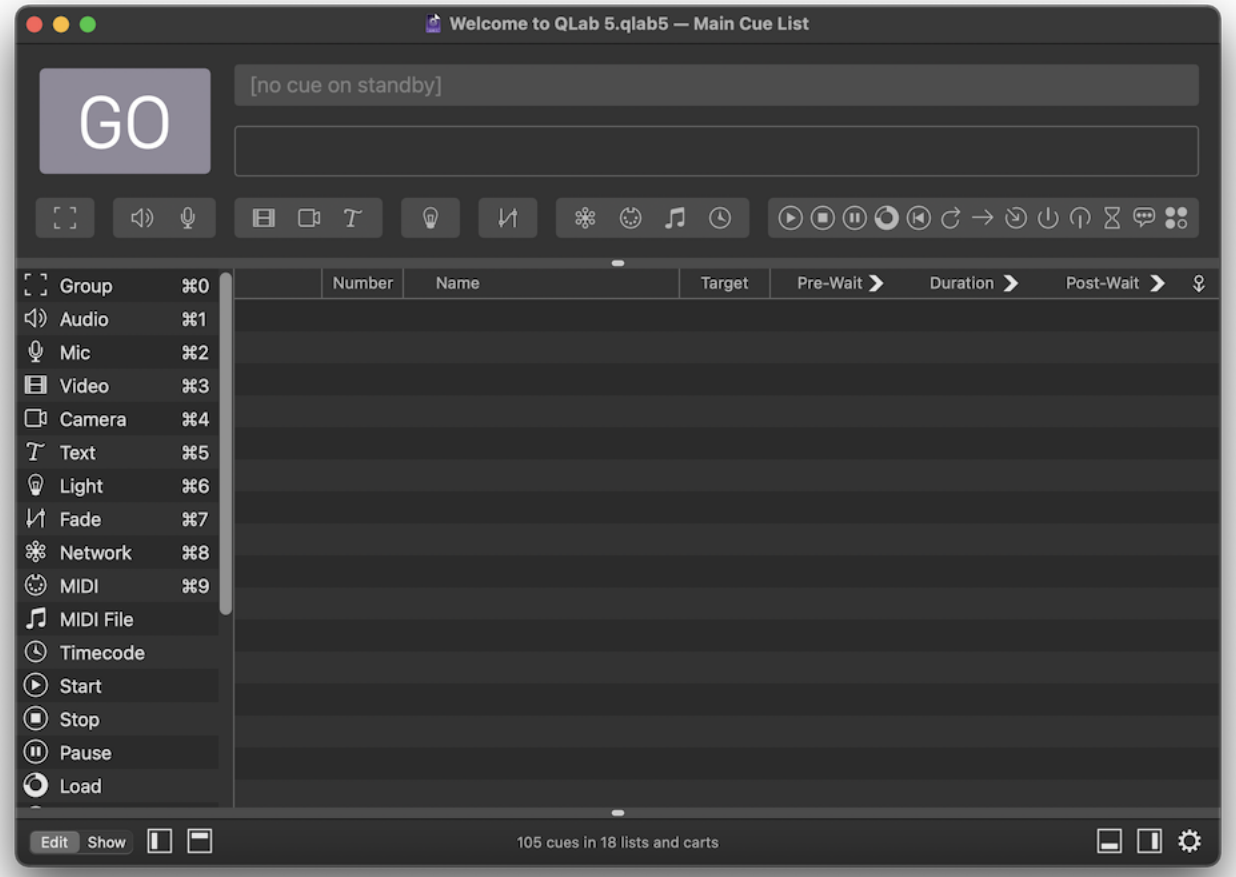
On the left side of each row is a pause/resume button, on the right side is a panic button, and throughout the whole row is a progress bar, colored green when the cue is playing and yellow when the cue is paused, which gives a visual indication of the progress of the cue.

If you hover your mouse over an active cue, the leading edge of the progress bar will show a thick yellow line, which can be dragged left or right to scrub the cue backward or forward.

Active cues are listed in the order in which they were started. By default, each cue that's started gets added to the bottom of the list, so the most recent cue is always at the bottom. You can reverse this order, putting the most recent cue at the top, in Workspace settings → General → Display.

The Toolbox

The toolbox is an optional miniature sidebar on the left side of the workspace window. You can open or close the toolbox by selecting *Toolbox* from the **View** menu, by using the keyboard shortcut **⌘K**, or by clicking the ▮ button in the footer.



The toolbox provides an alternative view of the toolbar; the two are very nearly identical. However, you can re-order cues in the toolbox. This lets you keep the cue types that you use most often close at hand. If you re-order the cues, notice that the **⌘-number** keyboard shortcuts remain assigned to the first ten cues. In this way, you can assign **⌘-number** keyboard shortcuts to the cues you prefer.

Reordering cues in the toolbox will reorder them in the Cues menu, but not in the toolbar.

---

1. The following characters may not be used in OSC messages, and therefore should not be used in cue numbers if you plan to use OSC: space, number sign `#`, asterisk `*`, comma `,`, forward slash `/`, question mark `?`, brackets `[` and `]`, or braces `{` and `}`.↩
2. Temporary properties are changes created by Target cues, Devamp cues, or any live OSC commands.↩

Workspace Settings

The Workspace Settings window can be accessed by choosing *Workspace Settings → Settings Window* from the **File** menu, or by using the keyboard shortcut ⌘, (command-comma). Settings in this window belong to the front-most workspace, not to QLab as a whole. Changes made here will not affect other workspaces, and will save and travel with the workspace if it's moved to another computer.

Settings are grouped into ten sections listed on the left side of the window. Many of those sections are divided into several tabs. You can also access each section directly from the *Workspace Settings* sub-menu of the **File** menu.

Importing and Exporting Workspace Settings

On the left side of the footer of the Workspace Settings window are two buttons for importing and exporting workspace settings.

Clicking **Import...** will allow you to import workspace settings from a saved QLab Settings file, another open workspace, or from QLab's default settings. In all three cases, QLab presents a list of checkboxes allowing you to choose which settings to import. You can use the up and down arrow keys to move up and down the list, and check or uncheck boxes with the mouse or the spacebar. You can also tap **A** on the keyboard to toggle all the checkboxes at once, and **C** to uncheck all checkboxes at once.

Imported settings will replace extant settings.

Clicking **Export...** will allow you to save the workspace settings from this workspace into a QLab Settings file which can then be used to import settings into another workspace. When you click **Export...**, QLab will present a list of checkboxes which looks and behaves like the list used for importing.

You can also use drag-and-drop in a few ways to import and export settings:

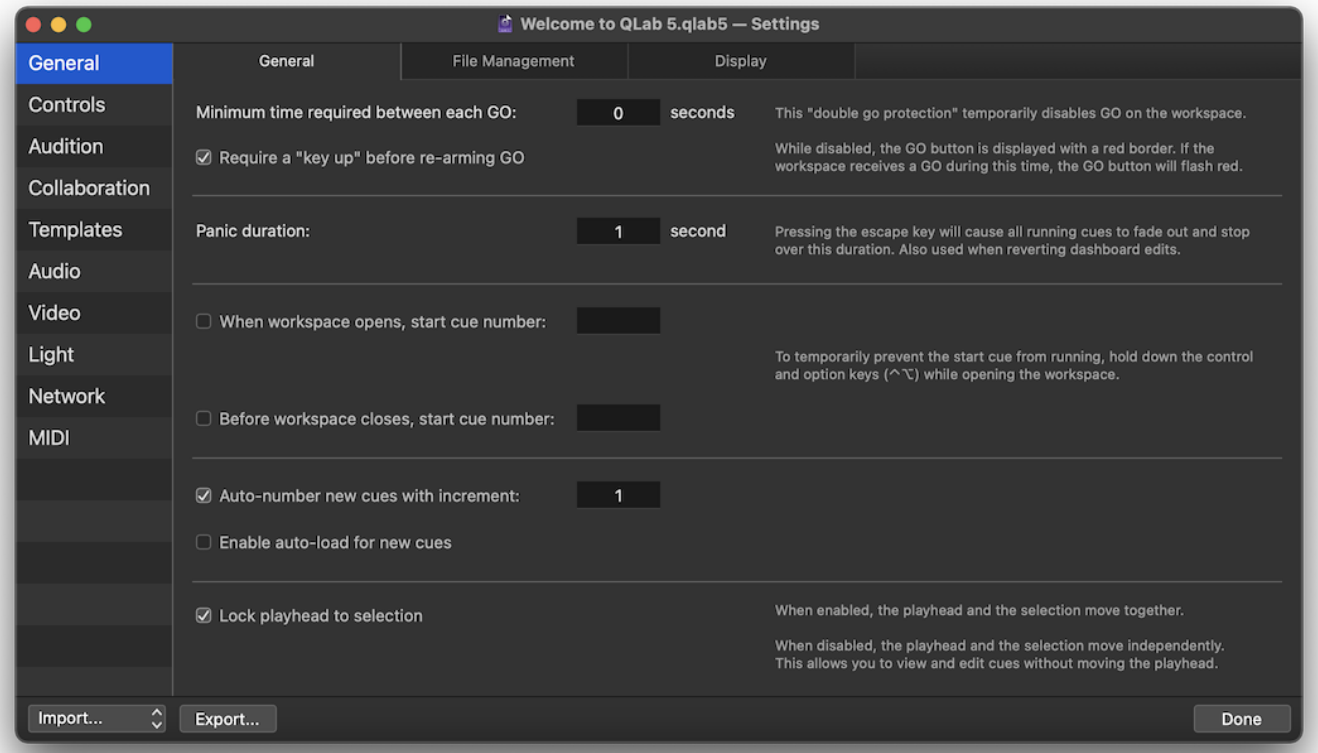
- You can drag a section from the list on the left side of the window to the Finder to create a QLab Settings file for that section of settings.
- You can drag a section from the list on the left side of the window into a Workspace Settings window belonging to another workspace to import that section of settings.
- You can drag a QLab Settings file from the Finder into any patch list, and if the file contains patches of the correct type, those patches will be imported to the workspace *without* replacing the existing patches.

You can also import and export settings from the *Workspace Settings* sub-menu of the **File** menu.

General

The three tabs in **General** settings apply to workspace-wide behavior.

The General tab



Minimum time required between each GO can be set to any duration, including zero, to help prevent accidentally starting cues. When a number greater than zero is entered, a red border will appear around the GO button after each GO to indicate that double-GO protection is in effect. If QLab receives a command to GO during this time, the area surrounding the GO button will flash red to indicate that the command was received, but disregarded. After the specified time elapses, the red border will disappear indicating that GOs are once again being accepted.

Double-GO protection applies to mouse clicks on the GO button, pressing the keyboard shortcut for GO (**space**, by default), and workspace MIDI, MSC, and OSC messages which directly trigger the "GO" action. Individual cue triggers and GOs via AppleScript are *not* protected by this mechanism.

Require a "key up" before re-arming GO applies only to the keyboard shortcut assigned to GO; if this box is checked, it compels QLab to wait for the key to be released before accepting another command to GO. This can help prevent problems caused by unreliable keyboards or weird KVM switches.

Panic duration is the amount of time it takes for cues to fade out and stop when the workspace receives the command to panic. If the workspace receives another command to panic while it's in the middle of panicking, everything which is in the midst of fading out will stop immediately.

When workspace opens, start cue number. If this box is checked and a cue number is entered, that cue will automatically start when the workspace is opened. If that cue begins a cue sequence, the sequence will play as normal. The playhead is not affected by this and will remain at the beginning of the cue list as usual.

To temporarily prevent the cue from starting, hold down the control and option keys (^⌥) while opening the workspace. You can also set QLab to prompt before running the workspace open cue.

Before workspace closes, start cue number. If this box is checked and a cue number is entered, QLab will present a dialog box when the workspace closes allowing you to choose to run that cue, close without running the cue, or cancel closing the workspace.

Auto-number new cues with increment causes newly created cues to be automatically numbered sequentially. QLab tries to be clever about this; if the increment is `1` and you have cues 2, 4, and 6, and add a cue after cue 6, the new cue will be numbered `7`. If you add a cue between cue 2 and cue 4, the new cue will be numbered `3`. If you then add another cue between cues 2 and 3, the new cue will be numbered `2.5`.

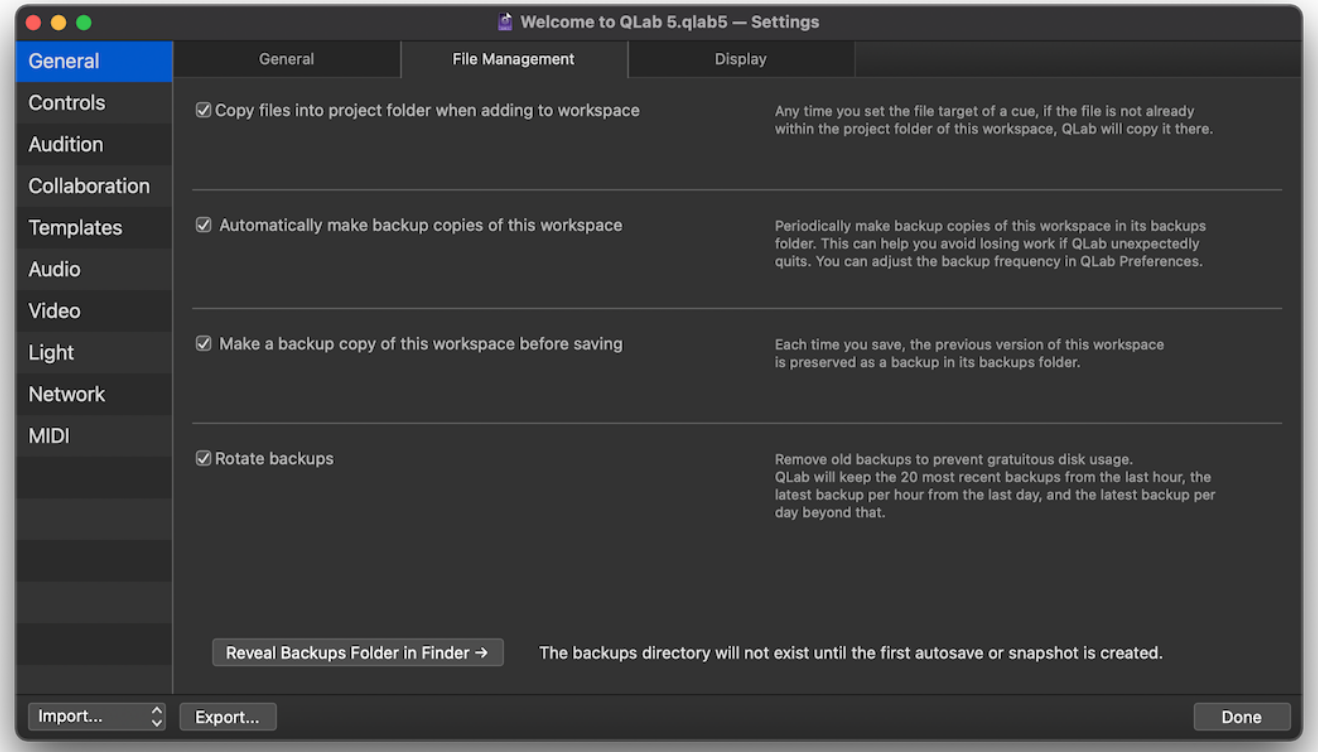
Cues which are created via OSC messages, AppleScript, or copy/paste are not automatically numbered.

Enable auto-load for new cues automatically sets all newly created cues to auto-load. Existing cues are not affected when this box is checked or unchecked.

Lock playhead to selection causes the playhead and the selection to move together; anything that causes either the playhead or the selection to move will cause both to move. When this box is not checked, the playhead and selection move independently. This makes it easy to view and edit one cue while QLab stands by on another cue.

The File Management tab

The three checkboxes in this tab control the way QLab automatically handles the files targeted by cues as well as files generated by the workspace. All three boxes are checked by default.



Copy files into project folder when adding to workspace. When this box is checked, any time you set the file target of a cue, QLab will copy that file into the workspace's project folder. If there is already an identical copy of the file target in the folder, QLab will not made an additional copy but will reassign the target of of the cue to use the copy of the file that's already in the folder. By keeping this box checked, you can always be confident that the project folder contains all necessary file targets to run your show. Because of that, you can easily move your workspace to another computer by simply copying the whole project folder.

You can alternately choose to direct QLab to copy media files whenever you wish by unchecking this box, and then choosing *Workspace Files > Copy media files into project folder* from the **File** menu.

Automatically make backup copies of this workspace. When this box is checked, QLab periodically creates backup copies of your workspace named `{the name of your workspace} (Backup {year-month-day_hourminutesecond}).qlab5backup`. These copies are stored in a folder named `{the name of your workspace} backups`. QLab never saves or changes your workspace file itself, it only creates copies. If your workspace document is ever lost or damaged, you can use a backup copy to recover your lost data.

You can configure the autosave interval in QLab Preferences to anywhere between five and 600 seconds (ten minutes.) Once set, QLab will watch your workspace for changes, and then wait for this amount of time to pass after the last change before autosaving. The reason for this delay is to minimize the possibility of the autosave interrupting your work. If you work continuously for a very long time, QLab will always save at least once every ten minutes.

> **Very important thing:** automatic backups can only be made after you've manually saved the workspace. Automatic backup *does not occur* on unsaved, untitled workspaces.

Make a backup copy of this workspace before saving. When this box is checked, QLab watches for you to manually save and automatically saves a copy of the workspace as it was *before* you made the changes that you are currently saving. These copies are named `{the name of your workspace} (Backup {year-month-day_hourminutesecond})` and are also stored in the backups folder.

If you open your workspace called "Hamlet" on August 30th, 2022 at 12:00 pm and begin work, then save at exactly 12:05, your workspace will obviously include all the changes you made between 12:00 and 12:05. At that moment, if this checkbox is checked, QLab will also create a backup called `Hamlet (Backup 2022-08-30_120000).qlab4backup` which reflects the exact state of your workspace when you opened it at 12:00.

If you do more work, then manually save again at exactly 12:12 pm, QLab will create another backup named `Hamlet (Backup 2022-08-30_120500).qlab5backup` which reflects the exact state of your workspace the *last time* you saved, at 12:05.

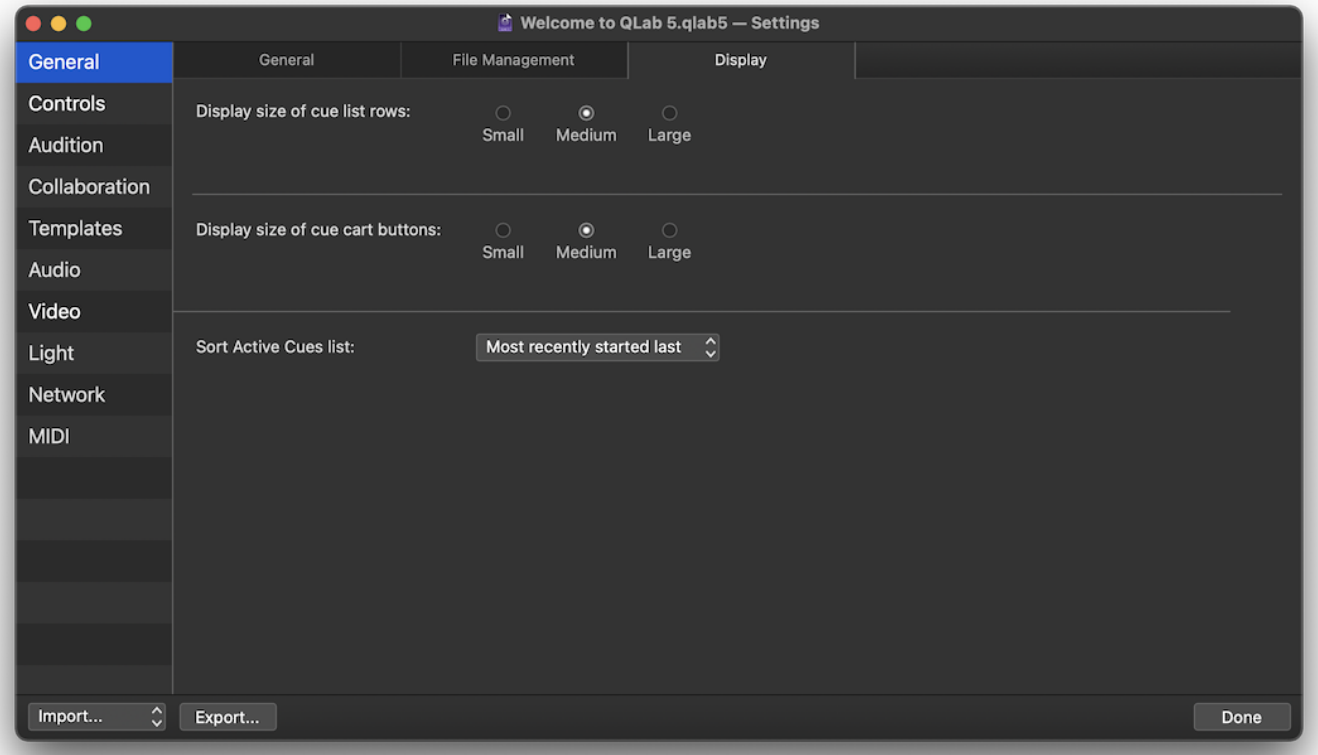
In order to keep from creating too many backups to be useful, QLab will only create one backup per minute. If you manually save your workspace more than once within a single minute, only the last backup created during that minute will be saved.

This makes it easy to return to an earlier state of your workspace if you discover that you need to revert lots of work, such as when returning from a dinner break and hearing the director say that dreaded phrase, "you know what, let's forget everything we did since lunch."

Rotate backups. When this box is checked, QLab automatically deletes old backups to prevent gratuitous disk usage. QLab will keep up to the 20 most recent backups within the last hour, the latest backup per hour from the last day, and the latest backup per day beyond that. The idea here is to make it simple for you to have very granular backups from your most recent work, somewhat granular backups over the last day (to help you, say, figure out what you were doing before lunch break), and a daily "end of day" backup reaching back indefinitely.

The **Reveal Backups Folder in Finder →** button at the bottom of the tab opens a Finder window showing the folder that contains your workspace and its backups folder. The current size of the backups folder is displayed next to the button.

The Display tab



The first set of small/medium/large buttons controls the display size of cue list rows.

The second set controls the display size of cue cart buttons.

Choosing “small” will maximize the information available on screen. Choosing “large” will maximum legibility, particularly at a distance. “Medium” is, unsurprisingly, a balance between the two.

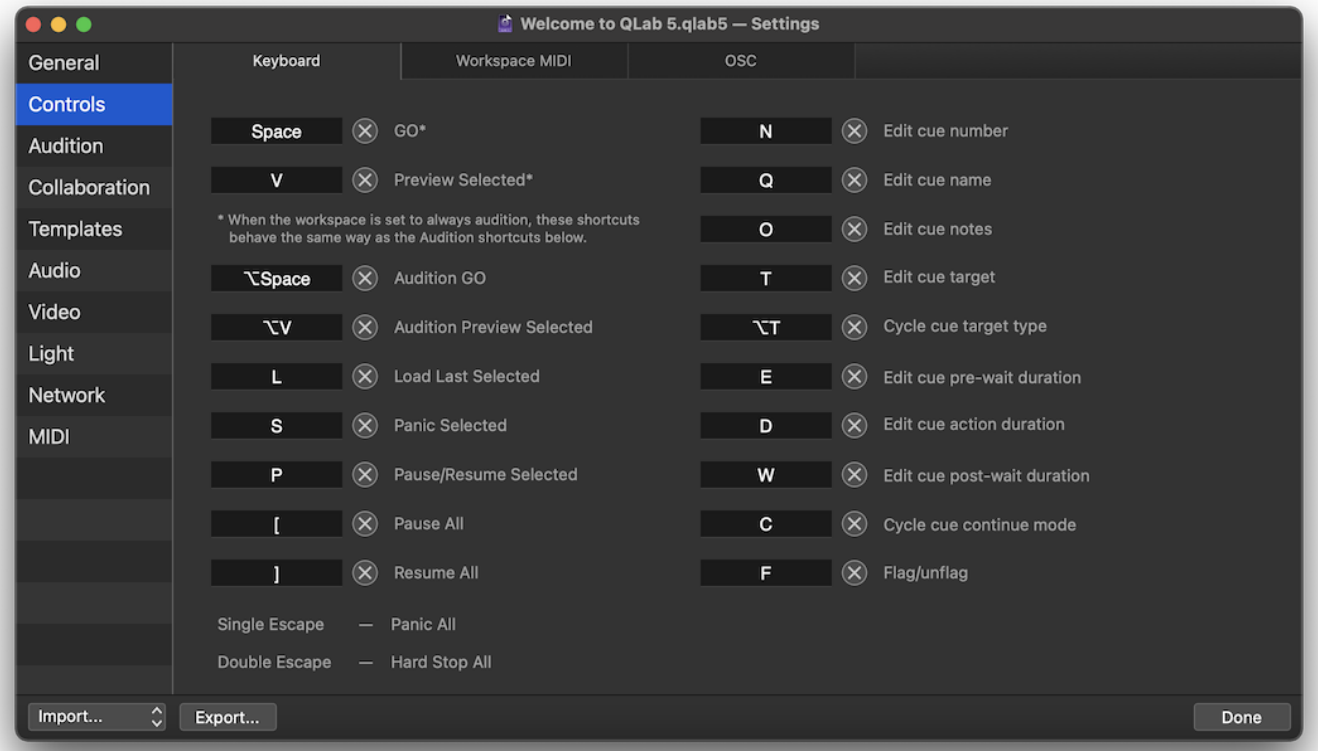
The *Sort Active Cues list* control lets you choose how the currently playing or paused cues are ordered in the Active Cues tab of the Workspace sidebar, sorting the most recent cues to the bottom or to the top.

Controls

The three tabs in **Controls** settings pertain to keyboard shortcuts, MIDI messages, and OSC messages which are used to directly control the workspace or basic functions of QLab.

The Keyboard tab

QLab allows you to change keyboard shortcuts for core control and editing features to suit your needs. Like all other workspace settings, these shortcuts apply only to the workspace being edited.



To remove a keyboard shortcut from a command, click the ⊗ button next to that shortcut. To change a shortcut, click in the text field for that shortcut and type the desired shortcut, optionally including one or more modifier keys:

Symbol	Meaning
⌘	command
⇧	shift
^	control
⌥	option

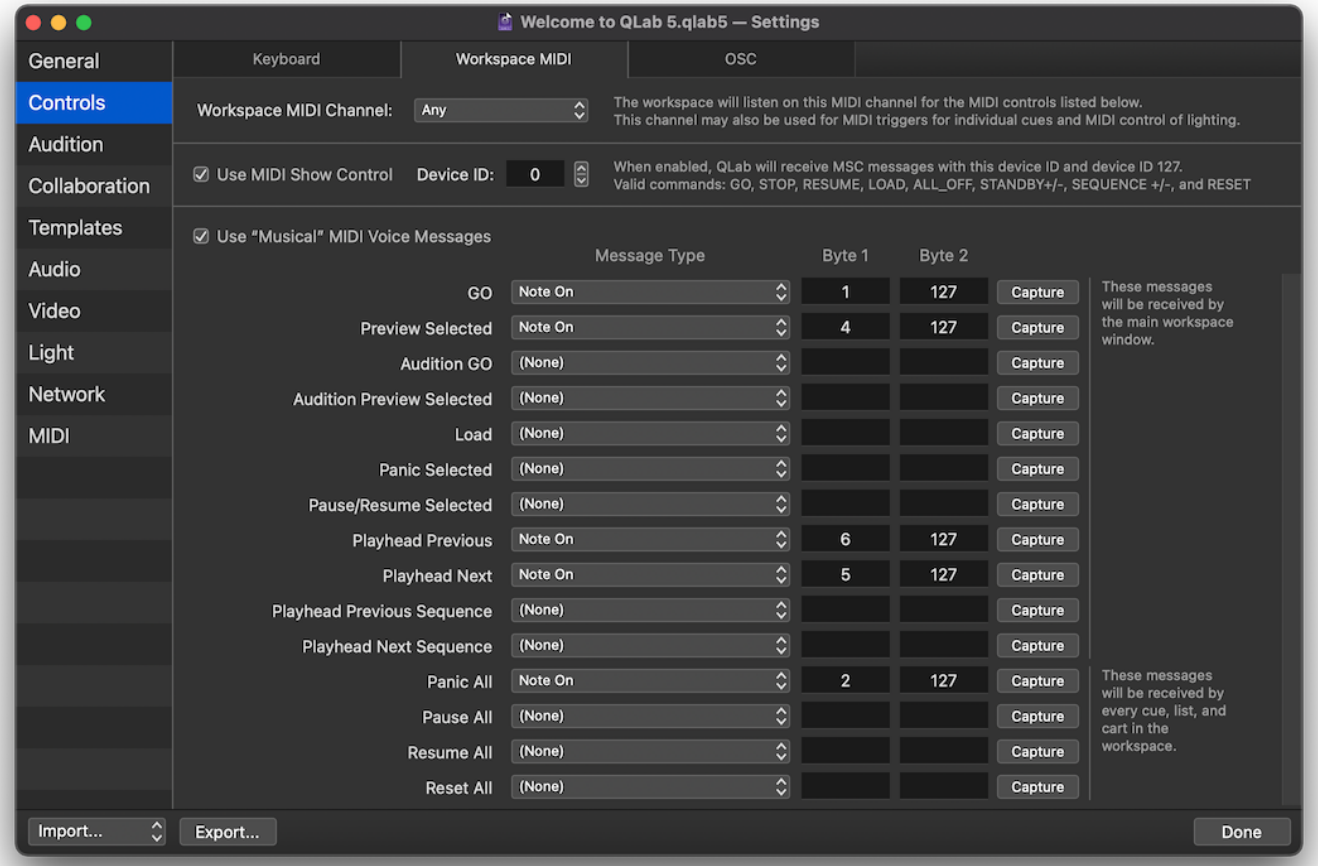
QLab will not prevent you from using a keyboard shortcut that has another meaning (in case you’re doing it deliberately,) so take care when assigning shortcuts.

Note that a single press of the **escape** key is always assigned to panic, and a double press of the **escape** key is always assigned to hard stop. This is a safety measure; in an emergency, any person familiar with QLab can confidently use the **escape** key to stop all playback from any QLab system.

The up and down arrow keys cannot be used for shortcuts, but the left and right arrow keys can be used.

The Workspace MIDI tab

This tab allows you to define the way your workspace will respond to incoming MIDI and MIDI Show Control messages.



Workspace MIDI Channel can be set to a single channel or to "any". The workspace will listen on the specified channel for MIDI messages that have been assigned to the commands below. Cues' MIDI triggers and the Light Dashboard can also listen on the workspace channel, or they can be set to listen to a separate channel irrespective of this setting.

Use MIDI Show Control enables the workspace to respond to the following MSC messages:

MSC message	QLab function
GO	GO.
GO `{cue_number}`	Start cue `{cue_number}` individually. The playhead is not moved.
STOP	Pause all
STOP `{cue_number}`	Pause cue `{cue_number}`.
RESUME	Unpause all
RESUME `{cue_number}`	Unpause cue `{cue_number}`.
LOAD	Load the currently standing-by cue.
LOAD `{cue_number}`	Load cue `{cue_number}`.

MSC message	QLab function
ALL_OFF	Panic.
STANDBY+	Move the playhead down one cue.
STANDBY-	Move the playhead up one cue.
SEQUENCE+	Move the playhead down one cue sequence.
SEQUENCE-	Move the playhead up one cue sequence.
RESET	Stop all cues and move the playhead to the top of the cue list.

QLab listens for MSC commands categorized as *Lighting (General)*, *Sound (General)*, and *Video (General)*.

Device ID should be unique for each device on an MSC network, or at least unique for all devices within a given category. When a workspace is set to use MIDI Show Control, it will respond to any message sent to this device ID, as well as any message sent to device ID 127 which is the "all call" identifier in the MSC specification.

MIDI channels and MSC device IDs have nothing to do with each other; all MIDI voice messages ignore MSC device IDs, and all MSC messages ignore MIDI channels.

Use "Musical" MIDI Voice Messages will enable the commands in the table below. There, you can manually assign an incoming MIDI message to some or all of the listed commands, or click on the **Capture** button next to a command to have QLab listen for the next incoming MIDI message on the workspace MIDI channel, and assign that message to that command.

Message Type can be Note On, Note Off, Program Change, or Control Change.

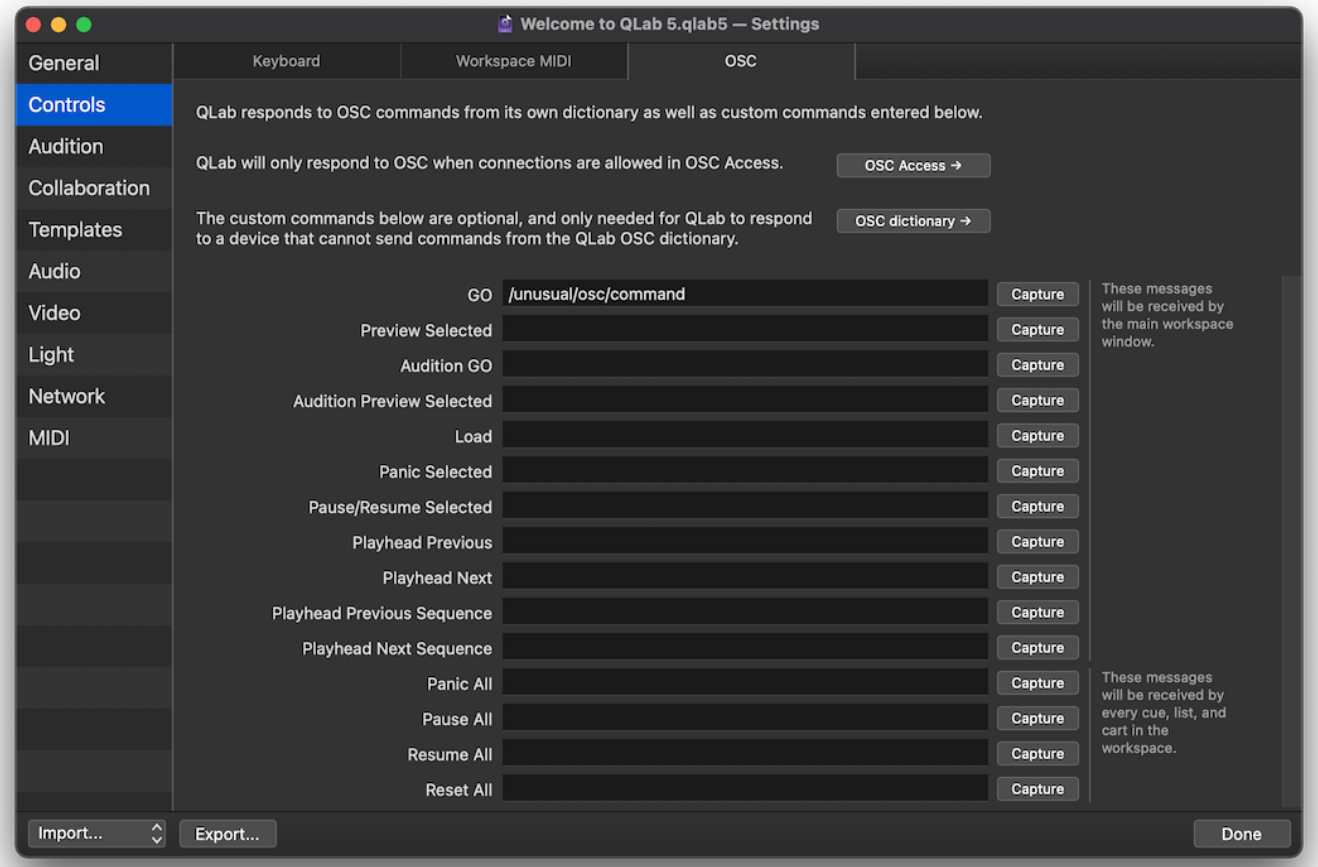
Byte 1 for Note messages represents the note number. For Program Change, it represents the program number. For Control Change, it represents the control number.

Byte 2 for Note messages represents the velocity. You can use any number from 0 to 127, inclusive, and you can use greater-than (>) or less-than (<) symbols to indicate a range. You can also use the word "any" to have QLab respond to any note velocity. For Program Change, this value is ignored. For Control Change, this value represents the control value.

The OSC tab

QLab has a very robust OSC dictionary which offers direct access to a wide variety of commands, settings, and functions via OSC messages. Sometimes, however, you may come across a device which you want to use to control QLab, and this device may not be able to send customized OSC messages.

To accommodate such devices, you can assign OSC messages to each of the commands listed in the table by typing or pasting in the OSC message, or by clicking the **Capture** button and letting QLab listen for the next incoming OSC message.



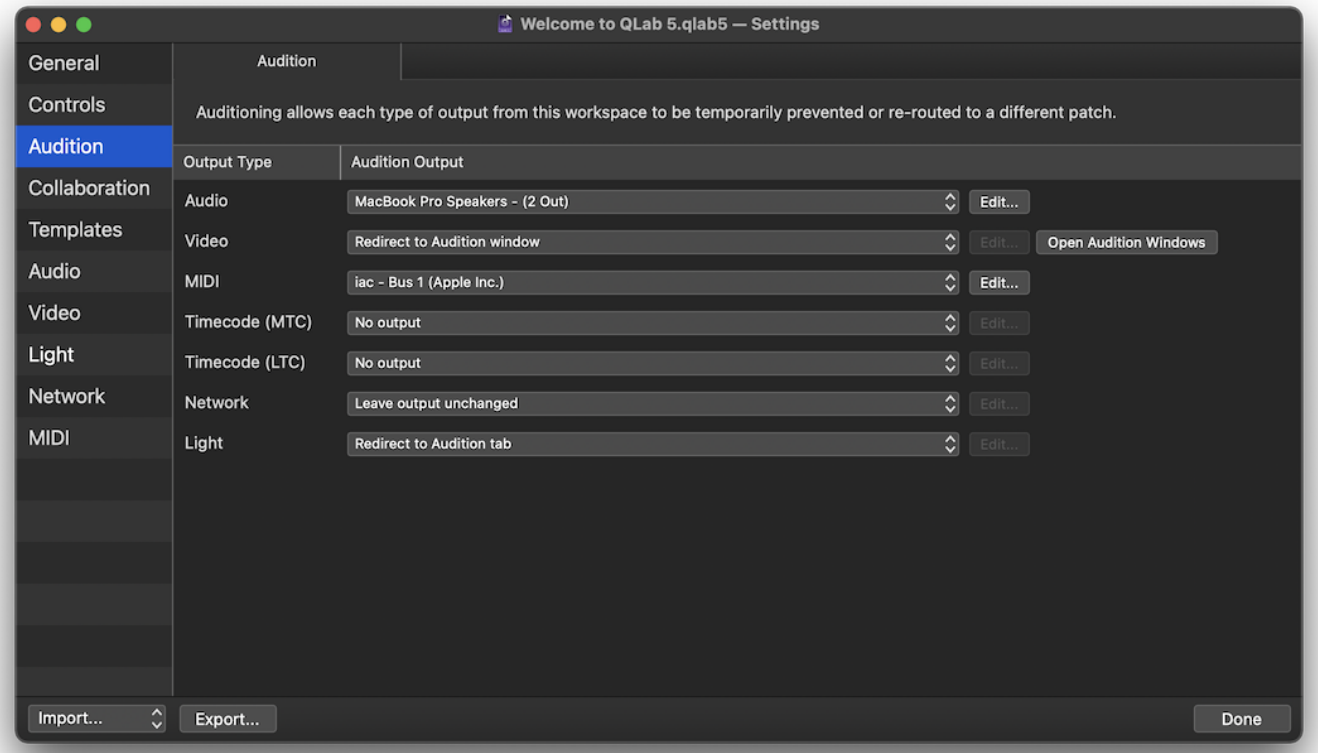
You do not need to use this section if your sending device can send OSC messages from QLab's dictionary.

As the text in this tab states, you need to enable incoming OSC messages in [the OSC Access tab of Workspace Settings → Network] in order to use OSC messages here.

Audition

These controls allow you to define the behavior of the different types of outputs from QLab when cues are auditioned. Each type of output has two to four of the following options:

- **Leave output unchanged** - output of this type will behave the same way in cues which are auditioned as it will in cues which are played normally.
- **No output** - output of this type will be entirely prevented from cues which are auditioned.
- **Alternate patch** - output of this type will temporarily use the specified output patch instead of the patch that's programmed into the cue.
- **Audition window** (video only) - output of this type will be displayed in a monitor window instead of sent to the video stage that's programmed into the cue.
- **Audition tab** (light only) - output of this type will be displayed in the Audition tab of the Light Dashboard, and not output via Art-Net or USB DMX.

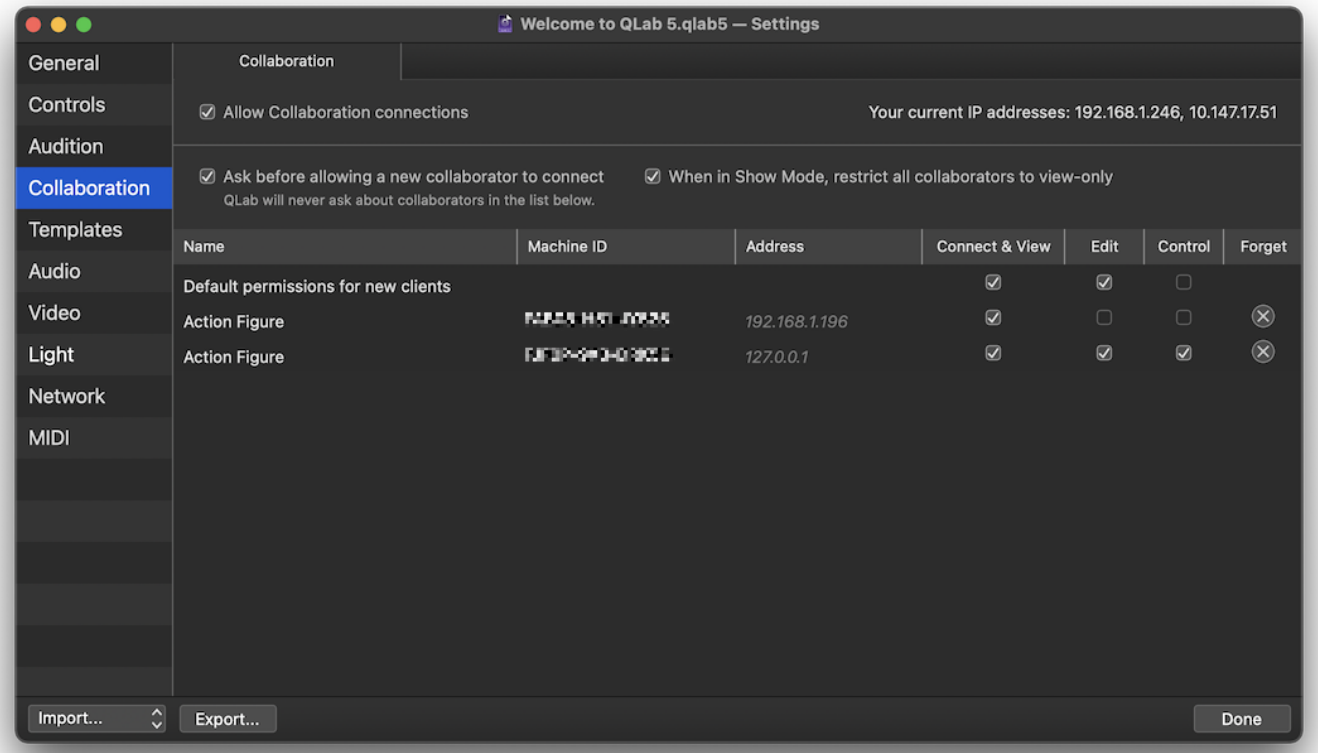


If an output type is set to an alternate patch, you can click the **Edit...** button to edit the selected patch.

You can learn more about auditioning cues from the Auditioning Cues section of this manual.

Collaboration

Other computers on your local network can use QLab to remotely connect to and collaborate on your workspace, through the power of QLab Collaboration, and this tab of Workspace Settings allows you to configure access permission for those connections.



Allow Collaboration connections. When the box is checked, Collaboration connections will be admitted based on the rest of the settings in this tab. When this box is unchecked, Collaboration connections will not be accepted, regardless of any other settings. If any collaborators are connected to the workspace when the box becomes unchecked, those collaborators will be immediately disconnected.

The upper right corner of the tab displays the current IP address or addresses of your Mac.

Ask before allowing a new collaborator to connect. If this box is checked, a message will appear on screen when a new collaborator tries to connect to your workspace, and you will have the option to allow or deny that connection, as well as to assign access permission for the connection. If you allow the connection, the collaborator will be added to the list below with the level of access that you assign, and will thereafter be allowed to connect to the workspace without the approval message.

When in Show Mode, restrict all collaborators to view-only. If this box is checked, all collaborators will be prevented from editing the workspace, moving the playhead, starting cues, or stopping cues whenever the workspace is in Show Mode, even if those collaborators would otherwise be allowed more expansive access. This is a useful way to allow full remote collaboration some of the time, but prevent accidental edits or playback events from remote collaborators during a performance.

The Collaboration section of Workspace Settings looks different on a Collaboration remote. You can learn more about this from the Using A Remote section of the Collaboration page of this manual.

The collaboration access table

The first row in the collaboration access table allows you to set the default access permissions for new collaborators. When a request to collaborate appears, clicking the default button (or pressing the enter or return key) in the connection message will admit the collaborator and assign these permissions. Then, a new row will appear in the table for that collaborator, identified by their Collaboration Name, Machine ID, and IP address.

You can optionally edit a Collaborator's access permissions after they've connected by using the checkboxes in the table. For example, you might allow your stage manager's Mac to connect with *view* permission only, and allow your assistant designer to connect with *view* and *edit* permission.

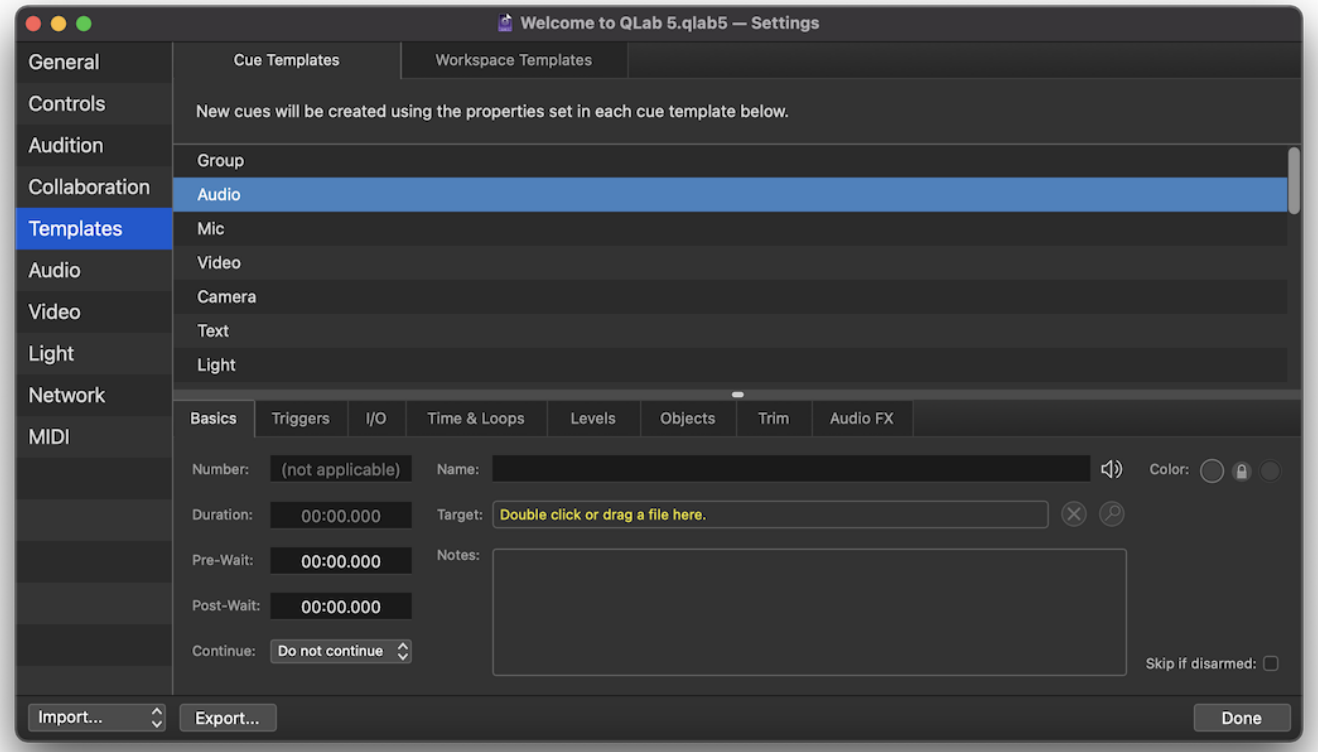
You can disconnect a client using the ⊖ button, or disconnect and forget a client using the ⊗ button. A forgotten client can reconnect, but will be treated as a new first-time collaborator once more.

Templates

The two tabs in **Templates** settings pertain to QLab's behavior when creating new cues and workspaces.

The Cue Templates tab

Cue Templates allow you to adjust the default settings of newly created cues. The Cue Templates section shows a list of all of the cue types in QLab and a copy of the inspector below that list. You can select one of the cue types and use the inspector to make any adjustments you like, including cue names, notes, targets, and everything else. Newly created cues in this workspace will use these settings as their default.



The Workspace Templates tab

Workspace Templates allow you to create your own set of starting conditions for new workspaces. This isn't really a workspace setting per se, but a shortcut is available here to open the Workspace Template Manager.

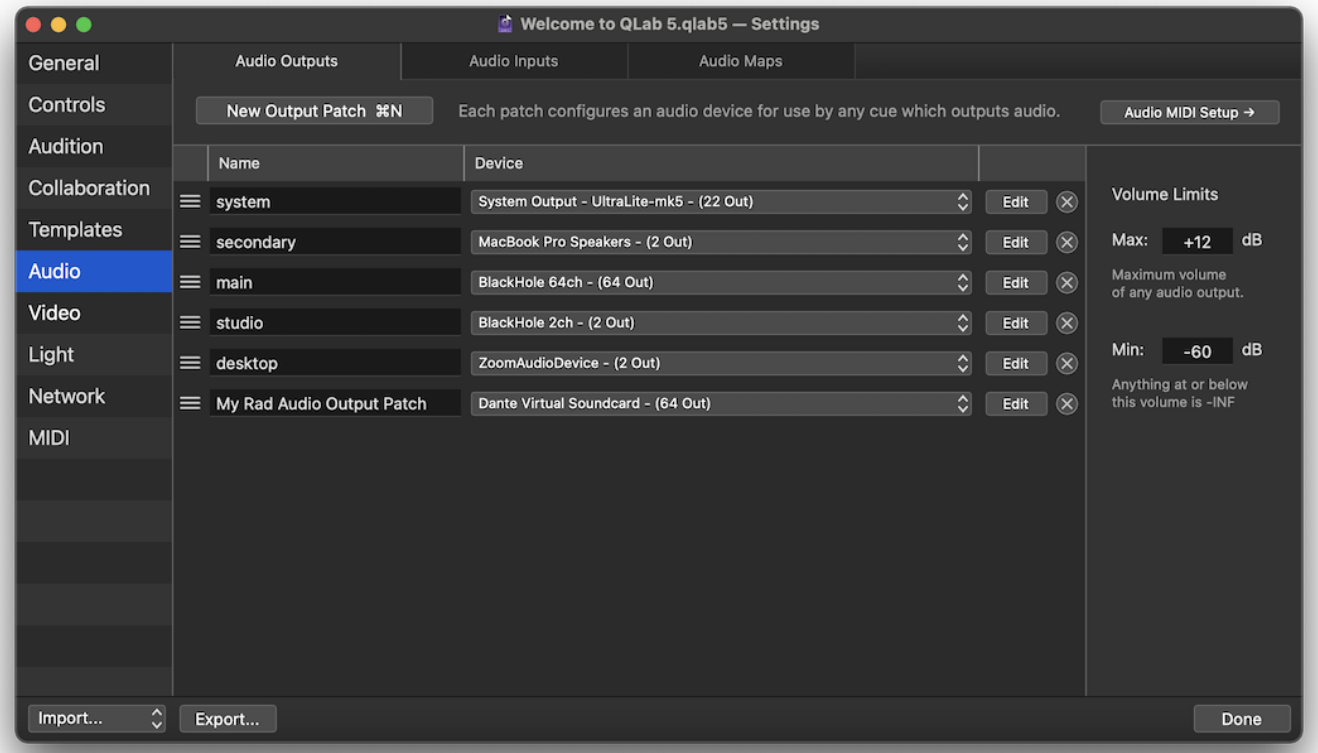
You can learn more about this from the Workspace Templates section of this manual.

Audio

The three tabs in **Audio** settings pertain to audio input and output patching, routing, effects, volume limits, and object audio.

The Audio Outputs tab

An **audio output patch** is the mechanism that QLab uses to route audio from cues to your audio output hardware. Any cue which can generate audio (Audio cues, Mic cues, Video cues, Camera cues, and Timecode cues set to LTC mode) must be assigned to a single audio output patch, and the configuration of the patch defines how the output from that cue will behave.



Workspaces in QLab 5 can contain any number of audio output patches. You can add an audio output patch by clicking on the **New Output Patch** button or by using the keyboard shortcut ⌘**N** while the Audio Output tab is open and in the foreground. A newly created blank workspace will automatically start off with a single audio output patch for each audio output device connected to your Mac at the moment you create the workspace.

Audio devices must have output channels to work with audio output patches.

Several actions are available in the patch table:

- **Delete** a patch by clicking on the ⊗ button on the right side of the row for that patch.
- **Reorder** a patch by dragging it up or down in the patch list. You can click anywhere on the background of the row for that patch, but it's easiest to click on the left side where the ≡ icon is displayed.
- **Duplicate** a patch by holding down the option key (⌥) and dragging the patch up or down in the patch list.
- **Copy** a patch into another workspace by dragging it into that workspace's patch list.
- **Export** a patch by dragging it into the Finder.

You can hold down the shift key (⇧) while clicking to select a range of patches, or the command key (⌘) while clicking to select two or more patches, allowing you to act on all the selected patches at once.

You can learn about editing audio output patches from the Audio Output Patch Editor section of this manual.

Volume Limits

The *Max* and *Min* volume limits for a workspace define the range of all audio in the workspace. Audio level controls, like the ones found in the Levels tab of the inspector for an Audio cue, will use the limits defined here, and setting audio levels via Fade cues, AppleScript, and OSC will similarly be limited to this range.

The maximum volume limit can be set anywhere from `-30` dB and up. `+12` is the default setting.

The minimum volume limit can be set anywhere between `-180` dB and `-40` dB, inclusive. `-60` is the default setting.

These levels are not (and in fact cannot be) measures of actual absolute loudness (dB SPL). They are only relative measures of loudness within QLab. A source file normalized to `-3` dBFS will output from QLab at `-3` dbFS if all level controls that it passes through within QLab are set to `0`. What happens

downstream of QLab cannot be known to QLab, so that `-3` dBFS signal may translate to 40 dB SPL on a very tiny sound system, or 120 db SPL on a very large sound system.

Setting the maximum volume limit

> **Note:** It is essential to understand that setting the maximum limit too high makes it possible to raise audio levels above a safe volume. Please be exceedingly careful when adjusting this control, and indeed when first playing back any sound with a new system.

There is no specific rule or guideline for setting the maximum volume limit, but it should be set as low as possible while allowing for its use.

One productive way to set this limit is to take the quietest audio file that you plan to use in your show, and play it as loudly as you can imagine wanting to. Then, set the maximum volume limit to that level. That said, if you're also using loud audio files in your show, setting a higher maximum limit opens the door to a little danger.

Setting the minimum volume limit

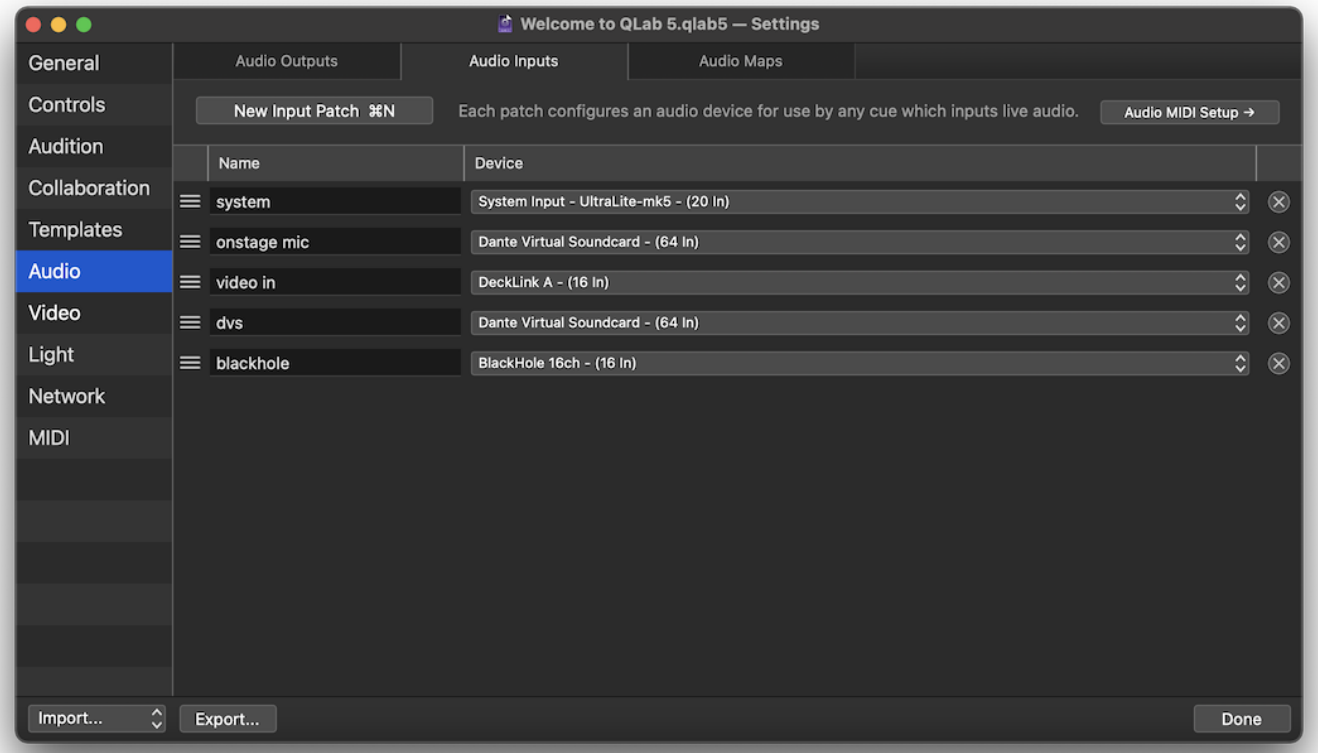
When QLab is used in a sound system that has been configured for optimal dynamic range, properly setting the minimum volume limit allows smooth fades without the sound dropping out at the end of a fade, or popping in at the beginning. Gain staging is a complex topic beyond the scope of this manual, but we can address the QLab part of the equation.

1. Arrange for some time to have full acoustic control of the space that you're working in. You will need as much quiet as possible for this process.
2. Get your sound system up and running and adjusted to your preference.
3. In Workspace Settings → Audio → Audio Outputs, set the minimum volume limit to `-180` dB.
4. In the cue list, create an Audio cue and target either a test file such as pink noise or a 1 kHz tone, or a piece of music that has relatively even loudness throughout.
5. Start the cue at a reasonable listening level, then slowly drag the main level control of the Audio cue down, getting quieter and quieter.
6. When you feel you no longer hear the sound, stop dragging the fader and walk around the space listening.
7. If you discover you can hear the sound anywhere, go back and drag it down a little more. Repeat until you are certain that the sound is inaudible.
8. Whatever the main level control is set to on that cue is your actual minimum volume. Enter that level as the minimum volume limit.

Now, when you fade an Audio cue in from silence or fade out to silence, QLab will be using the full range of volume available in your sound system.

The Audio Inputs tab

An **audio input patch** is the mechanism that QLab uses to route live audio from external sources into Mic and Camera cues.



Workspaces in QLab 5 can contain any number of audio input patches. You can add an audio input patch by clicking on the **New Input Patch** button or by using the keyboard shortcut **⌘N** while the Audio Input tab is open and in the foreground. A newly created blank workspace will automatically start off with a single audio input patch for each audio input device connected to your Mac at the moment you create the workspace.

Audio devices must have input channels to work with audio input patches.

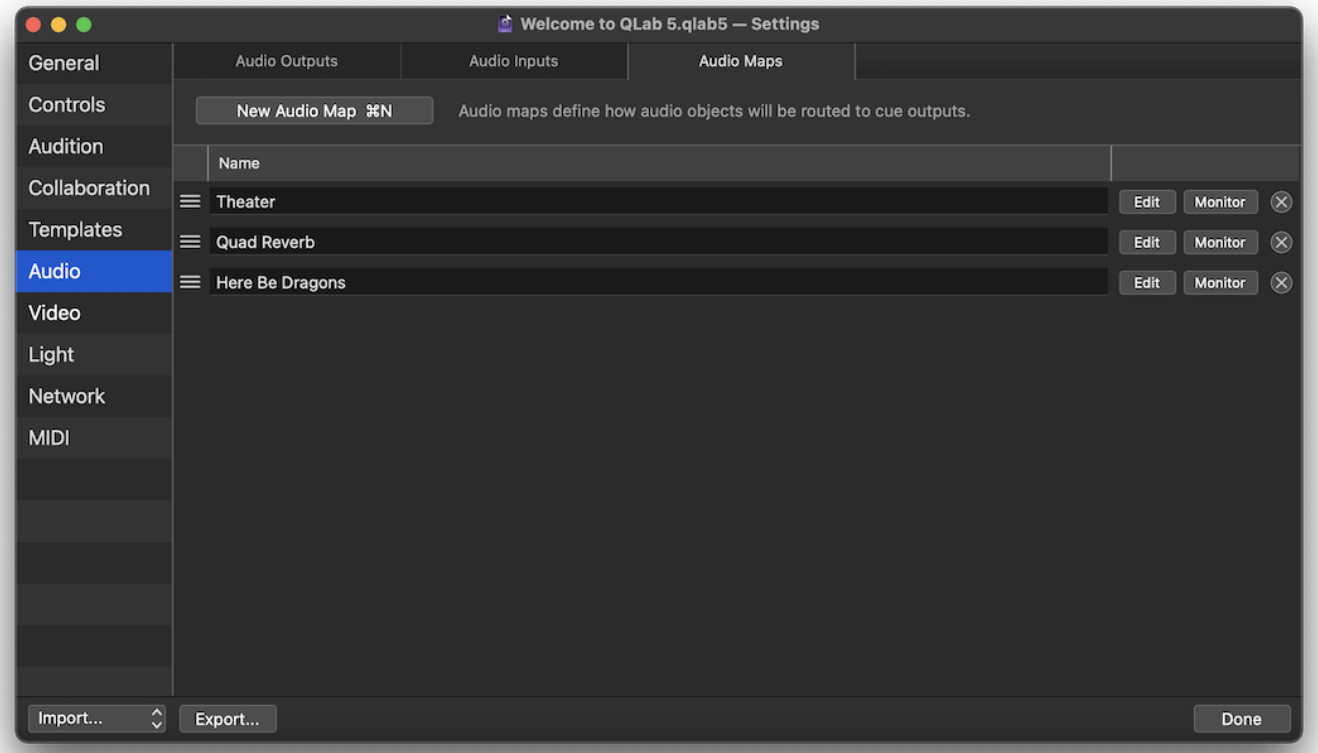
Several actions are available in the patch table:

- **Delete** a patch by clicking on the ⊗ button on the right side of the row for that patch.
- **Reorder** a patch by dragging it up or down in the patch list. You can click anywhere on the background of the row for that patch, but it's easiest to click on the left side where the ≡ icon is displayed.
- **Duplicate** a patch by holding down the option key (⌥) and dragging the patch up or down in the patch list.
- **Copy** a patch into another workspace by dragging it into that workspace's patch list.
- **Export** a patch by dragging it into the Finder.

You can hold down the shift key (⇧) while clicking to select a range of patches, or the command key (⌘) while clicking to select two or more patches, allowing you to act on all the selected patches at once.

The Audio Maps tab

Audio maps are one of the core components of object audio.



Workspaces in QLab 5 can contain any number of audio maps. You can add an audio map by clicking on the **New Audio Map** button or by using the keyboard shortcut ⌘**N** while the Audio Maps tab is open and in the foreground.

Several actions are available in the map table:

- **Delete** a map by clicking on the ⊗ button on the right side of the row for that patch.
- **Reorder** a map by dragging it up or down in the list. You can click anywhere on the background of the row for that map, but it's easiest to click on the left side where the ≡ icon is displayed.
- **Duplicate** a map by holding down the option key (⌥) and dragging the patch up or down in the list.
- **Copy** a map into another workspace by dragging it into that workspace's map list.
- **Export** a map by dragging it into the Finder.

You can hold down the shift key (⇧) while clicking to select a range of maps, or the command key (⌘) while clicking to select two or more maps, allowing you to act on all the selected maps at once.

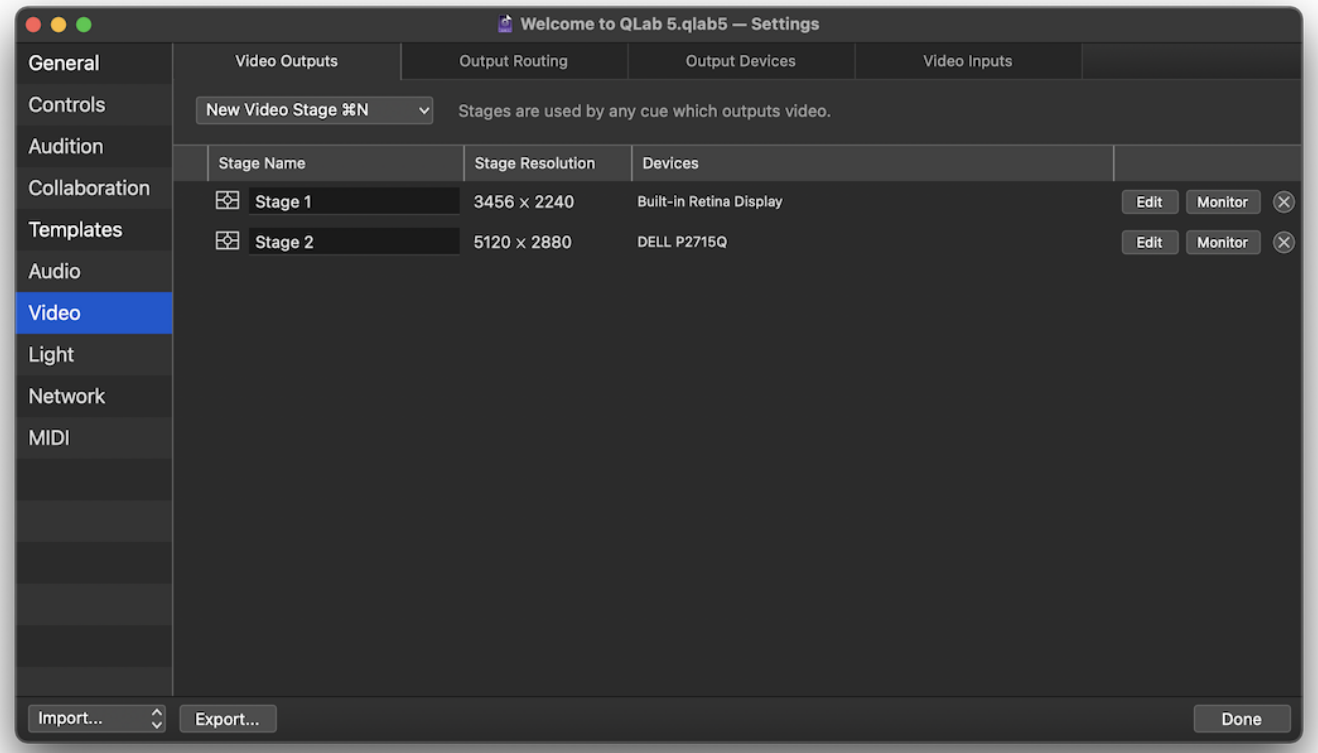
You can learn about editing audio maps from the Audio Map Editor section of this manual.

Video

The four tabs in **Video** settings pertain to video input and output patching, and the configuration of video output devices.

The Video Outputs Tab

A **stage** is the mechanism that QLab uses to route video from cues to your video output hardware, as well as to non-physical video outputs via Syphon and NDI. Any cue which can generate video (🎞 Video cues, 📹 Camera cues, and 𝑇 Text cues) must be assigned to a single video output stage, and the configuration of that stage defines how the output from that cue will behave.

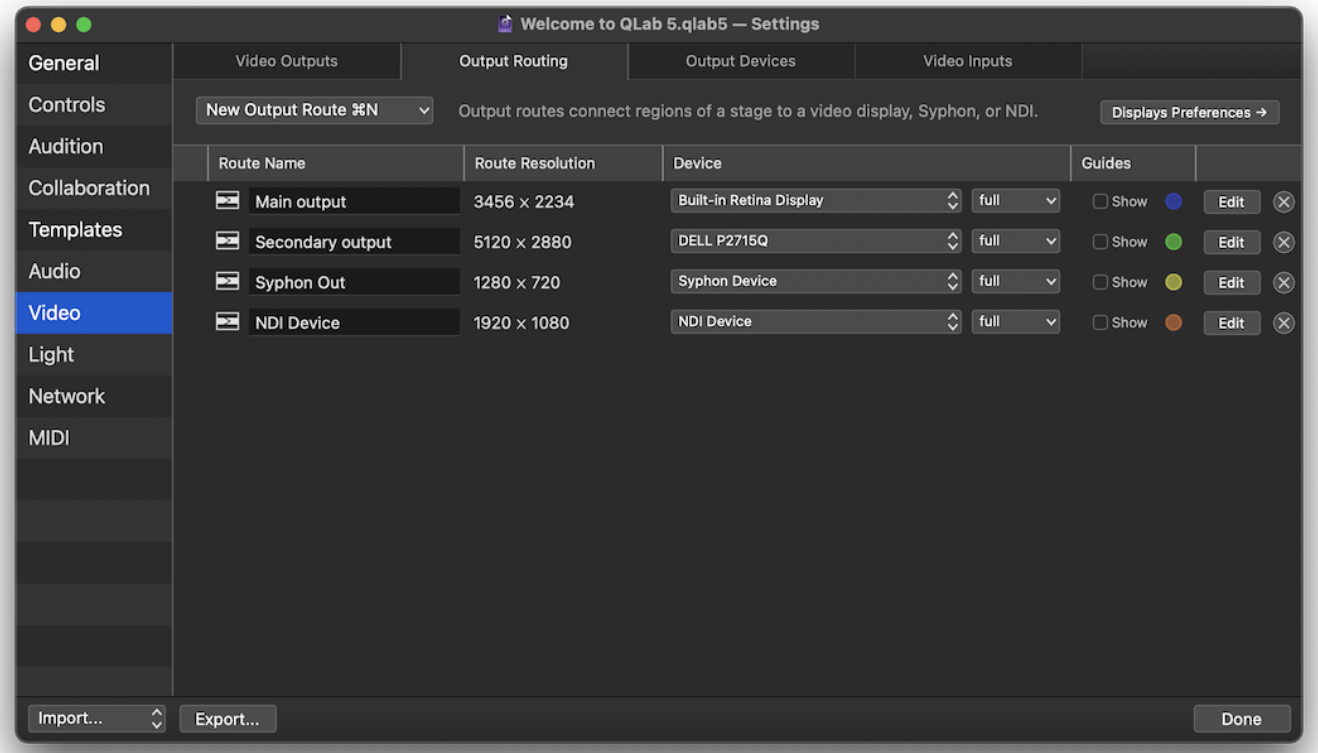


Workspaces in QLab 5 can contain any number of stages. You can create a new stage by clicking on the **New Video Stage** button or by using the keyboard shortcut ⌘**N** while the Video Outputs tab is open and in the foreground. A newly created blank workspace will automatically start off with a single stage for each video device connected to your Mac at the moment you create the workspace.

You can learn more about this tab from the Video Output section of this manual.

The Output Routing Tab

A **route** is a connection between a region of a stage and an output device such as a projector, screen, Syphon output, or NDI send. Because a route exists independently from both the stage and the output, you can make adjustments to the route or to the output device that the route uses without the stage "knowing" about it. This allows you to handle system changes without having to make cueing changes.

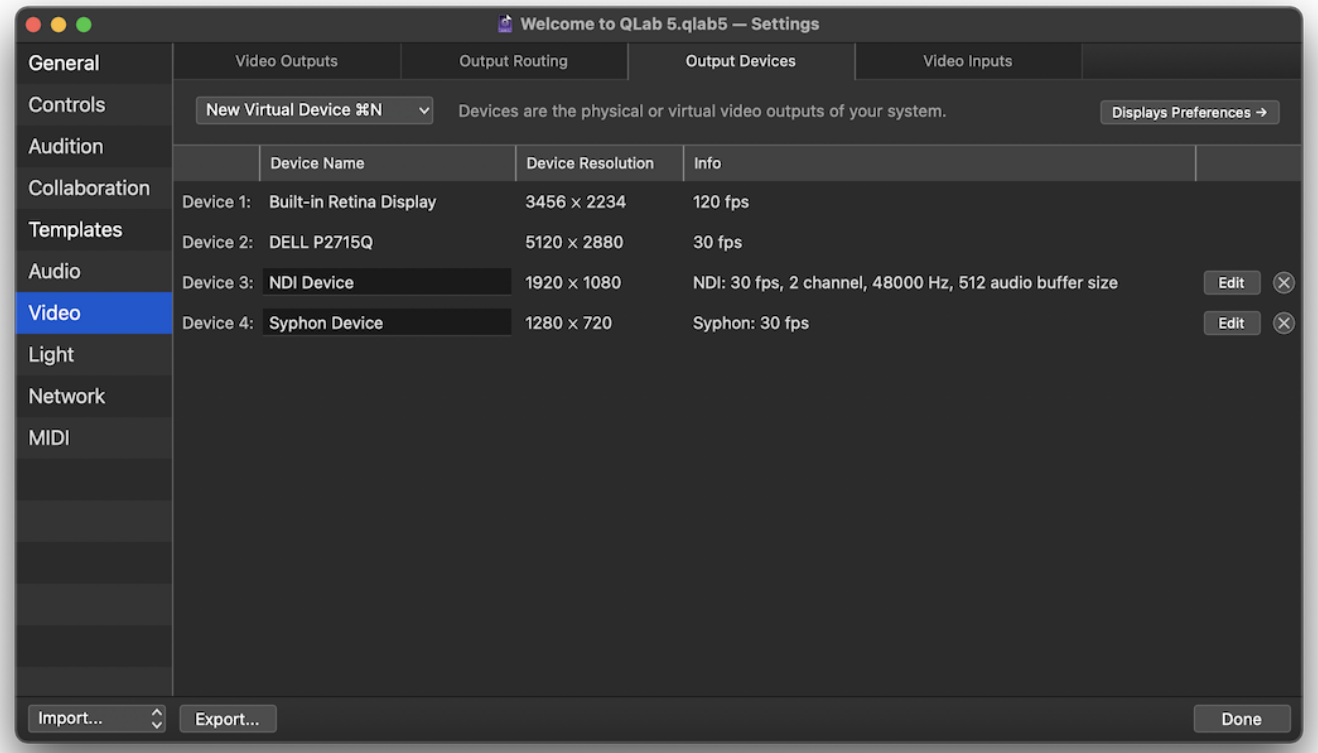


You can create a new route by clicking on the **New Output Route** button or by using the keyboard shortcut **⌘N** while the Output Routing tab is open and in the foreground. A newly created blank workspace will automatically start off with a single route for each video device connected to your Mac at the moment you create the workspace.

You can learn more about this tab from the Video Output section of this manual.

The Output Devices Tab

A **device** is a physical or virtual endpoint for video. Displays built into or connected to your Mac are devices, as are video projectors, LED wall controllers, and televisions. Compatible Blackmagic PCI cards and USB-connected boxes are devices too; Blackmagic cards with multiple independent outputs are actually multiple devices built into one. Finally, each Syphon or NDI output is its own device.



The Output Devices tab lists all the devices available to your workspace. Virtual devices and Blackmagic Designs devices can be adjusted using the **Edit** button on the right side of the list. Virtual devices can be deleted from the workspace by using the ⊗ button.

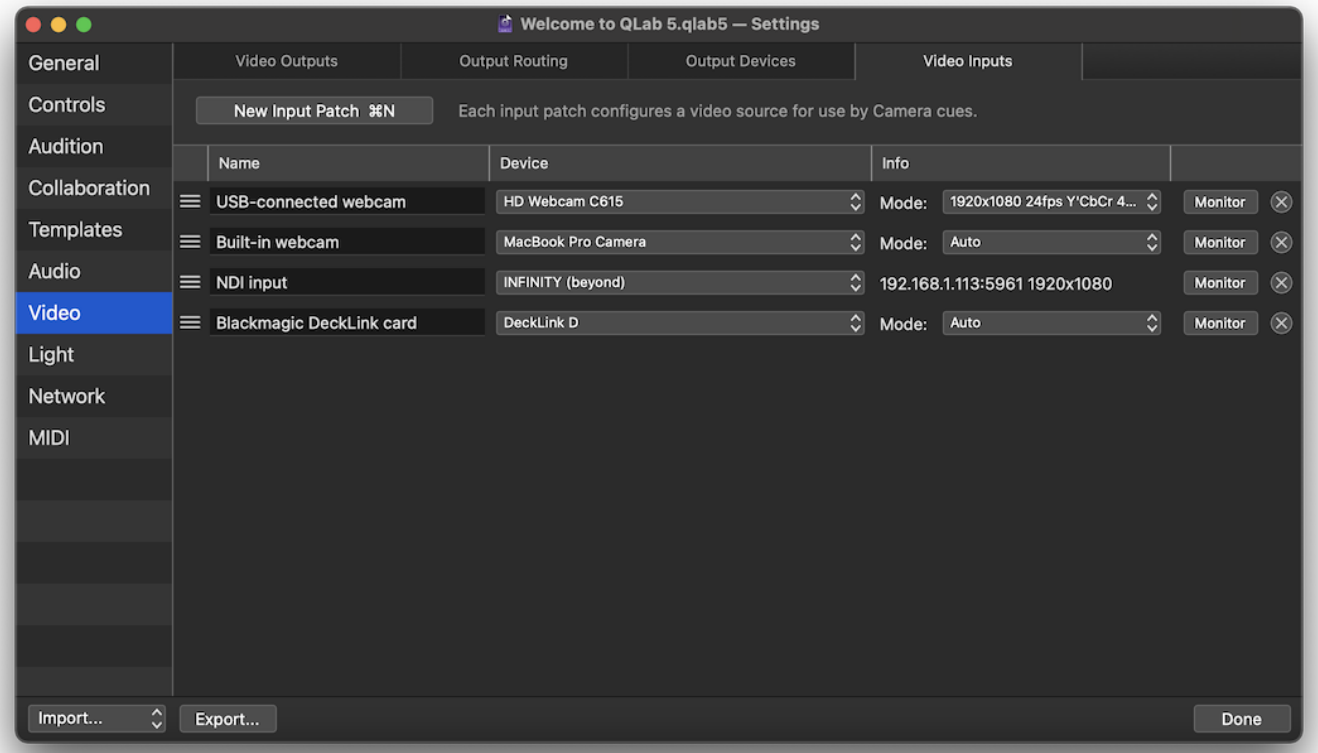
You can create a new Syphon or NDI device by clicking on the **New Virtual Device** button or by using the keyboard shortcut **⌘N** while the Output Devices tab is open and in the foreground.

The Video Output Device table consists of four columns.

1. **Device name** is inherent to physical devices, and editable for virtual devices.
2. **Device resolution** displays the pixel dimensions of the video device's raster as width × height.
3. **Info** displays any other available information about the device. This always includes refresh rate, but can also include other information depending upon the type of device.
4. The **Edit** button, which is only available for Blackmagic devices and virtual devices, allows you to adjust the parameters of the device which vary depending on the type of device. The ⊗ button, which is only available for virtual devices, deletes the device when clicked.

The Video Inputs Tab

A **video input patch** is the mechanism that QLab uses to route live video from external sources into Camera cues.



Workspaces in QLab 5 can contain any number of video input patches. You can add an video input patch by clicking on the **New Input Patch** button or by using the keyboard shortcut **⌘N** while the Video Input tab is open and in the foreground. A newly created blank workspace will automatically start off with a single video input patch for each video input device connected to your Mac at the moment you create the workspace.

Several actions are available in the patch table:

- **Delete** a patch by clicking on the ⊗ button on the right side of the row for that patch.
- **Reorder** a patch by dragging it up or down in the patch list. You can click anywhere on the background of the row for that patch, but it's easiest to click on the left side where the ≡ icon is displayed.
- **Duplicate** a patch by holding down the option key (⌥) and dragging the patch up or down in the patch list.
- **Copy** a patch into another workspace by dragging it into that workspace's patch list.
- **Export** a patch by dragging it into the Finder.

You can hold down the shift key (⇧) while clicking to select a range of patches, or the command key (⌘) while clicking to select two or more patches, allowing you to act on all the selected patches at once.

The Video Input Patch table consists of four columns.

1. **Name** is optional and can be any text. Note that if you plan to use OSC to control a patch by name, you must give the patch an OSC-compatible name, i.e. one without spaces or special characters reserved by the OSC spec.
2. **Device** lets you select the video input device to use with the patch. QLab can use any USB video device class webcam or input device (i.e. most normal webcams), any IIDC-compliant camera or input device (these are typically referred to as DV devices), any Blackmagic Design DeckLink, UltraStudio, or Intensity device, any Syphon source on your Mac, or any NDI source available to your Mac via its local network.
3. **Info** displays the IP address and port number of NDI sources, or the *Mode* menu for configuring the resolution and frame rate of webcams and Blackmagic devices. For Syphon sources and webcams with un-changeable settings, this column shows nothing.
4. The **Monitor** button opens a monitor window which lets you view the live input from the patch independently of any Camera cue. The ⊗ button deletes the patch when clicked.

Light

The three tabs in **Light** settings pertain to light patching, instrument definitions, and MIDI control of the Light Dashboard.

The Light Patch tab

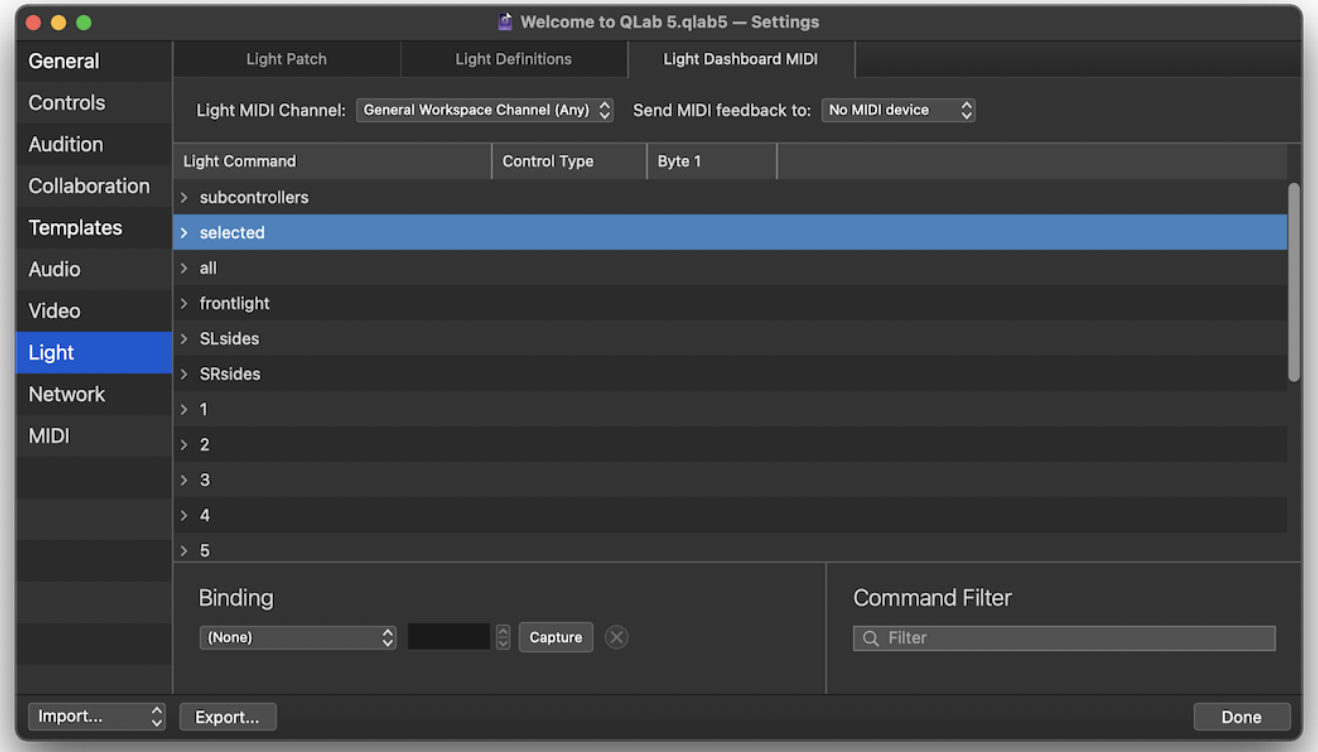
The Light Patch tab allows you to view and edit the instruments and light groups in the workspace. You can learn more about the Light Patch tab in the Light Patch Editor section of this manual.

The Light Definitions tab

The Light Definitions tab lists every light definition used in the workspace, and allows you to edit, copy, and delete definitions in the workspace. You can learn more about the Light Definitions tab in the Light Patch Editor section of this manual.

The Light Dashboard MIDI tab

This tab allows you to map incoming MIDI messages to light instrument parameters in the Light Dashboard. This lets you, for example, use MIDI control surfaces to directly control lights in QLab.



Light MIDI Channel can be set to follow the workspace MIDI channel, any single channel, or to "any". The Light Dashboard will listen on the specified channel for MIDI messages that have been assigned to the commands below.

Send MIDI feedback allows MIDI devices such as motorized fader banks to follow along. QLab will send MIDI messages to the specified device which match the MIDI messages that have been assigned below.

The Light Command table

This table contains one row for each instrument and light group in the workspace, as well as two special-case rows at the top, which will be discussed in a moment. As you add instruments and light groups to the workspace, this table will be automatically updated. Rows can be expanded to show the parameters they contain.

The first special-case row is the **subcontrollers** row, which contains any lighting subcontrollers in the workspace.

The second special-case row is the **selected** row, which represents whichever instruments and groups are selected in the Light Dashboard at a given moment. Any MIDI commands mapped to parameters in this row will be directed to the selected instruments and light groups. This allows you to flexibly use a MIDI controller to work with your lights, even if you have only a small controller and many lights.

After the special-case rows are rows for instruments and light groups.

When any row is selected, you can either manually assign a MIDI message to it using the controls under the **Binding** heading below, or you can click the **Capture** button and QLab will listen for the next incoming MIDI message on the MIDI channel specified at the top of the window.

You can assign a MIDI message to individual parameters within an instrument or light group, and you can also assign a message to the instrument or light group itself, in which case the message will be directed to the default parameter of that instrument or light group.

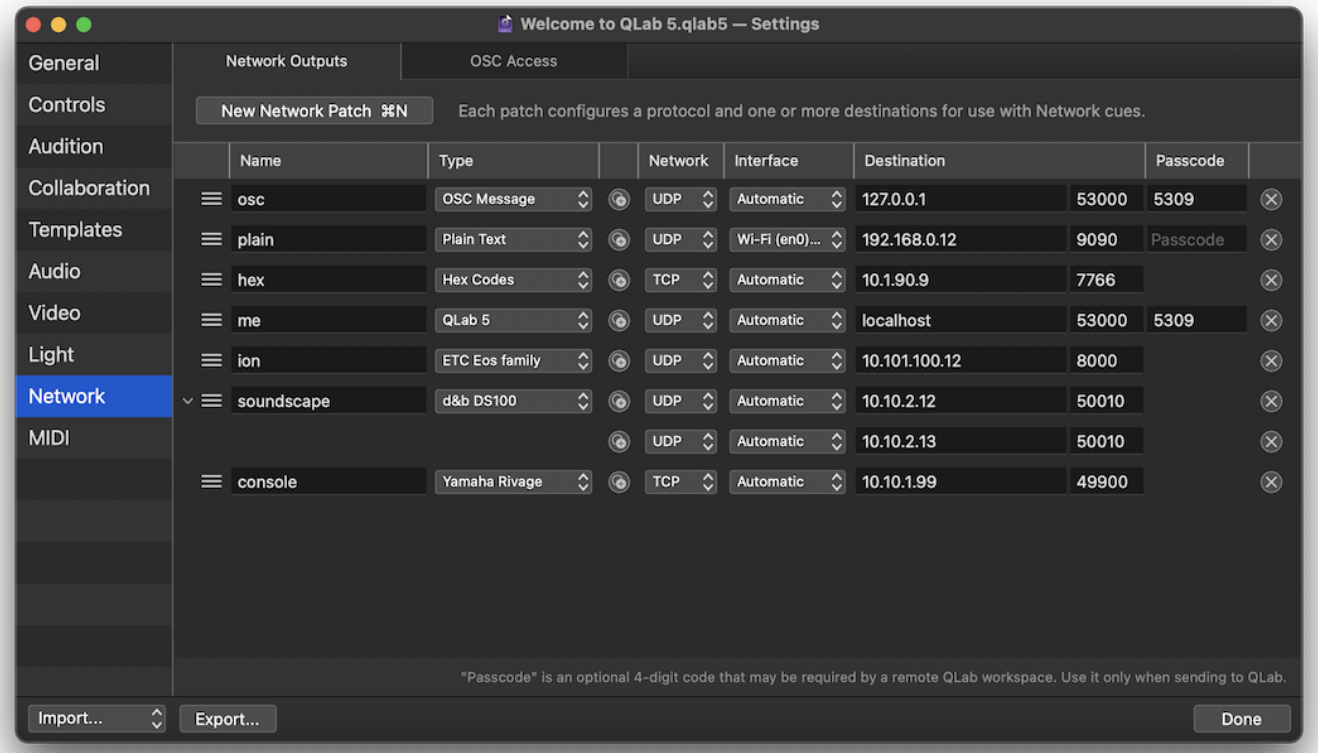
The **Command Filter** lets you search for the name of an instrument or light group, and temporarily hide everything else. This can make it easier to work with a complex workspace.

Network

The two tabs in **Network** settings pertain to Network cues, which send OSC and other network messages, and to settings that control how your workspace can be accessed over a network using OSC and plain text commands.

The Network Outputs tab

A network patch is the mechanism that QLab uses to route messages from Network cues to their destinations, both within the same computer that QLab is running on, and other devices on the network. Every Network cue must be assigned to a single network patch, and the configuration of the patch defines how the output from that cue will behave.



Workspaces in QLab 5 can contain any number of network patches. You can add a network patch by clicking on on the button labeled *New Output Patch* or by using the keyboard shortcut **⌘N** while the Network Outputs tab is open and in the foreground. A newly created blank workspace will automatically start off with a single network patch preconfigured to allow QLab to send OSC messages to itself.

Several actions are available in the patch table:

- **Delete** a patch by clicking on the ⊗ button on the right side of the row for that patch.
- **Reorder** a patch by dragging it up or down in the patch list. You can click anywhere on the background of the row for that patch, but it's easiest to click on the left side where the ≡ icon is displayed.
- **Duplicate** a patch by holding down the option key (⌥) and dragging the patch up or down in the patch list.
- **Copy** a patch into another workspace by dragging it into that workspace's patch list.
- **Export** a patch by dragging it into the Finder.

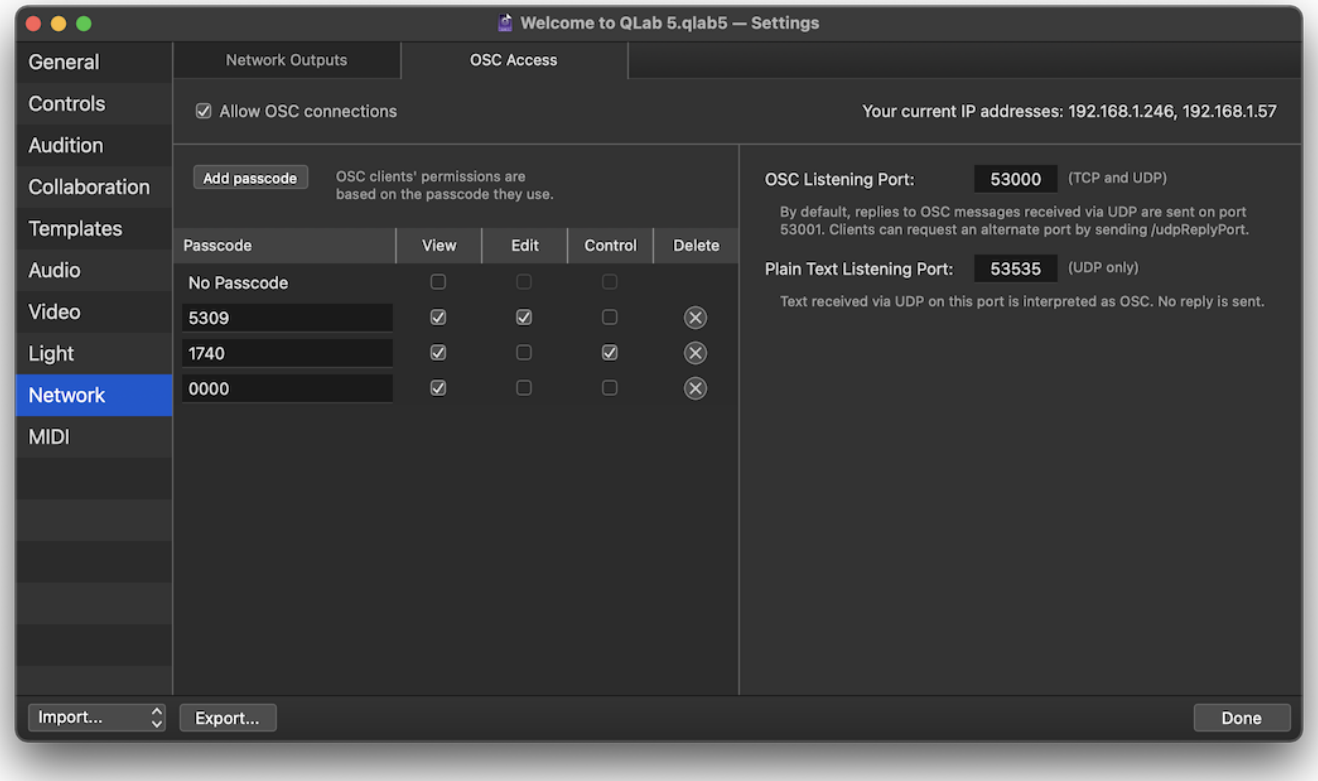
You can hold down the shift key (⇧) while clicking to select a range of patches, or the command key (⌘) while clicking to select two or more patches, allowing you to act on all the selected patches at once.

The patch table consists of nine columns.

1. The unlabeled column containing a ≡ button lets you drag patches to reorder them, copy them into other workspaces, or drag them to the Finder to save as a workspace settings file.
2. **Patch name** is optional and can be any text.
3. **Type** allows you to configure the patch to send generic OSC messages, plain text messages, or OSC messages using QLab's built in library of network device definitions. You can learn more about the various network device definitions available and how to use them in the Network cues section of this manual..
4. The unlabeled column containing a ⊕ button lets you add destinations to a patch, each with its own network, interface, IP address, port number, and passcode setting.
5. **Network** lets you select TCP or UDP transport for the patch.
6. **Interface** lets you select a specific network interface for the patch to use, or "automatic" to allow macOS to route messages using that patch. Automatic routing usually works, but if your Mac is connected to multiple networks using similar IP address schemes, selecting a specific interface may be necessary.
7. The **Destination** column contains both the IP address (such as `192.168.1.12`) or mDNS name (such as `qlab-computer.local`) of the destination device and the port number of the destination device. You can enter the word `localhost` in place of an IP address to have the patch route messages back into QLab with the lowest possible latency. These messages never leave QLab. You can enter `127.0.0.1` to route messages to other software on the same Mac.
8. **Passcode** allows you to use the patch to send messages to a QLab workspace that has a passcode set. This code is QLab-specific and should only be set when necessary.
9. The ⊗ button deletes the patch when clicked.

The OSC Access tab

This tab allows you to define how your workspace can be accessed via Open Sound Control (OSC) and plain text messages sent over UDP.



The upper left corner has a checkbox which turns OSC access on or off. When this box is unchecked, incoming OSC and plain text messages will have no effect on the workspace. When the box is checked, incoming OSC and plain text messages will be admitted based on the rest of the settings in this tab.

The upper right corner of the tab displays the current IP address or addresses of your Mac.

The OSC access table

Any program or device that sends OSC messages can communicate with your workspace using QLab's extensive OSC dictionary. QLab 5 allows you to set the scope of access for OSC connections using three permission switches:

- **View** access allows clients to view information about the workspace.
- **Edit** access allows clients to create and delete cues, change parameters of cues, and adjust workspace settings.
- **Control** access allows clients to move the playhead and start and stop cues.

The OSC access table allows you to configure the use of these three levels of permission for incoming OSC.

The first row in the OSC access table is labeled *No Passcode* and the boxes checked here set the access permission for OSC clients that connect to QLab without sending a passcode. If all three boxes are unchecked, which is the default, clients will not be able to connect without a passcode.

New workspaces have a single randomly generated passcode configured to allow full access. You can configure this passcode as needed using the view, edit, and control checkboxes, and change the passcode itself if you like. Passcodes must be four digit numbers.

The **Add Passcode** button adds a row to the table with a new, randomly generated passcode which you can, of course, edit as you prefer.

You can delete a passcode using the ⊗ button on the right side of the table.

Workspaces can have as many passcodes as you like. You can learn about how to connect to a QLab workspace using a passcode from this section of the OSC dictionary.

Listening Ports

To the right of the OSC access table are two text fields which allow you to define the ports that your workspace uses to listen for incoming messages. Ports are like doors in a computer's network connection. Each separate type of message needs its own port.

The **OSC listening port** is used for OSC messages sent using either TCP or UDP connections. The default port is `53000`, and you can change it to any valid port number, which is any whole number from 1 to 65,535. Be advised, though, that other software and network protocols may have a claim on certain port numbers, so do your homework before choosing a port. Generally speaking, one- and two-digit port numbers are spoken for. When messages come in via UDP, QLab sends replies on port `53001`, no matter what port you choose here for listening. Individual clients can request a custom UDP reply port by sending the message `/udpReplyPort x` where `x` is the requested port number.

The workspace also receives plain text via UDP using the **plain text listening port**, which is `53535` by default, and attempts to interpret it as OSC. For example, sending the text `/cue/selected/start` via UDP to QLab on this port will have the same result as sending the actual OSC command `/cue/selected/start` to the OSC listening port. No replies are sent to plain text messages.

When using plain text messages, it's wise to learn about and use the `/forgetMeNot` OSC message, which allows UDP clients to remain connected to QLab without having to re-send a passcode once per minute.

It's important to remember that QLab itself always listens on port `53000`. Any application-level OSC messages can still be received by QLab on port `53000` even if the workspace OSC listening port is changed.

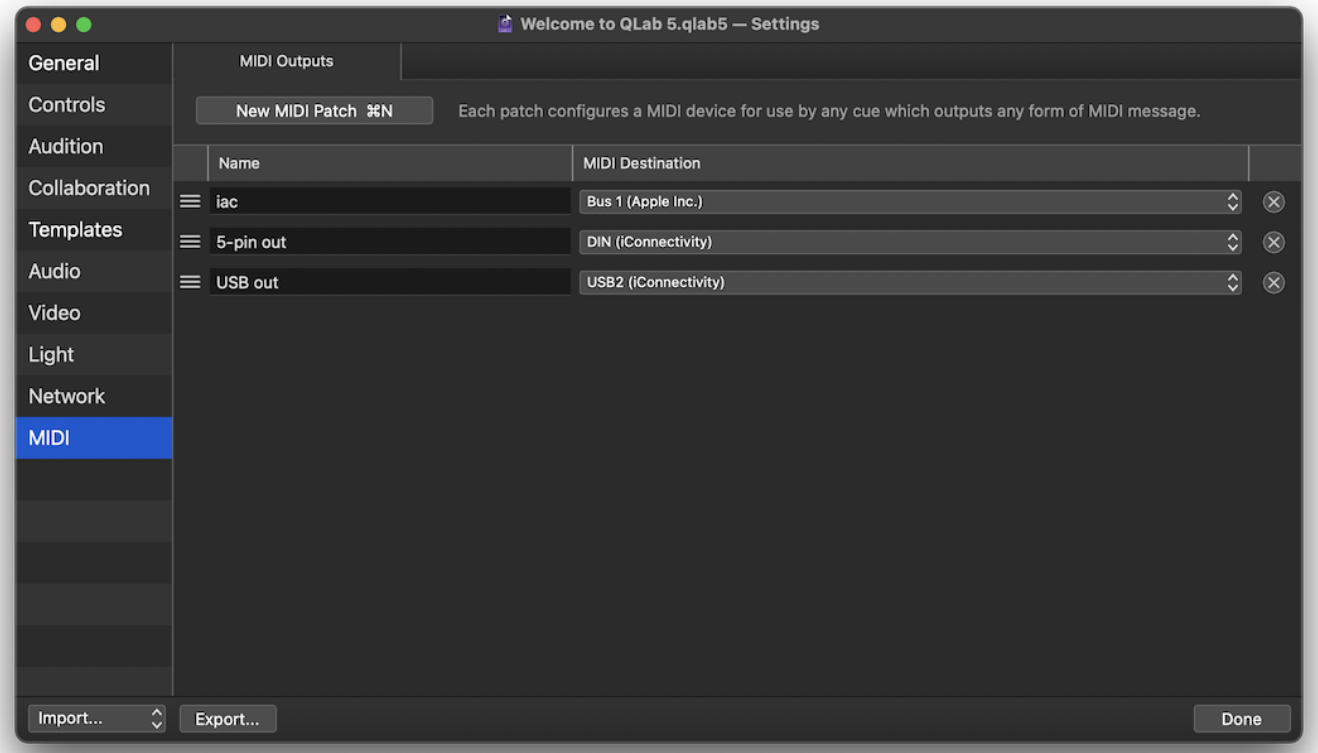
If you are using NDI for video input or output, QLab will warn you of port number collisions to the best of its ability. Nevertheless, you should take care to only use port numbers that you know are available.

MIDI

The two tabs in MIDI settings pertain to MIDI and MIDI File cues, which send MIDI Voice, MIDI Show Control, and MIDI System Exclusive (SysEx) messages, and to QLab's MSC show control broadcast feature.

The MIDI Outputs Tab

A MIDI patch is the mechanism that QLab uses to route messages from MIDI cues, MIDI File cues, Timecode cues using MTC, and MSC show control broadcast messages to their destinations. Every cue that sends MIDI must be assigned to a single MIDI patch, and the configuration of the patch defines how the output from that cue will behave.



Workspaces in QLab 5 can contain any number of MIDI patches. You can add a MIDI patch by clicking on on the button labeled *New Output Patch* or by using the keyboard shortcut **⌘N** while the MIDI Outputs tab is open and in the foreground. A newly created blank workspace will automatically start off with a single MIDI patch for each MIDI output available on your Mac at the moment you create the workspace.

Several actions are available in the patch table:

- **Delete** a patch by clicking on the ⊗ button on the right side of the row for that patch.
- **Reorder** a patch by dragging it up or down in the patch list. You can click anywhere on the background of the row for that patch, but it's easiest to click on the left side where the ≡ icon is displayed.
- **Duplicate** a patch by holding down the option key (⌥) and dragging the patch up or down in the patch list.
- **Copy** a patch into another workspace by dragging it into that workspace's patch list.
- **Export** a patch by dragging it into the Finder.

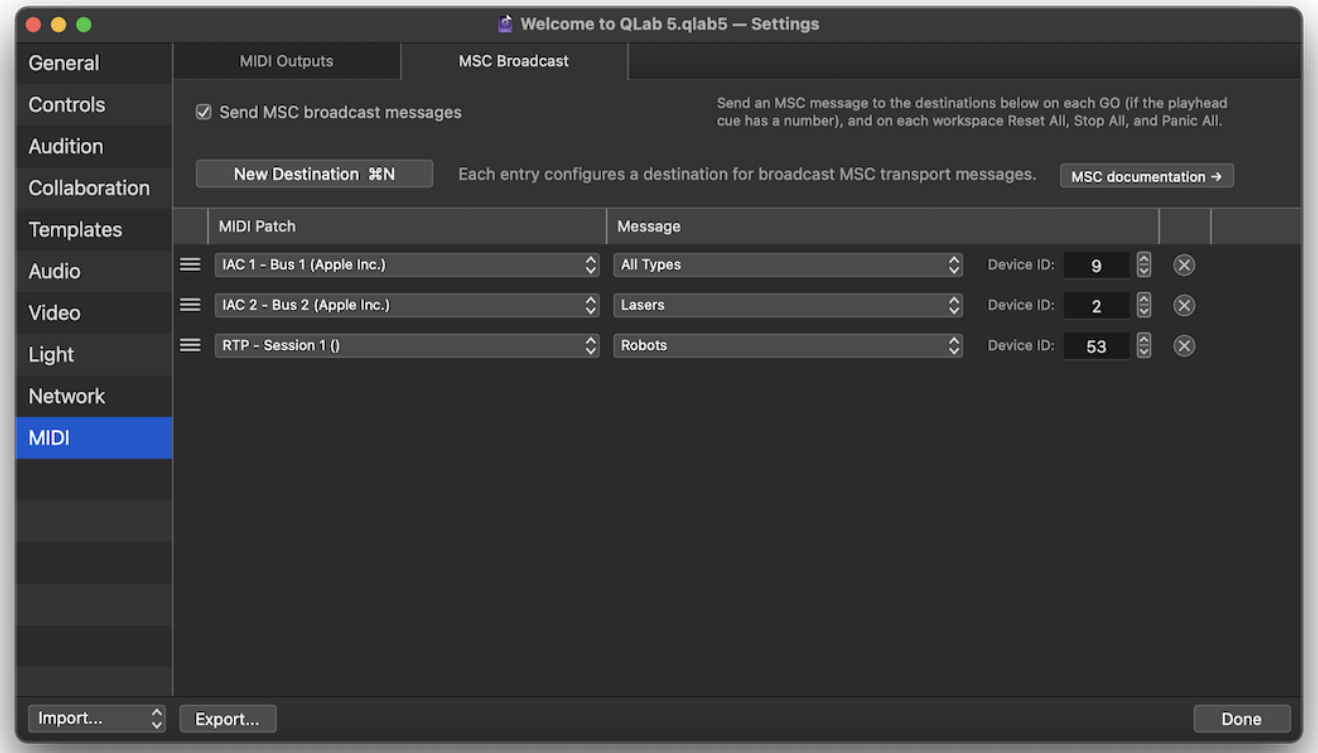
You can hold down the shift key (⇧) while clicking to select a range of patches, or the command key (⌘) while clicking to select two or more patches, allowing you to act on all the selected patches at once.

You can optionally give a name to each MIDI patch, which can be any text.

The pop-up menu for each patch displays all available MIDI destinations available on your Mac.

The MSC Broadcast Tab

This tab allows you to configure MSC broadcast outputs which allow other programs or devices to synchronize starting and stopping cues along with QLab.



Workspaces in QLab 5 can contain any number of MSC Broadcast destinations. You can add a destination by clicking on the **New Destination** button or by using the keyboard shortcut **⌘N** while the MSC Broadcast tab is open and in the foreground.

Several actions are available in the destinations table:

- **Delete** a destination by clicking on the ⊗ button on the right side of the row for that destination.
- **Reorder** a destination by dragging it up or down in the destination list. You can click anywhere on the background of the row for that destination, but it's easiest to click on the left side where the ≡ icon is displayed.
- **Duplicate** a destination by holding down the option key (⌥) and dragging the destination up or down in the destination list.
- **Copy** a destination into another workspace by dragging it into that workspace's destination list.
- **Export** a destination by dragging it into the Finder.

You can hold down the shift key (⇧) while clicking to select a range of destinations, or the command key (⌘) while clicking to select two or more destinations, allowing you to act on all the selected destinations at once.

MSC destinations require a MIDI Output patch, selected with the left pop-up menu, a message format, selected with the right pop-up menu, and a device ID, which is the device ID of the device(s) that will be receiving the MSC messages.

Workspace Templates

Workspace Templates allow you to create your own set of starting conditions for new workspaces.

Creating Workspace Templates

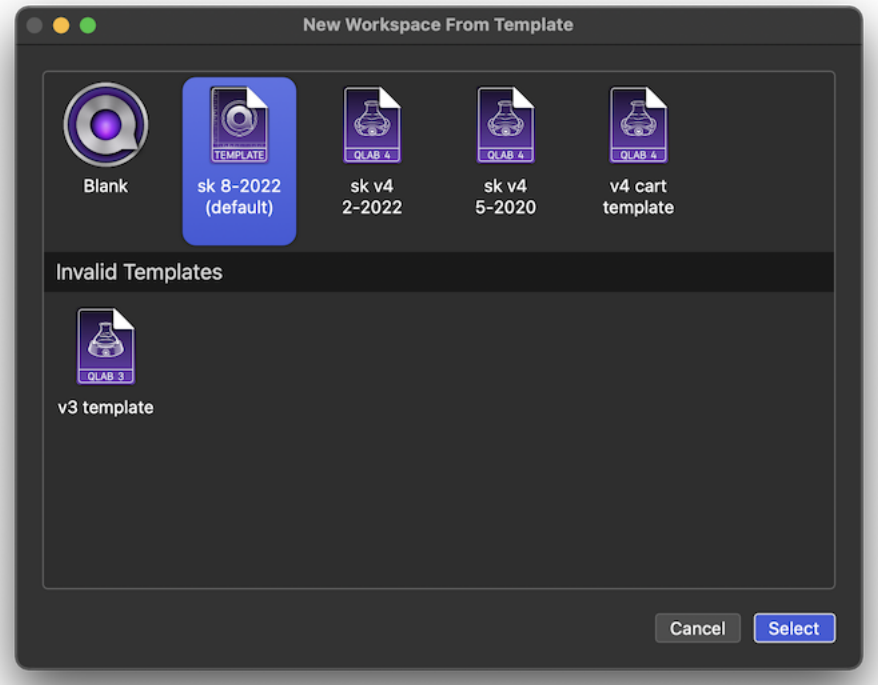
To create a workspace template, make a new workspace and adjust all its settings to suit your preference. If, for example, you prefer Fade cues to have a default duration of eight seconds, you can visit Workspace Settings → Templates → Cue Templates, select Fade from the list at the top, and set the duration to `8`.

You can also create cues, cue lists, or carts which will serve as jumping-off points. For example, if you have a set of Script cues that you routinely install into every workspace, you can place them in this workspace to include them in the workspace template.

Once you've arranged the workspace to your liking, save the workspace as a template by choosing *Save As Template* from the **File** menu. Pick any name you like. If you choose the same name as an existing template, you'll be asked if you want to replace the existing template.

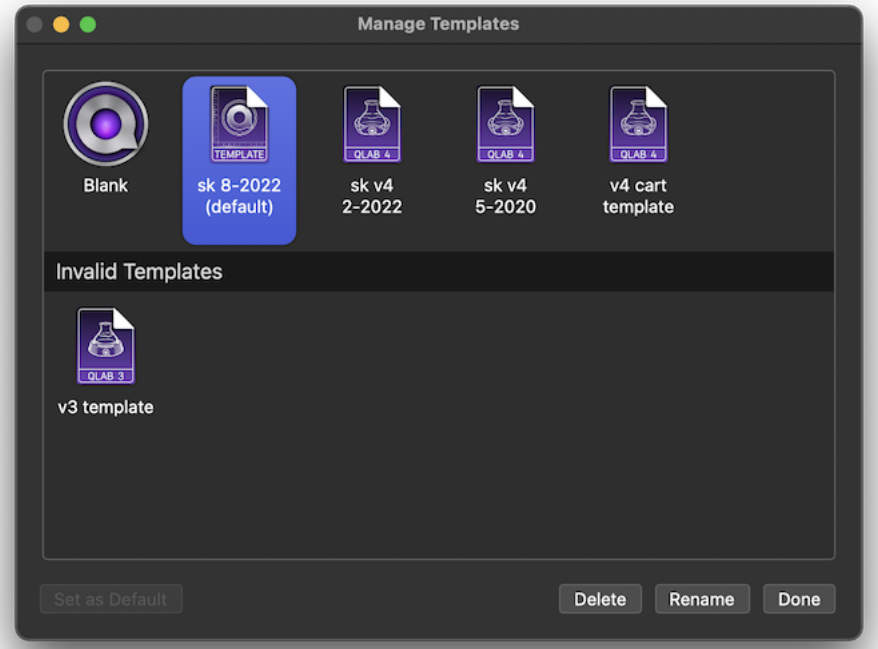
Making New Workspaces from Templates

To create a new workspace using your template, choose *New From Template* from the **File** menu, or use the keyboard shortcut ⌘⇧**N**. The Template Chooser will appear, and you can select the template you wish to use. Once you do, a new workspace will open up which is an exact copy of the template as you saved it.



Managing Templates

To delete or rename templates, or select a template as the default, choose *Manage Templates* from the **File** menu.



Whichever template you select as the default will be used whenever you choose *New Workspace* from the **File** menu, or use the keyboard shortcut ⌘**N**.

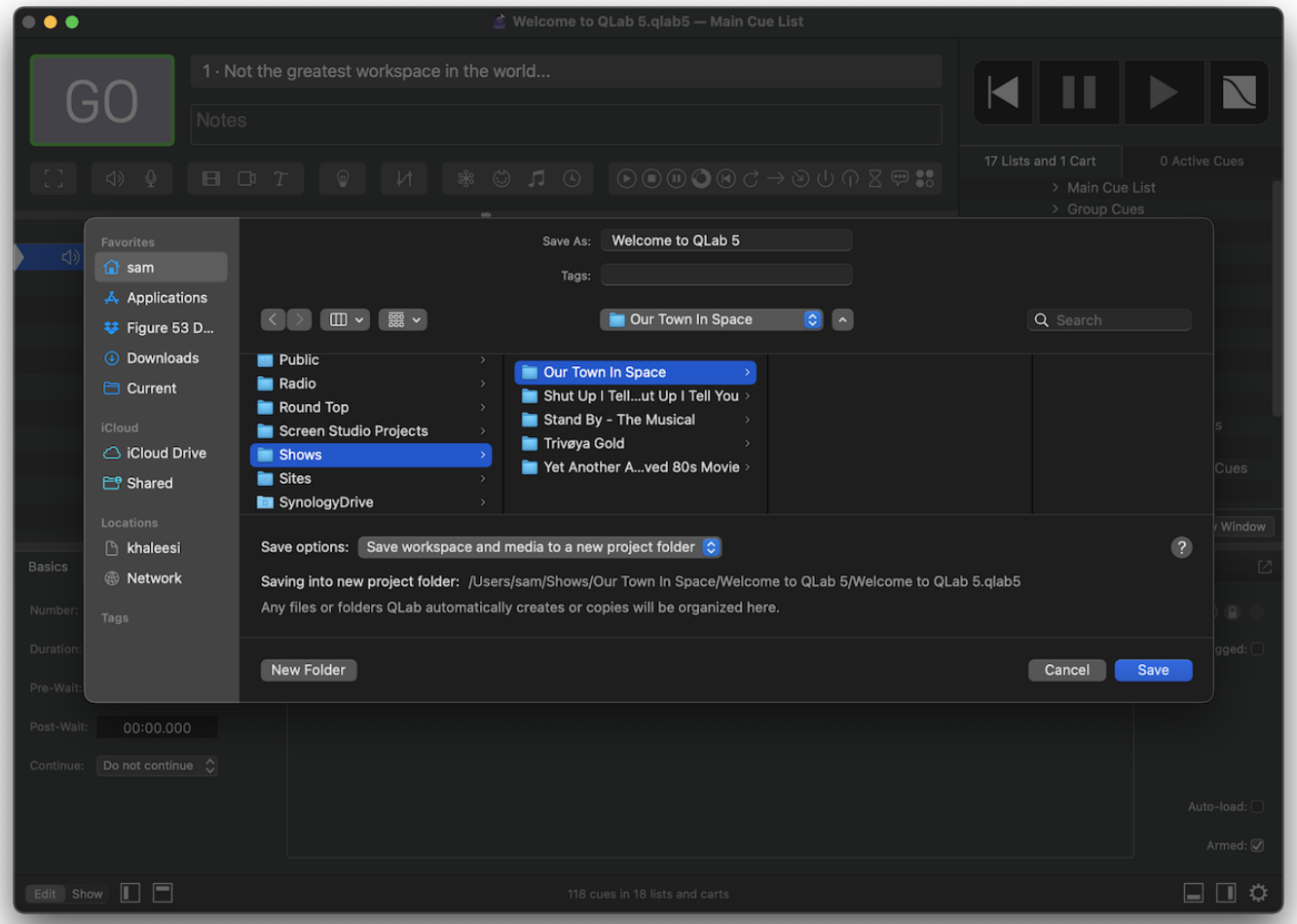
You can also right-click on a template to set it as the default, open a Finder window showing the folder that contains your templates, export the template (so you can copy it to another Mac), rename the template, or delete the template.

Files which appear to be workspace templates but which QLab cannot use, such as the QLab 3 workspace template in the screen shots above, are listed as invalid.

Managing Workspaces

When you first save a workspace in QLab 5, or when you choose *Save As...* from the **File** menu, QLab will present you with the standard macOS window that lets you choose where you want to save your workspace. Towards the bottom of that window is a pop-up menu with three choices; *Save workspace and media to a new project folder*, *Save workspace only*, and *Customize settings*.

Save workspace and media to a new project folder



When the default option of *Save workspace and media to a new project folder* is chosen, saving the workspace also causes a few other things to happen:

1. Instead of simply saving your workspace in the place that you select, QLab creates a **project folder** for your workspace. Your workspace is saved inside the project folder.
2. Any media files targeted by cues in your workspace are copied into the project folder. Audio files go into an “audio” folder, video files go into a “video” folder, and MIDI files go into a “midi” folder.
3. If the workspace is set to make backups, the “backups” folder that is created by those settings will also reside within the project folder.

Moving forward, any time you set the file target of a cue, QLab will copy that file into the workspace’s project folder, sorted into the appropriate subfolder. If there is already an identical copy of the file target in the folder, QLab will not made an additional copy but will reassign the target of of the cue to use the copy of the file that’s already in the folder.

Because of this, you can always be confident that the project folder contains all necessary file targets to run your show. If you need to move your workspace to another computer, you can simply copy the project folder and be confident that all necessary media targets will come along.

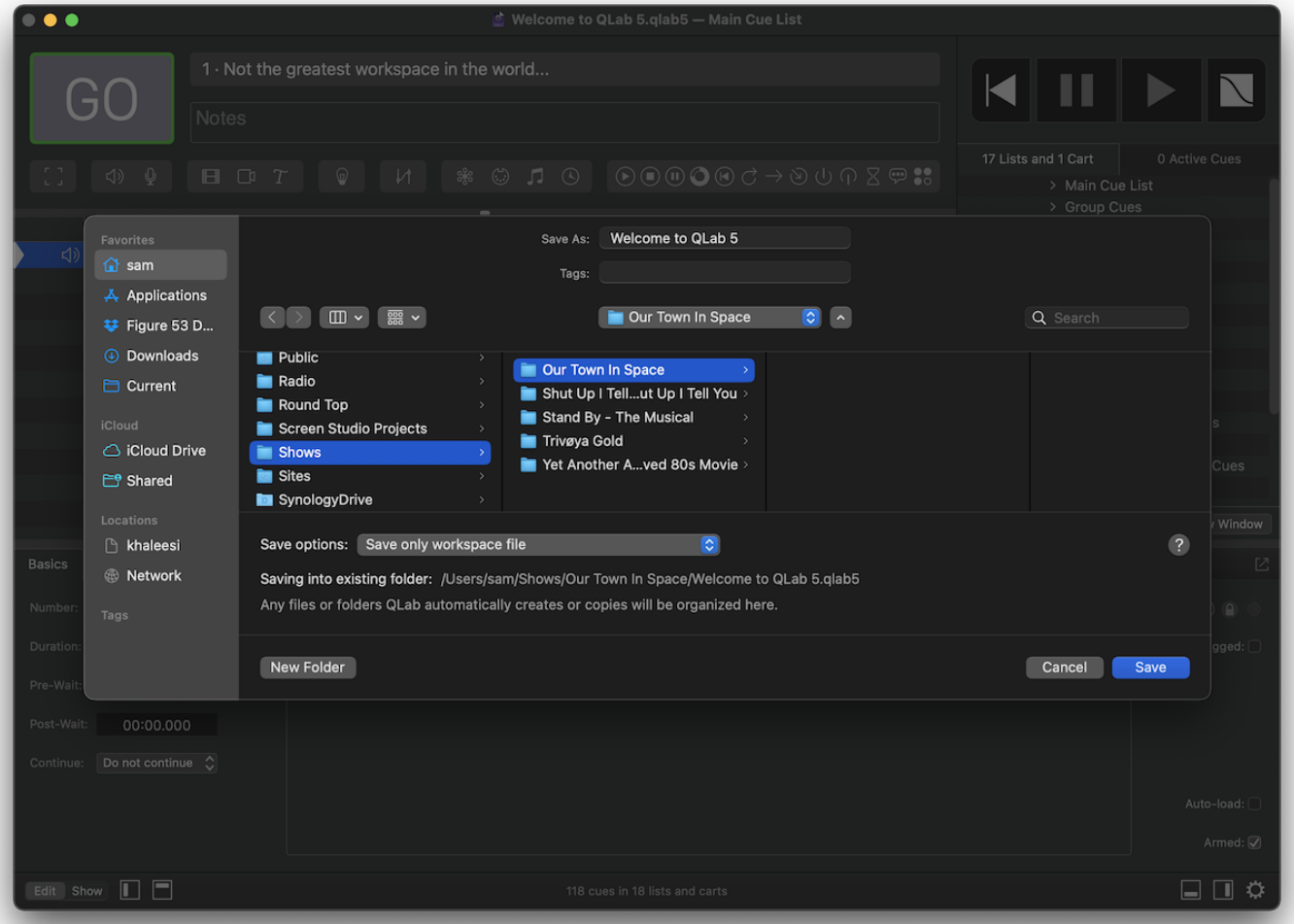
There are two types of external files that QLab *cannot* copy, however, and it's important to be aware of them:

- QLab cannot copy fonts used by Text cues.
- QLab cannot copy AudioUnits used for audio effects.

You'll need to be sure to manually install any fonts or AudioUnits that your workspace needs.

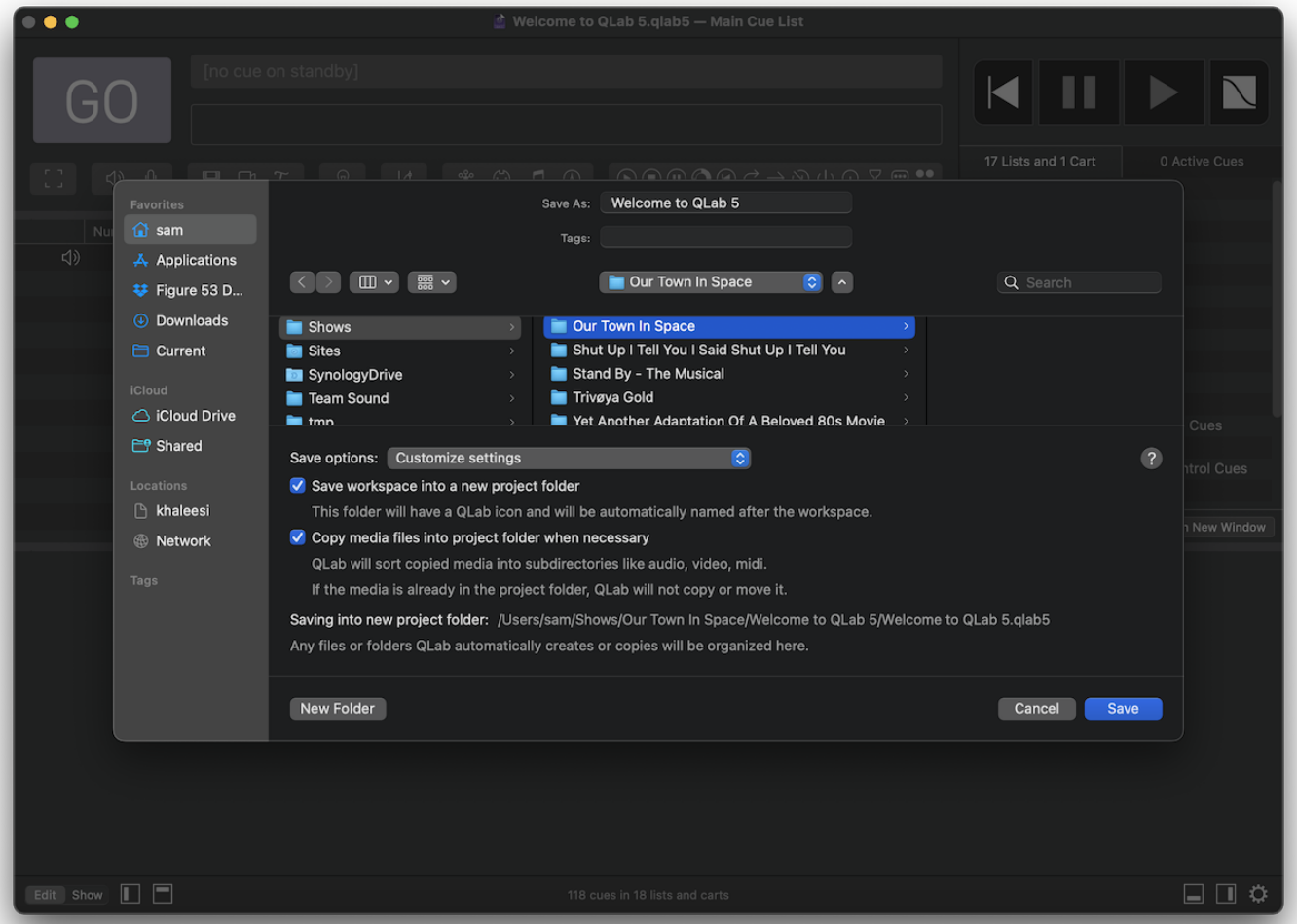
Save workspace only

If you select *Save workspace only* from the pop-up menu, QLab will save your workspace in the folder that you select, and do nothing else. If your workspace is set to copy files into project folder when adding to workspace and/or make backups, the folder you choose here will effectively be treated as the project folder.



Customize settings

If you select *Customize settings* from the pop-up menu, you'll be presented with additional choices.



Save workspace into a new project folder. Un-checking this box will prevent QLab from creating a project folder to enclose your workspace and its associated files.

Copy media files into project folder when necessary. Un-checking this box will prevent QLab from copying media files when your cues target them. This could be desirable when you are tight on disk space and know for sure you won't be moving the workspace to another computer, or if you simply prefer to handle these sorts of things by hand.

It's worth noting that *not* copying media can make it more challenging to use the workspace with collaboration remotes, since remotes can only "see" files that are in the same folder as the workspace, or in folders within that folder.

Unchecking both boxes will make QLab 5's saving behavior most similar to QLab 4's saving behavior, and is the same as choosing *Save workspace only*.

Cues

> In this section, every time a new tool, interface item, or concept that we feel is particularly essential is mentioned, it will appear in bold text. This is meant to help you notice that you're being introduced to a new idea. Thereafter, and throughout the rest of this documentation, bold text will be used for emphasis, to highlight keyboard shortcuts (like **⌘S**), and for indicating a menu name (such as the **File** menu.)

A cue is the most basic unit of action in QLab. A single cue causes a single event to occur when it is started, and the type of event which occurs is dependent upon the type of cue.



Here is a cue numbered `1` and named "Preshow announcement," which is selected (highlighted with the blue bar.) The color of the selection highlight may be different on different Macs; it can be configured in System Preferences → General.

The white triangle on the left edge of the cue list is called the *playhead* and it indicates that cue `1` is *standing by*. When the GO button is pressed, the cue that is standing by will be started, and the playhead will advance down the cue list, in this case to cue `2`, named "Opening song".

Making Cues

There are several ways to make a cue in QLab:

1. Click on one of the cue type icons in the toolbar.
2. Drag a cue type icon from the toolbar into the cue list.
3. Double-click on a cue type in the toolbox.
4. Drag a cue type from the toolbar into the cue list.
5. Select a cue type from the **Cues** menu.
6. Use one of the **⌘-number** keyboard shortcuts for the first ten cue types.
7. Drag one or more compatible audio or video files from the Finder into the cue list.
8. Drag a folder containing compatible audio or video files from the Finder into the cue list.
9. Hold down the option key while dragging a cue in the cue list to make a copy of it.

There are also ways to make cues using OSC and AppleScript, which you can learn more about in the OSC Dictionary and AppleScript Dictionary sections of this manual.

Setting File Targets, Cue Targets, and Settings Targets

An important basic rule of QLab is that some kinds of cues require a *target*, which is the thing that the cue acts upon. Some cues target files on your computer, some cues target other cues, some cues target other cues or workspace settings, and some cues have no target. Cues which require targets will not work unless one has been assigned, and they can only ever have one target at a time.

Audio, Video, and MIDI File cues - file targets

Audio, Video, and MIDI File cues require a *file target* which is a file on your Mac containing audio (for Audio cues,) video (for Video cues,) or MIDI data (for MIDI File cues.) There are several ways to assign a file target to cue:

1. Click the button for the cue in the Target column of the cue list.
2. Select the cue and type the keyboard shortcut for *Edit target*, which is **T** by default.
3. Double-click in the target field in the Basics tab of the inspector.
4. Drag and drop a file from the Finder onto the cue in the cue list.
5. Drag and drop a file from the Finder into the target field in the Basics tab of the inspector.

If a cue's file target is deleted or moved to the trash, the cue will break.

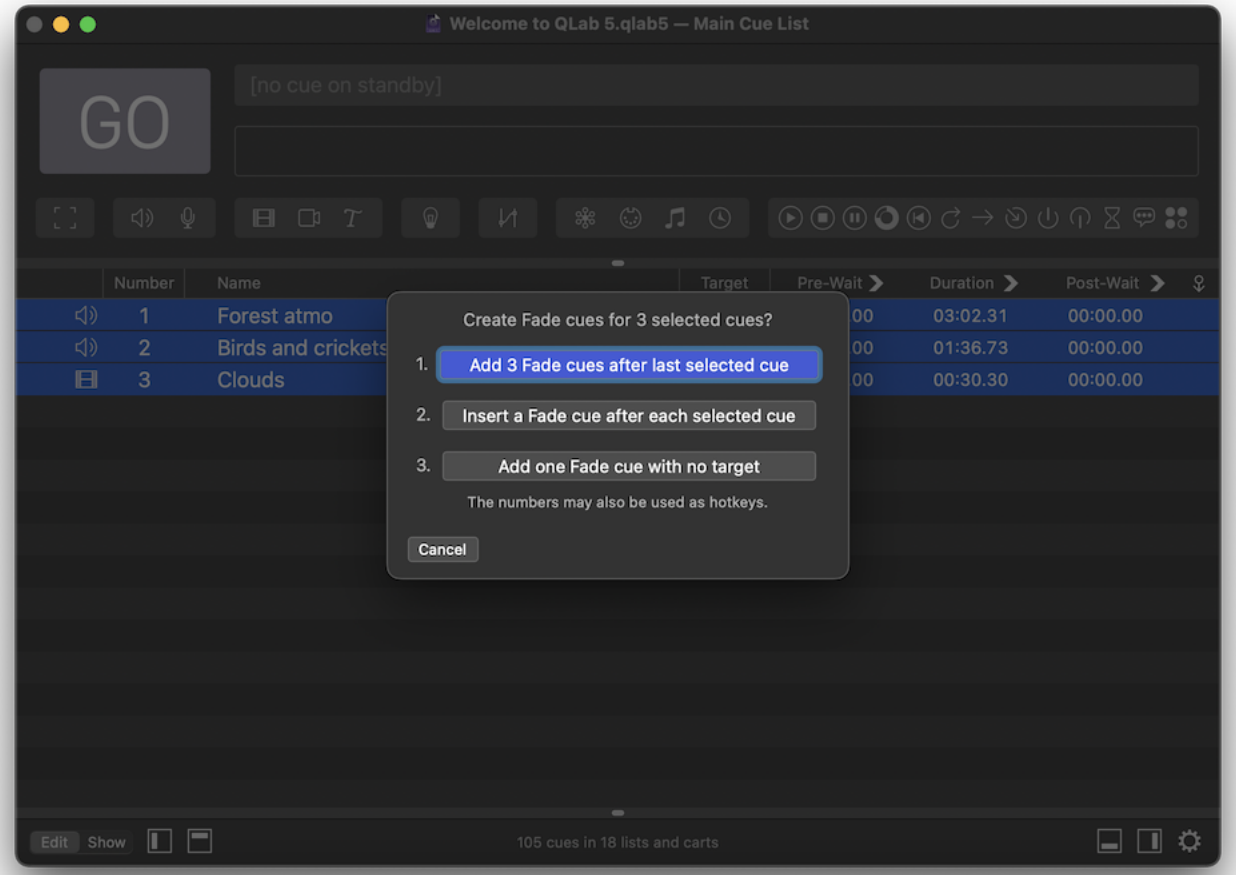
Other cues - cue targets

Cues which require a *cue target,* which is another cue, include Fade, Start, Stop, Pause, Load, Reset, Devamp, GoTo, Target, Arm, and Disarm. There are several ways to assign a cue target to a cue:

1. Double-click in the target column in the cue list and enter the cue number of the intended target.
2. Select the cue and type the keyboard shortcut for *Edit target*, which is **T** by default, then enter the cue number of the intended target.
3. Double-click in the target field in the Basics tab of the inspector and enter the cue number of the intended target.
4. Drag and drop the intended target cue onto the cue in the cue list.
5. Drag and drop the cue onto its intended target cue in the cue list.

Additionally, QLab has special behavior when you create a new cue that requires cue targets using the toolbar, toolbox, **Cues** menu, or **⌘-number** keyboard shortcut:

1. If no cues are selected, then the new cue is created without a target.
2. If one cue is selected, then the new cue is created targeting the selected cue.
3. If more than one cue is selected, QLab asks you whether you want to create multiple targeted cues, one after each selected cue; multiple targeted cues all after the last selected cue, or only one single new cue.



You can choose which behavior you want by clicking on the appropriate button, or pressing the corresponding number key on your keyboard. You can also hit the **enter** or **return** key on your keyboard to select the default choice, which is the highlighted button, or the **escape** key to dismiss the question and create no cues.

Whichever option you select will become the default option the next time QLab presents this question.

Fade cues and Reset cues - workspace settings targets

Fade cues and Reset cues have the additional ability to target *workspace settings* as well as cues. At present, audio output patches and audio maps are the two types of workspace settings that can be targeted. When a Fade cue or Reset cue targets an audio output patch or an audio map, it will have an effect on every cue which uses that audio output patch or audio map. To assign a workspace settings target:

1. Select the cue and navigate to the Basics tab in the inspector.
2. Click the pop-up menu labeled *Target* and choose **Patch** or **Map**.
3. Another pop-up menu will appear immediately to the right. Use that menu to select the patch or map that you want the cue to adjust.

You can also use the keyboard shortcut for *Cycle cue target type*, which is ⌥**T** by default, whenever one or more Fade or Reset cues is selected.

When a cue is set to target patches or maps, you can click the button for the cue in the Target column of the cue list or select the cue and type the keyboard shortcut for *Edit target*, which is **T** by default, to pop up a menu to select a target patch or map.

Types of Cues

Group cues contain other cues, and have several different modes which cause different behaviors. Group cues have different icons for their different modes, all of which have the same basic shape. You can read more about Group cues here.

Audio cues play pre-recorded audio using your Mac's built-in speakers, headphone jack, or another audio device. They require a target which must be a compatible audio file. You can read more about Audio cues here.

Mic cues route live audio through QLab. You can read more about Mic cues here.

Video cues play pre-recorded moving or still images on one or more projectors, displays, or screens connected to your Mac. They require a target which must be a compatible video or image file. You can read more about Video cues here.

Camera cues route live video through QLab, using the same outputs as Video cues. You can read more about Camera cues here.

Text cues display formatted text using the same outputs as Video cues. You can read more about Text cues here.

Light cues control lights and other DMX-controllable equipment connected to your Mac using Art-net or a USB-DMX device. You can read more about Light cues here.

Fade cues adjust one or more parameters of another cue in your workspace. They require a target, which must be an Audio, Mic, Video, Camera, or Text cue. Fade cues have different icons for absolute mode and relative mode. You can read more about Fade cues which target Audio and Mic cues here. You can read more about Fade cues which target Video, Camera, and Text cues here.

Network cues send OSC messages and some other types of messages using your Mac's network connection. You can read more about Network cues here.

MIDI cues send MIDI Voice, MIDI Show Control, or MIDI System Exclusive (SysEx) messages using a MIDI device connected to your Mac. You can read more about MIDI cues here.

MIDI File cues play pre-recorded MIDI files using a MIDI device connected to your Mac. They require a target which must be a MIDI file. You can read more about MIDI File cues here.

Timecode cues either generate Linear Timecode (LTC) and play it using an audio device connected to your Mac, or generate MIDI Timecode (MTC) and play it using a MIDI device connected to your Mac. You can read more about Timecode cues here.

Start cues tell another cue to start playing. They require another cue as a target. You can read more about Start cues here.

Stop cues tell another cue to stop playing. They require a target which can be any cue. You can read more about Stop cues here.

Pause cues tell another cue to pause. They require a target which can be any cue. You can read more about Start cues here.

Load cues tell another cue to load. They require a target which can be any cue. You can read more about Load cues here.

Reset cues tell another cue to reset. They require a target which can be any cue. You can read more about Reset cues here.

Devamp cues tell a cue which is looping to get to the end of the current loop, and then exit the loop. They require an Audio or Video cue as a target. You can read more about Devamp cues here.

Go To cues move the playhead to their target. They require a target which can be any cue. You can read more about Go To cues here.

Target cues tell another cue to change its target. They require any cue which can accept targets as their target. You can read more about Target cues here.

Arm cues arm their target cue. They require a target which can be any cue. You can read more about Arm cues here.

Disarm cues disarm their target cue. They require a target which can be any cue. You can read more about Disarm cues here.

Wait cues do nothing by themselves, but can be useful as part of a cue sequence. You can read more about cue sequences here.

Memo cues are placeholders which have no effect when played. They can be used to provide in-line notes to the person operating QLab or any similar reason.

Script cues execute AppleScript. You can read more about Script cues here.

Rules for cues

Once started, cues continue to play until they reach the end of their programmed action, or until they're told to stop. For an Audio or Video cue, a cue's action is the duration of the media file which it targets unless you program it otherwise. For example, an Audio cue that targets a one minute long audio

file will play that whole file and therefore last for one minute, but the cue can be programmed to play only a portion of the file and thus last for less than one minute, or to repeat some or all of the file and thus last for more than one minute. For a Fade cue, the default duration is five seconds, although you can change that default in Workspace Settings → Templates → Cue Templates → Fade.

By default, cues which are running will ignore additional commands for them to start. You can alter this behavior on a cue-by-cue basis in the Triggers tab of the inspector.

When you start a cue, the playhead will advance down the cue list. On the next press of the GO button or its keyboard shortcut (**space** by default), the next cue will start. If you're accustomed to using a non-theatrical playback program such as Apple Music or Spotify, this behavior can be disorienting at first. Fear not, for this is how QLab is meant to work.

Batch editing cues

When more than one cue is selected, QLab allows you to simultaneously edit properties that the selected cues all have in common. Not all properties of cues are batch-editable, but you can expect future releases of QLab to expand the list of batch-editable properties.

Whenever multiple cues are selected, the inspector shows only tabs which contain batch-editable properties. When the selected cues are of mixed types, only the Basics and Trigger tabs will be shown since those are the only tabs that all cues have in common.

You can set a batch-editable property in all selected cues to the same value by using the contextual menu (right-click, control-click, or two-finger-click), the inspector, or the keyboard shortcut for editing that property.

You can set a batch-editable property of an individual cue within the selected set by using the mouse in the cue list to edit the property. The other selected cues will not be edited.

It's important to remember that batch editing a property sets that property to the same value in all selected cue. For example, if you have selected two Audio cues whose main audio levels are set to different values, batch editing the main audio level will set both cues to that same new level.

The Inspector

The inspector is a tabbed interface which lets you view and edit the attributes of the selected cue or cues. You can find it beneath the cue list, and it can be shown or hidden while in Edit Mode by selecting *Inspector* from the **View** menu or using the keyboard shortcut ⌘**I**. You can pop the inspector out of the main window by clicking on the ↗ button on the right side of the inspector tab bar. The inspector is disabled when the workspace is in Show Mode.

The tabs shown in the inspector vary depending on the type of cue or cues selected. All cue types, as well as cue lists and cue carts, have the Basics tab and Triggers tab which are described below. Other tabs are described throughout this manual.

Along with their name, tabs can display indicators to give you clues about what's going on inside them.

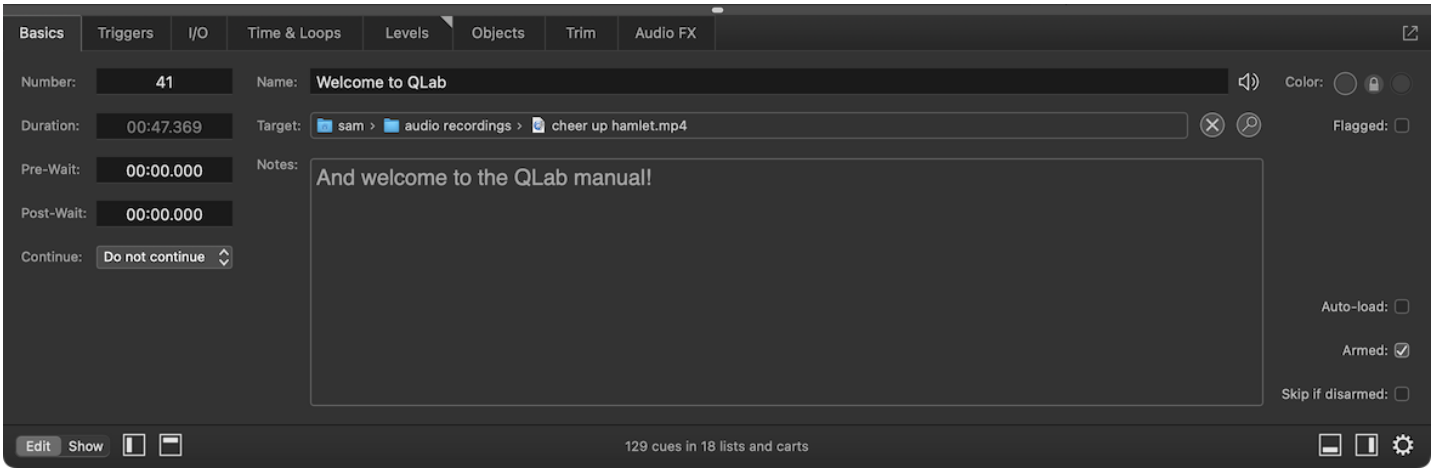


If you are using collaboration, a colored underline appears in inspector tabs when collaborators are looking at them. In the screen shot above, a collaborator is looking at the Time & Loops tabs.

The Levels and Objects tabs display a grey dog-ear indicator whenever they have a level set above -INF, essentially attempting to answer the question, "is this cue making sound because of the routing in this tab?" In the screen shot above, the cue has levels set in the Objects tab, and not in the Levels tab.

The Basics Tab

Perhaps unsurprisingly, the Basics tab shows basic properties which every cue has. Many of these properties are also editable from the Cue list.



Number

A cue number may be any text, or may be empty. All cue numbers in a given workspace must be unique. Cue numbers do not need to be consecutive, nor do they need to be digits. Some examples of acceptable cue numbers are: `1`, `1.5`, `A`, `AA`, `A.5`, `Steve`, or □. It's important to note that while QLab allows you to use digits, letters, punctuation, and emoji in cue numbers, you'll need to stick to proper numbers if you plan to use MIDI, MSC, or OSC to remote-control cues.

> **Cue numbers aren't necessarily numbers?!** That's right. While several millennia of human history have demonstrated that numerals are the clearest and most flexible way to uniquely identify items in an arbitrarily large ordered set, theatrical tradition and circumstance have normalized the use of other indices as well, notably letters. Wherever possible, QLab is designed to avoid making assumptions about how you work and why you do the things you do. Some folks like using letters for cues, by George, and QLab shall not stand in their way! They're still called *cue numbers*, though, because they have to be called *something* and "cue numbers or letters or what-have-you" does not make for attractive screen shots.

You can change the number of a cue in this text field, by double-clicking in the cue's number column in the cue list, or by selecting the cue and using the keyboard shortcut for editing cue numbers which is **N** by default. Changing a cue's number will not affect any cues that target it, or really anything else in

fact.

It's important to realize that since cue numbers are text strings, `1`, `1.0`, and `1.00` are technically three different unique cue numbers in QLab.

Reordering cues in the list does not automatically renumber them, and QLab does not care about "out of order" cue numbers.

Duration

The duration of a cue is the length of time it takes for the cue to complete, not counting its pre-wait or post-wait. The duration of some cues cannot be edited directly, but for those that can, they can be edited in this field, by double clicking and typing in the duration column in the cue list, or by using the keyboard shortcut for editing durations which is **D** by default.

Cues which have an instantaneous action, such as a Pause cue or a MIDI cue that sends a single Note On message, have no listed duration.

Pre Wait

Pre-wait is an optional delay that transpires before the cue starts. For example, a cue with a pre-wait of `3` would start running three seconds after being told to start. This can be useful for matching up cues in QLab with outside events or for making cues time out precisely against each other.

The pre-wait of a cue can be edited in this text field, by double clicking and typing in the pre-wait column in the cue list, or by using the keyboard shortcut for editing pre-wait which is **E** by default.

Post Wait

The post-wait of a cue is meaningful only when the cue's continue mode is set to auto-continue. When a cue is set to auto-continue, it starts the next cue in the cue list when it is itself started. If the first cue has a post-wait time, QLab waits for the post-wait to elapse and then starts the second cue. A cue's post-wait begins to elapse at the same time as the cue's duration begins to elapse.

The post-wait of a cue can be edited in this text field, by double clicking and typing in the post-wait column in the cue list, or by using the keyboard shortcut for editing post-wait which is **W** by default.

> **A Note About Time:** All times in QLab are precise to the millisecond (0.001 seconds), but are usually displayed on screen using only hundredths of a second (0.01) or sometimes even tenths of a second (0.1). The reason for this may be unexpected; displaying multiple, independent timers on a computer is surprisingly computationally expensive, so much so that running timers with three decimal places versus two has a meaningful impact on performance (and battery life, where applicable.) Rest assured that the imprecision is only visual, not technical, and serves the greater good of better performance.

Continue

This pop-up menu displays and allows you change the cue's continue mode:

- **Do not continue** is the default mode. When a cue is set this way, starting it does not automatically cause any other cues to start as well.
- **Auto-continue.** If a cue is set to **auto-continue** (↓), starting that cue will start the following cue as well. If the first cue has a post-wait, the post-wait will be honored before the next cue is started.
- **Auto-follow.** If a cue is set to **auto-follow** (⇊), then the next cue will be started as soon as the first cue completes. When you set a cue to auto-follow, QLab will automatically show a post-wait time equal to the duration of the cue. This cannot be edited, and serves as a visual reminder of the auto-follow.

The continue mode of a cue can also be editing by clicking in the cue's continue mode column in the cue list and using the pop-up menu which appears, or by using the keyboard shortcut for editing continue mode which is **C** by default.

Name

A cue name may be any text, or may be empty. For many cue types, QLab fills in a default name which reflects an essential attribute about the cue.

Cue type	Default name
Audio	The file name of the target audio file.
Video	The file name of the target video or image file.
Text	The text entered in the Text tab of the inspector.

Cue type	Default name
Light	If the Light cue contains a single light command, the default name will be that full command. If the Light cue contains more than one command, the default name will be "fade x lights" where x is the number of instruments that have commands in the cue.
Fade	"fade" plus the name of the cue being faded. If the fade is set to stop target when done, the default name is "fade and stop" plus the name of the cue being faded.
Network	The network command sent by the cue.
MIDI	The type of MIDI message sent by the cue.
MIDI File	The file name of the target MIDI file.
Timecode	The starting frame of timecode sent by the cue.
Start, Stop, Pause, Load, Reset, Go To, Arm, Disarm	The type of cue plus the name of the target cue.
Wait	"wait" plus the duration of the cue.

You can change the name of a cue in this text field, by double clicking in the cue's name column in the cue list, or by selecting the cue and using the keyboard shortcut for editing cue names which is **Q** by default. Unlike cue numbers, cue names do not have to be unique.

Changing the name of a cue will also change the name of any cues that target it, if those cues are using their default names. For example, if an Audio cue is named "fanfare", a Fade cue which targets that Audio cue will have the default name "fade fanfare." If you change the name of the Audio cue to "entrance music," the Fade cue will change its name to "fade entrance music." If you give the Fade cue a unique name, changing the Audio cue's name will not change the Fade cue's name.

Deleting the name of a cue will reset it to its default name, if that type of cue has one, or to empty if not.

Default names appear in the Basics tab in darker text than unique names.

Target

This field behaves in one of three ways depending on the type of cue that is being inspected.

File Targets

If the cue being inspected accepts file targets, such as Audio cues and Video cues, this field shows the full path to the file target of the cue. In this case, you can double-click on the field to open a dialogue allowing you to select a target file, or you can drag a file from the Finder and drop it onto the field.

If the cue has a target, you can click the button to clear the target, and the button to reveal the target file in the Finder.

Cue Targets and Patch Targets

If the cue being inspected accepts cue targets, such as and Start cues and Devamp cues, or both cue targets and patch targets, such as Fade cues and Reset cues, a menu appears allowing you to choose between cue targets and patch targets. If the menu is set to *Cue*, the Target field shows the cue number of the target cue. To the right of the field, the cue name of the target cue is shown. You can enter a cue number here to set the target. If the menu is set to *Patch*, the Target field is replaced by a pop-up menu which allows you to select a patch to fade.

If the cue has a target, you can click the button to clear the target, and the button to move the selection to the target cue.

No Target

If the cue being inspected does not accept a target, this field is disabled and QLab displays *(not applicable)*.

Notes

Text entered in this field is shown in the notes field up in the masthead of the workspace window whenever the selected cue is standing by, so it is the perfect place for notes or special instructions to your operator. Text in the Notes field is searchable using the Find tool. It can be styled and formatted, and can include emoji and images.

You can edit a cue's notes here in this field, or in the masthead when the cue is standing by. You can also select a cue and use the keyboard shortcut to edit notes, which is **O** by default.

Color

Cues can be given a highlight color to make them stand out in the cue list. There are twenty colors to choose from here, divided into eight “basic” colors and twelve “extended” colors. The color doesn’t alter the behavior of the cue in any way, so feel free to use color coding in whatever way is most useful to you. The expanded selection of colors available in QLab 5.2 and later allows a variety of color coding schemes that are compatible with a variety of color perception differences.

Second Color

You can optionally set a cue to display a different color after it’s be started for the first time by clicking on the icon to unlock it. This allows you to choose a color in the second circle.

The cue will display the second color after it’s started, and will continue to display the second color until after the workspace is closed and reopened, after the cue is reset via a Reset cue or OSC command, or after the workspace is reset using the Reset button in the workspace sidebar or the associated OSC command.

Flagged

Flagging a cue is a way of marking it for later. You might, for example, flag cues during a run through of your show as a way of notating which ones need to be reviewed after the run through. Checking this box will display a flag ( ) in the status column of the cue list (if the cue is in a list) or within the cue cell (if the cue is in a cart.) It will also add the cue to the Warnings tab of the Workspace Status window.

If a Group cue contains one or more cues that are flagged, the Group cue will display a hollow flag icon ( ) to help you find your flagged cues when the Group is collapsed (not showing its contents.)

Auto-load

If this box is checked, this cue will automatically be loaded after the previous cue is played. Loading a cue is not necessary, but can help keep playback smooth if your workspace is working at or close to the limits of the available computing power.

You can set all new cues to auto-load by default in Workspace Settings → General.

Armed

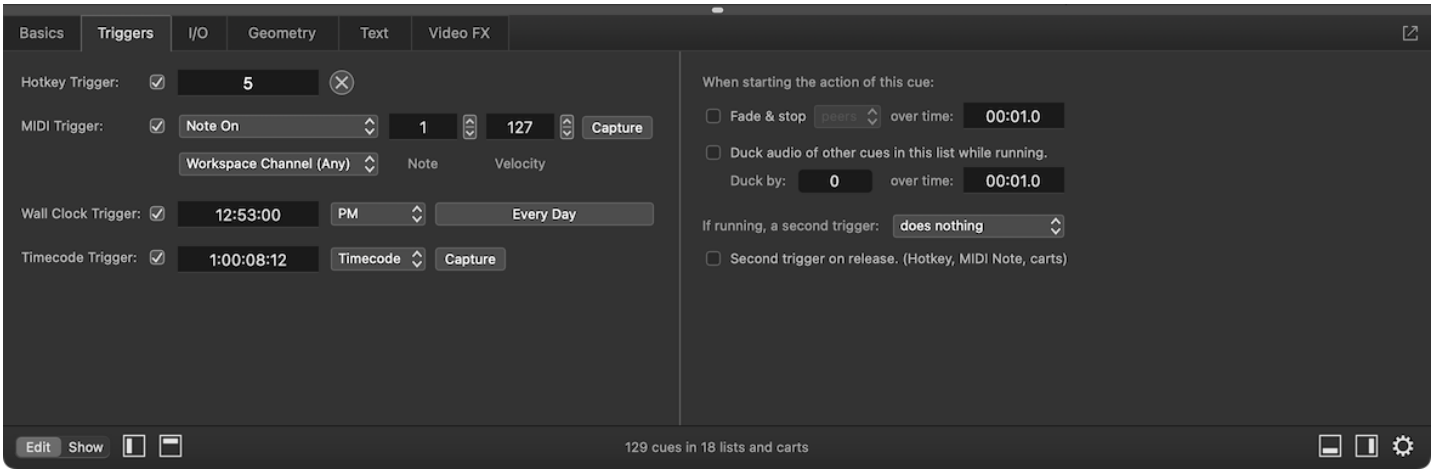
Cues are armed by default. When disarmed, cues will not execute their action, but an auto-continue or auto-follow will be triggered if present. Disarming a cue is useful, for example, for temporarily silencing a cue while allowing the surrounding cues to operate as-is. Disarmed cues are shown with a diagonal pattern in the cue list.

Skip if disarmed

If this box is checked and the cue is disarmed, it will be entirely skipped by QLab as though it does not exist. If the cue is part of a sequence, checking this box will likely cause the sequence to behave differently. When the cue is armed, this checkbox has no effect. Disarmed and skipped cues are shown with a vertical curvy line pattern in the cue list.

The Triggers Tab

The Triggers tab is divided into two sides.



The left side gives you access to four ways to directly start a cue. By configuring any of these triggers, you can set the cue to be started at any time, regardless of the position of the playhead and separately from the GO button. You can use these triggers in any combination. To activate a trigger, check the checkbox next to it. To deactivate a trigger, uncheck the box.

The right side of the Triggers tab gives you several options for shaping QLab's behavior when the cue runs. Note that these right side options apply to the cue no matter how the cue is started, including when it's started by the GO button, OSC messages, or AppleScript.

Hotkey Trigger

Nearly any key on the keyboard that's not being used for something else can be assigned to start a cue when pressed. There are a few exceptions:

- You cannot use the **Esc** key, which is permanently assigned to Panic (stop) all cues.
- You cannot use the up or down arrow keys (although you *can* use the left and right arrow keys.)
- You cannot use a modifier key (command, control, option, function, or shift) by itself, but you can use one or more modifiers in combination with any otherwise valid key.
- You cannot use any keys that have been assigned functions in Workspace Settings → Controls → Keyboard.

To assign a hotkey trigger, check the box to enable the trigger, and then click in the text field and type the key or key combination that you want to assign to the selected cue. To clear the assignment, click on the ⊗ button to the right of the text field.

Hotkey triggers behave just like regular keyboard shortcuts for menu items; for example, if you set the hotkey for a cue to **J**, as in the screen shot above, any time you press **J** on the keyboard, that cue will start even if it is not currently standing by. Hotkeys are particularly useful for starting Network and Script cues that act on selected cues, since using the hotkey means you don't need to select the Network or Script cue in order to start it.

MIDI Trigger

MIDI triggers allow you to start cues using external hardware or software using MIDI voice messages.

To assign a MIDI trigger, check the box to enable the trigger and then either program in the message manually, or click the **Capture** button to record an incoming message. The screen shot above shows a cue with a MIDI trigger set to Note On, note 1, velocity 127, channel "any". When that MIDI message comes in on any MIDI channel, the cue will start even if it is not standing by.

MIDI trigger values can include > and < operators, or be set to "any". This is particularly helpful for MIDI Note On messages, where a specific note velocity won't necessarily be achieved with each press of the MIDI controller. If you set the velocity (byte 2) to "any", QLab will respond to the specified MIDI note regardless of velocity.

Wall Clock Trigger

The wall clock trigger will start a cue at a specific time of day and, optionally, only on certain days of the week.

To assign a wall clock trigger, check the box to enable the trigger and then enter a time in the field. Be sure to enter the hours, minutes, and seconds; if you type "10:30" QLab will interpret that as ten minutes, thirty seconds past midnight, not as ten hours, thirty minutes, and no seconds. You can select either AM, PM, or 24-hour time as suits your preference, and you can click on the button labeled *Every Day* to select the days of the week on which you want the trigger to be active. The screen shot above shows a cue with a wall clock trigger of 12:53 PM every day. As long as the workspace is open, this cue will run every day right at 12:53 PM.

Please note that computer clocks will drift if left to their own devices, so if your show computer is offline or if you've disabled online clock setting, be sure to check your clock's accuracy at least weekly.

Timecode Trigger

Cues can be started from incoming LTC (SMPTE) or MTC timecode, but only if timecode has been enabled for the enclosing cue list or cart. You can enable timecode for a list or card in the Timecode tab of the inspector when that list or cart is selected.

To assign a timecode trigger, check the box to enable the trigger and then enter a time in the field. The drop-down menu to the right of the field lets you choose to enter the timecode trigger using either the timecode format of `hour:minutes:seconds:frames` or the real-time format of `hours:minutes:seconds:decimals`. The screen shot above shows a cue with a timecode trigger set at `1:00:08:12`.

You can also click the **Capture** button while timecode is incoming to record the current frame as the timecode trigger.

If a timecode trigger is set while timecode is currently incoming, that timecode will need to be stopped and restarted before the trigger is active.

Fade & Stop Others Over Time

When this box is checked, starting the cue will fade and stop other cues over the given time. The drop down menu lets you choose which other cues will be faded and stopped. You can choose one of three options:

- **Peers.** When this option is selected, the cue's *peers* will be faded and stopped. A cue's peers are those at the same hierarchical level in the cue list. For a cue in a list or cart, the other cues in that list or cart are its peers. For a cue within a Group, the other cues within that same Group are its peers.
- **List or cart.** When this option is selected, all other cues within the same list or cart will be faded and stopped.
- **All.** When this option is selected, all other cues within the entire workspace will be faded and stopped.

Duck/Boost Audio of Other Cues In This List While Running

When this box is checked, all other cues in the same cue list or cart that have audio will be ducked or boosted by the given level, fading over the given time.

The label text automatically changes to "duck" or "boost" based upon the number you enter. Negative numbers cause a duck, positive numbers cause a boost.

If this cue has a pre-wait time, the duck or boost begins after the pre-wait has elapsed.

Second Triggers

A second trigger is any signal to start that is received by a cue while it's already running, including triggers from the left side of the Triggers tab, the GO button, OSC messages... everything. When a cue receives a second trigger, it can be set to behave in one of six ways:

- **Second trigger does nothing.** The cue will ignore all signals to start while it's running. This is the default setting.
- **Second trigger panics.** The cue will panic (fade out and stop) when it receives a second trigger.
- **Second trigger stops.** The cue will stop when it receives a second trigger.
- **Second trigger hard stops.** The cue will hard stop (stop, and also immediately stop any audio effects on the cue) when it receives a second trigger.
- **Second trigger hard stops & restarts.** The cue will hard stop and then immediately restart when it receives a second trigger.
- **Second trigger devamps.** If the cue is looping when it receives a second trigger, it will exit the loop at the conclusion of the current iteration. If the cue is not devamp-able (or not devamp-able at the time of a second trigger,) then the second trigger will do nothing. This option is new to QLab 5.

There are two additional options in this list when the selected cue is a Group cue in Playlist mode, and you can learn more about those options in the section on the Playlist Tab in the Group cue section of this manual.

The checkbox marked *Second trigger on release* pertains only when the cue is started using a hotkey trigger or MIDI trigger, or when the cue is in a cart. If so, and this box is checked, then QLab responds to the release of the trigger as though it's a second trigger. This is a straightforward way to get sampler-like behavior out of QLab. If you assign a hotkey to a cue, set that cue to stop or panic on a second trigger, and check this box, then pressing the hotkey will start the cue, and releasing the hotkey will stop it.

Secondary Inspectors

QLab allows you to open multiple inspector windows, each of which can be independently assigned to inspect a specific cue.



In the screen shot above, the main inspector is showing the Time & Loops tab of the selected cue, cue `41`, and a secondary inspector has been opened which is showing the Levels tab of cue `42`.

You can open secondary inspector windows by...

- ...choosing *Inspector for Selected Cue* from the **View** menu
- ...using the keyboard shortcut ⇧⌘I
- ...right-clicking (or control-clicking, or two-finger-clicking) on any cue and choosing **Open in new inspector window** from the contextual menu.

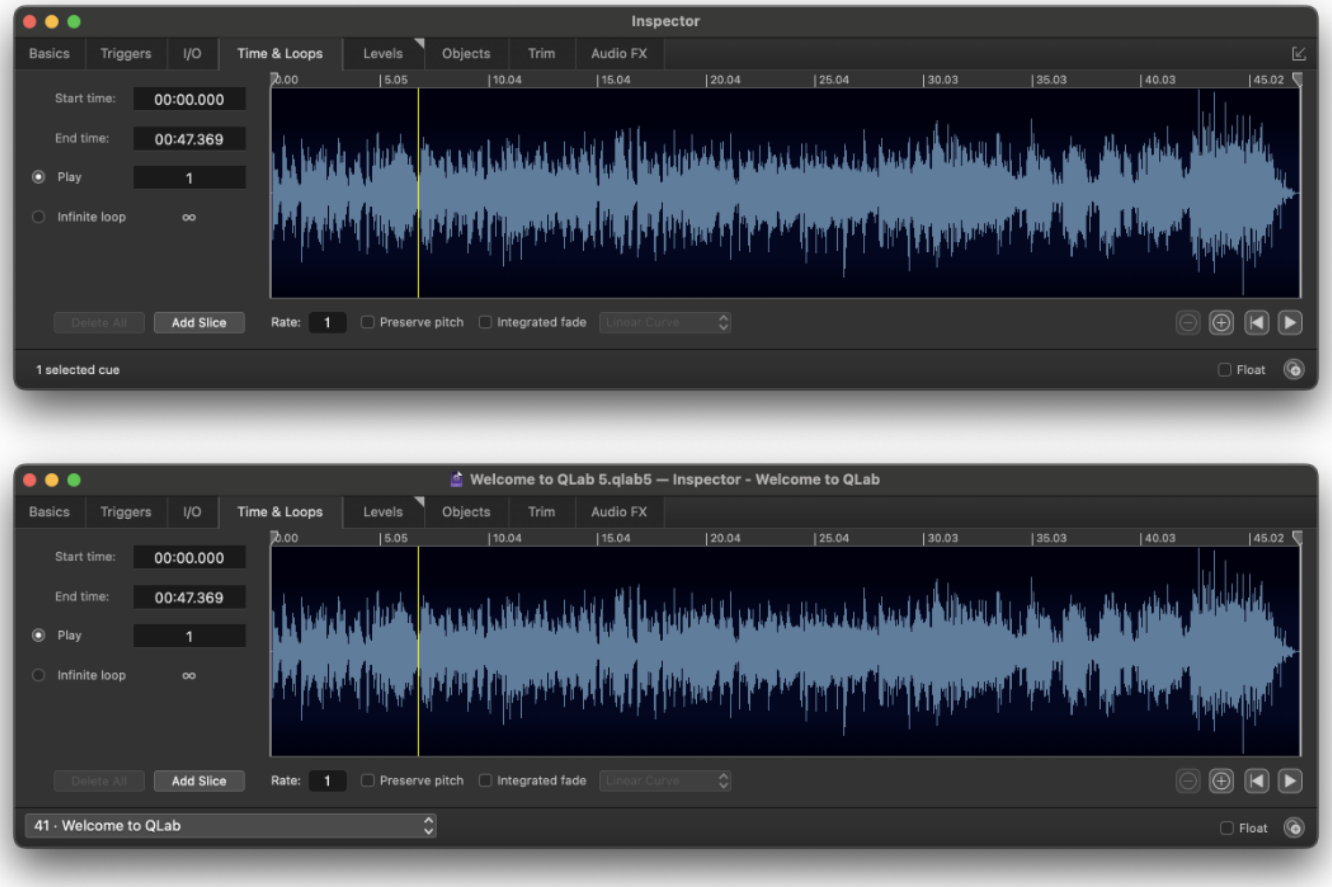
If you have multiple cues selected, QLab will open one inspector window for each cue.

Unlike the main inspector, which always inspects the selected cue in the main workspace window, secondary inspector windows do not automatically change which cue they're inspecting. The left side of the footer of secondary inspector windows contains a pop-up menu showing the cue number and name of the cue being inspected. You might call this menu the inspector selector. If you look closely, you might call yourself the inspector selector detector. If you choose to ignore this menu, which is fine, you might be the inspector selector neglector.

The right side of the footer contains a checkbox labeled *Float* which causes that secondary inspector to float above other windows, and a Clone button which creates another secondary inspector.

Inspector Inspector

When the main inspector is popped out, it can be difficult to distinguish from secondary inspectors.



The screen shot above shows the main inspector on top and a secondary inspector below it. There are three clues to help you identify which is which:

- The main inspector window shows the name of the workspace and the selected cue; the secondary window shows the name of the workspace, the word "Inspector," and the name of the cue being inspected.
- The main inspector window has a button to pop it back in to the main workspace window; secondary inspectors do not have this button.
- The main inspector window shows the number of selected cues in the lower left corner; secondary inspectors show the inspector selector.

Textual Editing in QLab

Textual editing in QLab behaves according to the basic rules and expectations of standard macOS text entry, and can be subdivided into four basic categories: text, numbers, times, and decibel values.

Editing Text

Most places that accept text will accept ASCII characters, unicode characters, and system-supported emoji. Please note that OSC does not support unicode or emoji, so if you want to use OSC to control something in your workspace, you should only use ASCII characters to identify that thing. For example, even though you can name a cue □, which QLab has no trouble with, the OSC message `/cue/ □ /start` will not work because OSC cannot interpret non-ASCII characters.

Editing Numbers

Numbers in QLab are generally distinguished by whether they can include decimal values or not. Parameters which cannot include decimal values, often called *integers*, will generally allow you to enter a number with a decimal value, but QLab will throw away the decimal portion automatically.

Parameters which can include decimal values, called *real numbers* by math people or *floating point numbers* or simply *floats* by computer people, can also accept values without decimals. In this case, QLab will assume a decimal value of `.0`.

Editing Times

Times in QLab can be entered in a few ways to make it easy and quick to get things done. You can enter times as:

- `seconds`
- `minutes : seconds`
- `hours : minutes : seconds`

In all cases, QLab accepts decimal values down to the millisecond (`0.001`) and automatically reformats the time you enter as hours (if necessary) : minutes : seconds. If you enter, say, `100` seconds, QLab displays that as `01:40.00`. If you enter `65:02.5`, QLab displays that as `1:05:02.5`.

Editing Decibels

Audio levels are expressed in decibels (DBFS, to be precise) which are just numbers with decimal values that have special meaning (i.e. they specifically represent audible volume, not just some numerical value.)

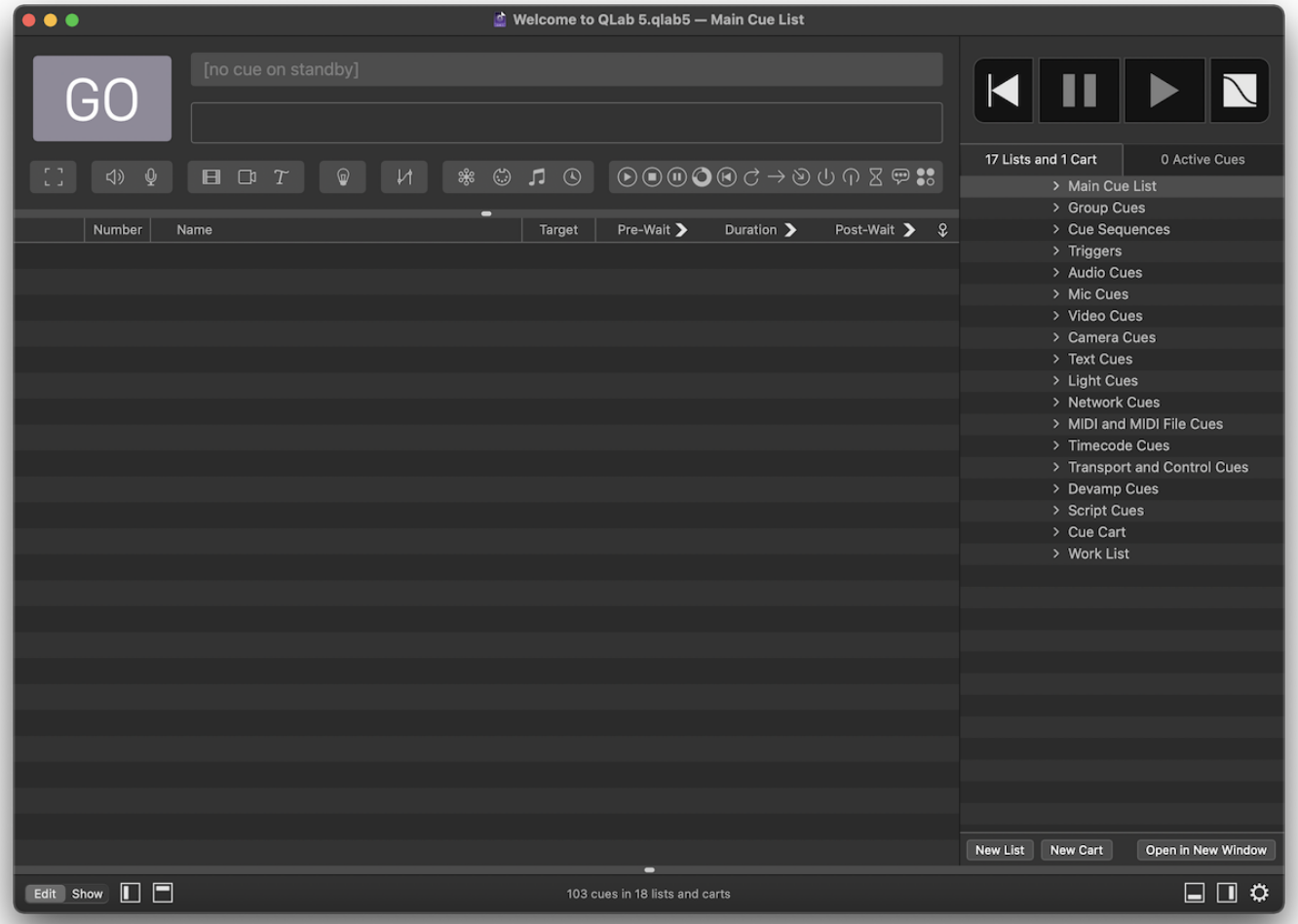
To help you edit levels quickly and to err on the side of caution, if you edit an audio level in QLab and omit the `+` or `-` sign, QLab will assume you mean `-`, so entering `12` in an audio level will automatically correct to `-12`. If you want to enter a positive value, you have to enter it using the `+` sign.

Cue Lists

A cue list is a linear collection of cues, ordered visually from top to bottom. When a cue in a cue list receives the instruction to GO, it starts playing and the playhead moves to the next cue or cue sequence. While it's easy to jump around in a cue list, the basic design of cue lists assumes that you will use it in a more or less linear way.

A workspace in QLab can contain an unlimited number of separate cue lists. By default, all new QLab workspaces have one cue list, titled "Main Cue List" and all newly created cues go in that list.

To see the cue lists in a workspace and to add or remove cue lists, open the sidebar by choosing *Lists / Carts & Active Cues* from the **View** menu, or use the keyboard shortcut ⌘**L**. You can also click on the ▌ button on the right side of the workspace window footer. The sidebar has two tabs; the left tab shows cue lists (and cue carts too, but we're talking about cue lists right now.)



You can create a new list by clicking on the **New List** button. You can delete a cue list by selecting it and choosing *Delete* from the **Edit** menu or by using the keyboard shortcut ⌘⌫ (delete, sometimes labeled "backspace").

> **Note:** deleting a cue list deletes all the cues within that list too. Be careful! But also, rest easy knowing that you can *Undo* cue list deletion.

You can open additional windows to view separate cue lists and carts by selecting the list and clicking the **Open in New Window** button towards the bottom of the sidebar, or by right-clicking (or control-clicking or two-finger-clicking) on the name of the list or cart.

Cue lists have cue names and cue numbers just like cues, and they follow the same rules and can be edited in the same ways.

Working with Cue Lists

The cue list whose cues are displayed in the main area of the workspace is called the *active cue list*. Only one cue list or cart can be active at a time, but the other cue lists and carts in a workspace can still be used even when they're not active.

The following things are true for the active cue list or cart only:

- Selected cues can appear in the main inspector window.
- The toolbar, Find tool, or Load to Time tool appear in the masthead.
- The commands in Workspace Settings → Controls → Workspace MIDI control playback and playhead positioning.
- The commands in Workspace Settings → Controls → OSC control playback and playhead positioning.
- Keyboard shortcuts from the **View** menu control the position of the playhead and the selection.
- OSC and AppleScript commands that refer to the active cue list, naturally, only have their effect on the active cue list.

The following things are true for all cue lists and carts, whether or not they are active:

- Cues' triggers will work.
- MIDI and OSC commands that refer to "all" cues, like Panic All and Pause All will work.
- OSC and AppleScript commands that do *not* specifically refer to the active cue list will work.

In short, cue lists and carts that are not *active* can still be quite active and are fully useable.

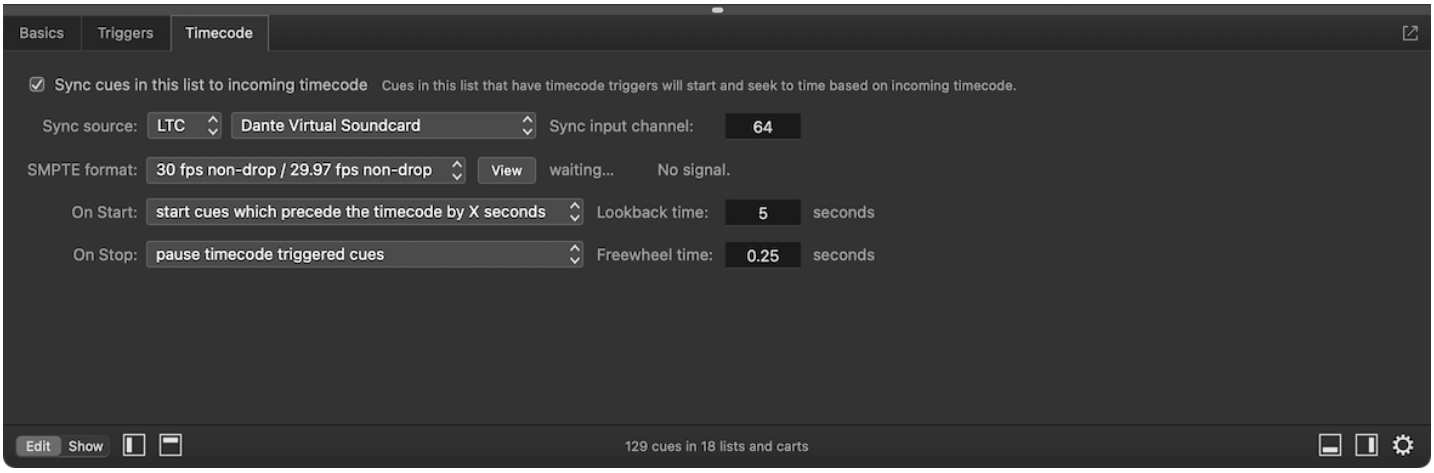
Starting a cue list is the same thing as pressing GO in a window displaying that list. This means that you can use the Triggers tab to assign a hotkey or a MIDI message to a specific cue list, turning that hotkey or MIDI message into a list-specific GO button.

Cues can target cue lists, or cues in other lists. For example, a Start cue in your main cue list could target a separate cue list and anytime that Start cue ran, it would effectively be like pressing GO on that list. A Fade cue in your main cue list could target that cue list, and any time that Fade cue ran it would fade any and every cue playing in that list.

When a cue list is selected, the inspector shows three tabs: Basics, Triggers and Timecode. Please refer to the Inspector section of this manual to learn about the Basics and Triggers tabs of the inspector.

The Timecode Tab

With an Audio, Video, or Bundle license installed, QLab can listen to incoming timecode and start cues using timecode triggers, discussed in the Triggers tab section of this manual. In order for timecode triggers on cues to work, however, incoming timecode must be enabled for the cue list that contains those cues.



To enable incoming timecode on a cue list, check the box labeled *Sync cues in this list to incoming timecode*.

Sync source lets you choose whether the cue list will listen for MTC (which stands for MIDI timecode,) or LTC (which stands for linear timecode or longitudinal timecode.) QLab does not support VITC (vertical integrated timecode) because we've frankly never heard of it being used outside of a film set.

> **Note:** LTC is sometimes referred to as SMPTE, which is similar to the way that a tissue is often called a Kleenex®, or a photocopy is often called a Xerox®. Strictly speaking, SMPTE, the Society of Motion Picture and Television Engineers, is the name of the organization that invented LTC, and LTC is the name of the timecode format. But we digress.

If you choose MTC, the following pop-up menu will populate with available MIDI devices to listen to. If you choose LTC, the pop-up menu will populate with available Audio devices, and an input channel selector will appear to the right. Enter a channel number here to have QLab listen for LTC on that input channel of the specified audio device.

SMPTE format. Once you've chosen the mode and the sync source, select the format of your incoming timecode. This is usually referred to as *frame rate*. It's important to be sure that every device on your timecode network is using the same format, or else your cues will not line up properly.

The **View** button to the right of the format pop-up will open the Timecode Status window to allow you to watch the incoming timecode and verify that everything is working properly. To the right of this button is a small textual indicator that shows the status of incoming timecode, or *waiting...* when there is no incoming timecode.

On Start. This pop-up menu tells QLab how this cue list should behave when it begins receiving timecode. There are five options, all of which pertain only to cues which have timecode triggers:

- **Do not start cues which precede the timecode.** This behavior is the most straightforward. When timecode begins, say with frame `1:02:00:00`, cues with timecode triggers at `1:02:00:00` or later will be started in time with their trigger. Any cues whose triggers are before `1:02:00:00` will not be started, except cues with durations which ought to still be running. For example, if a cue with a length of thirty seconds was set to start playing at `1:01:50:00`, it would start playing from ten seconds in when timecode started.
- **Start cues since most recent minute of timecode.** When timecode begins, QLab looks at the first frame that it receives and then reaches back and starts any cue whose timecode trigger falls within the previous minute of timecode. QLab refers to this as the *lookback time*. For example, if timecode starts at `1:02:00:00` then any cue whose trigger falls between `1:01:00:00` and `1:02:00:00` will be started.
- **Start cues since most recent hour of timecode.** Similarly to the previous option, QLab starts any cue whose timecode trigger falls within the previous hour of timecode. In our example, that's every cue from `0:02:00:00` through `1:02:00:00`.
- **Start cues which precede the timecode by X seconds.** This option provides a text field to enter a custom *lookback time* which QLab will use to start cues. If you enter, say, `5` seconds, then cues from `1:01:55:00` through `1:02:00:00` would be started.
- **Start all cues which precede the incoming timecode.** When timecode begins, QLab starts all cues whose timecode trigger precedes the first incoming frame.

Very important note: no matter which of these five options is selected, any cue with a duration which *ends* before the lookback time will not be started.

On Stop. This pop-up menu determines how cues which are triggered by timecode will behave when timecode stops: they can either pause, stop, or keep running. When the *pause* or *stop* options are selected, the *Freewheel time* sets the window of time during which a dropout of timecode is considered accidental, and will therefore be ignored. Freewheel time can be set from `0` to `2` seconds. For example, if the freewheel time is set to `1` second and timecode stops for half

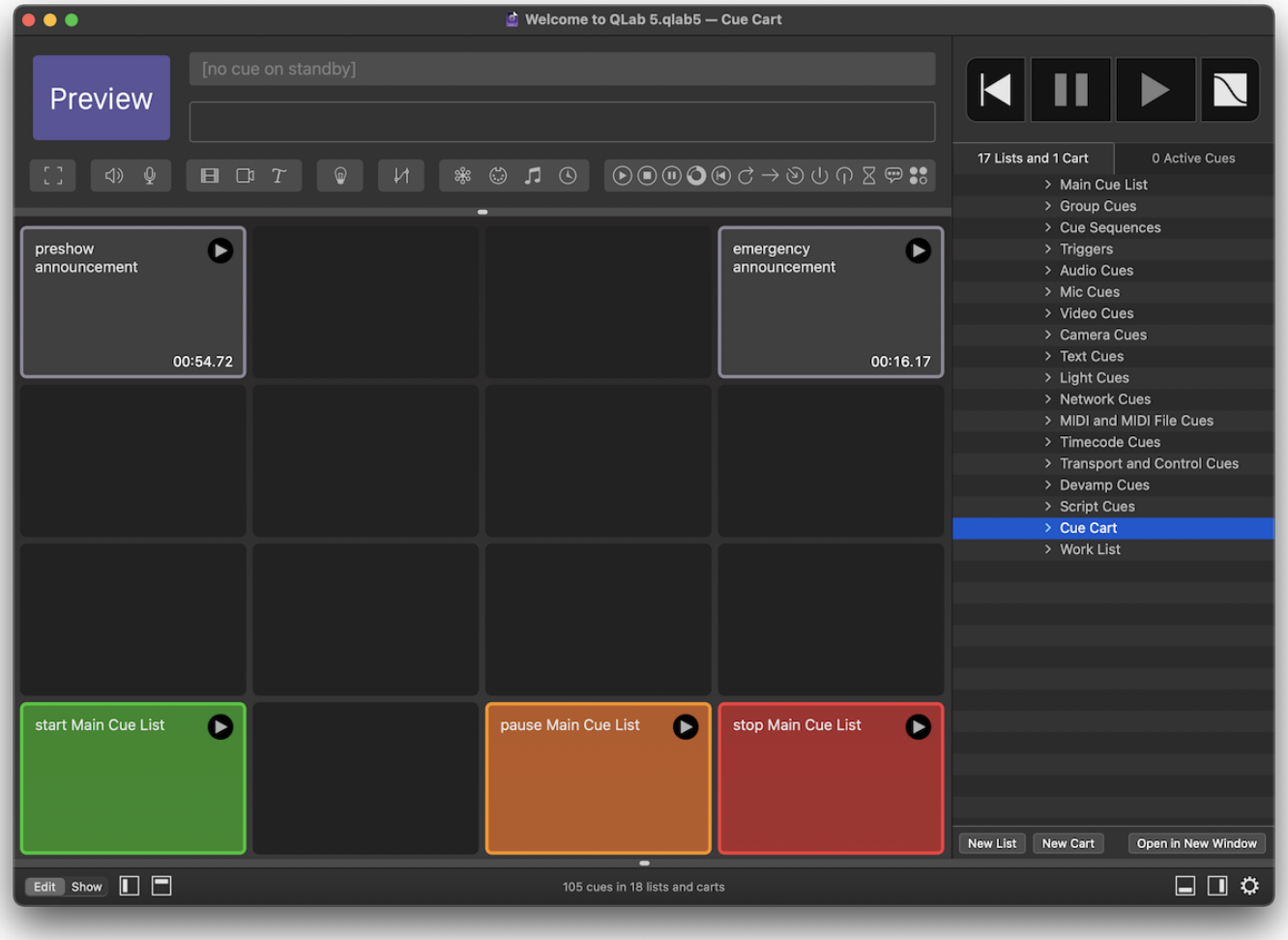
Cue Carts

A cue cart is a grid-shaped collection of cues. Cue carts have no playhead and no concept of ordering; a cue cart is meant to make it easy to play cues in any order you like, and to play individual cues as many times as you like.

Cue carts can contain all types of cues except for [ ] Group cues. Cues in a cart behave the same way they do in a list, with one key exception: cues in a cart cannot be set to auto-continue or auto-follow. Put another way, cue carts cannot contain cue sequences. You can, of course, use a ▶ Start cue in a cart to run a cue sequence in a separate list, but the cue sequence itself cannot be inside the cart.

A workspace in QLab can contain an unlimited number of separate cue carts.

To see the cue carts in a workspace and to add or remove cue carts, open the sidebar by choosing *Lists / Carts & Active Cues* from the **View** menu, or use the keyboard shortcut ⌘**L**. You can also click on the ▮ button on the right side of the workspace window footer. The sidebar has two tabs; the left tab shows cue carts (and cue lists too.)



You can create a new cart by clicking on the **New Cart** button. You can delete a cue cart by selecting it and choosing *Delete* from the **Edit** menu or by using the keyboard shortcut ⌘⌫ (delete, sometimes labeled “backspace”).

Note: deleting a cue cart deletes all the cues within that cart too. Be careful! But also, rest easy knowing that you can *Undo* cue cart deletion.

You can open additional windows to view separate cue lists and carts by selecting the list or cart and clicking the **Open in New Window** button towards the bottom of the sidebar, or by right-clicking (or control-clicking or two-finger-clicking) on the name of the list or cart.

Cue carts have cue names and cue numbers just like cues, and they follow the same rules and can be edited in the same ways.

The minimum size of cue art cells can be set to any of three sizes. The smallest size allows the cart to be smaller overall or to show the greatest number of cues on screen at once, but of course is harder to read from a distance. You can adjust the display size in Workspace Settings → General → Display Size.

Show Mode and Edit Mode

When the workspace is in edit mode, the GO button in the masthead is labeled "Preview," because cue carts do not have a playhead, so starting an individual cue does not have any of the sequential behavior that the GO button implies. Clicking on this Preview button starts the selected cue or cues. You can also start cues in a cart while in edit mode by clicking on the ▶ button in the upper right corner of the cue, or by selecting the cue and using the keyboard shortcuts for GO (**space** by default) or preview (**V** by default.)

When the workspace is in show mode, or when the cart is open in a secondary window, the masthead is not visible.

Working with Cue Carts

The cue cart whose cues are displayed in the main area of the workspace is called the *active cue cart*. Only one cue list or cart can be active at a time, but the other cue lists and carts in a workspace can still be used even when they're not active.

The following things are true for the active cue list or cart only:

- Selected cues can appear in the main inspector window.
- The toolbar, Find tool, or Load to Time tool appear in the masthead.
- The commands in Workspace Settings → Controls → Workspace MIDI control playback and playhead positioning.
- The commands in Workspace Settings → Controls → OSC control playback and playhead positioning.
- Keyboard shortcuts from the **View** menu control the position of the playhead and the selection.
- OSC and AppleScript commands that refer to the active cue list, naturally, only have their effect on the active cue list.

The following things are true for all cue lists and carts, whether or not they are active:

- Cues' triggers will work.
- MIDI and OSC commands that refer to "all" cues, like Panic All and Pause All will work.
- OSC and AppleScript commands that do *not* specifically refer to the active cue list will work.

In short, cue lists and carts that are not *active* can still be quite active and are fully useable.

Starting a cue cart loads all the cues in the cart.

When a cue cart is selected, the inspector shows three tabs: Basics, Triggers, and Grid Size. Please refer to the Inspector section of this manual to learn about the Basics and Triggers tabs of the inspector.

The Grid Size Tab

The grid size tab lets you set the number of rows and columns in the cart, from a minimum size of 1 × 1 up to a maximum size of 15 × 15. If there are cues in a cart, QLab will not allow you resize the cart smaller than the minimum size necessary to accommodate those cues. To make the cart smaller in that case, you'll need to move or delete cues.

If a cart has forcibly been resized to be too small, for instance via AppleScript or OSC commands, the cart will generate a warning to make it easy for you to notice and address the issue.

Group Cues

The Group cue is a type of cue which contains other cues. The cues within a Group, called “child” cues, can be any type of cue including other Group cues, and will behave in one of several ways depending on the mode of the “parent” Group.

The default keyboard shortcut to create a Group cue is ⌘**0**. If you create a Group cue while two or more other cues are selected, those cues will be placed inside the newly created Group cue. Once a Group cue is created, it can be collapsed or expanded for visual simplicity using the grey triangular indicator to the left of the Group cue’s name. Group cues behave exactly the same way whether they are collapsed or expanded.

You can collapse all Group cues in the workspace by using the keyboard shortcut **<** (⇧ **comma**). You can expand all Group cues in the workspace by using the keyboard shortcut **>** (⇧ **period**).

The Mode Tab

Group cues have five **modes** which can be set in the Mode tab of the inspector.

Timeline

Timeline Groups have a green border with square corners.

When a Timeline Group is started, all its children start simultaneously. If the Timeline Group cue was standing by and started via a GO, the playhead will advance to the next cue after the Timeline Group cue.

If any of the child cues have pre-waits, of course, those pre-waits will start counting down when the Timeline Group is started. Since all child cues start at once, their order within the Timeline Group cue doesn’t matter, technically. This means you can order cues inside a Timeline Group cue however you prefer.

You can learn more about Timeline Group cues in the Timeline tab section below.

Playlist

Playlist Group cues have an orange outline with square corners.

When a Playlist Group cue is started via a GO, its first child cue is started and the playhead advances to the first cue after the Playlist Group cue. Children of the Playlist Group cue are then played sequentially, one at a time, with optional crossfading, looping, and shuffling.

You can learn more about Playlist Group cues in the Playlist tab section below.

Start First And Enter

Start First And Enter Group cues have a blue outline with rounded corners.

When a Start First And Enter Group cue is started via a GO, its first child cue starts and the playhead advances to the next child cue within the Start First And Enter Group cue. When the last child cue is started, the playhead will advance to the first cue after the Start First And Enter Group cue.

You may have observed that this is not at all different from how the same list of cues would behave if there were no Group cue, and you’d be entirely correct. This mode is essentially an organizational tool to visually separate cues into different sections within the cue list, and to hide or show an entire batch of cues with a single click (on the grey triangular disclosure indicator to the left of the Group cue’s name.)

Start First

Start First Group cues have a blue outline with square corners.

When a Start First Group cue is started via a GO, its first child cue starts and the playhead advances to the next cue after the Start First Group cue. Therefore, other cues within the Start First Group cue will get skipped over unless they are connected with auto-continues or auto-follows. By using auto-follows and/or auto-continues, the children within the Start First Group cue can be made into a cue sequence, which will progress independently of the playhead.

Since the playhead advances to the cue after the Start First Group cue, you can continue to press GO, starting other cues and advancing the playhead, while the cues within the Start First Group cue are running.

Start Random

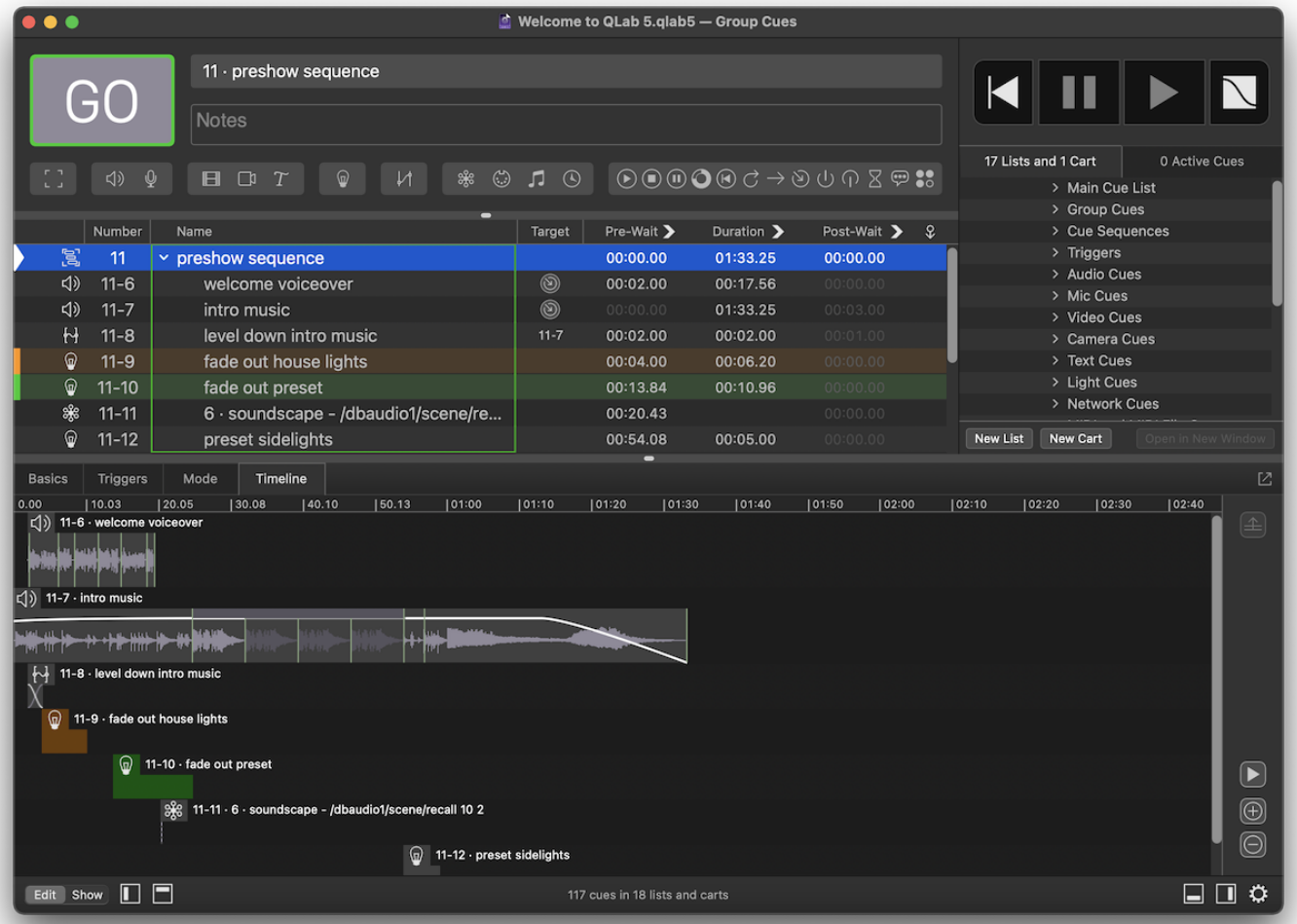
Start Random Group cues have a purple outline with square corners.

When a Start Random Group cue is started via a GO, a randomly selected child cue that is both armed and not currently playing is started, and the playhead advances to the first cue after the Start Random Group cue. If the Start Random Group cue is started more than once, each child cue that is armed will be started once before any of the cues is started a second time. This form of cueing is often referred to as “round robin” triggering. This is admittedly not truly random, but “start-random-cue-that-isn’t-disarmed-or-playing-and-be-sure-to-start-each-child-once” mode doesn’t quite have the same snappy sound to it.

The internal “memory” of a Start Random Group cue is reset each time the workspace is opened.

The Timeline Tab

Timeline Group cues have a Timeline tab in the inspector which displays the Timeline Group cue’s children in a visual timeline editor similar to the timeline found in many audio and video editing programs.



Each child cue in the Timeline Group cue appears in its own row in the timeline, in the same order as the cues are arranged in the Timeline Group cue. Cues show their type icon, their cue number, and their cue name. If a cue has a color, that color is also reflected in the timeline.

If an Audio cue or Video cue in the Timeline Group cue has slices, those slices are displayed. If a slice repeats a finite number of times, the slice is drawn the appropriate number of times with a grey bar to help make it visually clear which section of the cue is repeating. If a slice repeats infinitely, it's only drawn once and the grey bar displays an infinity sign (∞). If a slice is set to a count of zero, it is not drawn. Integrated fade curves are also drawn.

Cues can be adjusted in the Timeline tab in several ways:

Drag

When the pointer is hovering over a cue, it will turn into a hand. Click and drag left or right to move the cue in the timeline, thus adjusting its pre-wait time. While a cue is being dragged, blue guidelines appear indicating the start and end of all the other cues in the Group, as well as any slices within those cues. The dragged cue will snap to these guidelines in order to make it easy to align cues with each other. If you want to focus on the guidelines belonging to only a single cue, drag over that cue's "lane." The guidelines for all other cues will be hidden, and you can then drag left or right as normal. Dragged cues also snap to the current playback time which is indicated by the vertical yellow line. This makes it easy to pause playback at a desired moment, then drag a cue to exactly that moment.



To temporarily disable snapping, hold down the command key (⌘) while dragging.

Nudge

When one or more cues are selected in the timeline, the following keyboard shortcuts are available to nudge those cues forward and backward in the timeline by adding to or subtracting from their pre-waits:

- ⌥← - reduce pre-waits by 0.1 seconds.
- ⌥→ - increase pre-waits by 0.1 seconds.
- ⇧⌥← - reduce pre-waits by 0.01 seconds.
- ⌥⇧→ - increase pre-waits by 0.01 seconds.

Trim

When the pointer is hovering at the beginning or end of a cue which has a duration, it will turn into a bi-directional horizontal arrow. Click and drag left or right on the start of the cue adjust its start time and pre-wait time simultaneously, or on the end of the cue to adjust the end time of the cue. These adjustments necessarily also adjust the duration of the cue.



Slip

When the pointer is hovering over an Audio or Video cue, hold down the option key (⌥) to *slip* the cue, adjusting its start and end time without adjusting its pre-wait time or duration. Note that only cues without slices can be slipped.



Scale

When the pointer is hovering over the bottom edge of a cue, it will turn into a bi-directional vertical arrow. Click and drag up or down to scale the display of the cue. This adjustment is purely visual and does not alter the behavior of the cue in any way.

To scale the whole timeline view, instead of just a single cue, you can scroll vertically while holding down the option key (⌥), or use the keyboard shortcuts ⇧⌘- and ⇧⌘=.

Pin

Cues can be pinned to the top of the timeline view in order to make it easier to compare them to other cues lower down in the Group. Select a cue and click ⏏ to pin it to the top of the timeline. If the selected cue is already pinned, click ⏏ to unpin the selected cue. You can hold down the command key (⌘) while clicking to select multiple cues.



On the right side of the timeline view, beneath the pin/unpin button, are three more buttons:

- Click ▶ to preview the selected cue.
- Click ⊕ to zoom in on the timeline.
- Click ⊖ to zoom out on the timeline.

You can also zoom in and out on the timeline by using pinch gestures on a trackpad, by scrolling horizontally while holding down the option key (⌥), or by using the keyboard shortcuts ⌘- and ⌘=.

Timeline Groups as Cue Sequences

Timeline Group cues are the easiest way to create a cue sequence in which each cue is timed relative to the beginning of the sequence.



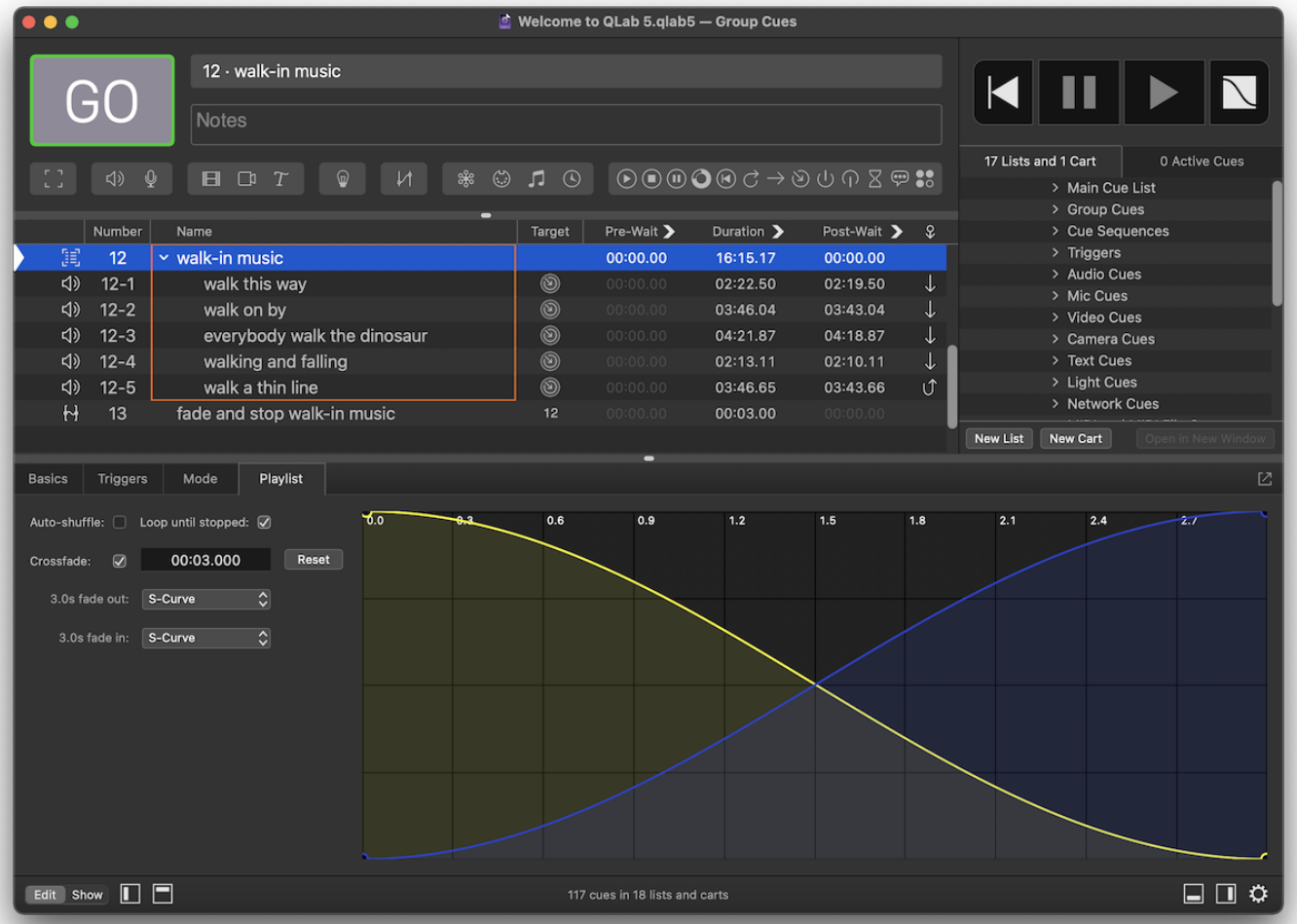
Number	Name	Target	Pre-Wait	Duration	Post-Wait
11	preshow sequence		00:00.00	01:33.25	00:00.00
11-6	welcome voiceover		00:02.00	00:17.56	00:00.00
11-7	intro music		00:00.00	01:33.25	00:03.00
11-8	level down intro music	11-7	00:02.00	00:02.00	00:01.00
11-9	fade out house lights		00:04.00	00:06.20	00:00.00
11-10	fade out preset		00:13.84	00:10.96	00:00.00
11-11	6 · soundscape - /dbaudio/scene/recall 10 2		00:20.43		00:00.00
11-12	preset sidelights		00:54.08	00:05.00	00:00.00

In this screen shot, cue `11` is a Timeline Group cue. When cue `11` starts, that causes cues `11-6` through `11-12` to start as well. Since all the children except `11-7` have pre-waits, those elapse before the main action of those cues begins. The total result is that at the moment of GO on cue `11`, cue `11-7` starts playing the intro music, then two seconds later cue `11-8` causes the intro music to fade down in level, then the welcome voiceover plays after that, and so on.

The Playlist Tab

Children of Playlist Group cues are treated as a discrete set of cues which play one at a time in sequence. Quintessential uses of a Playlist Group cue include pre-show, post-show, and intermission music, still-image slide shows, or sets of MIDI or OSC commands that need to be sent together but not simultaneously. Any type of cue can be placed inside a Playlist Group cue.

The Playlist tab in the inspector allows you to define the behavior of the Playlist Group cue.



Cues inside a Playlist Group cue are necessarily and automatically set to auto-continue with a non-editable post-wait time equal to their duration.

Auto-shuffle

When this box is checked, the children of the Playlist Group are randomly reordered each time the Group is loaded or started. When the cues are shuffled, their cue numbers and names are displayed in italics to remind you that they are not in their true order. This random order is not saved, and unchecking the *auto-shuffle* box restores the cues to their true order.



Loop until stopped

When this box is checked, the Playlist Group cue loops itself, automatically restarting its first child when it's last child finishes playing. When this box is checked, the last child in the Playlist Group cue displays a U-shaped arrow (↺) as its continue mode, indicating that when that cue completes, the Playlist Group cue will loop back around to the first child cue.



If a Playlist Group cue is set to loop, it must contain at least one cue with a non-zero duration. If it contains only zero-duration cues, the Playlist Group and its children will report as :fig-broken-5: broken. Setting least one child to a duration greater than zero will un-break all the cues.

Crossfade

When this box is checked, QLab will crossfade between cues in the Playlist Group cue. This only applies to cues with audio levels or video geometry; it's the main audio level and the opacity parameters that are crossfaded. When a Playlist Group cue is set to crossfade, several more controls are enabled and a crossfade curve display appears on the right side of the tab.

The duration of the crossfade can be as long or as short as you like, but any cue in the Playlist Group cue that is shorter than the crossfade duration will break since you cannot crossfade through a cue over more time than the cue lasts.

Pop-up menus allow you to select the fade curve shape of the fade out and fade in independently. The fade out curve is shown in yellow and the fade in curve is shown in blue.

Editing fade curves

- The start time and end time of each fade curve are editable by dragging the control points in the corners of the fade curve display.
- When using *Custom* or *linear* fade curve shapes, you can click along the fade curves to add control points. These can be dragged around to adjust the shape of the fade.
- When a control point is selected, the time for that control point is displayed at the bottom left of the inspector. You can type a time into that field to precisely adjust the time of the control point.
- When editing a control point that isn't the start or end point of a curve, the fade percentage is also displayed and can also be editing. This control describes the level of the audio and/or opacity at that moment in the fade, where 100% is the level that the audio and/or opacity are set to in the fading cue, and 0% is silence and/or 0% opacity.
- When a control point is selected, the fade curve that the control point belongs to is drawn with a solid line. The opposite curve is drawn with a dotted line.

Second Triggers

By default, Playlist Group cues automatically play through their children sequentially, starting each child cue when the previous child completes. However, you can also set a Playlist Group cue to advance to the next child cue, or step back to the previous child cue, using the Second Triggers pop-up menu in the Triggers tab of the inspector.

If a Playlist Group cue receives a GO while one of its children is playing, and its second trigger behavior is set to *plays next*, the Playlist Group cue will start the child cue after the one that's currently playing, crossfading if that option is set, and looping around from the last child cue to the first if that option is set.

If the behavior is set to *plays previous*, a GO will cause the Playlist Group cue to start the child cue previous to the one that's currently playing, likewise respecting crossfading and looping settings.

These options are particularly useful when the Playlist Group cue contains cues with an indeterminate length, such as an infinitely looping Audio cue, a Video cue targeting a still image, or a Text cue.

Broken Cues

Group cues can become broken for the following reasons:

A cue inside the Group is broken

Fix all the broken cues inside the Group to clear this warning.

A cue inside a Timeline Group is set to auto-continue or auto-follow

Auto-continue and auto-follow are not permitted for child cues of Timeline Groups. Set all child cues to *do not continue* to clear this warning.

A cue inside a Playlist Group is too short for the Group's crossfade settings

Adjust the length of the child cue or the length of the crossfade to clear this warning.

Cue Sequences

When two or more cues are started together from one single press of the GO button (or one single incoming MIDI command, MSC command, OSC command, hotkey press, etc.), this is called a **cue sequence.** Plainly put, cue sequences are the most straightforward way that you can have QLab do more than one thing in response to a single command.

Cue sequences can be built in three different ways:

- By connecting cues with auto-continues,
- By connecting cues with auto-follows,
- By putting cues into a Group cue set to Timeline mode.

You can download an example workspace which explores cue sequences here.

Auto-continue

A cue set to auto-continue will cause the following cue to start after the initial cue's post-wait time has elapsed. By default, cues have no post-wait time, so by default an auto-continue will cause the two cues to start simultaneously. If a post-wait is added to the first cue, the post-wait will begin to elapse at the moment that the cue is started. Once the post-wait elapses, the second cue will start.



This screen shot shows a cue sequence made up of three cues connected with auto-continues. Cue 1 ("intro music") is set to auto-continue and has a post-wait of 3 seconds. Cue 2 ("level down intro music") is also set to auto-continue and has a post-wait of 1 second. When cue 1 is standing by and you press GO, QLab will:

- Start cue 1 immediately, then,
- Three seconds later, start cue 2, then,
- One second later, start cue 3.

To set a cue to auto-continue, select that cue and then look in the Basics tab of the inspector. Click on the *Continue* pop-up menu in the bottom-left corner and choose *Auto-continue*. A straight arrow (↓) will appear in the far-right column of that cue's row in the cue list. To remove the auto-continue, select *Do not continue* from the pop-up menu.

Auto-follow

A cue set to auto-follow will cause the following cue to start after the initial cue has completed.



This screen shot shows a cue sequence made up of three cues connected with one auto-continue and one auto-follow. Cue 21 ("intro music") is set to auto-continue with no post-wait. Cue 22 ("level down music") has a pre-wait of 3 seconds, a duration of 2 seconds, and is set to auto-follow. When cue 21 is standing by and you press GO, QLab will:

- Start cue 21 and cue 22's pre-wait immediately, then,
- Three seconds later, start cue 22, then,
- Two seconds later, start cue 23.

If you change the length of cue 22 to eight seconds and then run the cue sequence again, QLab will start cue 21 and cue 22's pre-wait, wait eight seconds, and then start cue 23. Cue 23 will always start after cue 22 has completed.

To set a cue to auto-follow, select that cue and then look in the Basics tab of the inspector. Click on the *Continue* pop-up menu in the bottom-left corner and choose *Auto-follow*. An arrow with a flagged top ( ) will appear in the far-right column of that cue's row in the cue list. To remove the auto-follow, select *Do not continue* from the pop-up menu.

Other ways to edit continue mode

You can also adjust the continue mode of a cue by...

- ...clicking in the Continue column in the cue list to bring up a pop-up menu,
- ...using the keyboard shortcut for editing the continue mode of the selected cues, which is **C** by default,
- ...right-clicking (or control-clicking, or two-finger-clicking) on the cue in the list and choosing **Edit → Continue Mode** from the contextual menu.

Group Cues

The Group cue, whose job it is to contain other cues, is a great way to make cue sequences. Not every sequence needs to use a Group cue, and not every Group cue necessarily creates a sequence, but the two go together very often and very nicely.

The other modes of the Group cue can be used for cue sequences as well, which you can learn more about them from the Group cue section of this manual.

Disarmed Cues in Cue Sequences

Disarmed cues in a sequence do not interrupt or change the flow of events in a sequence. Their pre-waits, post-waits, and follows are still respected, but the cue itself does not execute. Disarmed Audio cues play no audio, disarmed Video cues play no video, disarmed MIDI cues send no messages, and so on.

GO, Start, Trigger, and Preview

There are several ways to start a cue in QLab, and some of these ways can change whether or not a cue sequence plays as normal.

If a cue in a cue sequence is started via:	... then:
GO	The cue and the rest of the sequence play as programmed and the playhead moves to the first cue after the sequence.
A hotkey trigger, MIDI trigger, Timecode trigger, or Wall Clock trigger	The cue and the rest of the sequence play as programmed, but the playhead does not move.
Audition GO	The cue and the rest of the sequence play as programmed, with all cues playing to the outputs specified in Workspace Settings → Audition, and the playhead moves to the first cue after the sequence.
Preview	The cue itself plays in isolation. The other cues in the sequence are not played, and the playhead does not move.
Audition preview	The cue itself plays to the output specified in Workspace Settings → Audition. The other cues in the sequence are not played, and the playhead does not move.

Chapter 3: Tools

3.1 The Tools Menu
3.2 Find
3.3 Paste Cue Properties
3.4 The Launcher Window
3.5 Monitor Windows
3.6 Auditioning Cues
3.7 Override Controls

The Tools Menu

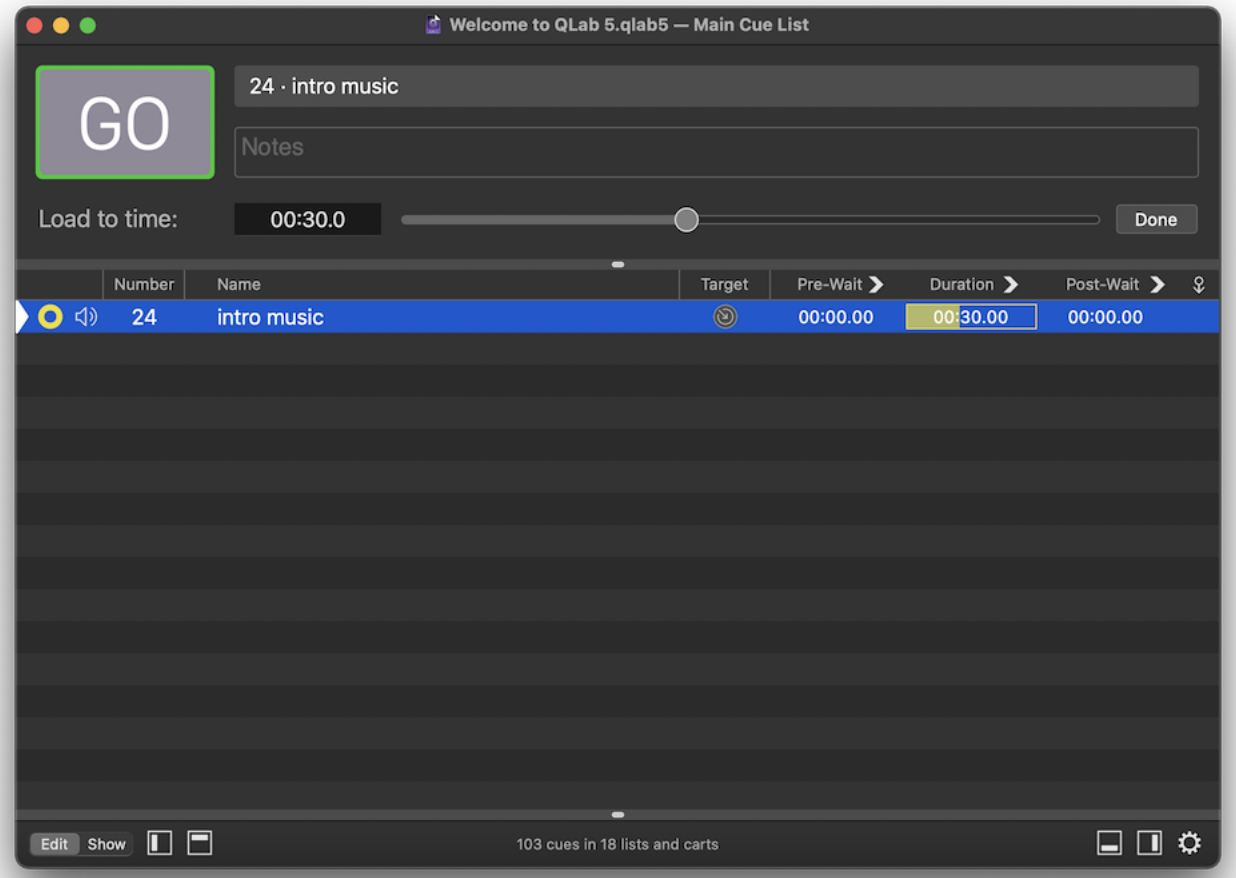
The **Tools** menu contains items related to assisting and managing your *workflow*, which is to say the process of building cues and cue sequences and operating QLab. The menu is context-sensitive, and can display different tools when different windows in QLab are active. When a cue list or cart window is active the **Tools** menu shows its default set of items.

Load To Time

The *Load to Time* tool allows you to scrub through the standing-by cue (or cue sequence) to a particular time before starting the cue. When the cue or sequence is started after loading-to-time, it will begin playing from that time. For example, if you are rehearsing a bit of your show that happens thirty seconds into the start of a cue, you can load that cue to `0:30` and then start it right at the moment you want to rehearse.

To access this tool, select *Load to Time* from the **Tools** menu, or use the keyboard shortcut ⌘**T**. *Load to Time* is one of the tools that occupies the toolbar, and selecting the menu item or using the keyboard shortcut cycles the toolbar through three states:

1. When the *Load to Time* tool isn't displayed, the command will display it.
2. When the *Load to Time* tool is displayed but the cursor isn't in its text box, the command will move it there.
3. When the *Load to Time* tool is displayed and the cursor is in the text box, the command will close the *Load to Time* tool and move focus back to the cue list.



To use the *Load to Time* tool, type a number into the text field or drag the slider to load the selected cue or cue sequence to your desired time.

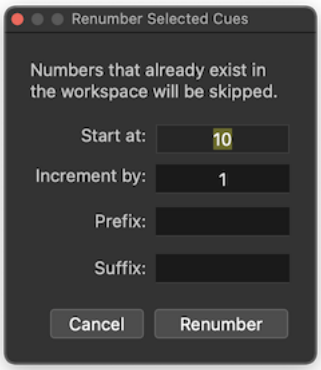


Negative Times

If you type a negative value into the field, QLab will load to that amount of time back from the end of the cue or sequence, if possible. For example, if you type `-10` into the field and press enter, QLab will load the selected cue to ten seconds before the end of the cue.

Cue Numbering

Renumber selected cues... (⌘R) will open a tool window that allows you to programmatically renumber all of the currently selected cues. There are four text fields in the renumber tool. The first two must be filled in, and the second two are optional.



- **Start at** is the first number you want QLab to use. Any whole or decimal number can be used here.
- **Increment by** is the amount you want QLab to add to create successive numbers. In the screen shot above, the selected cues will be renumbered to `10`, `11`, `12`, and so on. Any whole or decimal number can be used here.
- **Prefix** is optional. Any text entered here will be prefixed to the new cue numbers. If you entered `Test-` in the screen shot above, the selected cues would be renumbered `Test-10`, `Test-11`, `Test-12`, and so on.
- **Suffix** is optional. Any text entered here will be appended to the new cue numbers. If you entered `backup` in the screen shot above, the selected cues would be renumbered `10backup`, `11backup`, `12backup`, and so on.

The renumber tool will skip over any existing cue numbers when it runs. Using the screen shot above as an example, if you selected three cues before using the renumber tool, they would be renumbered `10`, `11`, and `12`. If there was already a cue `11` somewhere else in the workspace, however, the selected cues would be renumbered `10`, `12`, and `13`.

Delete numbers of selected cues (⌘D) will, as stated, delete the cue numbers of all selected cues. This will not alter cue targets or anything else at all, just the cue numbers.

Jump

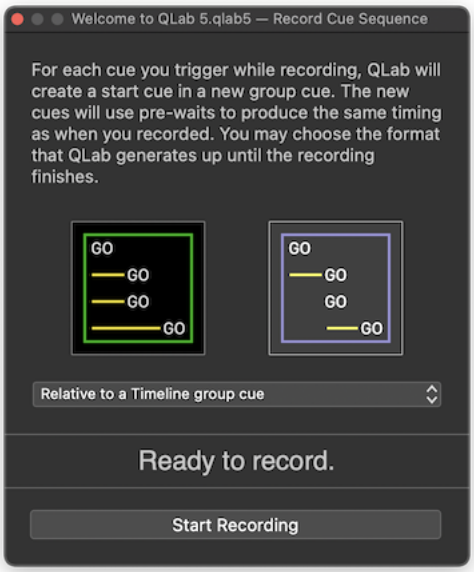
Jump to selected cues' targets (⌥⌘J). When the cue or cues that are selected target other cues in the workspace, this will move the selection to those target cues.

A similar tool from QLab 4, *Jump to cue*, has been renamed *Select cue…* and moved to the **View** menu.

Record Cue Sequence…

The sequence recorder tool allows you to play through cues by hand, and have QLab record your timing. When you stop recording, QLab will create a Group cue filled with Start cues which target the cues that you played while recording. The Start cues will have pre-wait times which exactly match your timing.

To use this tool, choose *Record cue sequence…* from the **Tools** menu. Then, choose whether you want to create a Timeline Group cue, with pre-waits on the Start cues relative to the beginning of the sequence, or a Start First Group cue with auto-continues on the Start cues and pre-waits relative to the previous Start cue. Then, click *Start Recording* and run the cues that you want to include in the sequence.



When you're done running your cues, click *Stop Recording*. QLab will create the new Group cue, full of Start cues, below the selected cue. Your original cues remain untouched. If you want to re-record the sequence, simply delete the new Group cue and try the process again. If you like the sequence and want to use it in your show, you can move the original cues into another cue list to keep them out of the way. It's important to remember that they cannot be deleted, however, since the sequence is made up of Start cues which target the original cues.

Always Audition

When *always audition* is turned on, clicking the GO button or using its keyboard shortcut (**space** by default) will behave as an audition GO instead of a regular GO. Likewise, previewing a cue by using the button in the Time & Loops tab or by using the keyboard shortcut (**V** by default) will behave as an audition preview instead of a regular preview.

Always audition has no effect on workspace MIDI controls or MSC messages, workspace OSC controls, or commands sent using OSC or AppleScript.

You can learn more about auditioning cues from the Auditioning Cues section of this manual.

Live Fade Preview

When *live fade preview* is turned on, any adjustments you make to a Fade cue or a Network cue in 2D Fade mode will be immediately reflected, as long as the target cue is playing. So, if you play an Audio cue, and then create a Fade cue targeting that Audio cue, any adjustments you make in the Fade cue will be audible.

When *live fade preview* is turned off, you can make adjustments to a Fade cue or a Network cue without hearing or seeing those changes until you run the cue. This can be helpful while building cues in rehearsal, to allow you to run an Audio or Video cue for the performers to work with, and then prepare a Fade cue for future use without disturbing them.

Beginning with QLab 5.4, *live fade preview* is a property of the workspace, not QLab as a whole. If you have multiple workspaces open at the same time, each will maintain its own *live fade preview* status.

Highlight Related Cues

When *highlight related cues* is turned on, QLab will highlight all cues which target or which are targeted by the selected cue. QLab uses a grey highlight for related cues, to distinguish from the regular highlight color as well as the yellow highlight color used by the Find tool.

Black Out and Restore Desktop Backgrounds

You can choose *Black out desktop backgrounds* from the **Tools** menu to have QLab set the desktop background of every screen connected to your Mac to plain black. This is, perhaps obviously, particularly helpful when working with Video cues. QLab will remember the existing desktop backgrounds before replacing them with black, and you can restore them by choosing *Restore saved desktop backgrounds*.

Tools for Group Cues

When a Group cue is selected, the **Tools** menu also shows a Group-specific item, **Shuffle order of cues in Group**. This item only operates on Playlist Group cues.

Tools for Audio and Video Cues

When an Audio or Video cue is selected, the **Tools** menu also shows the following file-target-specific items:

- **Open target file in external editor.** This will open the target media file of the selected cue in whichever application is designated as the default for that media type. For example, by default, .MP3 files will open in iTunes, .MOV files will open in QuickTime Player, and so forth. You can change these defaults by selecting a media file in the Finder, choosing *Get Info* from the **File** menu, and changing the assignment under the heading "Open with:"
- **Reveal target file in Finder.** This will open a window in the Finder showing the folder which contains the target media file of the selected cue.
- **Open file target search tool.** This will open a window that helps you address the problem of cues missing their file targets. This can happen when a workspace is moved from one computer to another, or when media files are reorganized on disk while the QLab workspace is closed. The search tool allows you to specify a folder which QLab will search inside of, looking for files that match the missing file targets of cues in the workspace. You can confirm or reject each match, and then have QLab automatically relink one or multiple cues automatically.
- **Paste audio levels.** This will bypass the *Paste Cue Properties* sheet and simply paste the audio levels of a previously copied cue, if possible.
- **Paste integrated fade shape.** This will bypass the *Paste Cue Properties* sheet and simply paste the integrated fade curve of a previously copied cue, if possible.

Tools for Fade Cues

When a Fade cue is selected, the **Tools** menu also shows the following fade-specific items:

- **Paste audio levels.** This will bypass the *Paste Cue Properties* sheet and simply paste the audio levels of a previously copied cue, if possible.
- **Paste integrated fade shape.** This will bypass the *Paste Cue Properties* sheet and simply paste the fade curve of a previously copied cue, if possible.
- **Set Parameters From Target.** This will display the Paste Cue Properties sheet, allowing you to choose properties of the Fade cue's target cue to copy into the Fade cue.
- **Set Audio Levels From Target (⇧⌘T).** This will bypass the *Paste Cue Properties* sheet and simply copy the Audio Levels tab of the Fade cue's target, if applicable.
- **Set Video Geometry From Target (^⌥⌘V).** This will bypass the *Paste Cue Properties* sheet and simply copy the Video Geometry tab of the Fade cue's target, if applicable.
- **Revert Fade Action (⇧⌘R).** When this command is invoked after running a Fade cue, QLab reverts the levels of the target cue to whatever they were before the Fade cue ran *except* for levels which have been otherwise changed. That is to say, the only adjustments that are reverted are the ones that the selected Fade cue caused.

Tools for the Light Dashboard

When the Light Dashboard is the frontmost window, the contents of **Tools** menu is re-populated with items pertinent to lighting. You can learn about those items from the page on the Light Dashboard in the Lighting section of this documentation.

Tools for Light Settings (Patch menu)

When Workspace Settings → Light is being viewed, the **Tools** menu is replaced by the **Patch** menu, which contains the following light patch-specific items:

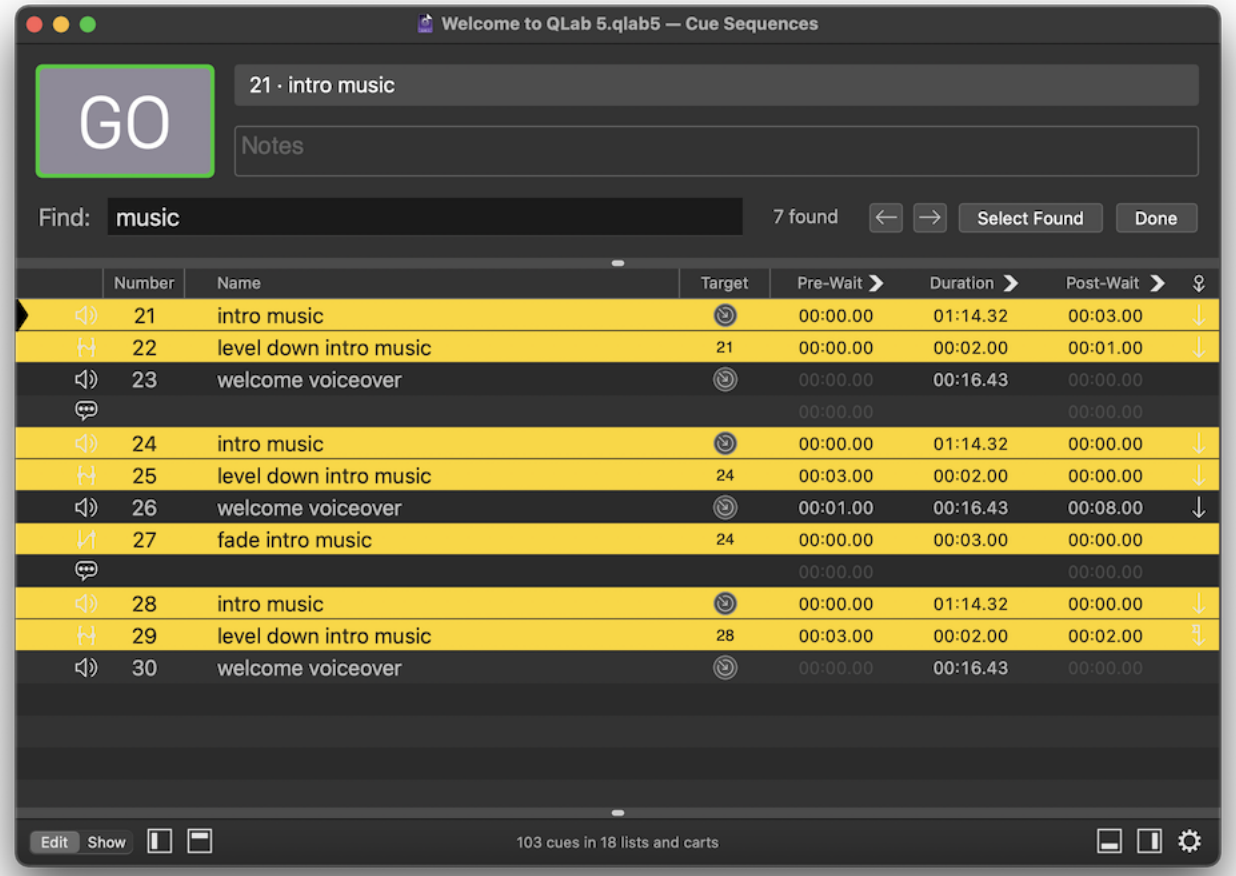
- **Unpatch all** unpatches all instruments in the workspace, removing all address and output assignments.
- **Unpatch selected** unpatches only the selected instruments, leaving the rest alone.
- **Auto-patch all** opens a dialog box which allows you to select the DMX output you want to use, as well as the starting address. If you select Art-net for output, you can also choose a starting universe, sub-net, and net. QLab then assigns addresses to all instruments according to those settings.
- **Auto-patch selected** behaves like *auto-patch all*, but only operates on the selected instruments.
- **Select all instruments** does precisely what it seems like it does. You can also select all instruments by using the keyboard shortcut **⌘A** while viewing the light patch.

Find

You can search your workspace by selecting *Find* from the **Edit** menu in the toolbar, or by using the keyboard shortcut ⌘**F**. QLab will search cue names, cue numbers, the file names of Audio and Video cue targets, cue notes, the contents of Text cues, the contents of Network cues, and the contents of Script cues. Searches are limited to one cue list or cue cart at a time.

Find is one of the tools that occupies the toolbar, and selecting the menu item or using the keyboard shortcut cycles the toolbar through three states:

1. When the *Find* tool isn't displayed, the command will display it.
2. When the *Find* tool is displayed but the cursor isn't in its text box, the command will move it there.
3. When the *find* tool is displayed and the cursor is in the text box, the command will close the *Find* tool and move focus back to the cue list.



Clicking the left and right arrow buttons will jump the selection to the previous or next found cue.

Clicking *Select Found* will select all found cues at once.

Clicking *Done* will clear the search and close the *Find* tool.

If a found cue is standing by, the playhead is drawn in black instead of the usual grey in the interest of visibility.

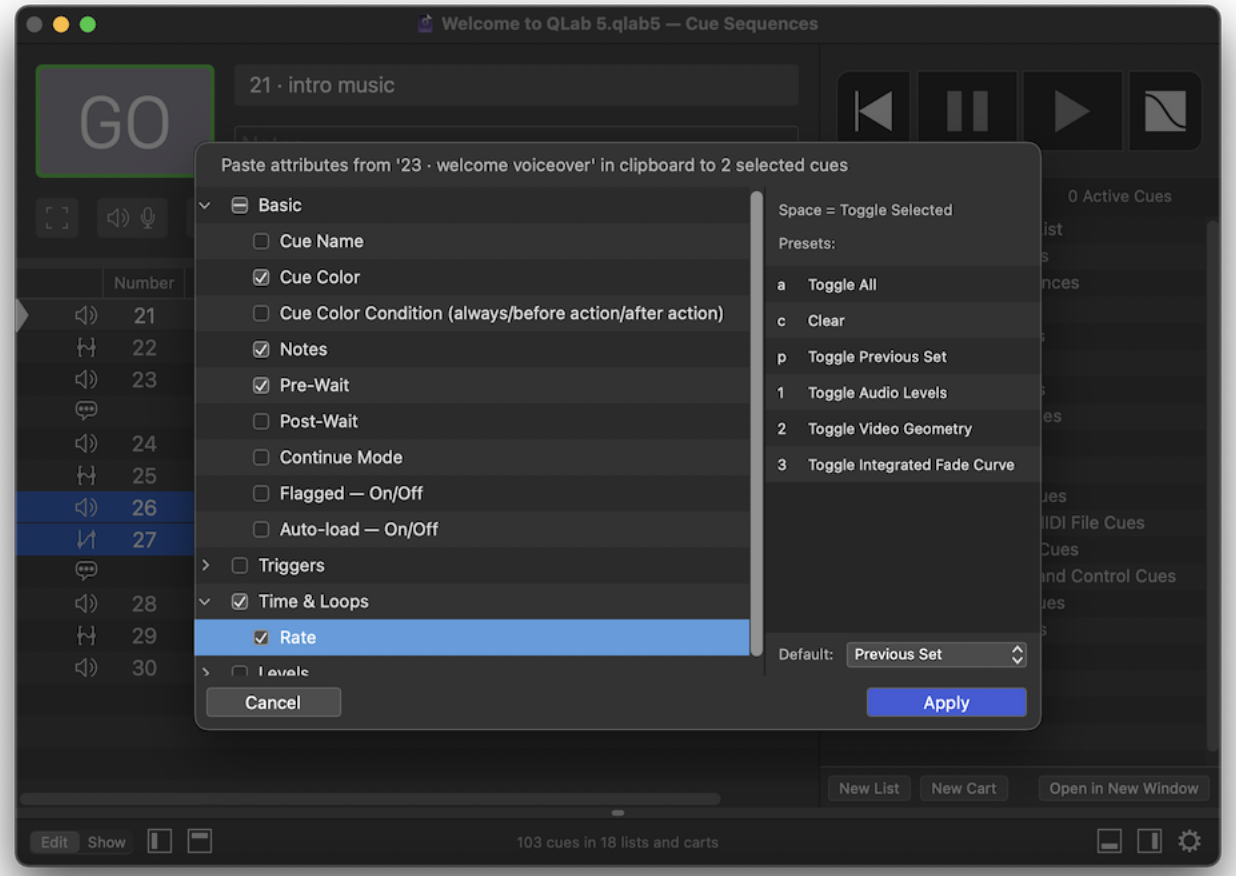
Paste Cue Properties

QLab allows you to copy the properties of one cue to other cues in order to simplify the process of making changes to groups of cues, or duplicating the behavior of one cue in another.

To do this, first select the cue whose properties you want to duplicate, and copy that cue by choosing *Copy* from the **Edit menu** or using the keyboard shortcut ⌘**C**.

Next, select the cue or cues that you want to paste the properties onto, and choose *Paste Cue Properties...* from the **Edit menu** or use the keyboard shortcut ⇧⌘**V**.

QLab will display a sheet showing a categorized list of properties to paste.



Categories will be hidden if they are not relevant to the selected cue. In this screen shot, for example, only the categories that are paste-able onto both Audio cues and Fade cue are visible, because both those cue types are selected.

You can navigate the list of properties with the mouse, using the disclosure triangles to unfold each category and checkboxes to select properties, or you can use the arrow keys to navigate and the space bar to check or uncheck boxes.

Hitting the **enter key** or clicking *Apply* will paste the selected properties onto the selected cue or cues. Hitting **escape** or clicking *Cancel* will close the sheet without making any changes.

On the right side of the sheet is a list of presets to quickly toggle sets of checkboxes that are commonly used together. Simply press the key corresponding to a preset to toggle the checkboxes belonging to that preset. The *p* key (short for *p*revious) always corresponds to the most recently used

set of checkboxes.

At the bottom of the list of presets is a drop-down menu that lets you select a default preset. The preset chosen from this menu will dictate which checkboxes are checked when *Paste Cue Properties* is invoked. So, for example, if you pretty much always use *Paste Cue Properties* to paste audio levels, you could select “Audio Levels” from this drop-down. Then, whenever you invoke *Paste Cue Properties*, the audio checkboxes will be checked to begin with, so you can just hit enter and be done with it.

The Launcher Window

The Launcher window, which appears by default when you open QLab 5, provides a centralized place to get started working with QLab.



On the left side of the window, you'll see the exact version of QLab that you're using, followed by four links to the web.

- **Documentation** links to this manual.
- **Tutorials** links to the Tutorials section of this manual.
- **Support** links to https://qlab.app/support/ which tells you all about how to get technical support for QLab.
- **https://qlab.app** links to the main QLab website.

Beneath those links are four buttons which correspond to four of the most commonly used first actions after QLab opens.

- **Open Workspace** displays a window which lets you browse for and open a preexisting workspace.
- **New Workspace** creates a new workspace using the default workspace template.
- **Connect to Workspace** displays the QLab Collaboration connection window, which allows you to connect to a workspace on another Mac.
- **Licenses** opens the license manager window, where you can install, remove, or just check in on your licenses.

On the right side of the window, you'll find two tabs.

The Recent Workspaces tab lists the most recently opened workspaces, which is the same list found when you select *Open Recent...* from the **File** menu. Double click on a workspace to open it, or click once and then either click the **Open Workspace** button at the bottom of the window, or use the **return** or **enter** key on your keyboard.

To clear this list, choose *Clear Menu* from the bottom of the list of recent workspaces in the **File** menu.

The Templates tab lists all the workspace templates on your Mac. Double click on a template to create a new workspace using that template, or click once and then either click the **New Workspace from Template** button at the bottom of the window, or use the **return** or **enter** key on your keyboard.

When this tab is active, you can click on the **Manage** button to delete or rename workspace templates, or to select a new default template.

Launcher Window behavior

If QLab is set to open the Launcher window at launch, which can be configured in QLab Preferences, it will appear whenever you launch QLab directly. It will not appear if you launch QLab by opening a workspace. The Launcher window closes itself whenever you use it to open a workspace or create a new one, and will reappear when the last workspace has been closed.

To open it at any time, you select *Launcher Window* from the **Window** menu.

Monitor Windows

Monitor windows allow you to view video that passes through your workspace outside of its regular context. There are four types of monitor windows in QLab, each designed to help with a different situation.

All monitor windows have a **float** checkbox, which sets the window to float on top of other windows, and a **clone** button which creates a copy of the monitor window. Multiple copies of a monitor window can be helpful if you have several screens connected to your Mac, each for a different teammate to look at.

Video Output Monitor Windows

A video output monitor window shows a live copy of the output that's going to a particular video output stage. All video which is visible on a stage is likewise visible in the monitor window that belongs to that stage.

Output monitor windows are helpful as confidence monitors, when the person operating QLab cannot see an audience-facing display but still needs to view the output.

You can open an output monitor window using the **Monitor** buttons found in Workspace Settings → Video → Video Output or the I/O tab of a Video, Camera, or Text cue.

Video Input Monitor Windows

A video input monitor window shows the live signal from a video input patch. This can help you ensure that your video input device is working properly, or make sure a camera's framing is correct before starting a Camera cue that uses that input.

You can open an input monitor window using the **Monitor** buttons found in Workspace Settings → Video → Video Input or the I/O tab of a Camera cue.

Audition Monitor Windows

If a workspace is set to redirect video output to audition windows in Workspace Settings → Audition, any Video, Camera, and Text cues which are auditioned will display in an audition monitor window instead of on their assigned video output stage.

If the audition window for a stage is not visible when you audition a cue that needs it, QLab will open the window for you. You can also manually open audition monitor windows using the **Open Audition Windows** button in Workspace Settings → Audition.

Unlike earlier versions of QLab, audition monitor windows don't turn always audition on and off by virtue of being open or closed.

Map Monitor Windows

A map monitor displays the live status of an audio map and all objects within it. When a cue containing cue objects is playing, its objects are visible in the map monitor for the map that the cue is using. When the cue stops, its objects disappear from the map monitor. Map objects are always visible in the map monitor since they are always present and "active" even if no audio is passing through them.

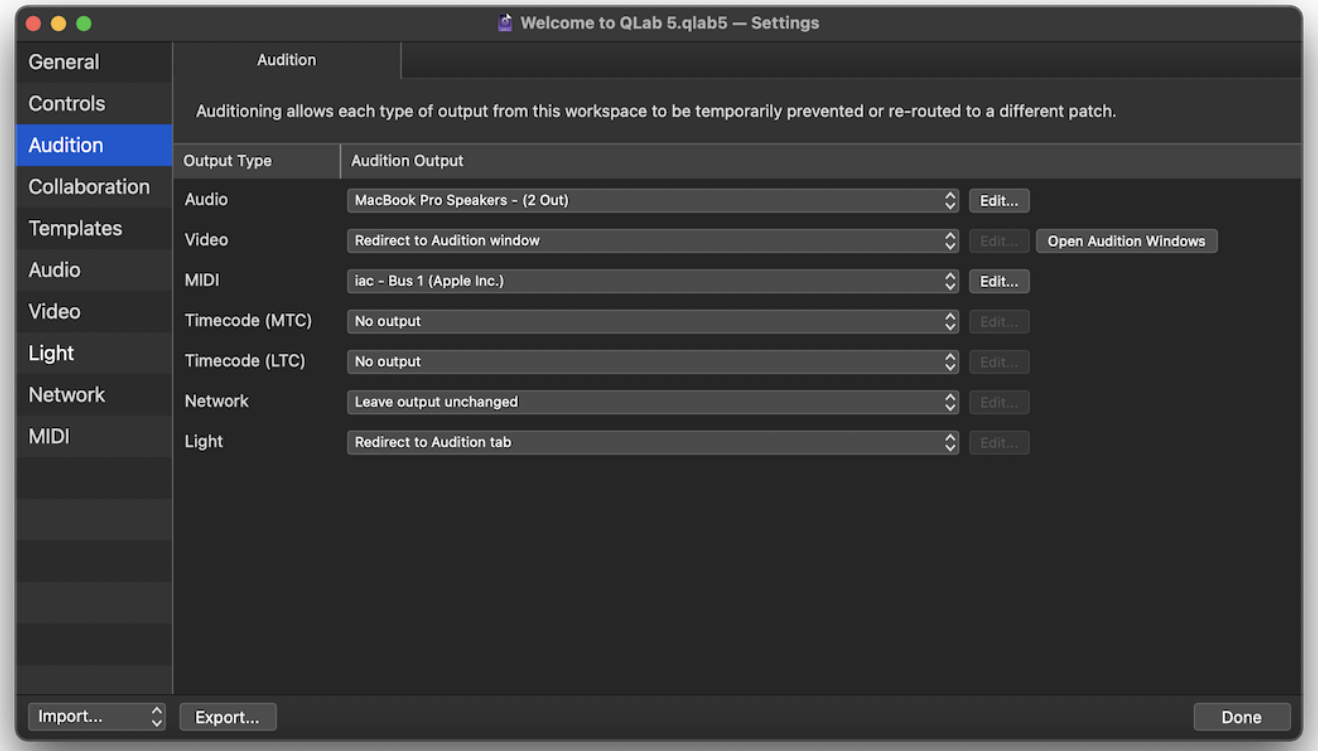
You can open an audio map monitor window using the **Monitor** buttons found in Workspace Settings → Audio → Audio Maps or the Objects tab of a cue that is using the audio map.

Auditioning Cues

Auditioning a cue means playing that cue outside of the context of your show as a whole, perhaps experimentally or while developing the cue for later use. In situations like this, it can be preferable to play the cue in a different way than normal. For example, you might want to audition an individual Audio cue in your headphones instead of through the sound system for everyone to hear. If you are working on a complex video system, but the lighting designer is focusing lights and the projectors must be turned off, you might want to audition all Video cues on your computer's screen so that you can get work done without disturbing your colleague.

Setting Up Audition

Workspace Settings → Audition is where you tell your workspace what you want it to do when you audition a cue of a particular type. There, you'll find a table which divides all output from QLab into seven categories:



- **Audio** means the audible output of Audio, Mic, Video, and Camera cues.
- **Video** means the visual output of Video, Camera, and Text cues.
- **MIDI** means the output of MIDI and MIDI File cues, including MIDI Voice, MSC, and SysEx messages.
- **Timecode (MTC)** means the output of Timecode cues set to MTC (MIDI timecode) mode.
- **Timecode (LTC)** means the output of Timecode cues set to LTC (linear timecode) mode.
- **Network** means all output of all Network cues.
- **Light** means all DMX output from QLab, including from Light cues, the Light Dashboard, and the DMX status window.

Each type of output has a pop-up menu next to it which contains two to four of the following options:

- **Leave output unchanged** - output of this type will behave the same way in cues which are auditioned as it will in cues which are played normally.
- **No output** - output of this type will be entirely prevented from cues which are auditioned.
- **Alternate patch** - output of this type will temporarily use the specified output patch instead of the patch that's programmed into the cue.
- **Redirect to Audition window** - (Video only) output of this type will be displayed in an audition-only monitor window instead of sent to the video stage that's programmed into the cue.
- **Redirect to Audition tab** - (Light only) output of this type will be displayed in the Audition tab of the Light Dashboard and not sent to DMX output hardware.

Note that using an alternate patch to audition video requires a video license.

Output type	Applies to	Leave unchanged	No output	Alternate patch	Audition window/tab
Audio	Audio cues, Mic cues, and audio output from Video and Camera cues.	✓	✓	✓	✗
Video	Video output fron Video, Camera, and Text cues.	✓	✓	✓	✓
MIDI	MIDI cues, MIDI File cues.	✓	✓	✓	✗
Timecode (MTC)	Timecode cues set to MTC.	✓	✓	✓	✗
Timecode (LTC)	Timecode cues set to LTC.	✓	✓	✓	✗
Network	Network cues.	✓	✓	✗	✗
Light	Light cues.	✓	✗	✗	✓

Using Audition

There are two workspace-wide controls for auditioning cues: *audition* GO and *audition preview selected*, both of which have keyboard shortcuts which can be customized in Workspace Settings → Controls → Keyboard. When these controls are used, any cues that they start will play through the audition output that has been selected for their output type.

The default keyboard shortcut for audition GO is ⌥**space**. The default keyboard shortcut for audition preview is ⌥**V**.

If a cue sequence is standing by, audition GO will cause the entire cue sequence to run according to its regular timing, with every cue in the sequence playing through the audition output that has been selected for its output type.

Cues which are started normally while other cues are auditioning will play through their normal outputs.

- GO will start the standing-by cue/cue sequence through normal outputs and move the playhead to the next cue/cue sequence.
- **Preview** will start the selected cue or cues through normal outputs, skipping any pre-waits, ignoring any auto-continues and auto-follows, and leave the playhead where it is.
- **Audition** GO will start the standing-by cue/cue sequence through audition outputs and move the playhead to the next cue/cue sequence.
- **Audition preview** will start the selected cue or cues through audition outputs, skipping any pre-waits, ignoring any auto-continues and auto-follows, and leave the playhead where it is.

If a cue is in the midst of auditioning and it's instructed to start normally, which is to say to play through normal outputs and *not* audition, then the cue will immediately stop and restart through normal outputs. You do not need to manually stop it first. The idea here is to make it as easy as possible to audition a cue, then play it normally as quickly as possible.

Always Audition

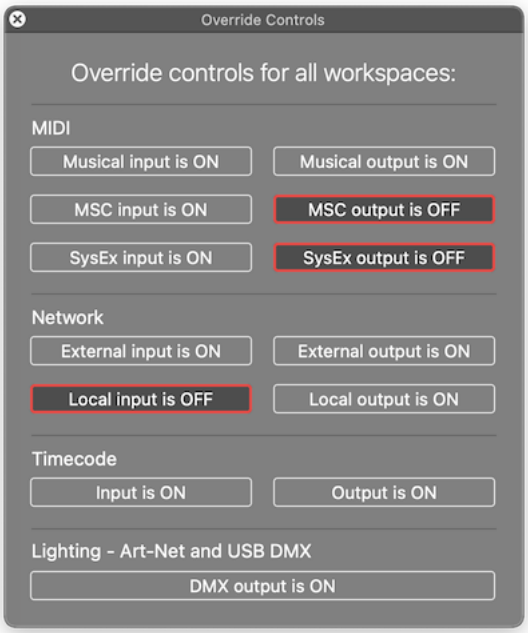
Sometimes, you might prefer to audition all cues for a while, such as in the above example of a projection designer working while the video system is switched off.

In this case and others like it, you can select *Turn on always audition* from the **Tools** menu. When always audition is turned on, the GO button will become an **Audition** button, colored blue to help clarify that the button is in a different mode. Additionally, the word *Audition* will appear in the workspace footer. When *always audition* is on, clicking the **Audition** button or using its keyboard shortcut (**space** by default) will behave as an audition GO instead of a regular GO. Likewise, previewing a cue by using the ▶ button in the Time & Loops tab or by using the keyboard shortcut (**V** by default) will behave as an audition preview instead of a regular preview.

Always audition has no effect on workspace MIDI controls or MSC messages, workspace OSC controls, or commands sent using OSC or AppleScript.

Override Controls

The Override Controls window can be opened by choosing *Override Controls* from the **Window** menu or by using the keyboard shortcut ⇧⌘**O**. This window provides a way to temporarily suspend input and output of various types of signals to and from QLab.



You can independently override input and output of each of the following types of signals:

- “Musical” MIDI voice messages
- MIDI Show Control (MSC)
- MIDI SysEx
- Network messages to and from other devices on the network
- Network messages within QLab and between QLab and other software on the same Mac
- Timecode
- Art-Net and USB DMX

Because QLab does not respond to incoming Art-Net or DMX, there is no Art-Net and USB DMX input override.

When input overrides are engaged, QLab will display a message in red text in the footer of the workspace. When output overrides are engaged, any cue whose output is overridden will show a red circle icon in the status column (⃠) to indicate that the cue will not output anything when it’s run.

Override controls are global, which means that they apply to all open workspaces.

Chapter 4: Workflow Tools

4.1 The Workspace Status Window
4.2 The Timecode Status Window

The Workspace Status Window

The Workspace Status window contains six tabs which are designed to assist with programming cues and troubleshooting your workspace.

The Warnings Tab

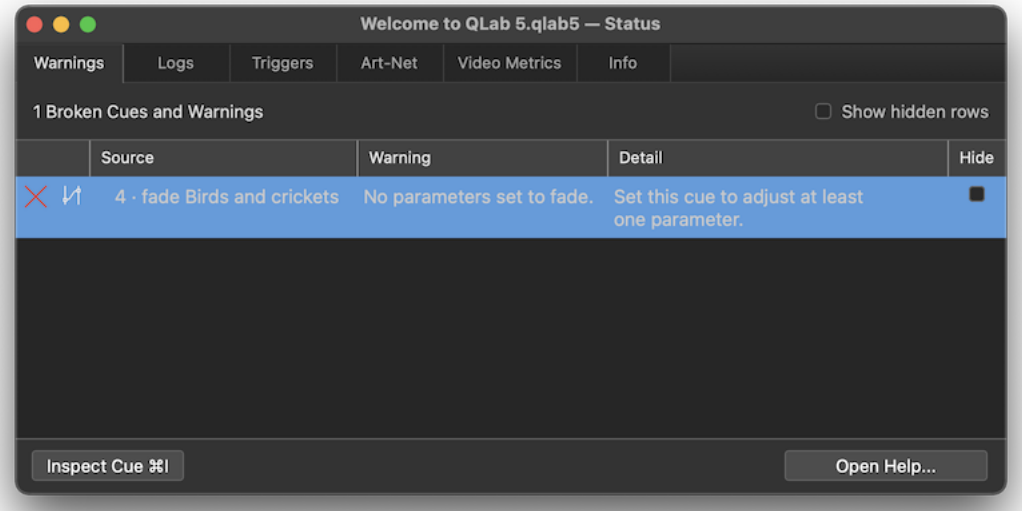
Warnings are created by cues and workspaces settings that have a detectable problem. Sometimes these problems will definitely cause trouble in performance, such as an Audio cue that has no target file, but other times these problems might not cause trouble. For example, a Light cue which contains several light commands that are functional and a single light command which controls a light that is unpatched might be a problem or might not, but only you, the human person, can know for sure.

There are several kinds of warnings:

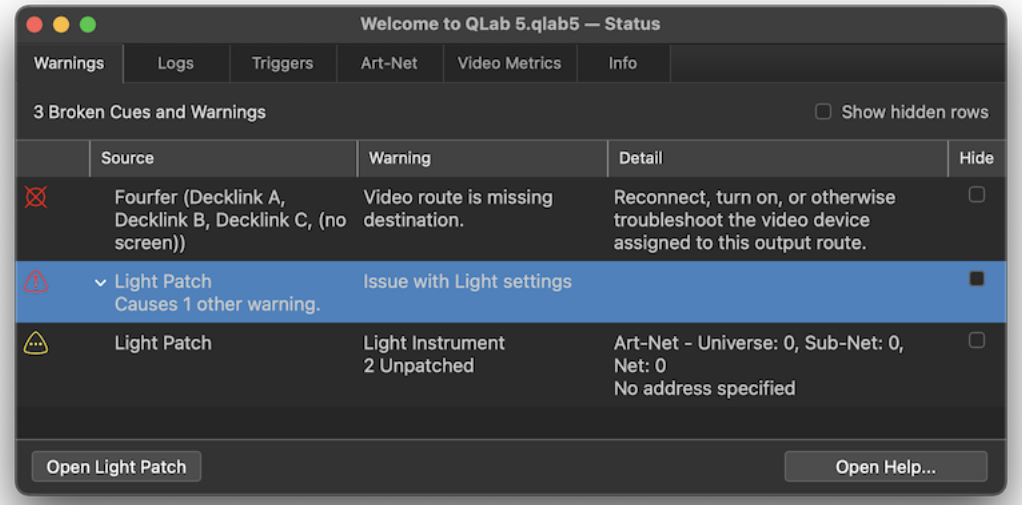
- **Broken cues** cannot be played because they lack some necessary bit of information, or because they depend upon a workspace setting which is incompletely or improperly configured. Broken cues will not play when they're started, but if they are part of a cue sequence QLab will attempt to leave |the sequence intact by running the broken cue's pre-wait, duration, post-wait, and auto-continue or auto-follow. QLab's goal is to prevent a single broken cue from causing any trouble outside itself.
- **Flagged** cues count as warnings so that you can easily work through the list of flagged cues to address the reason for their flag. A flag doesn't change the behavior of a cue, it's just a marker for you to use however you see fit.
- **Non-breaking warnings** are created when QLab has detected a problem but cannot determine whether the problem is important or not. A misconfigured light instrument definition is an example of a non-breaking warning; as long as there is no light instrument using that definition, it won't cause a problem in your show.
- **Breaking warnings** are created when QLab has detected a problem that can be determined to prevent a show from running correctly, or when the problem causes other warnings. A misconfigured audio patch that causes cues to break is a good example of a breaking warning.
- **Disconnection warnings** stem from missing hardware or software that was previously connected to QLab, but which cannot be found. For example, a MIDI output patch that was assigned to use a MIDI interface will generate a disconnection warning if the MIDI interface is unplugged.

Using the Warnings Tab

When a broken cue is selected, the **Inspect Cue** button in the lower left corner of the window causes QLab to select that cue in the main workspace window and, if the workspace is in edit mode, display the inspector so that you can immediate edit the cue and address the reason for the cue's brokenness.



When a non-cue warning is selected, the button in the lower left corner of the window adapts to the selected warning. In the screen shot below, the button is labeled **Open Light Patch** and clicking it will open the Light Patch tab in Workspace Settings → Light.



No matter what type of warning is selected, the **Open Help...** button in the lower right corner of the window opens a new window in your default web browser and brings you to the section of this manual that's most relevant to the problem. In the screen shot showing the broken cue, the **Open Help...** button brings you to the section on fading audio levels. In the screen shot showing the unpatched light instrument, the **Open Help...** button brings you to the section on the Light Patch.

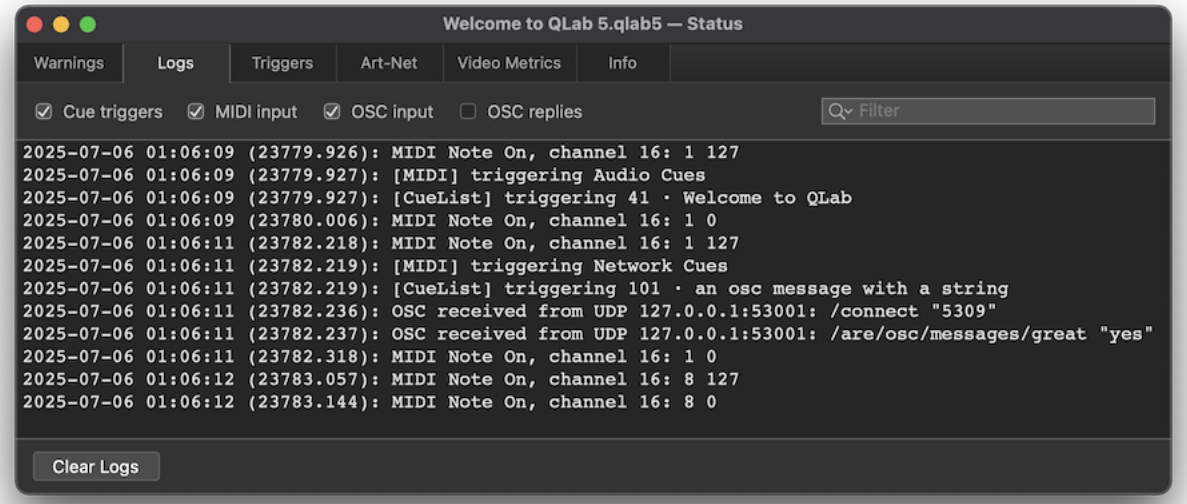
Hide and Show hidden

You can check the box in the *Hide* column to hide a warning from view. Hidden warnings will only be shown when the *Show hidden* checkbox at the top right of the window is checked. This allows you to focus on distinct groups of warnings while choosing to ignore others. For example, you may wish to hide a few flagged cues from the warnings list because you know you won't be addressing them until tomorrow and you want a clear view of the other warnings in your workspace.

When warnings are hidden, the header text in the upper right corner of the window will be adjusted to remind you.

The Logs Tab

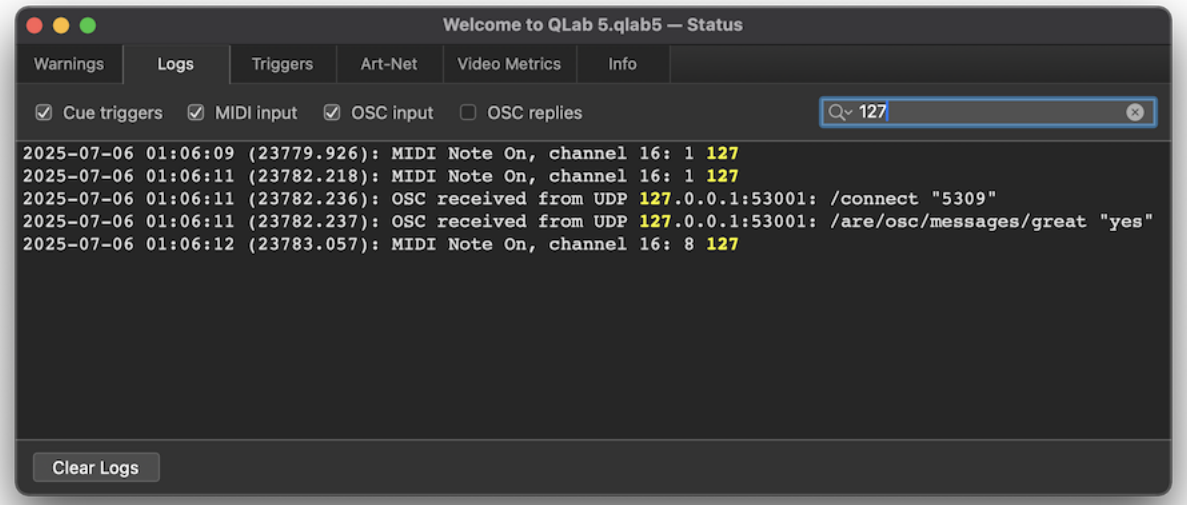
The Logs tab has four checkboxes for enabling or disabling logging of specific kinds of events in QLab. Please note that these logs are entirely separate and different from your Mac's Console logs and unrelated to the logging level set in QLab Preferences.



- **Cue triggers.** QLab will report any time a cue, cue list, or cue cart is started via a GO or a trigger. It will also report anytime a cue trigger is blocked via workspace double-GO protection.
- **MIDI input.** QLab will report any MIDI messages received by the workspace.
- **OSC input.** QLab will report any OSC messages received by the workspace.
- **OSC output.** QLab will report any OSC messages that it generates, including messages sent by Network cues, show control broadcast, and messages generated in response to OSC messages sent from other workspaces, devices, or programs.

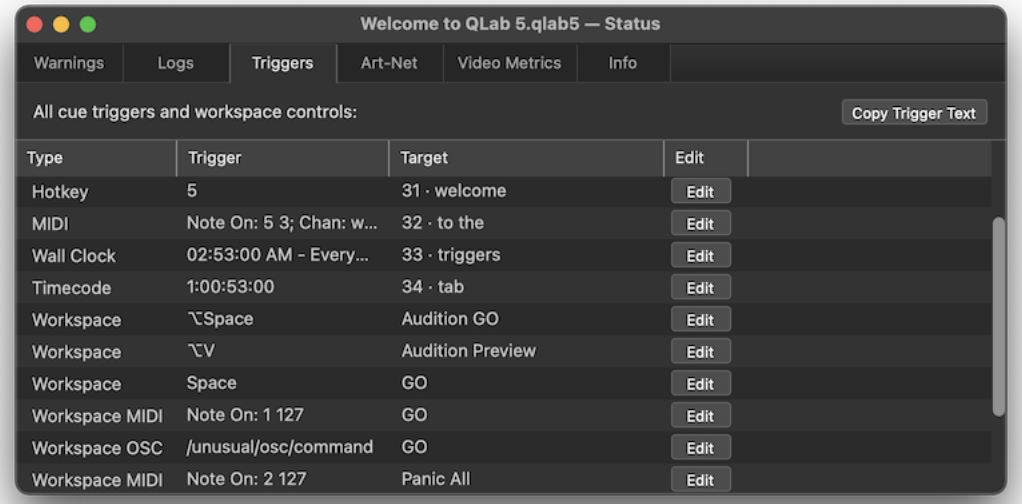
Beginning with QLab 5.5, the state of these checkboxes will be saved with the workspace, so if they are checked when the workspace is saved and closed, they will be checked when the workspace reopens.

The filter field allows you to temporarily filter the contents of the log text so that you can focus on certain messages while troubleshooting.



The Triggers Tab

The Triggers tab provides a central place to view and edit all the cue triggers and workspace controls in your workspace.

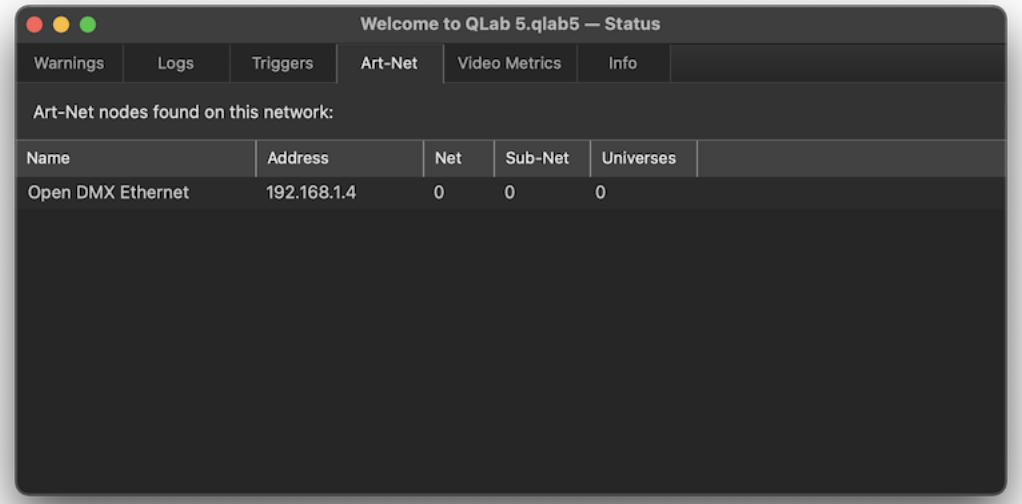


The list of triggers can be sorted by clicking on the column headers in the table.

Clicking the **Edit** button next to a cue trigger will select that cue in the main workspace window and display the Triggers tab of the inspector. Clicking the **Edit** button next to a workspace trigger will open the Workspace Settings window showing the appropriate tab of Controls settings.

The Art-Net Tab

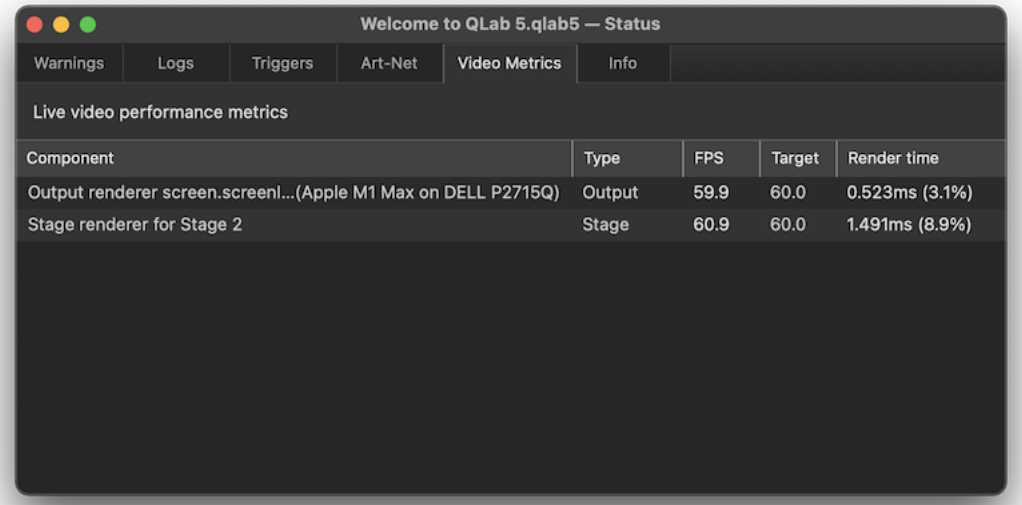
If your workspace contains at least one lighting instrument, QLab will scan your local network for available Art-Net devices. If any are found, they will be listed in this tab with their name, IP address, and available Art-Net output information.



Some devices do not support *discovery* which is the part of the Art-Net specification that facilitates this scanning and reporting. If your Art-Net device is connected and properly configured but still does not appear in this list, you may find that it works anyway, If it doesn't, you may find that it will work if you enable Art-Net broadcast mode in QLab Preferences.

The Video Metrics Tab

The Video Metrics tab displays the framerate, target framerate, and render time for each stage and video output device currently in use by your workspace.



The **FPS** column shows the actual current frame rate. It is normal for this number to drift slightly higher or lower than the target frame rate, both because clock timing can drift slightly and because the sampling rate of the metrics tab can be slightly offset from the refresh cycle of the stage or device.

The **Target** column shows the target FPS of the stage or device. For video devices, this is determined by the device's clock. For stages that use a single video device, that device's clock is the clock for the stage. For a stage that uses multiple devices, QLab determines the most suitable clock to follow. Often it's as simple as "whichever hardware video device is used by this stage or output that has the highest refresh rate."

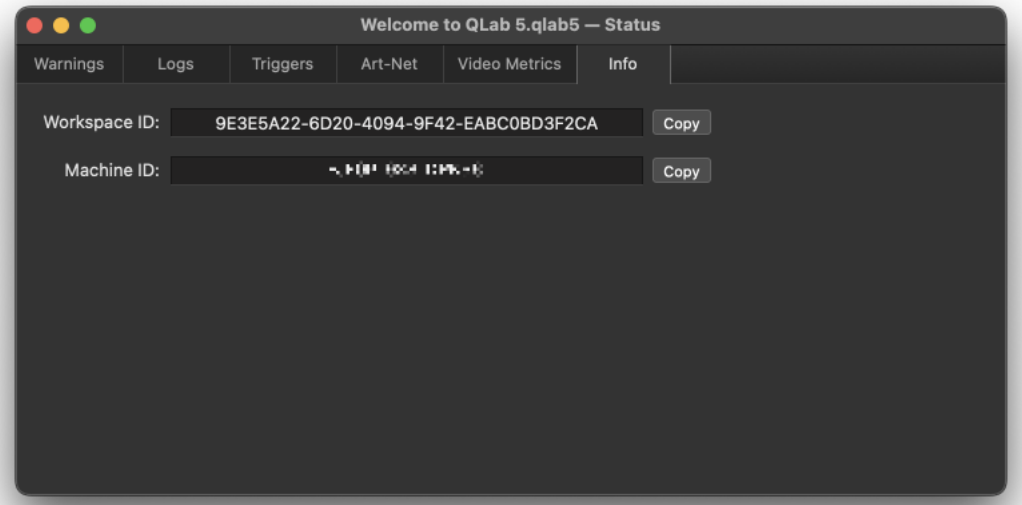
If the current frame rate drops below 90% of the target FPS, it will be drawn in orange. If it drops below 80%, it will be drawn in red.

The **Render time** column shows the amount of time QLab is taking to render one frame of video for the given stage or device, both as an absolute figure in milliseconds and as a percentage of one frame as defined by the stage or device. For example, in the screen shot above, the device has a target refresh rate of 60 FPS, and the render time of one frame of video for that output is 0.523 milliseconds, which is 3.1% of 1/60 of a second.

It's important to note that a frame rate below the target frame rate isn't necessarily a sign of trouble. For example, some Apple displays have a variable refresh rate up to 120 Hz, and so stages or outputs which use those displays will always show a target rate of 120, even if the screen is actually running at 60 FPS or even lower. As with all metrics, the numbers don't always tell the whole story.

The Info Tab

The Info tab shows basic information about your workspace.



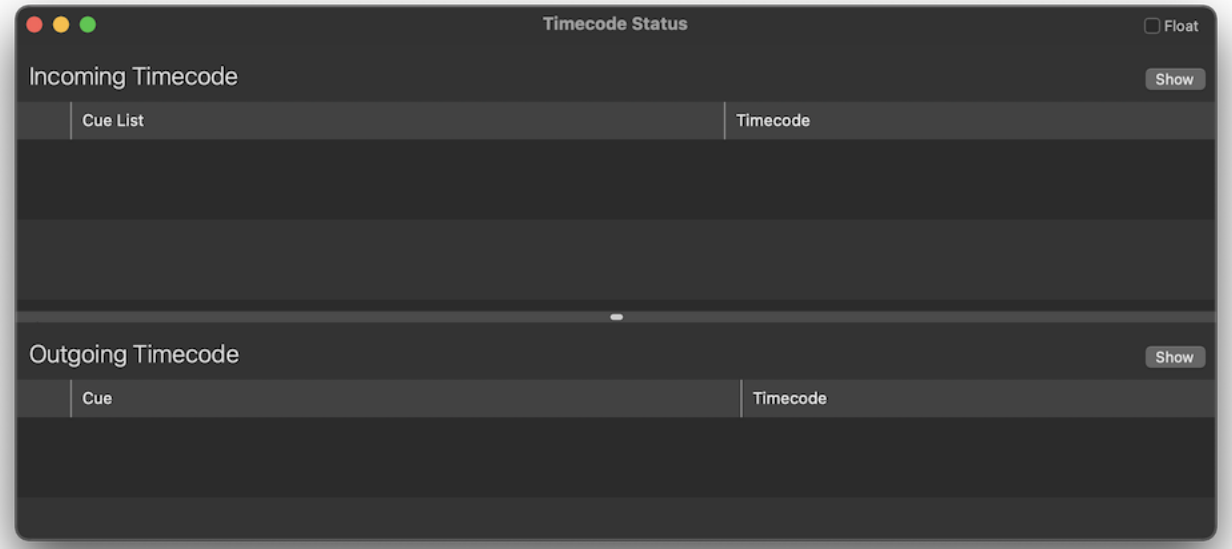
Workspace ID. The workspace ID is a unique identifier for this workspace, which you can select and copy in order to use in AppleScripts, OSC messages, or any other situation in which you need to know the unique ID of the workspace.

Machine ID. This number is unique to each Mac, and is the identifier that the license system uses to associate license seats with Macs. You may need to grab this number if you're communicating with support@figure53.com about license questions.

The Timecode Status Window

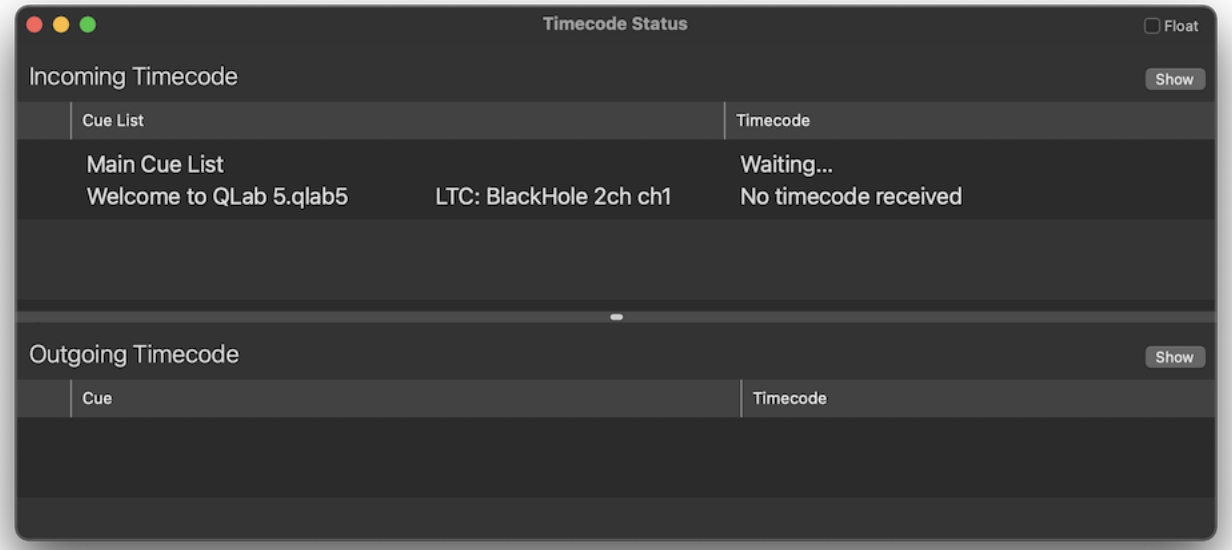
The Timecode Status window gives you a centralized place to view the status of all the incoming and outgoing timecode connected to the open workspaces on your Mac.

If no Timecode cues are running, and no cue lists are set to sync to incoming timecode, the window will be empty.

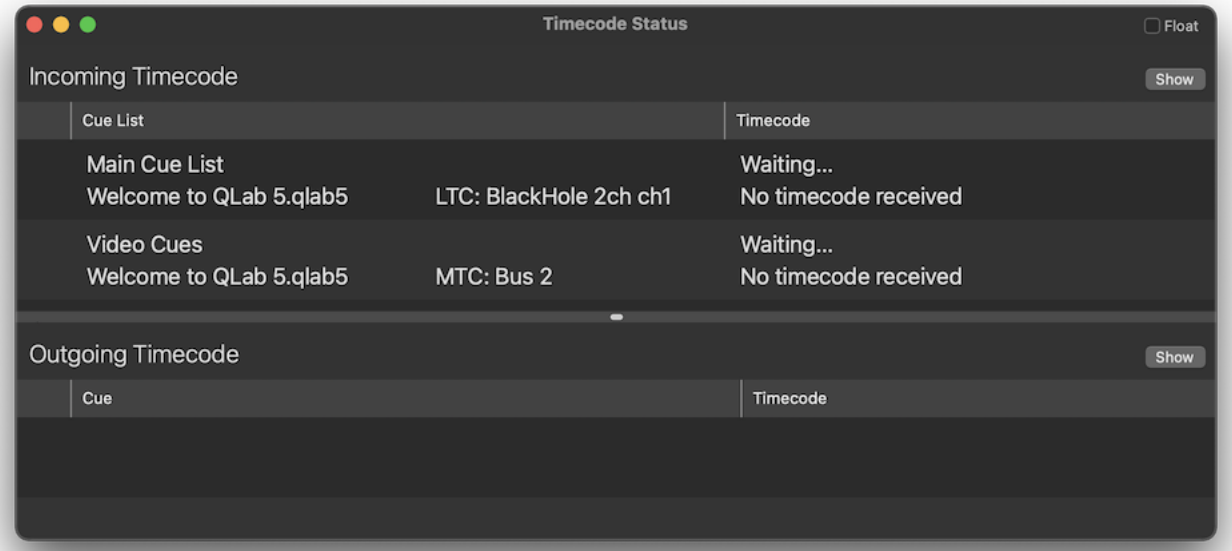


Incoming Timecode

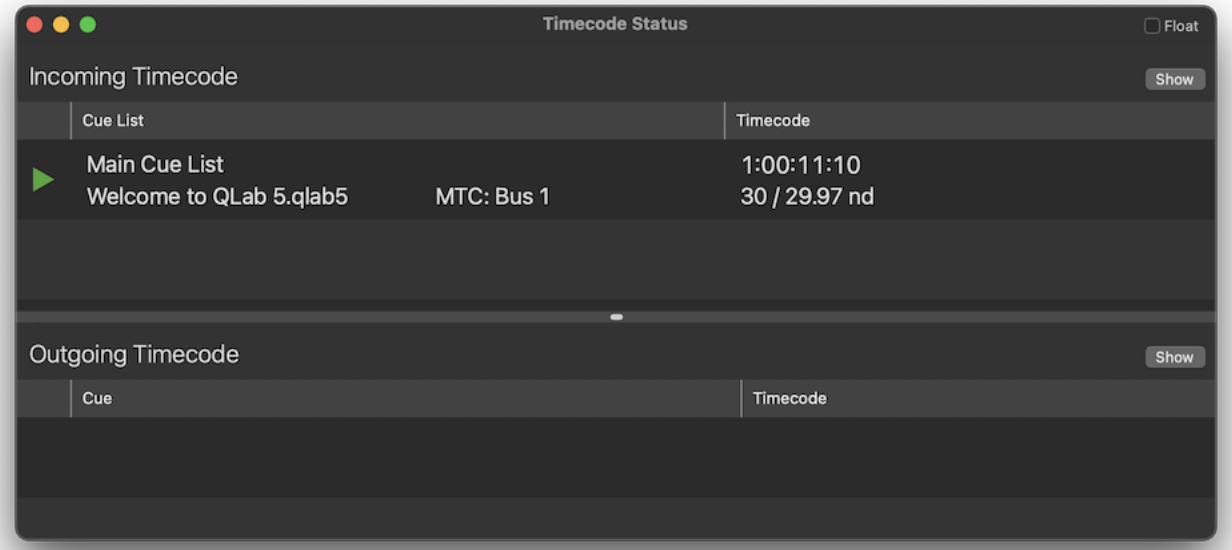
If a cue list has been set to sync to incoming timecode, that cue list will be displayed in the upper portion of the window, along with the name of the workspace that contains it, the form of timecode that the list wants (LTC or MTC), and the input on which that timecode is expected.



If multiple lists are set to sync to timecode, they're all listed one after another.

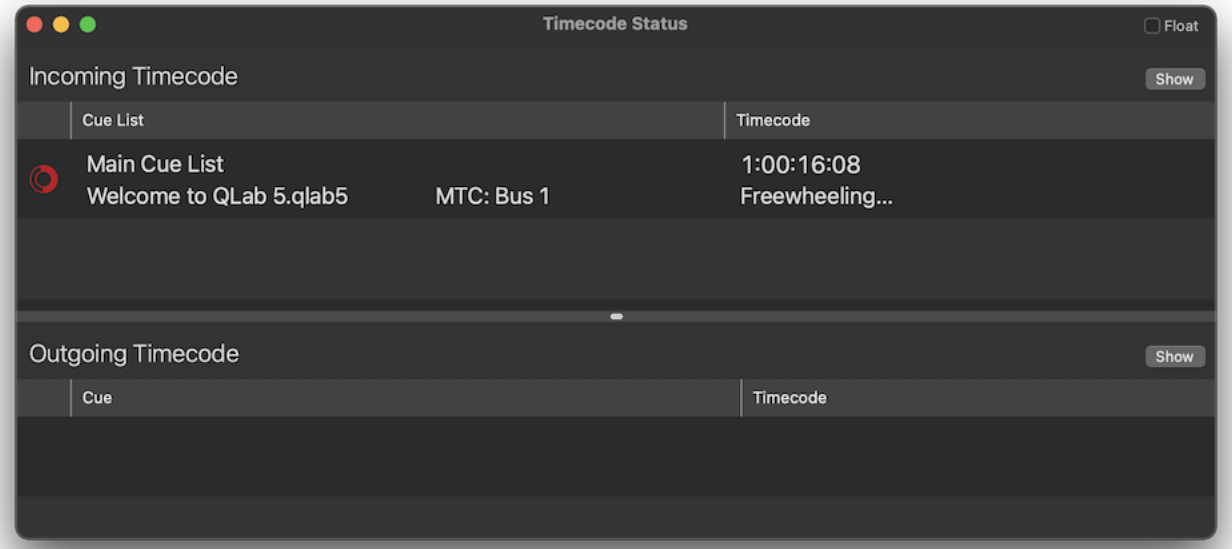


Once timecode starts coming in, the cue list displays a ▶ running status icon and the right-most column displays the timecode.



You can click the upper **Show** button while timecode is incoming to display a large and resizable timecode monitor window, suitable for viewing at a distance. If there are multiple incoming streams of timecode, the most recently started one will appear in this monitor window.

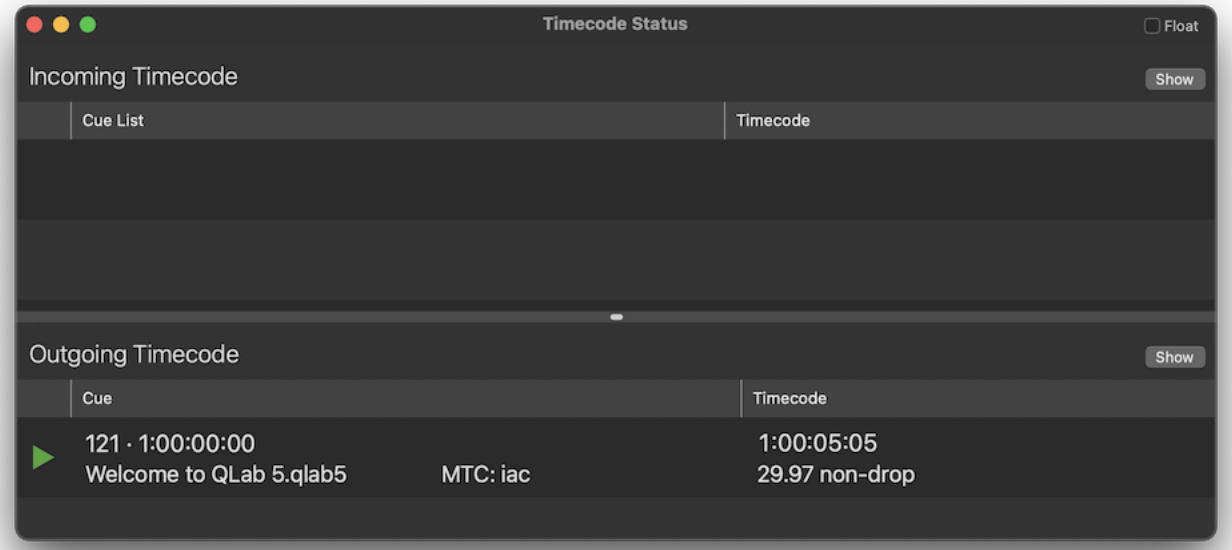
When incoming timecode stops, the display changes based on how the list has been configured in its Timecode tab. If the list is configured not to freewheel at all, the display simply changes back to the "waiting" stage. If the list is configured to freewheel *and* to stop timecode-triggered cues when timecode stops, a red disc shows the freewheel time elapsing.



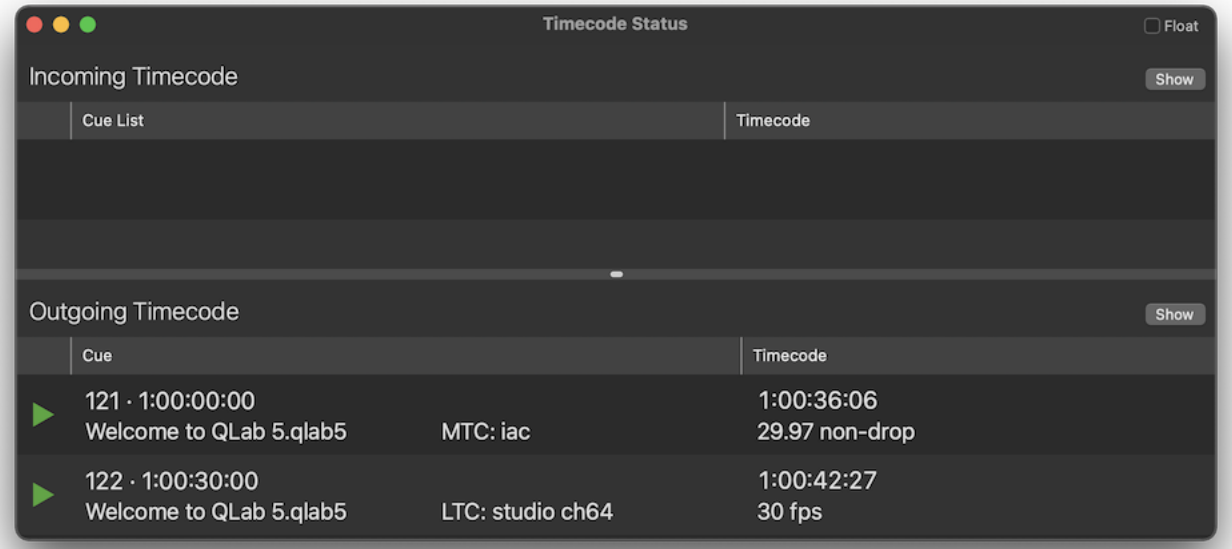
The disc is shown in yellow if the list is configured to pause timecode-triggered cues when timecode stops, and in grey if the list is configured to do nothing when timecode stops.

Outgoing Timecode

When a Timecode cue runs, that cue's details appear in the lower portion of the window.



If multiple Timecode cues run at once, they all appear.



You can click the lower **Show** button while timecode is outgoing to display a large and resizable timecode monitor window, suitable for viewing at a distance. If there are multiple outgoing streams of timecode, the most recently started one will appear in this monitor window.

Chapter 5: Audio

5.1 Intro to Audio
5.2 Object Audio in QLab
5.3 Audio Cues
5.4 Mic Cues
5.5 The Audio Output Patch Editor
5.6 The Audio Map Editor
5.7 Fading Audio & Audio Effects

Intro to Audio

Audio is the heart of QLab.

Audio can come into QLab from an audio file, an audio track within a video file, a live input on an audio device connected to your Mac, or the audio track within an incoming NDI stream.

Audio can go out from QLab using the built-in speakers or headphone jack on your Mac, any macOS-compatible[1] audio interface (usually connected via USB or Thunderbolt), or network-based audio output systems like Dante, AVB, and NDI.

> **Note:** Timecode cues set to LTC output also generate audio in the form of an LTC signal from scratch. Since LTC is not meant to be a signal that human beings listen to, at least not on purpose, it's treated rather differently in QLab. Therefore, this section of the manual does not apply to Timecode cues which you can learn more about from the Timecode cue section of this manual.

Audio cues and Video cues generate output by reading audio from an audio file such as an AIFF, WAV, or other compatible file type, or by reading audio data from a compatible video file.

Mic cues and Camera cues generate output by reading audio from a live audio input such as a microphone or line input connected to your Mac's audio interface, or from audio contained in an NDI stream.

Once inside QLab, audio is treated more or less the same way no matter how it arrived.

Each cue that generates audio has a **cue matrix mixer**, which is unique to that cue and can be found in the Audio Levels tab of the inspector. The cue matrix mixer routes audio from the cue to the **patch matrix mixer** of the audio output patch that the cue is using. The patch matrix mixer, in turn, routes audio to the outputs of the audio device that the patch is using.

What Is A Matrix Mixer?

A matrix mixer is a mixer in which the signal from each input can be sent to each output at an individually set level. You can visualize a matrix mixer as a set of rows and columns. In QLab, each row represents an input, and each column represents an output. The intersection of each row and column is called a *crosspoint*; the point at which the row and column cross each other. Crosspoints are effectively volume knobs which control how much signal flows from the row into the column.

The matrix mixers in QLab consist of four sets of controls:

- **Input level controls** set the overall level of individual inputs. Adjusting an input level control will adjust that signal's volume in every output.
- **Output level controls** set the overall level of individual outputs. Adjust an output level control will adjust the level of all signals mixed into that output, but have no effect on those signals' level in other outputs.
- **Crosspoint level controls** set the level that an input feeds into an output.
- **The main level control** sets the overall level of the output from the whole matrix.

The Cue Matrix Mixer

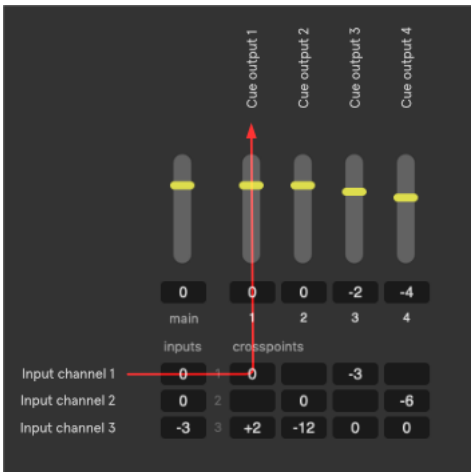
These images show a close-up view of the **cue matrix mixer** in QLab (with some additional annotations). The top row, row 0, contains the output level controls. Each successive row represents an input with audio entering the mixer on the left edge and flowing to the right.

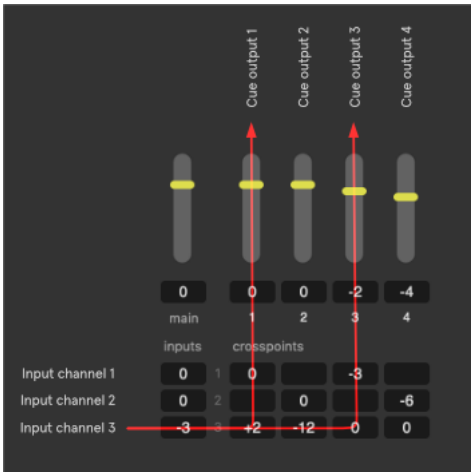
The left-most column, column 0, contains in the input level controls. Each successive column represents a cue output with audio entering each output via the crosspoints in the middle of the matrix and flowing upwards to the output level controls.

In the top left corner, at row 0/column 0, is the main level control which sets the overall level for the whole cue.

In this example, the cue has three input channels routed to four cue outputs. If the cue is an Audio or Video cue, the inputs represent channels of audio in the target file. If the cue is a Mic cue or Camera cue, the inputs represent input channels on the source audio device.







In QLab, like most digital audio environments, a level of `0` represents *unity gain* which means "this level control makes no change in level." Whatever level comes in is the level that goes out. Therefore, we see that input channel 1 is routed to cue output 1 with no change in level, because the input, crosspoint, and output level controls for that signal path are all set to `0 dB`. If input channel 1 is a recording of test tone that measures, say, `-14 dB` in the audio file, then a measurement of cue output 1 would also be `-14 dB` assuming no other sound was being played.

The whole point of the matrix, however, is that every level control can be set to any value. Input channel 3, for example, is set to `-3 dB` at the input, meaning its overall level is `3 dB` quieter than the level recorded in the file. The crosspoint at the intersection of input 1 and output 1 is set to `+2 dB`, so the sum total level of input 3 in cue output 1 is `-3 + 2` or `-1 dB` in cue output 1.

Input channel 3 is also routed to cue output 3, with the crosspoint at unity. Cue output 3's output level is set to `-2 dB`, however, so the sum total level of input 3 in cue output 3 is `-3 + -2` or `-5 dB` in cue output 3.

Much of the time, particularly with smaller sound systems, this flexibility is not necessary and level adjustments to cues can be done largely with the main level control alone. When you need to get more exact, however, the full flexibility of the matrix mixer is there for you.

The Patch Matrix Mixer

In the **patch matrix mixer**, each row represents the *cue outputs* from the cue matrix mixer. All the audio from all the cues is summed together and feeds into this matrix from the left side. The columns represent the *device outputs*, the outputs that are offered by your audio interface.

In these images, the audio patch is using a virtual audio device called Dante Virtual Soundcard which has 64 output channels, but to keep this discussion simpler we'll only consider the first eight outputs. The first image shows each cue output routed to a single device output at unity gain, so each cue output is effectively a direct path to one corresponding device output.

This is a great way to start working with a multichannel sound system in QLab, but it's certainly not the only way. For example, imagine this assignment of device outputs:

1. Main house left speaker
2. Main house right speaker
3. Main center speaker
4. Subwoofer
5. Onstage effect speaker 1
6. Onstage effect speaker 2
7. House left front fill
8. House right front fill

In this situation, device outputs 7 and 8 are feeding speakers that cover a section of audience that cannot hear the main speakers fed by outputs 1, 2, and 3. It might be nice if those outputs could automatically receive an appropriate mix if the signals being sent to those other device outputs, and voila, that is possible!

The second image shows cue output 1 routed to device output 1 at unity, and also routed to device output 7 at `-3 dB` (assuming that the front fills need a bit less oomph than the mains.) Cue output 2 repeats this with cue output 8. Cue output 3 routes to both 7 and 8 at `-6 dB` to reinforce the center channel.

You can learn more about the capabilities of audio output patches from the section on the Audio Output Patch Editor in this manual.

Levels Inside QLab





While levels in QLab are of course measured in decibels, it's important to understand that QLab necessarily uses *dBFS* (decibels relative to full scale) and not *dB SPL* (decibels of sound pressure level). QLab cannot know the absolute loudness of a sound for a number of reasons, but the simplest one is that QLab cannot know about the hardware that is being used beyond QLab. You might be playing QLab into a pair of tiny headphones, or you might be plugged into a million-watt sound system.

When a control is set to a level of `0 dB` inside QLab, it simply means that the signal entering that control will be the same level when it leaves. If the control is set to `-3 dB` then the signal exits the control 3 dB quieter than it entered. That difference of `-3`, however, might translate into a larger or smaller difference depending upon the rest of the sound system.

The World Beyond QLab

In the most technical sense, what QLab knows as the device outputs are not necessarily the actual outputs of the audio device. Device manufacturers are responsible for creating the software drivers for their devices, which tell the Mac about the outputs and other capabilities of the hardware. The Mac, in turn, tells QLab. So, the device might have its own internal routing or other special features which could make for a discrepancy between what QLab sees and what's really there. For example, the iConnectivity device in the images above is listed as having eight outputs, but some of those outputs do not correspond to physical connections on the outside of the device. That's something that QLab cannot know.

By and large, any devices we've seen that do things like this are well designed and documented, so all that's needed to get the full picture is to read the manual for your audio interface and make sure you understand what's going on. If you find yourself having trouble, we encourage you to contact the manufacturer of the device before reaching out to us.

If the audio interface has a software control panel or virtual mixer interface, that control panel or mixer acts as a final post-QLab set of controls for the hardware.

The simplest example of this is the headphone jack on your Mac, which has volume controls on the keyboard and in the menu bar. These volume controls come "after" QLab, and therefore behave as an ultimate arbiter of the overall output volume. Other audio interfaces use other software controls, but the principle is the same.

Synchronization and Simultaneity

Because QLab is designed for a live playback environment, in which different things may happen at different times and each performance of the same workspace may be slightly different, QLab uses an elastic approach to synchronization which is a bit unusual when compared to other audiovisual software. The end goal of this approach is to allow you to not think about sync at all if you don't want to worry about it, and to allow you to get specific, predictable behavior if you do want to worry about it.

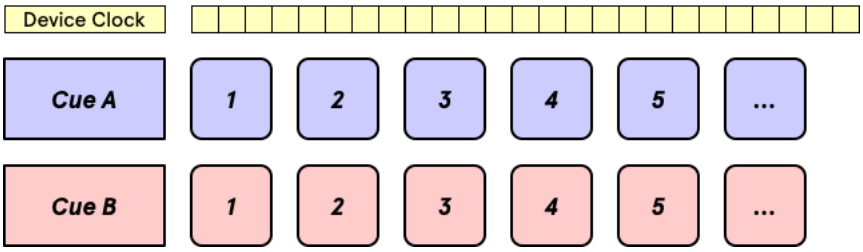
If you do not want to worry about it, feel free to skip over this section.

If you do want to worry about it, there are two guidelines to remember when working on your cues:

1. Playback of cues which have audio is controlled by the clock of the audio device to which they are assigned[2], so cues which are assigned to the same audio output patch will maintain sample-accurate synchronization.
2. Cues which are started at the same time via a single action will start simultaneously[3].

Since there are two guidelines with two possible conditions each, there are four possible cases:

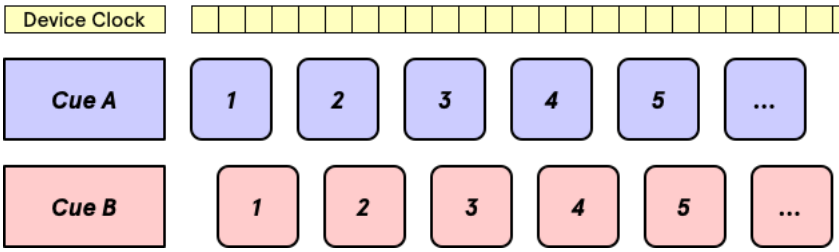
In sync and simultaneous



In this example, Cue A and Cue B are both assigned to the same audio output patch and both started simultaneously, either because they are both contained within a Timeline Group with identical pre-wait times, or because the first cue is set to auto-continue with no post-wait and the second cue, which is started via the auto-follow, has no pre-wait.

The simultaneous start causes the first sample from each cue to be played at the same time, and the clock of the audio device guarantees that samples from each cue will be played at the same cadence. As a result, the cues will start at the same time, and stay in sync for as long as they both run.

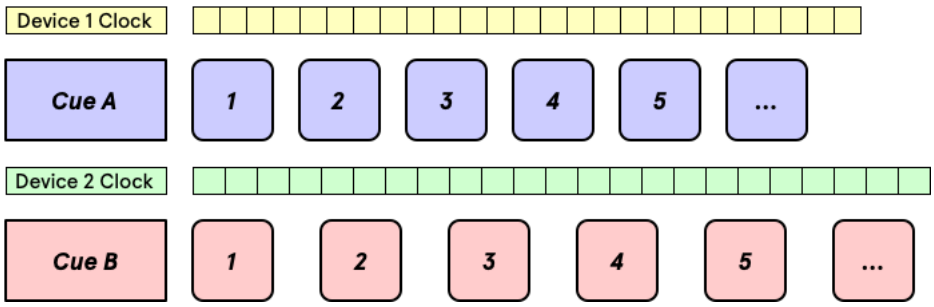
In sync, but not simultaneous



In this example, Cue A and Cue B are both assigned to the same audio output patch but they are not started simultaneously. This could be because one of them has a pre-wait; wait times use the system clock, not the audio device clock, and so wait times and audio playback cannot be perfectly synchronized. It could also be because they are not connected at all, and are started by two separate presses of the GO button, or one is started via the GO button and the other is started by a MIDI trigger or something else like that.

Because the start is not simultaneous, the first sample of each cue is played at a different time, but because both cues are still assigned to the same audio output patch the audio device clock still guarantees that samples from each cue are played at the same cadence. As a result, the two cues will remain in sync for as long as they both run, offset by the exact same amount of time throughout.

Not in sync, but simultaneous



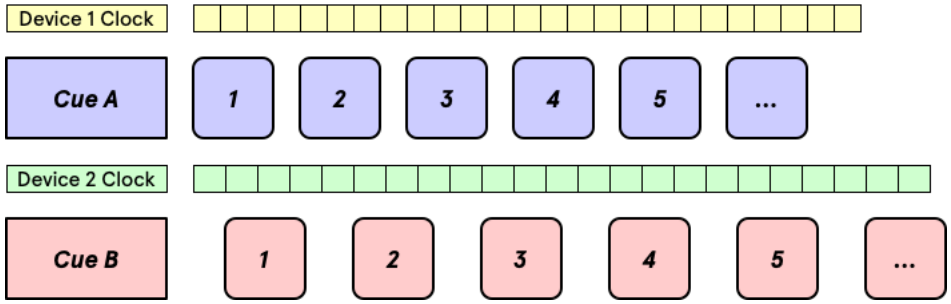
In this example, Cue A and Cue B are assigned to different audio output patch and both started simultaneously.

The start is simultaneous, so the first sample of each cue is played at the same time. Since the two audio output patches use different audio device clocks, though, the time between samples is different for the two cues. As a result, they will not remain in sync and will drift apart over time. The longer

the cues are, the more audible the difference will become.

If sync is needed in this situation, using audio devices that have word clock connections can solve the problem by provided an external mechanism to sync the two clocks, thus converting this scenario into a “in sync and simultaneous” scenario.

Not in sync, not simultaneous



In this example, Cue A and Cue B are assigned to different audio output patch and not started simultaneously.

As a result, the first sample of each cue is played at a different time, and the cadence of each cue is different because they follow different clocks. These two cues have basically nothing to do with each other.

---

1. Most technically, an audio interface needs to be *Core Audio compliant* to work with QLab. There are some audio interfaces that work with Macs in general, but only with specific software. Those devices are in the tiny minority however. If an audio interface appears in the Sound section of System Preferences, it will almost definitely work with QLab.↩
2. This can be changed for Video cues, but don’t worry about that for now.↩
3. In both a philosophical sense and a computer science sense, it’s hard to quantify what “simultaneous” truly means, since measuring time is done in discrete chunks, whereas Time Itself is a fluid continuum, as far as we know, and one which behaves Very Weirdly if you examine it closely. In order to have a reasonable conversation about it without graduate level coursework as a prerequisite, this manual will assume a meaning of “simultaneous” which is confined to human-level perception... if two things happen so close together that a typical organic human person perceives them to be simultaneous, then we will say that they *are* simultaneous.↩

Object Audio in QLab

Object audio is a way of mixing audio that is more flexible and more visual than traditional output channel-based mixing. It takes many forms and goes by many names, and despite being several decades old as a concept, it has only entered the mainstream of live event audio in recent years.

QLab 5.5 adds an object mixer to its array of audio tools. The object mixer exists alongside QLab's channel-based mixer, and you can use either one or both at any time, in any cue.

The briefest possible definition of object audio

The "regular" way of mixing audio, called channel-based mixing, allows you to route one or more audio inputs to one or more audio outputs. Whenever multiple signals are routed to the same output, they are combined so that you hear both of them. To make a sound seem to move in space, you raise the level of that sound in one output while lowering it in another output.

The tool to do this is called a *mixer*, which consists of several level controls, usually knobs or faders. In QLab, most level controls are text boxes which behave a little bit like faders. If a level control in QLab says `-3`, it's sort of like there's an invisible fader right there that's set to `-3` dB. The mixers in QLab are all *matrix mixers*, which just means that every input can be routed to every output at an independent and variable level.

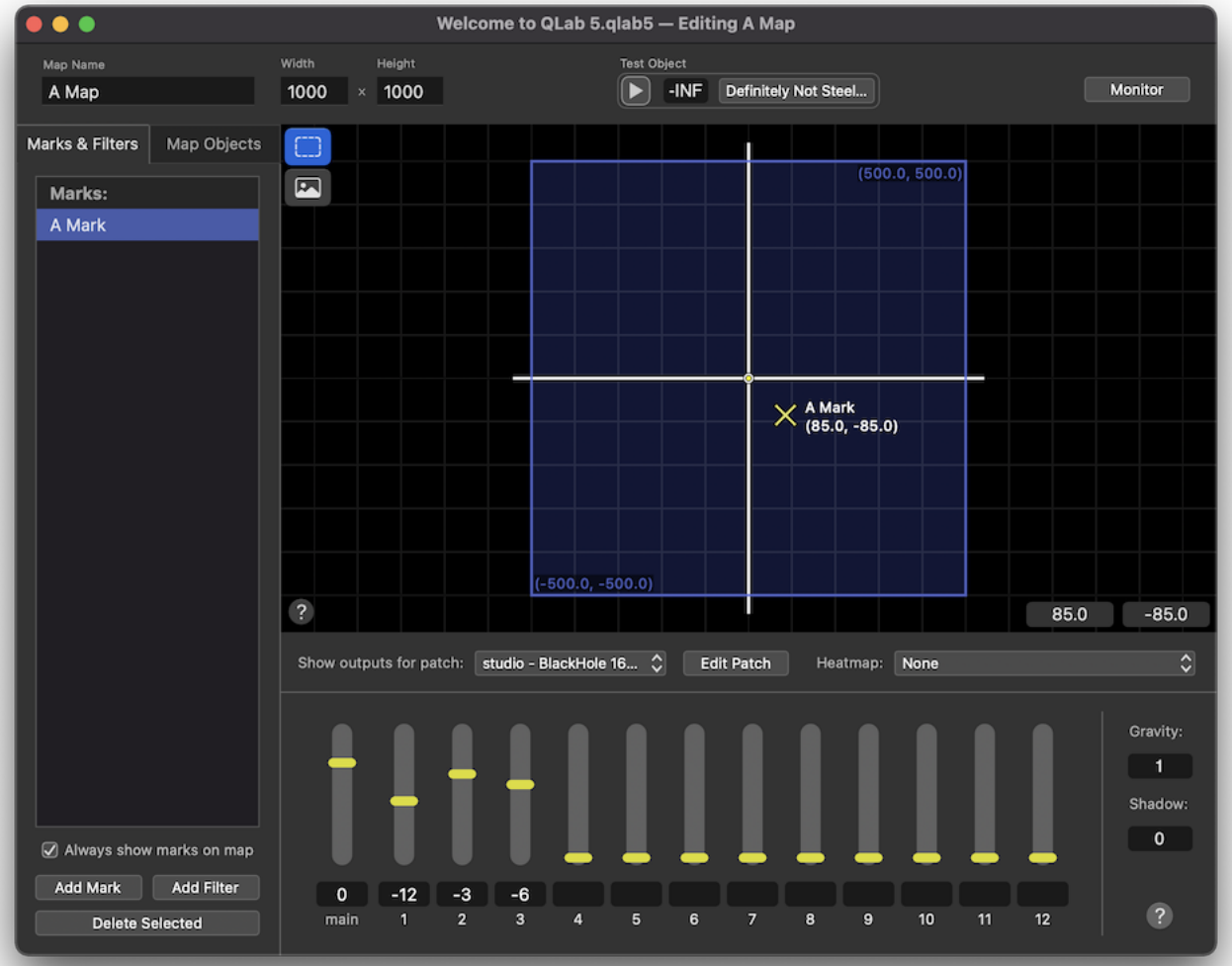
Object audio mixing replaces or supplements the mixer with a different kind of tool, an *object mixer*. The end result is the same; audio signals enter the object mixer and are routed to one or several outputs. The difference is, instead of a set of sliders and knobs, each of which moves along a single axis to control a single level, object mixers present a flexible graphical interface to the person doing the mixing. In this interface, inputs are represented by *objects* which can be moved around within the mixer freely. The position of an object within the object mixer determines how its sound is routed.

From the audience perspective, there is no way to tell what tool or tools were used to mix the audio. The audience just hears sound coming out of speakers. The power of object audio is that this differently designed mixing tool lets you make adjustments to routing very easily that would be difficult, or at least time-consuming, to make using a channel mixer.

Maps, Marks, and Objects

The three essential components of QLab's object audio mixer are **maps**, **marks**, and **objects**.

A map is a virtual space that your objects live within. Maps can be literal representations of the physical layout of your sound system, but they do not have to be. Marks are locations in a map which represent a specific set of cue output levels.



Marks are not outputs; no audio flows through them. Instead, a mark simply says, "when an object is placed *here,* it should sound like *this.*" In this screen shot, if an object is placed directly on top of this mark, then the audio from that object will be routed according to the levels at the bottom of the screen.

Objects can belong to cues or to maps. Map objects are part of the map, and can receive audio from any cue or from several cues at once. Cue objects can only receive audio from the cue that they belong to. Cue objects effectively only exist while the cue that they belong to is playing, paused, or loaded.

Why Use Object Audio?

Imagine fading a sound from left to right in a very simple theatrical sound system. It's not very complicated to do; start with the sound up in the left speaker and not in the right, then turn it down in the left speaker while turning it up in the right speaker.

If you add a center channel to the sound system, you now need two fades: down in the left and up in the center, then down in the center and up in the right.

But what if you want to fade a sound across, say, twelve outputs? Making eleven Fade cues is annoying, but probably manageable. But then, imagine the director asking for that movement to take seven seconds longer. Now you need to divide seven by eleven to see how much time to add to each Fade cue. Once you need to do that two or three times, you may wish to tear your hair out.

Object audio lets you use a single Fade cue to create a path for the object to follow, and QLab automatically calculates how loud the object should be in each output at every moment. The creative process is much simpler, which means you spend less time fiddling with Fade cues and more time hearing your cues.

Audio Cues

Audio cues allow you to play sound files with precise control over timing, levels, and routing. Audio cues must have a file target, which is a sound file on your computer, and an audio output patch, which connects QLab to a sound output destination such as your computers' speakers or headphone jack, an audio interface, or a network-based output such as Dante, AVB, or NDI.

Audio Files

Audio cues may target any file type supported by Core Audio (Apple's audio framework), but we recommend the following types:

- AIFF - the Audio Interchange File Format
- WAV - the Waveform Audio File format
- CAF - Core Audio Format
- AAC - Advanced Audio Coding format
- MP4 - MPEG-4 part 14 container format (which is usually just an AAC audio file wearing a different set of clothes.)
- M4A - MPEG-4 Audio container format (which is also just AAC.)

QLab also supports MP3 files, by their very design they incur a variable and unpredictable delay when decoding and playback begins. This makes exact timing impossible when using MP3 files. For this reason, and also because they usually don't sound very good, we do not recommend using them unless you are certain that their limitations are not relevant to your show.

Audio cues can also target any file type supported by Video cues; the audio portion of the file will be used and the video portion of the file will be ignored.

Without getting too far into the technical weeds, AIFF, WAV, and CAF files will give you the best audio quality and the least technical trouble and those are the formats that we recommend most highly.

Audio File Details

QLab can target multichannel audio files and will recognize the first 24 channels in an audio file[1].

QLab supports audio file targets at any sample rate from 8 kHz up to 192 kHz, and any bit depth from 8-bit up to 32-bit[2].

Through the low-level audio frameworks built into macOS, QLab automatically and seamlessly performs any necessary sample rate and bit depth conversion on the fly to match the requirements of the audio output patch. This means that you do not need to choose a single sample rate and bit depth to work with; workspaces can target audio files of various rates and depths, and you can use audio outputs with various rates and depths.

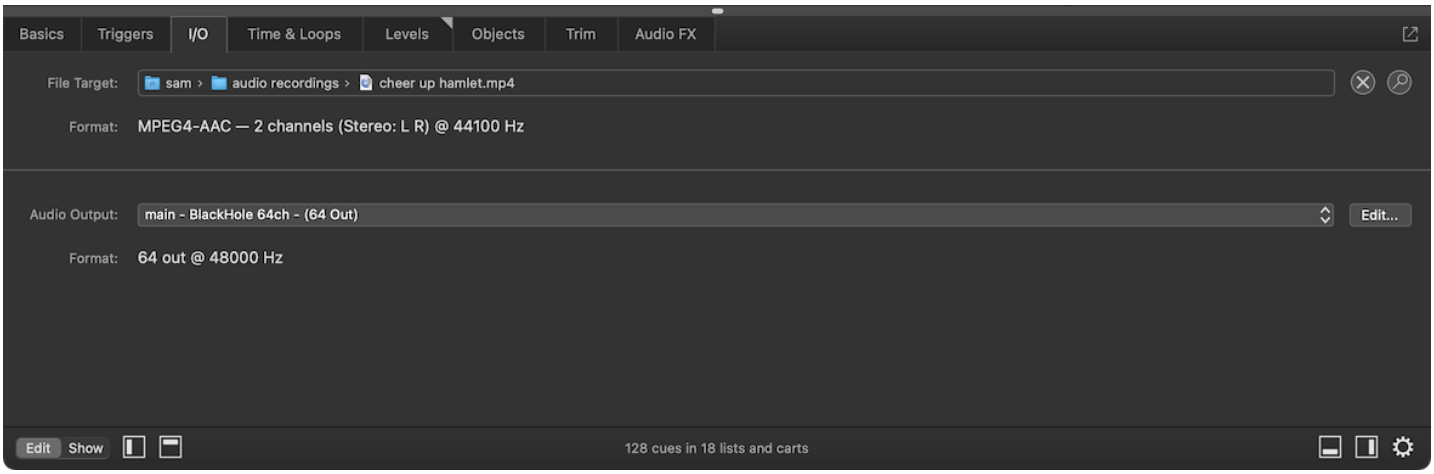
This on-the-fly conversion is a very efficient process and in most situations you will not need to concern yourself with the matter. If you are trying to squeeze every available drop of performance out of your system, you can convert all your audio to the same sample rate and bit depth as your output hardware is using.

The Inspector for Audio Cues

When an Audio cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as the following tabs:

The I/O Tab

The I/O tab lets you assign a target audio file and audio output patch, and view details about them as well.



File Target. If a target audio file is assigned, this field shows the path to that file. You can double-click on the field to select a file target, or you can drag and drop a file in from the Finder.

Clicking the ⊗ button removes the target file from the cue (though it does not delete the file itself, of course.)

Clicking the 🔍 button reveals the target file in the Finder.

These controls are also visible in the Basics tab; you can use either one or both interchangeably.

Format displays some technical details of the target audio file, including the type of file and codec in use, number of channels, and sample rate.

Audio Output. This pop-up menu allows you to select an audio output patch for the cue to use. Clicking on the menu allows you to select one of the audio output patches already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Audio Settings to edit patch list...* to quickly get to Workspace Settings → Audio → Audio Outputs, or choose *New patch with audio device* to quickly generate a new audio output patch and select it for use.

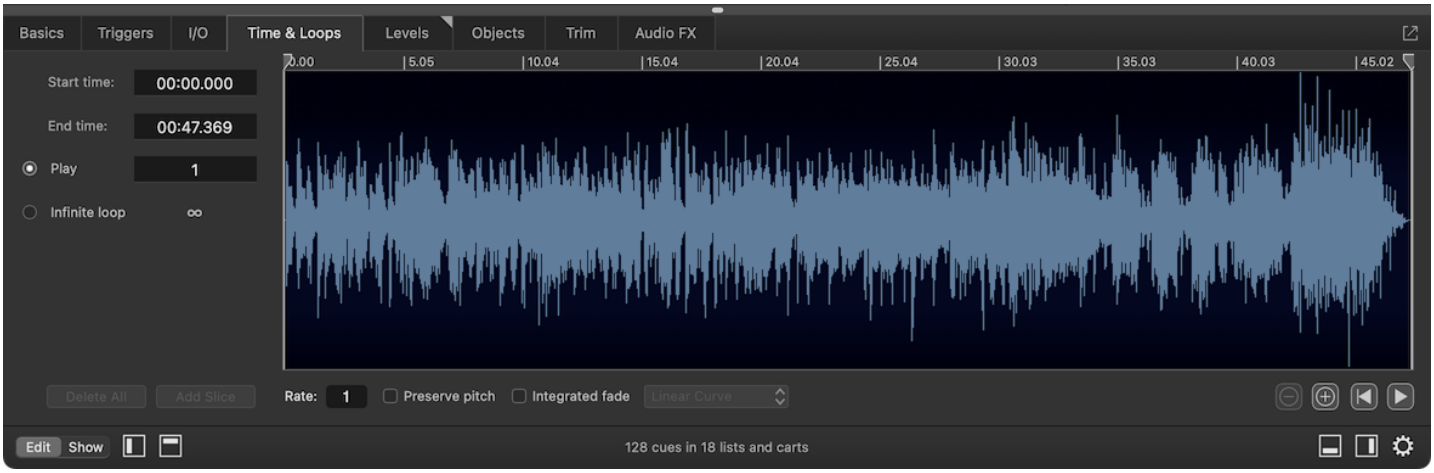


The **Edit...** button opens the audio output patch editor for the currently selected audio output patch, for quick access.

Format displays the number of channels and sample rate of the audio device used by the selected audio output patch. Note that for virtual audio devices such as Existential Audio's BlackHole or the venerable Soundflower, the sample rate may not display accurately at all times, or even at all. Virtual devices are not always able to provide this information; it's not necessarily a sign of a problem.

The Time & Loops Tab

The Time & Loops tab lets you adjust time-based behaviors of the cue and view a waveform representation of the target audio file.



Start Time and End Time

Start time defaults to `00:00.000` and represents the timestamp in the target audio file at which QLab will start playing the file. You can change the start time of the cue in three ways:

1. You can type a new value into the text field.
2. You can drag the start time marker, which is the downwards-pointing grey triangular handle at the top left side of the waveform view.
3. You can click anywhere in the waveform view, or preview the cue and then pause it, then use the keyboard shortcut ⇧**I** ("I for in") to set the start time to the current playback time.

End time defaults to the end of the target audio file, and so displays the timestamp of the final sample of audio in the target file. You can change the end time of the cue in three ways:

1. You can type a new value into the text field.
2. You can drag the end time marker, which is the downwards-pointing grey triangular handle at the top right side of the waveform view.
3. You can click anywhere in the waveform view, or preview the cue and then pause it, then use the keyboard shortcut ⇧**O** ("O for out") to set the end time to the current playback time.

If you click and drag within the waveform view to select a section of the file, you can use both ⇧**I** and ⇧**O** one right after the other to quickly crop the cue to the selected region.

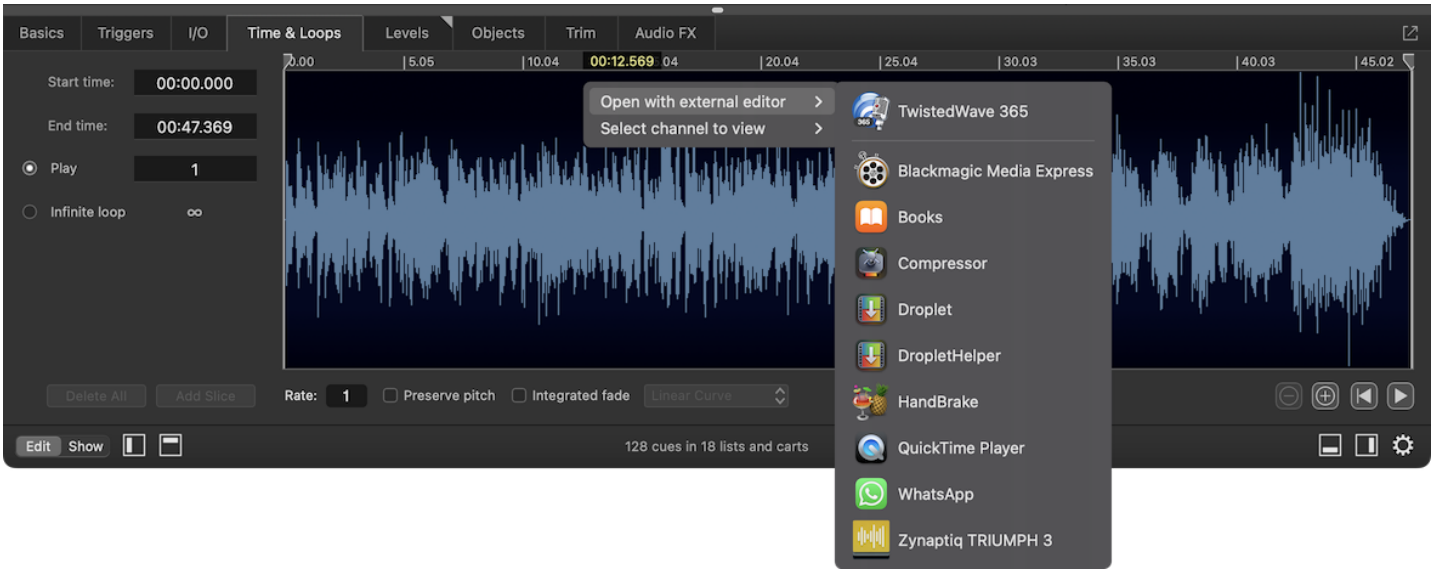


Play count and infinite loops

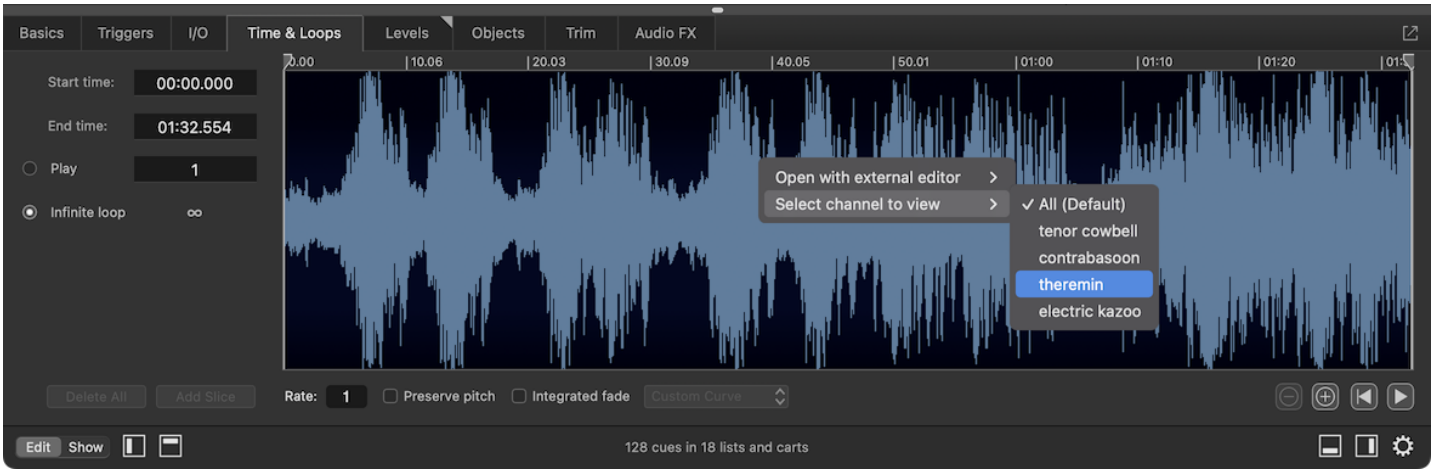
Play count defaults to `1` and is the number of times that the target audio file will be played when the cue is run. You can enter any whole number in the text field to loop the target audio file that number of times, or select *Infinite loop* below to loop the target audio file indefinitely.

Waveform contextual menu

Right-clicking (or two-finger-clicking, or clicking while holding down the option [⌥] key) within the waveform view brings up a contextual menu which lets you open the target audio file in an external audio editor or select which channel of audio you wish to view.



The “open with external editor” submenu lists all programs on your Mac which self-identify as an editor for audio files.



The “select channel to view” submenu allows you to choose between viewing a sum of all channels, which is the default, or any single channel. The channel names in this menu reflect names set in the Levels tab.

Slices

The *play count* and *infinite loop* options allow you to loop the entire target audio file, but QLab allows you to loop specific sections of the file as well. To this, you create *slices* within the cue, and set each slice to loop as needed.

To create a slice, click in the waveform view and then click the **Add Slice** button to the left of the waveform, or use the keyboard shortcut **M**. A green marker, called a slice marker, will appear in the waveform at the spot where you clicked. A section of the cue between two slice markers, or between a marker and the start or end of the cue, is a slice.



The green numbers which appear along the bottom of the waveform view are play counts for each slice. The play count will default to 1, but you can easily edit the count of an individual slice by double-clicking the number at the bottom of the slice and entering a value. To loop a slice infinitely, type any letter. To seamlessly skip a slice, type `0`. The slice will receive a darkened tint, and QLab will skip over that slice while playing the cue. At least one slice of a cue must have a play count greater than zero.

You can click the green handle at the top of a slice marker and slide it left and right along the waveform to adjust its position, or click on the handle and then enter a value manually in the text field that appears on the left side of the waveform view. Slice markers cannot be closer together than `.05` seconds.

If you click and drag within the waveform to select a section of the file, the **Add Slice** button (and the **M** shortcut) will create slice markers on both sides of the selected section.

If the target audio file is an AIFF or WAV file which contains markers, those markers will automatically appear in QLab as slice markers. Markers closer together than `.05` seconds will be discarded by QLab, though the markers in your file will remain untouched.

To delete a selected slice, select it and hit ⌫ (delete) on the keyboard, or drag its slice handle upwards out of the waveform view.

> ProTools users will find that markers from ProTools projects are not included in bounced files, which makes QLab's automatic marker importing somewhat less useful. Fortunately, there is a workaround as long as you have a two-track editor that allows importing markers (such as TwistedWave):
>
> 1. Bounce or export your audio as usual.
> 2. In ProTools, choose *Export...* from the **File** menu and export your session info as text.
> 3. Open your audio file in your two-track editor and import the text file you created from ProTools. Et voila!

The **Delete All** button deletes all slices from the cue. Markers in the target audio file are not deleted.

Vamping and devamping

Looping and slicing cues starts to get really interesting when used in combination with the Devamp cue, which allows you to dynamically exit loops while cues are playing. You can learn more about this from the section on Devamp cues in this manual.

Rate and pitch

You can adjust the playback rate of the Audio cue by typing a value in the *Rate* text field to the right of the waveform, or by clicking in that field and dragging up or down. By default, adjusting the rate will adjust pitch as well in a manner similar to changing the playback speed of analogue tape. If you check the *Preserve pitch* box, however, the pitch will not be changed along with the rate.

Rate 1 = 100% (normal speed) - no pitch shift Rate .5 = 50% (half speed) - pitched down one octave Rate 2 = 200% (double speed) - pitched up one octave The minimum rate is `0.03`, and the maximum rate is `33`.

It is worth noting that checking the *Preserve pitch* box requires slightly more processor power than leaving the box unchecked.

The Integrated fade envelope

The integrated fade envelope allows you to adjust the overall volume of the cue graphically over its duration. This can be useful for evening out dramatic volume differences, or modulating volume for creative effect.

To use the integrated fade envelope, check the box underneath the waveform view labeled *Integrated fade*. A yellow line will appear along the top of the waveform view, and you can click and drag along the line to create control points.

The pop-up menu next to the *Integrated fade* checkbox allows you choose between *Custom Curve*, which makes the integrated fade envelope curve smoothly between and around control points, or *Linear Curve*, which creates sharp angles precisely at each control point.

The pop-up menu also gives you the option to *Lock fade to start/end*, which automatically stretches the integrated fade curve to fit within the start and end time of the cue. With this option not set, the integrated fade curve will remain locked to the natural start and end time of the target audio file, regardless of the start and end times set in the cue.

Note that if you edit the envelope while the cue is playing, you won't hear your changes until you stop and restart the cue.

You can edit and navigate between control points using the arrow keys:

Command	Key
Select previous control point	←
Select next control point	→

Command	Key
Move selected control point up	⌥↑
Move selected control point down	⌥↓
Move selected control point left	⌥←
Move selected control point right	⌥→
Move selected control point up a small amount	⇧⌥↑
Move selected control point down a small amount	⇧⌥↓
Move selected control point left a small amount	⇧⌥←
Move selected control point right a small amount	⇧⌥→

Preview, Reset, and Zoom

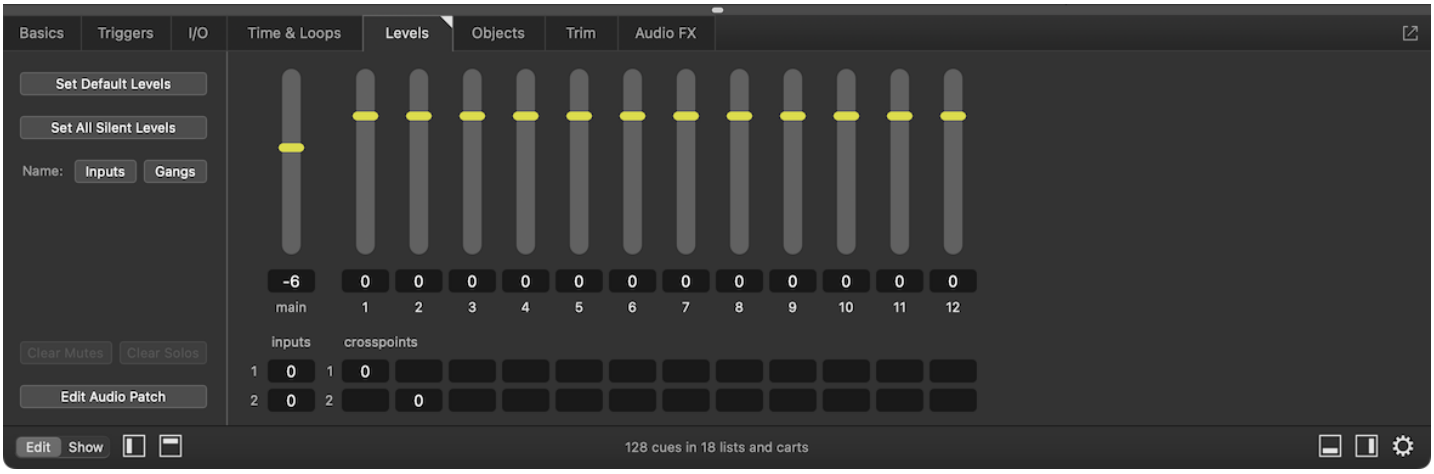
You can use the ⊖ and ⊕ buttons to zoom in and out on the waveform. If you're using a trackpad, you can zoom in and out by placing the cursor over the waveform view and pinching in and out, or by holding down the ⌥ (option) key and scrolling vertically. If you're using a mouse or trackball with a scroll wheel, you can zoom in and out by placing your mouse cursor over the waveform view and scrolling.

The ⏮ button resets the cue, stopping it and resetting all temporary properties such as conditional cue colors.

The ▶ button starts playing the cue without advancing the playhead or triggering any auto-follow or auto-continue. The default keyboard shortcut for previewing a cue is **V**. While the cue is playing, this button changes to a ⏸ pause button.

The Levels Tab

The Levels Tab lets you adjust volume levels for the cue.

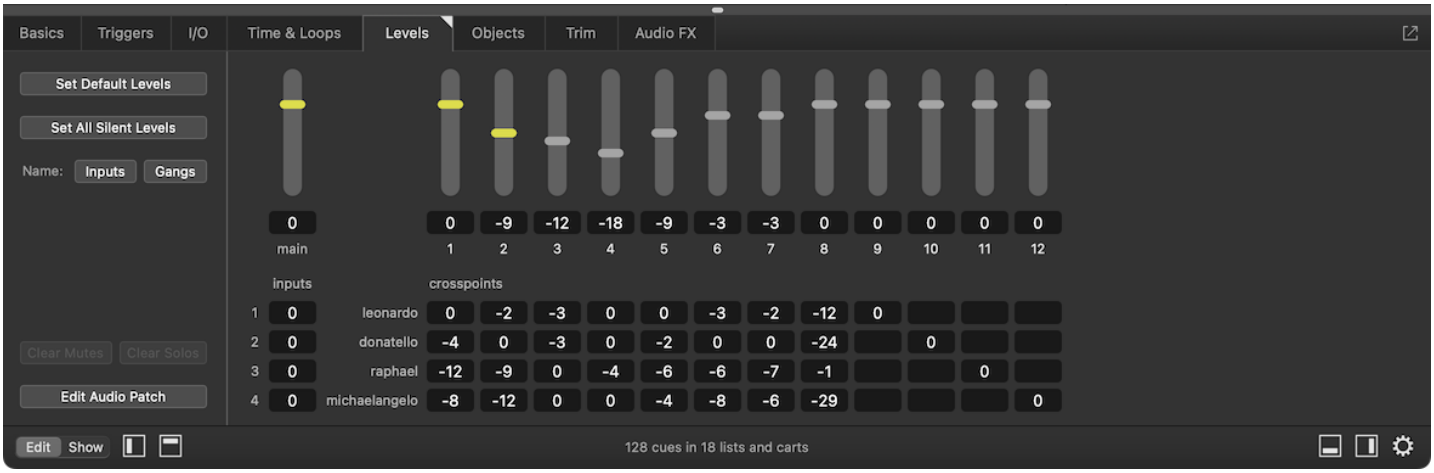


Controls

Set Default Levels. This button sets all levels of the cue to match the levels in the Audio cue template, which can be found and adjusted to suit your needs in Workspace Settings → Templates → Cue Templates.

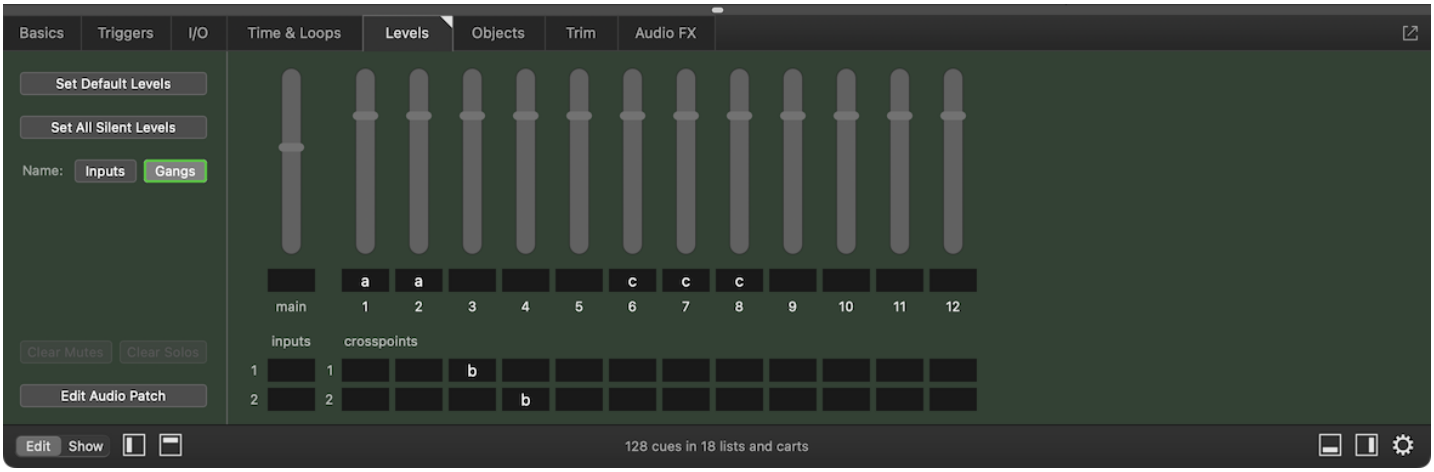
Set all silent. This button does exactly what it seems: it sets all levels for the current Audio cue to -INF.

Inputs. Clicking this button allows you to assign names to each input (row) in the cue.



Input names are unique for each cue, so they can reflect the specific details of that cue. Input names are visible in both the Levels tab and the Objects tab.

Gangs. When you click this button, all audio levels will be hidden and the mixer will switch to gang assignment mode. In this mode, you can type anything (for example, a single letter) into any level field. All fields which receive the same text will become part of the same gang, or level group.



When you click *Assign Gangs* again, the mixer will switch back to its regular view, and ganged fields will be shown with matching colored backgrounds so you can tell at a glance which fields are ganged. Then, you can simply click and drag one control to adjust every level in that gang by the same amount.



If you gang levels together while they're at different starting points and start moving them up or down, one level might reach its maximum or minimum level before another. Then and only then will the levels be adjusted disproportionately with one movement. Once the levels catch up to each other at their maximum or minimum, they will all move together.

Edit Audio Patch. This button opens the Audio Output Patch Editor for the audio output patch in use by the cue.

The Cue Levels Matrix Mixer

The cue levels matrix mixer is a grid made of rows and columns, like a spreadsheet. The main output level for the cue is in the top left, the cue outputs are the column headers, and channels in the audio file are the rows. The number of rows in the mixer is dictated by the number of channels in the target audio file.

Cue outputs which are assigned to device outputs have a yellow fader handle. Cue outputs which are not assigned to device outputs, and which therefore cannot be heard, have a grey fader handle.

Crosspoints are the level controls for routing a given input (row) to a given output (column).

Levels can be typed into each control, or you can click in a field and drag up or down. QLab won't allow you to drag above `0 dB` as a safety measure, but you can type a level above `0` manually, limited by the maximum level set in Workspace Settings → Audio. Since the majority of levels set are below `0 db`, QLab will interpret a number entered without a + or - sign as a negative number.

A level control showing no value can be assumed to be set at -INF (silent).

Adjustments made in the Levels tab take effect live and in realtime, so you can start the cue and then adjust levels by ear. Please be careful at the beginning of a project, since a very small movement of the mouse can cause a very large adjustment in level.

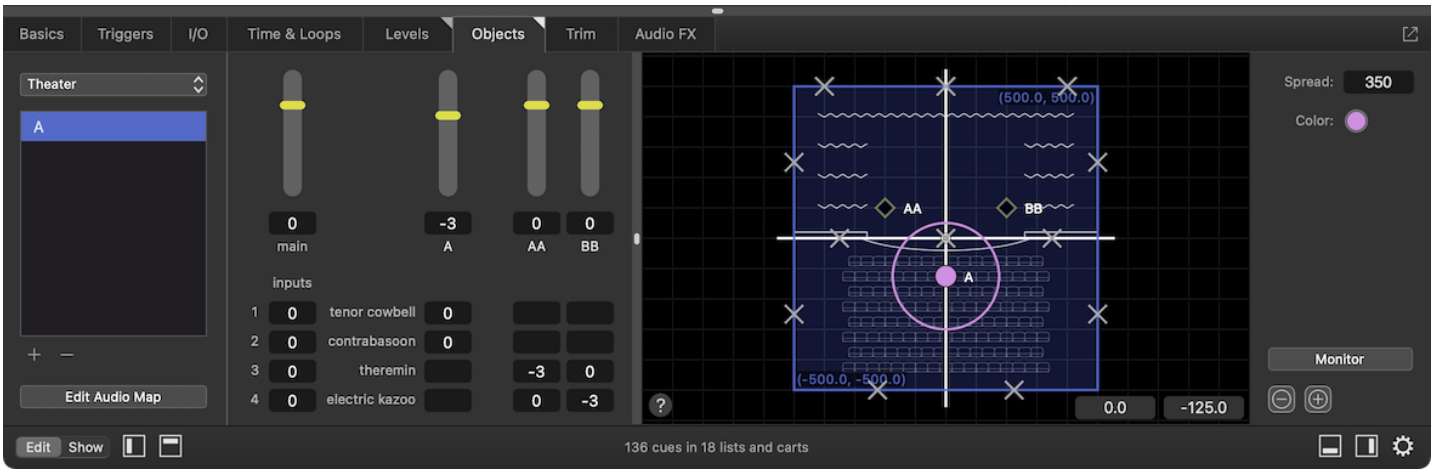
The main level control, in the top left corner of the mixer, is present in both the Levels tab and the Objects tab. Adjusting it in either tab will adjust it in both; it is the overall level control for all sound produced by the cue.

When at least one input channel is routed to at least one cue output at any level above -INF, a grey dog-ear indicator appears in the tab label. Essentially, the question this indicator is meant to answer is, "is this cue making sound because of the routing in this tab?" This indicator does not consider the main level of the cue, since fading in a cue from silence is so common, and in that case the question you really want to know the answer to is, "will the cue make sound because of this tab after I fade it in?"

Other than the main level control and input names, nothing that happens in the Levels tab has any effect on the Objects tab. The two tabs coexist, and you can use either one or both at the same time to route audio from the cue to your sound system.

The Objects Tab

The Objects tab lets you send audio from the cue to audio objects, both cue objects and map objects.



On the left side of the tab is a list of the cue objects belonging to the cue. By default, newly created cue objects are named with single upper-case letters, but you can always rename them to suit your needs.

The [+] button creates a new cue object. The [-] button deletes the selected cue objects. The number of cue objects that a cue can have is limited only by your Mac's processing capability.

The **Edit Audio Map** button opens the audio map editor for the cue's audio map.

The Objects Matrix Mixer

The objects matrix mixer works side-by-side with the cue levels mixer. You can route audio from the cue in both tabs at the same time, or you can use just one or just the other. In the objects matrix mixer, each column represents an object that is available to the cue. Cue objects are listed first, then map objects are listed. Rows in the matrix are channels in the audio file, just like in the cue levels matrix mixer.

Routing a channel (row) to an object (column) allows the audio in the channel to be output according to the position of the object on the map. Since this is a matrix mixer like any other, any input can go to any mixture of objects, and any object can receive any mixture of inputs.

The main level control, in the top left corner of the mixer, is present in both the Objects tab and the Levels tab. Adjusting it in either tab will adjust it in both; it is the overall level control for all sound produced by the cue.

It's important to remember that map objects are shared across all cues that use the same audio map, so any signal that you route to a map object may be mixed with signals coming from other cues.

All the rules and behaviors of the cue levels matrix apply to the objects matrix mixer.

Input names are shared across both mixers, but gangs and levels (other than the main level) are not shared.

When at least one input channel is routed to at least one cue object or map object at any level above -INF, a grey dog-ear indicator appears in the tab label. Essentially, the question this indicator is meant to answer is, "does this cue make sound because of the routing in this tab?" This indicator does not consider the main level of the cue, since fading in a cue from silence is so common, and in that case the question you really want to know the answer to is, "*will* this cue make sound because of this tab *after* I fade it in?"

The Map Canvas

The map canvas lets you position cue objects within the map. Map objects are visible here as well, but cannot be moved because they do not "belong" to the cue. To move the home position of a map object, visit the Audio Map Editor. To move a map object dynamically during your show, use Fade cues targeting the map that contains the object you wish to move.

The selected object or objects can be moved by clicking and dragging, or by using the text boxes in the lower right corner of the canvas.

The ? **help button** displays a list of the keyboard shortcuts and modifiers for working in the map canvas.

Other Controls

To the right of the canvas are two controls which let you set the **spread** and **color** of the selected cue object or objects, a **Monitor** button for opening the monitor window window for the current map, and buttons for zooming in and out on the canvas. You can also zoom in and out on the canvas by using standard macOS pinch and spread gestures, and by using the scroll wheel on a mouse or trackball while holding down the ⌥ (option) key.

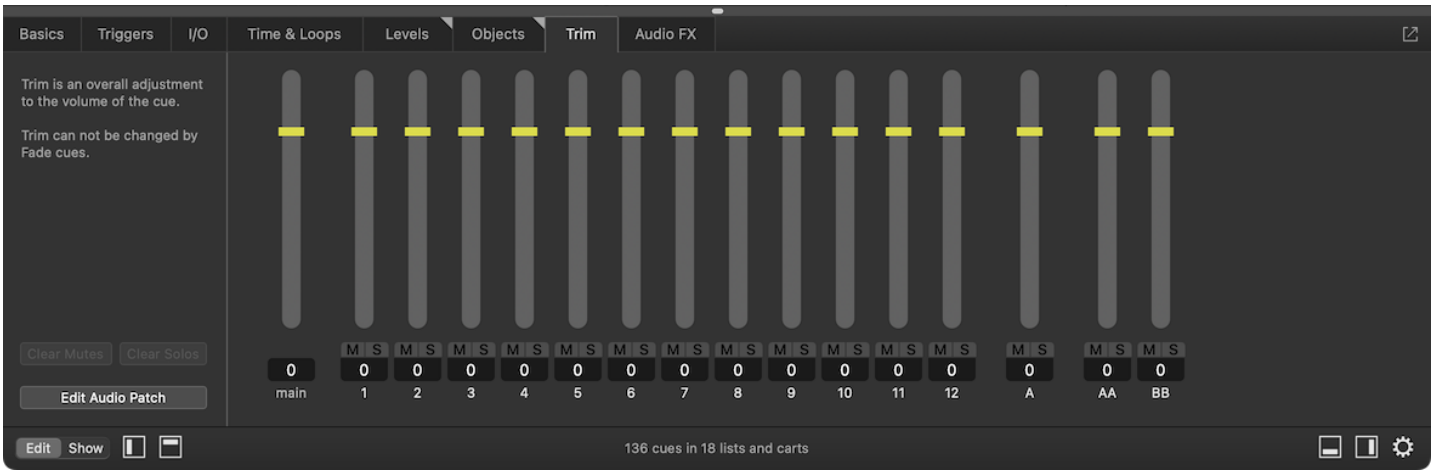
Spread is the apparent size of an object. Objects effectively occupy every point within their spread, so their level in each cue output equals the loudest point for that output within the object's spread. Increasing the spread of an object while sound is playing can raise the overall level of the object as the spread encompasses more marks, but the level of the object to *each* cue output will never be louder than the loudest level of that cue output in any one mark.

Color can be used to differentiate objects for logistical or aesthetic purposes. The list of available colors is the same as the list of cue colors.

Other than the main level control, nothing that happens in the Objects tab has any effect on the Levels tab. The two tabs coexist, and you can use either one or both at the same time to route audio from the cue to your sound system.

The Trim Tab

The Trim tab provides a set of "post-fader" overall level adjustment controls for the main output, cue outputs, and objects in a cue.



These controls are not adjustable using Fade cues, MIDI, OSC, or any other automation; their purpose is specifically to exist outside the realm of automation and other sources of control so that they are guaranteed to stay the way you set them.

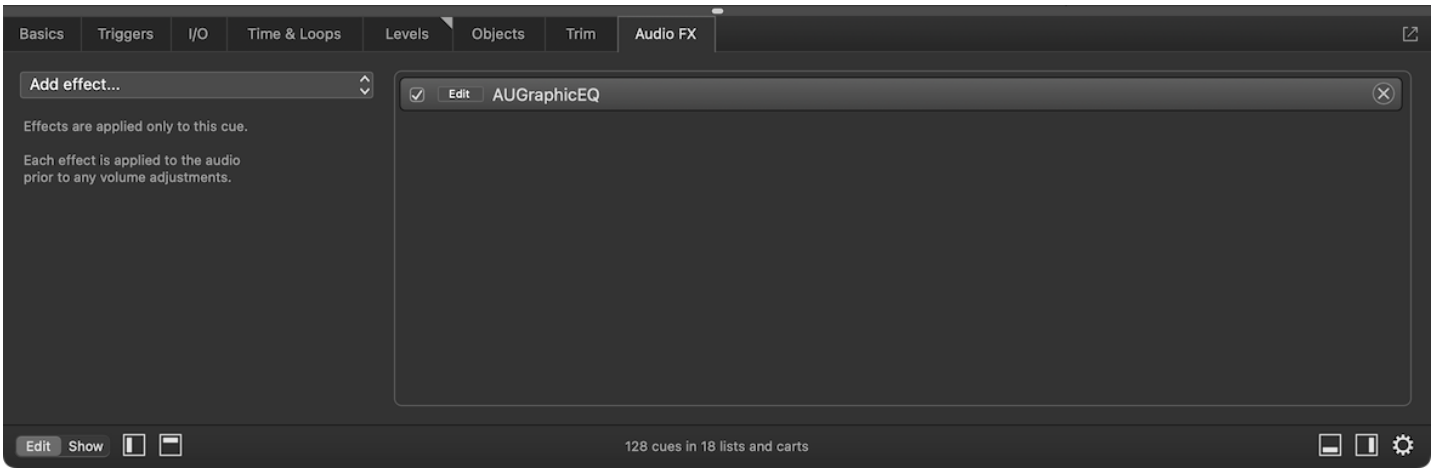
If you make a cue sequence with a bunch of fades and lots of intricate level movement, and then afterwards you decide that one output or object is too loud throughout the whole sequence, you can adjust the trim in the audio cue instead of adjusting that output in each and every fade cue; just one adjustment and you're done.

If you build a cue sequence with a set of audio files and then need to replace those audio files with new versions, but accidentally normalize the new versions differently so the new versions are all `3 dB` hotter than the old versions, you can correct for that using the Trim tab.

The controls for map objects only effect the level of audio coming from *this* cue into those map objects. Audio that is going to map objects from other sources will not be changed by the controls in the Trim tab.

The Audio FX Tab

With an Audio license installed, QLab can use AudioUnit effects installed on your Mac to process sound dynamically in realtime. For an AudioUnit to work in QLab, it must be compatible with the version of macOS that you're using, must be defined as an effect (i.e. MIDI synthesizers won't be available in QLab because they are generators, not effects), and must report a "tail time" (for example, a reverb's decay time.)



Additionally, the number of channels in the target audio file must match the number of channels supported by the AudioUnit. Different AudioUnits behave differently if there is a mismatch in channel count. Some will pass the audio straight through as though no effect is in use, some won't pass any audio at all, and some may have other unpredictable consequences. A good example of this behavior is exhibited by Apple's AUMatrixReverb, which requires a two-channel (stereo) source. If you use AUMatrixReverb on a mono audio track, the audio will pass through the AudioUnit unchanged.

To use an audio effect, select it from the pop-up menu labeled *Add Audio Effect*... When you select an effect, it will appear in the list to the right of the menu, and an AudioUnit editor window will open automatically. You may close the window and access it again easily anytime by clicking the **Edit** button next to the name of the effect in the list. The editor window looks different for each AudioUnit because each effect requires different controls. QLab uses the built-in interface created by the designer of the AudioUnit, which means the look and feel (as well as quality and usability) of AudioUnits can vary widely.

You can bypass an audio effect by unchecking the box to the left side of the effect, and you can remove an audio effect from the cue by clicking the ⊗ button to the right side. If the AudioUnit editor window is open, you can also turn the effect on or off by using the checkbox labeled *Enabled* in the top right corner of the effect window.

Audio effects will be applied to the cue in the order in which they appear in the effects list. To change this order, simply click on an effect and drag it up or down within the list.

You can also insert AudioUnits on cue outputs and device outputs. You can find out more about using AudioUnits on outputs in the Audio Output Patch Editor section of this manual.

Broken Cues and Warnings

Audio cues can become broken or for the following reasons:

No audio file selected

The cue has no target, and Audio cues require an audio file as a target. Select an audio file as the target of this cue to clear this warning.

Missing audio file

The cue had a target file assigned, but that file is missing. Perhaps the file was on a removable drive or network drive which is not currently connected. Re-locate the target file, or assign a new target file to clear this warning.

File target in Trash

The cue's target file is in the Trash, and files in the Trash cannot be used. Either move the target file out of the Trash, or assign a new target file to clear this warning.

File target lacks read permissions

This is a bit of a bizarre error that is technically possible, but very unusual. If the target file lacks read permissions, it cannot be used. Since it cannot be used, though, it probably could not have been selected as a target in the first place. Despite this seeming paradox, it still is technically possible that this could happen, and so QLab tries to handle it. Fix the target file's permissions, or assign a new target file to clear this warning.

No audio output patch

The cue has no audio output patch assigned. Assign an audio output patch to clear this warning.

Issue with audio output patch

The cue has an audio output patch assigned, but something is wrong with it. The audio output patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the audio output patch or select a new audio output patch to clear this warning.

Missing cue audio effect

The cue has an audio effect assigned, but that AudioUnit is not installed. Either remove the audio effect from the cue or install the missing AudioUnit to clear this warning.

Invalid slice play counts

The cue is sliced, and all slices have a play count of `0`. This means that no part of the target file will be played, which means the cue effectively does nothing. Either set at least one slice to a play count greater than zero, or delete this cue that does nothing anyway to clear this warning.

License required

An audio license is required to use more than the first two cue outputs, audio effects, or Timecode triggers. Install an audio license or adjust the cue to avoid using licensed features to clear this warning.

License required (in re. object audio)

An audio license is required to use object audio, but cues which use objects do not break if no license is present. Instead, those cues display a non-breaking warning ( ) to indicate that object audio will not work, although the rest of the cue will.

---

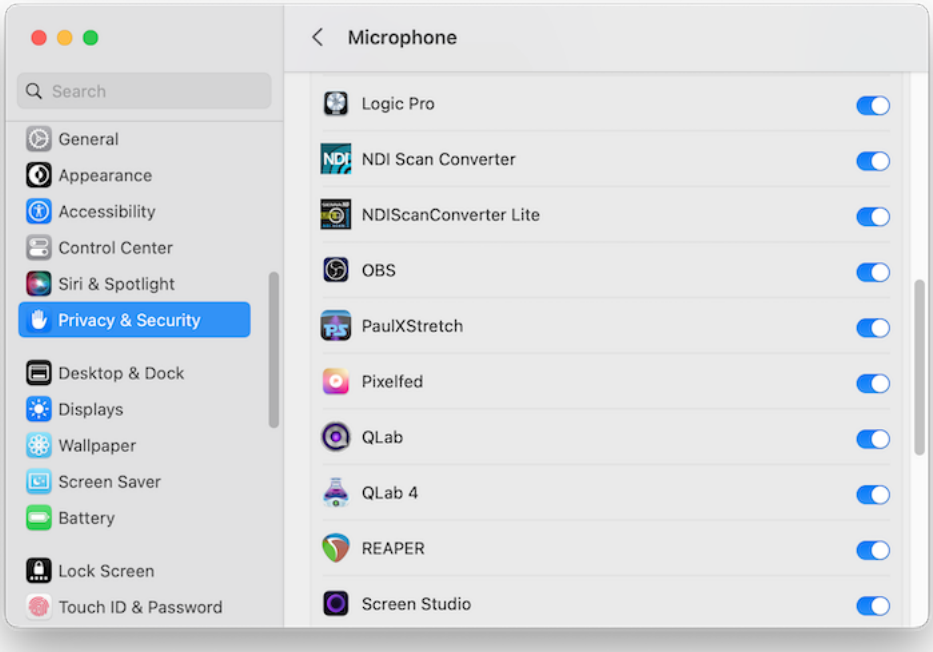
1. Requires an Audio license. Without an Audio license installed, QLab recognizes the first two channels in an audio file.
2. While reasonable people may differ on the audible effects of sample rates above 48 kHz and bit depths above 24-bit, what is undebatable is this: 96 kHz audio requires double the processing power of 48 kHz audio. We recommend avoiding sample rates above 48 kHz unless using them is required by other equipment in your system.

Mic Cues

Mic cues allow you to play live audio through QLab, using the same cueing, routing, and audio effects that are available to Audio cues. The source of this live audio can be any audio input available to your Mac, including the built-in microphone, the mic/line input jack, inputs on a connected audio interface, or incoming Dante or AVB channels. Mic cues run indefinitely (until they are stopped, of course) unless you give them a finite duration.

macOS Security

macOS requires you to proactively grant permission to each program that wants to use any microphone or audio input device in order to limit sneakiness and make it harder for programs to spy on you. You will need to grant QLab permission for microphone access in order to use Mic cues. If you're not automatically prompted to do this the first time you try, or if you need to change QLab's permission, you can do that by visiting *System Settings* → *Privacy & Security* → *Microphone*, finding QLab in the list, and flipping the switch on or off.

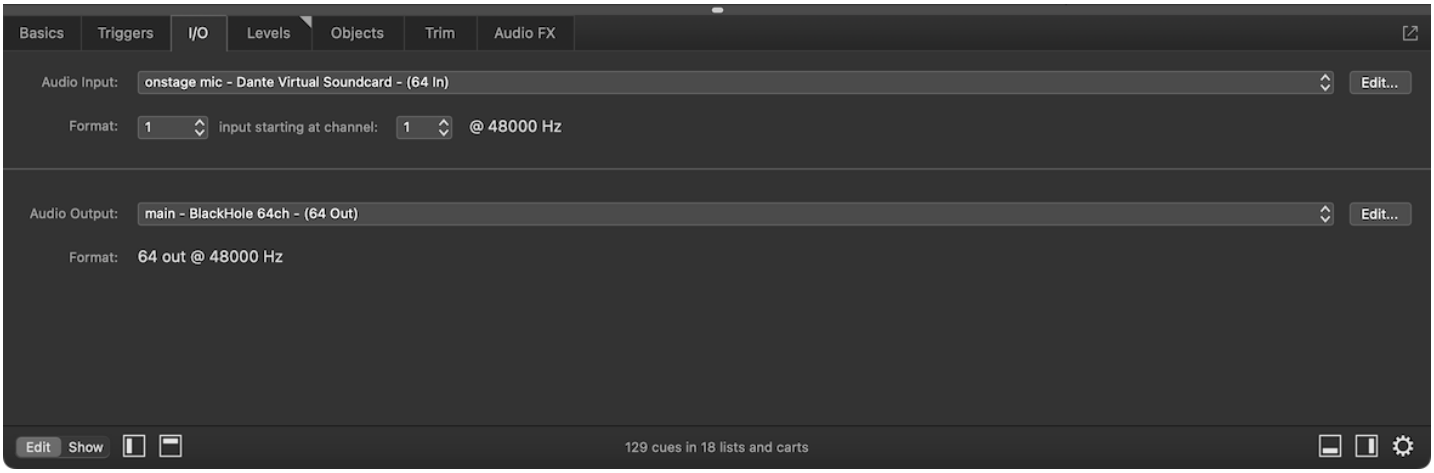


The Inspector for Mic Cues

When a Mic cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as the following tabs:

The I/O Tab

The I/O tab lets you configure the cue's audio source and destination, as well as the number of channels used by the cue.



Audio Input. This pop-up menu allows you to select an audio input patch for the cue to use as the source of audio in the cue. Clicking on the menu allows you to select one of the audio input patches already configured in the workspace, or *(unpatched)* if you want to deliberately prevent the cue from passing any audio when it's started. You can also choose *Open Audio Settings to edit patch list...* to quickly get to Workspace Settings → Audio → Audio Inputs, or choose *New patch with audio device* to quickly generate a new audio input patch and select it for use.

The **Edit...** button also opens Workspace Settings → Audio → Audio Inputs.

Format. This pair of pop-up menus allow you to configure the input channels of the cue. The first menu sets the number of channels, which can be anywhere from 1 to the number of input channels available in the audio input patch, up to a maximum of 24. In the screen shot above, you can see that the audio input patch uses a device with two inputs, so for this cue, the number of input channels could be 1 or 2. The second menu sets the number of the first input that the cue should use, which can be anywhere from 1 to the number of input channels available.

Audio Output. This pop-up menu allows you to select an audio output patch for the cue to use. Clicking on the menu allows you to select one of the audio output patches already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Audio Settings to edit patch list...* to quickly get to Workspace Settings → Audio → Audio Outputs, or choose *New patch with audio device* to quickly generate a new audio patch and select it for use.

The **Edit...** button opens the audio output patch editor for the currently selected audio output patch, for quick access.

Format displays the number of channels and sample rate of the audio device used by the selected audio output patch. Note that for virtual audio devices such as Existential Audio's BlackHole or the venerable Soundflower, the sample rate may not display accurately at all times, or even at all. Virtual devices are not always able to provide this information; it's not necessarily a sign of a problem.

Using different devices for input and output

In QLab 5, Mic cues (and Camera cues) can use different audio devices for incoming audio and outgoing audio. In order to compensate for differences in clocking and sample rates between the input and output device, QLab applies a synchronization algorithm which guarantees that the maximum drift between audio input and audio output stays within a very small, predicable range. The range is defined by this simple equation:

> Maximum drift ≤ (buffer size of input audio device) + (buffer size of output audio device) + (very small safety margin)[1]

A two-hour test[2] which recorded audio passing through QLab using this algorithm measured the total range of drift between the input and output between `0` and `3` milliseconds. That's not nothing, to be sure, but it's comfortably below the threshold of human perception for most use cases.

The Levels Tab

The Levels tab behaves the same as the Levels tab for Audio cues.

The Objects Tab

The Objects tab behaves the same as the Objects tab for Audio cues.

The Trim Tab

The Trim tab behaves the same as the Trim tab for Audio cues.

The Audio FX Tab

The Audio FX tab behaves the same as the Audio FX tab for Audio cues.

Broken Mic Cues

Mic cues can become broken for the following reasons:

No microphone access

QLab has not been given permission to access audio inputs. To clear this warning, open System Preferences → Security & Privacy, then click on the Privacy tab and select *Microphone* from the list on the left. Then, find QLab in the list on the right and check the box next to it. This step typically does not need to be repeated, but may need to be if you completely uninstall and then reinstall QLab or perform a major macOS version update on your Mac.

Audio input patch does not have enough input channels

The cue has been configured to use more channels than the input device has available. Either select an audio input patch which uses a device with enough input channels, or reduce the number of channels used in the cue to clear this warning.

Missing cue audio effect

The cue has an audio effect assigned, but that AudioUnit is not installed. Either remove the audio effect from the cue or install the missing AudioUnit to clear this warning.

No audio input patch

The cue has no audio input patch assigned. Assign an audio input patch to clear this warning.

Issue with audio input patch

The cue has an audio input patch assigned, but something is wrong with it. The audio input patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the audio input patch or select a new audio input patch to clear this warning.

No audio output patch

The cue has no audio output patch assigned. Assign an audio output patch to clear this warning.

Issue with audio output patch

The cue has an audio output patch assigned, but something is wrong with it. The audio output patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the audio output patch or select a new audio output patch to clear this warning.

License required

An audio license is required to use Mic cues. Install an audio license or remove the cue to clear this warning.

---

1. The "very small safety margin" is determined by CoreAudio, the low-level macOS framework that QLab uses for everything audio-related, and is not a specific value that QLab can know. It is larger for devices that use higher sample rates and bit depths, and smaller for lower sample rates and bit depths, and in any case it truly is quite small; somewhere in the sub-millisecond range.↩
2. In this test, the output device was set to 24-bit depth at a 48 kHz sample rate, and the input device was set to 24-bit depth at a 44.1 kHz sample rate. The devices were each configured to use their own internal clock, and were not configured as an aggregate audio device.↩

The Audio Output Patch Editor

The Audio Output Patch Editor, which is available on Macs with an Audio (or Bundle) license installed, allows you to customize the configuration, labeling, and behavior of an audio output patch. This allows you to create complex routing, matrixing, and output processing entirely within QLab.

To open the Audio Output Patch Editor, visit *Workspace Settings* → *Audio* → *Audio Outputs*, then click the **Edit** button next to the audio output patch that you wish to edit. Alternately, you can select to any cue which uses the audio output patch, navigate to the I/O tab of the inspector, and click the **Edit...** button on the right side next to the audio output patch menu, or navigate to the Levels tab of the inspector and click the **Edit Audio Patch** button.

At the top of the Audio Output Patch Editor are three controls:

- The **Patch Name** field lets you edit the name of the patch.
- The **Cue Outputs** field lets you adjust the number of cue outputs in the patch.
- The **Audio Device** pop-up menu lets you select the audio device used by the patch.

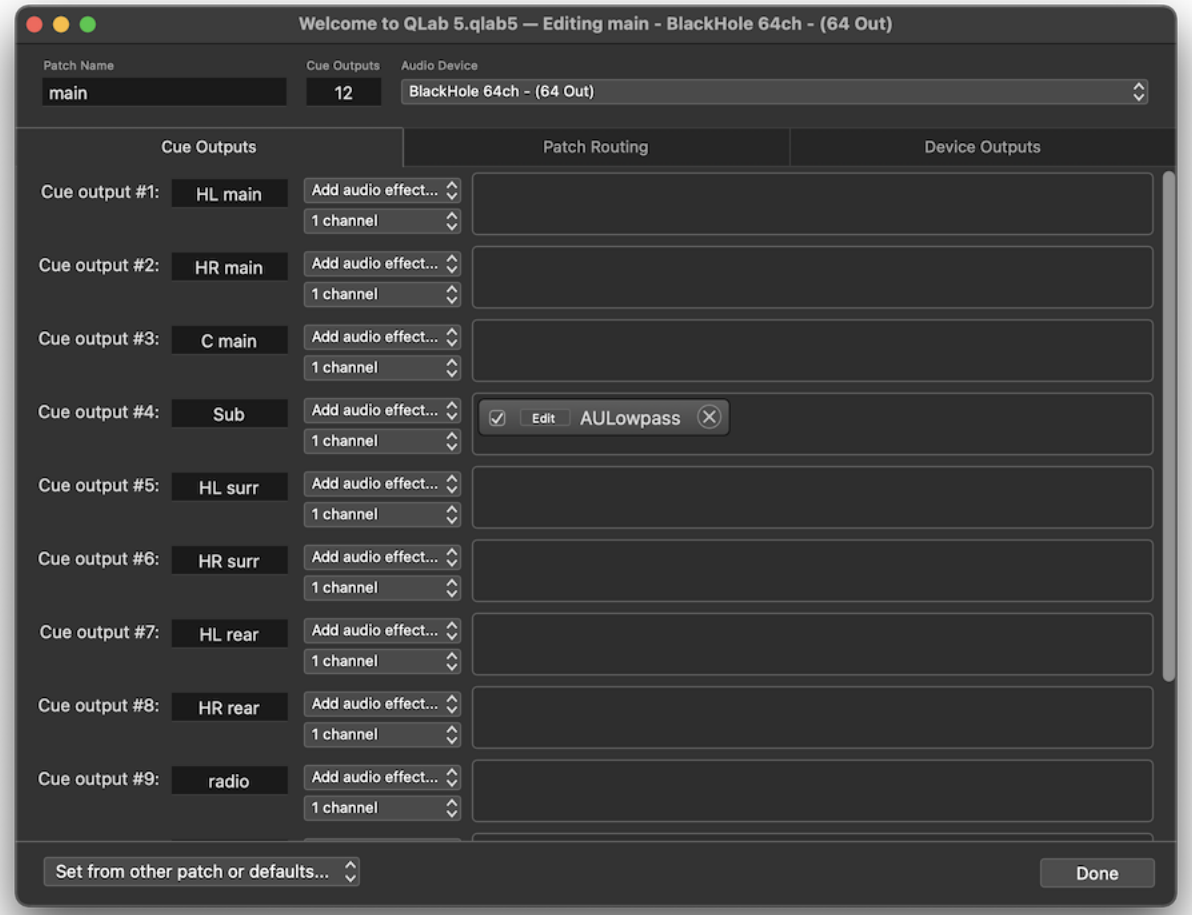
Audio output patches can have anywhere from one to 128 cue outputs. Larger patches require more computing power to run, so it's best to set this number as low as your design allows. Otherwise, QLab will have to expend processing power to look after outputs that you aren't using, thus reducing the amount of processing power available to do things like play cues and render effects.

Below these controls are three tabs: *Cue Outputs*, *Patch Routing*, and *Device Outputs*.

In the lower left corner of the window is a pop-up menu labeled *Set from other patch or defaults...* which allows you to copy settings from other audio output patches in the workspace, or reset settings to QLab's defaults. The menu contains four items, which together make up the entirety of the contents of the three tabs of the Audio Output Patch Editor.

In the lower right corner is a button labeled **Done** which commits the changes you've made, and closes the window.

The Cue Outputs tab



Each row in this tab represents one cue output; cue outputs are the columns in the cue matrix mixer found in the Levels tab of the inspector for Audio, Mic, Video, and Fade cues.

You can rename cue outputs by editing the text field which defaults to the number of the cue output. The text you enter will appear beneath the output sliders in the Levels tab of the inspector. There's not a lot of space there, so it's best to use fairly brief names. You can type ⌥**return** while editing to add a second line of text inside the text field.

To the right of the text fields are two pop-up menus which say *Add effect...* and *1 channel*. The first menu allows you to choose an available AudioUnit effect to insert on the cue output. All AudioUnits installed on the Mac which appear to be compatible with QLab will be listed under the drop-down.

Some AudioUnits, notably Apple's AUMatrixReverb, only work when they're supplied with two channels of input, so for this reason the *1 channel* menu can be used to group a cue output with the following cue output. This grouping-together is only relevant to AudioUnits on cue outputs, and has no effect anywhere else in QLab.

AudioUnits inserted on cue outputs appear in the area to the right of the pop-up menus. Effects are applied in order, left to right, and the number of effects per output is limited only by the processing power of your computer.

The Patch Routing tab

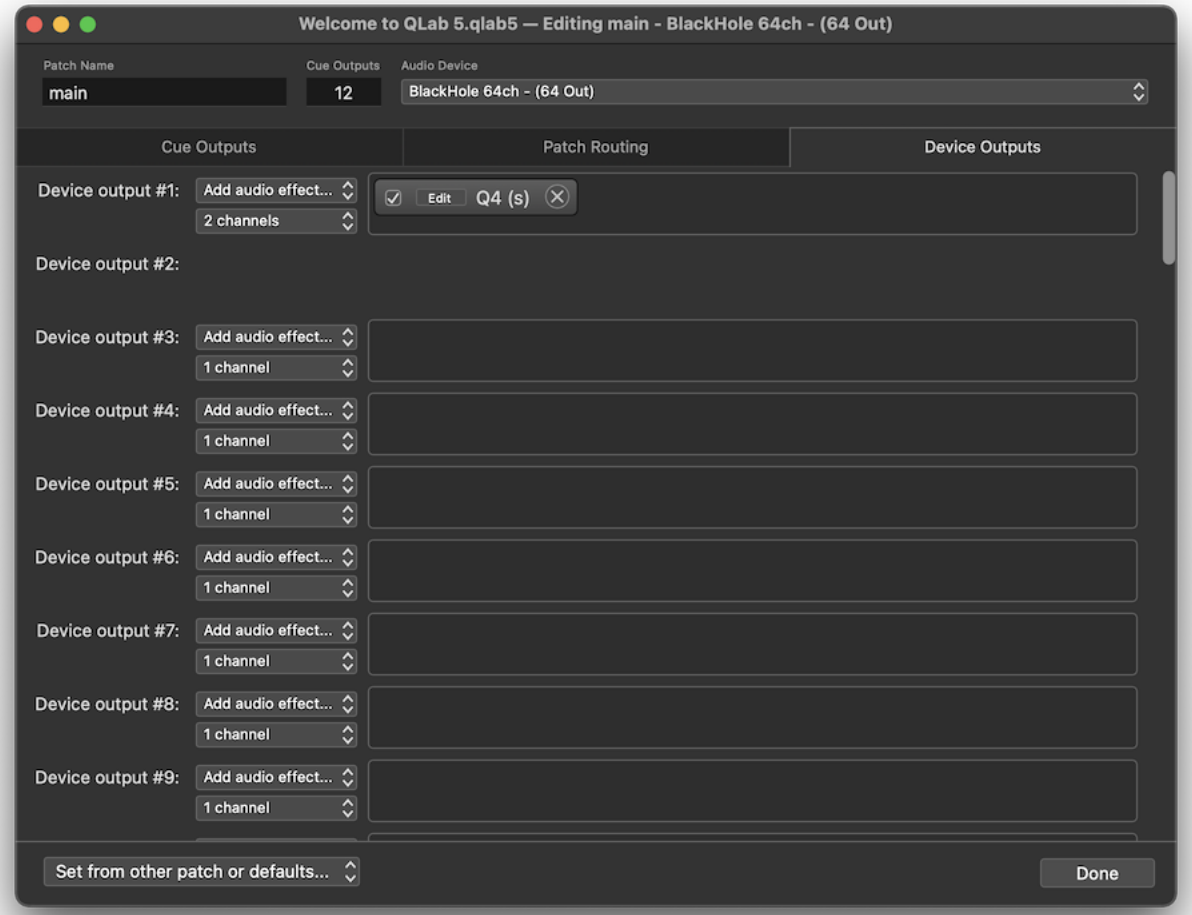


The matrix mixer in this tab allows you to do the actual routing of cue outputs to device outputs. Rows in this matrix mixer represent cue outputs and columns represent device outputs. The number of rows is dictated by the number of cue outputs in the patch, from 1 to 128, and the number of columns in the matrix mixer is dictated by the number of outputs on the device that is assigned to the patch.

The overall volume of all audio sent through the audio output patch can be adjusted using the main level control in the top left corner of this matrix mixer.

Any cue output can be routed to any device output, or to any combinations of device outputs. You can, for example, route two different cue outputs to one device output, and use effects on only one of the cue outputs in order to effectively create an auxiliary effects send for that output.

The Device Outputs tab

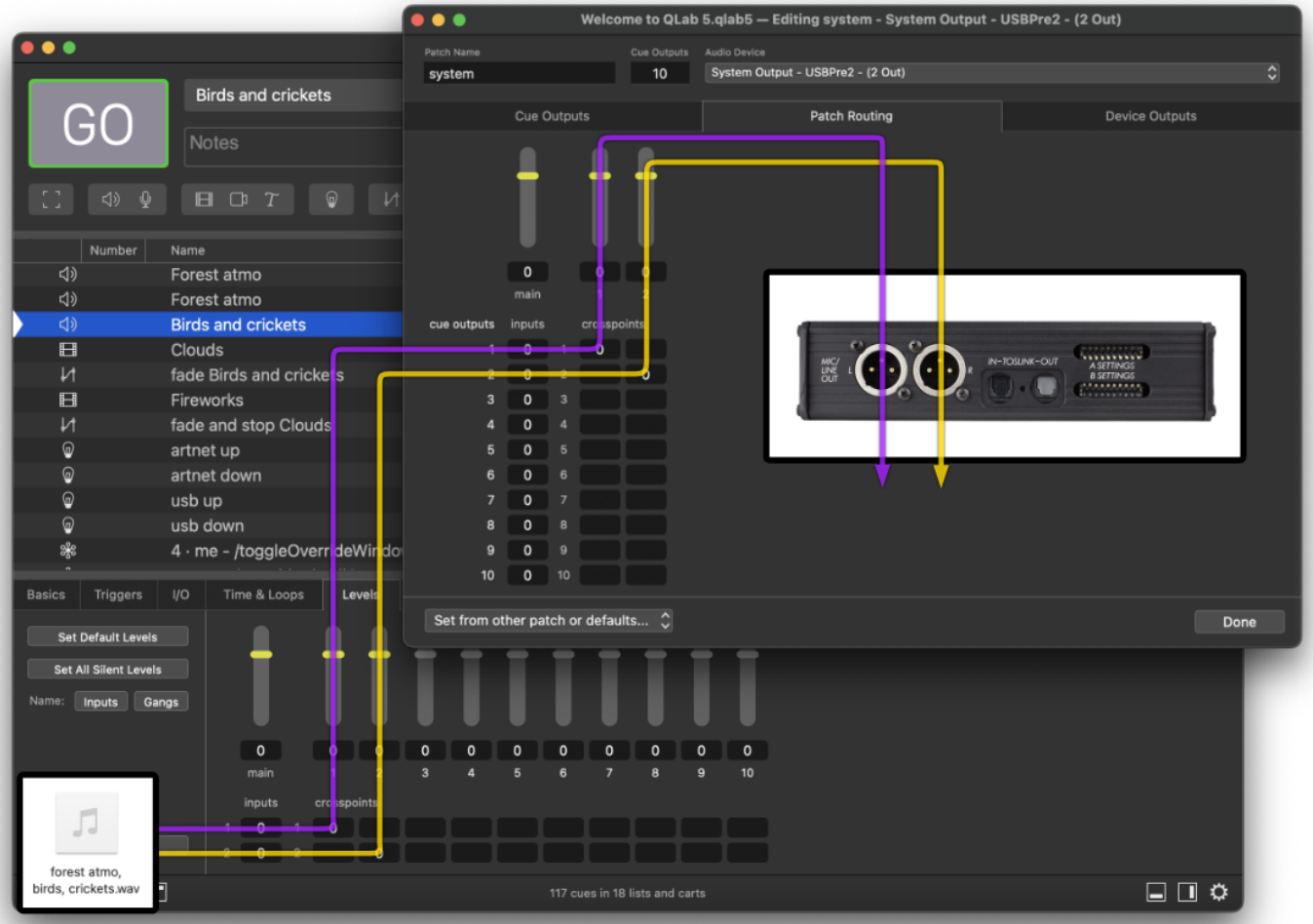


Each row in this tab represents one device output, which are the outputs made available by your audio device.

Device outputs cannot be named in QLab, but other than that the Device Outputs tab works precisely the same way as the Cue Outputs tab.

Signal Flow In QLab

This illustration demonstrates the signal flow of audio through QLab for a basic Audio cue.



In this example, the file target of the cue named “Birds and crickets” is a two-channel WAV file named `forest atmo, birds, crickets.wav`. The audio from the first channel of this audio file is represented by the purple arrow, and the second channel is represented by the yellow arrow.

Starting the the Levels tab of the inspector for the cue, the first channel of the target audio file goes from input 1 of the Audio cue, through crosspoint (1,1), and out through cue output 1.

Then, in the Patch Routing tab of the Audio Output Patch editor, the signals sent from cues to cue output 1 are received at input 1, routed through crosspoint (1,1), and out through device output 1.

Device output 1 represents the XLR output on the physical audio device, which can be verified using the Audio MIDI Setup application found in Applications → Utilities.

This routing can, of course, be changed as needed; these are only the default settings.

Audio Output Patch Warnings

Audio Output Patches can become broken or warned for the following reasons:

No audio output device selected.

⚠ The audio output patch has not had an audio device assigned to it, so it has nowhere to send audio. Assign an audio device to the patch in order to clear this warning.

Audio output device missing.

⛒ The audio device being used by the audio output patch cannot be seen by QLab, most likely because it's been turned off or unplugged, or because of some other problem within the device. Reconnect, turn on, or otherwise troubleshoot the audio device assigned to the patch in order to clear this warning.

Audio device has no outputs.

⚠ The audio output patch has been set to use an audio device with no output channels, such as the built-in microphone in most Macs. Assign an audio device with at least one output channel to clear this warning.

The Audio Map Editor

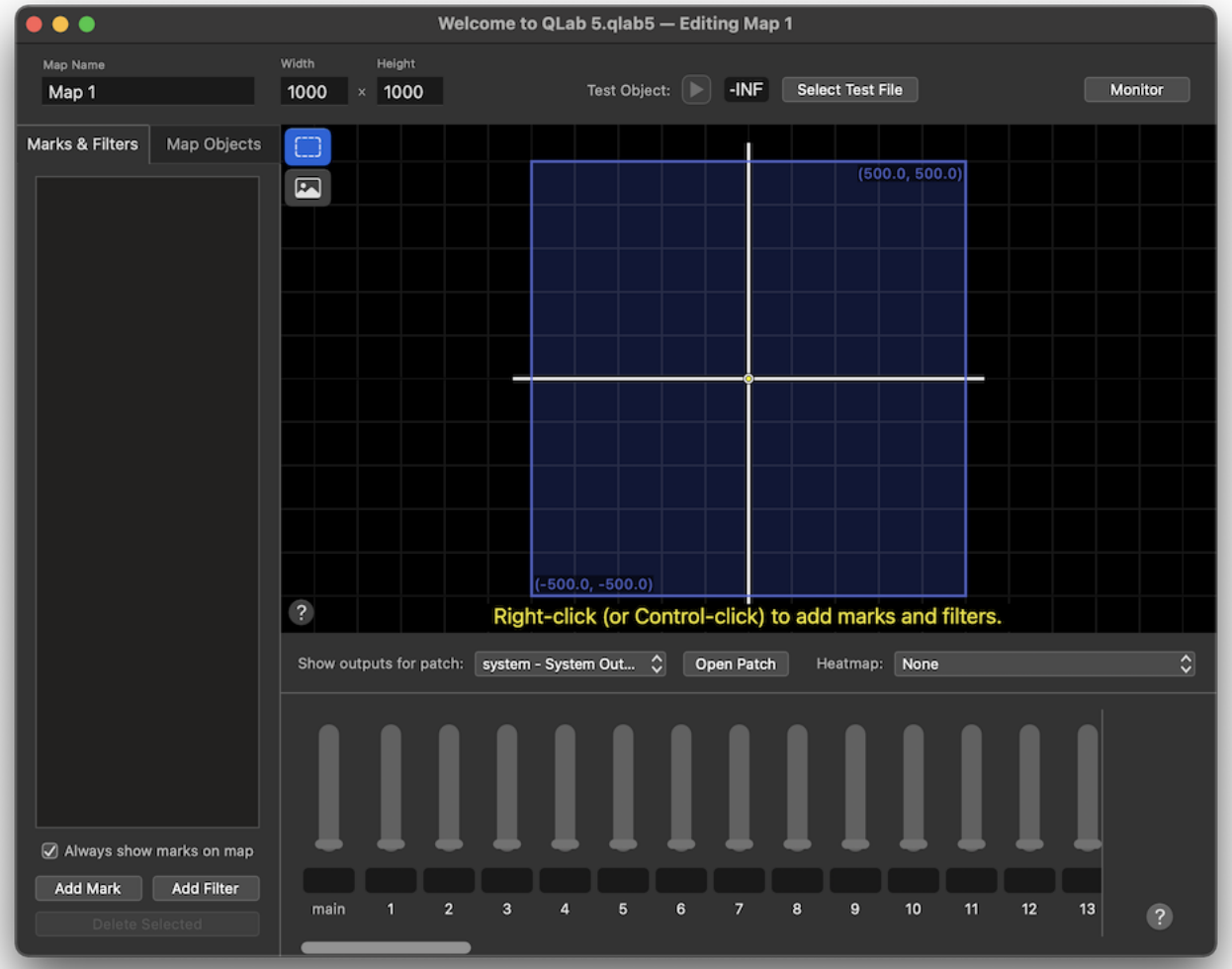
The Audio Map Editor, which is available with an Audio or Bundle license installed, allows you to customize the configuration and behavior of an audio map. If you do not have an appropriate license installed, you can use the Audio Map Editor to view maps, but not edit them.

If you have not yet looked at the introduction to Object Audio in QLab, now is a great time to do that.

Editing a map comprises these steps:

1. Set the **name** and **dimensions** of the map,
2. Optionally add a **background image,**
3. Add and configure **marks** to define how objects should sound across the map,
4. Optionally add **filters** to delineate sections of the map,
5. Optionally add **map objects** for cues to use later,
6. **Test** the map to make sure it behaves the way you want.

To open the Audio Map Editor, visit *Workspace Settings* → *Audio* → *Audio Maps*, the click the **Edit** button next to the audio map that you wish to edit. Alternately, you can select to any cue which uses the audio map, navigate to the Objects tab of the inspector, and click the **Edit Audio Map** button in the lower left corner.



At the top of the Audio Map Editor are several controls:

- The **Map Name** field lets you edit the name of the map.
- The **Width** and **Height** fields let you set the size of the map.
- The **Test Object** controls let you select an audio file to use as a test sound while working with the map, set its playback level, and play and pause it. The test object is explained in more detail below.
- The **Monitor** button opens a monitor window which lets you view the live status of the map and all objects within it.

The default size of an audio map is 1000 × 1000. While there is no upper or lower limit on map sizes, extremely small and extremely large maps will run into practicality considerations. Map dimensions are unit-less and do not inherently represent real physical space. If you want to use an audio map as a representation of real space, you could of course decide that map size units will correlate to centimeters or inches or parsecs or furlongs, and then arrange the various elements in your map according to that decision, but it's essential to understand that this is entirely optional. The object audio system in QLab is designed to operate in a spatially agnostic manner.

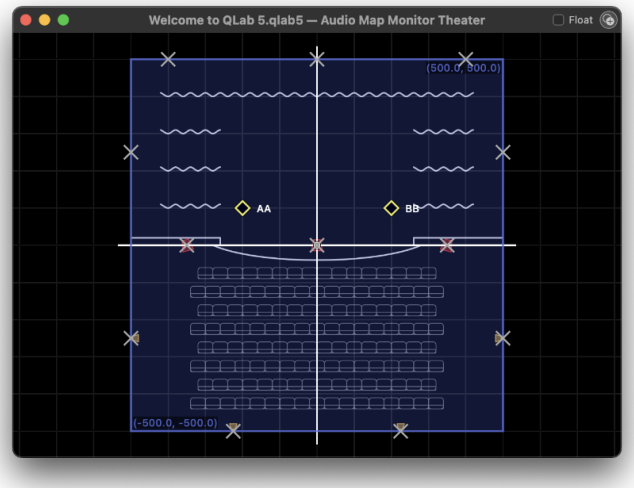
Objects can move outside the boundaries of the map, much the same way that 🎞 Video cues can be placed outside the boundaries of a display. The size that you choose for the map should be thought of as the size of the space you will typically work within.

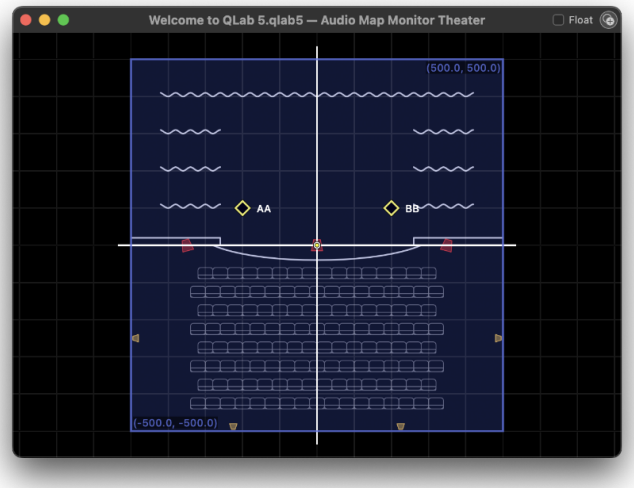
On the left side of the window are two tabs: *Marks & Filters* and *Map Objects*. The selected tab determines what you can edit and what you can see in the rest of the Audio Map Editor window.

Marks & Filters

The Marks & Filters tab displays a combined list of the marks and filters in the map, with marks above and filters below. You can double-click on a mark or filter to rename it, and drag marks and filters up and down the list to reorder them.

Below the list, the **Always show marks on map** checkbox lets you display or hide the × mark icons in cue inspectors, monitor windows, and the Map Objects tab. Only the marks' visibility is affected, not their behavior. The simplest example of when you might want to uncheck this box is when using a background image with your map that has visible details that line up the positions of your marks, like speakers.





The **Add Mark** button adds a new mark to the map. You can also add a mark by right-clicking (or control-clicking or two-finger-clicking) in the canvas.

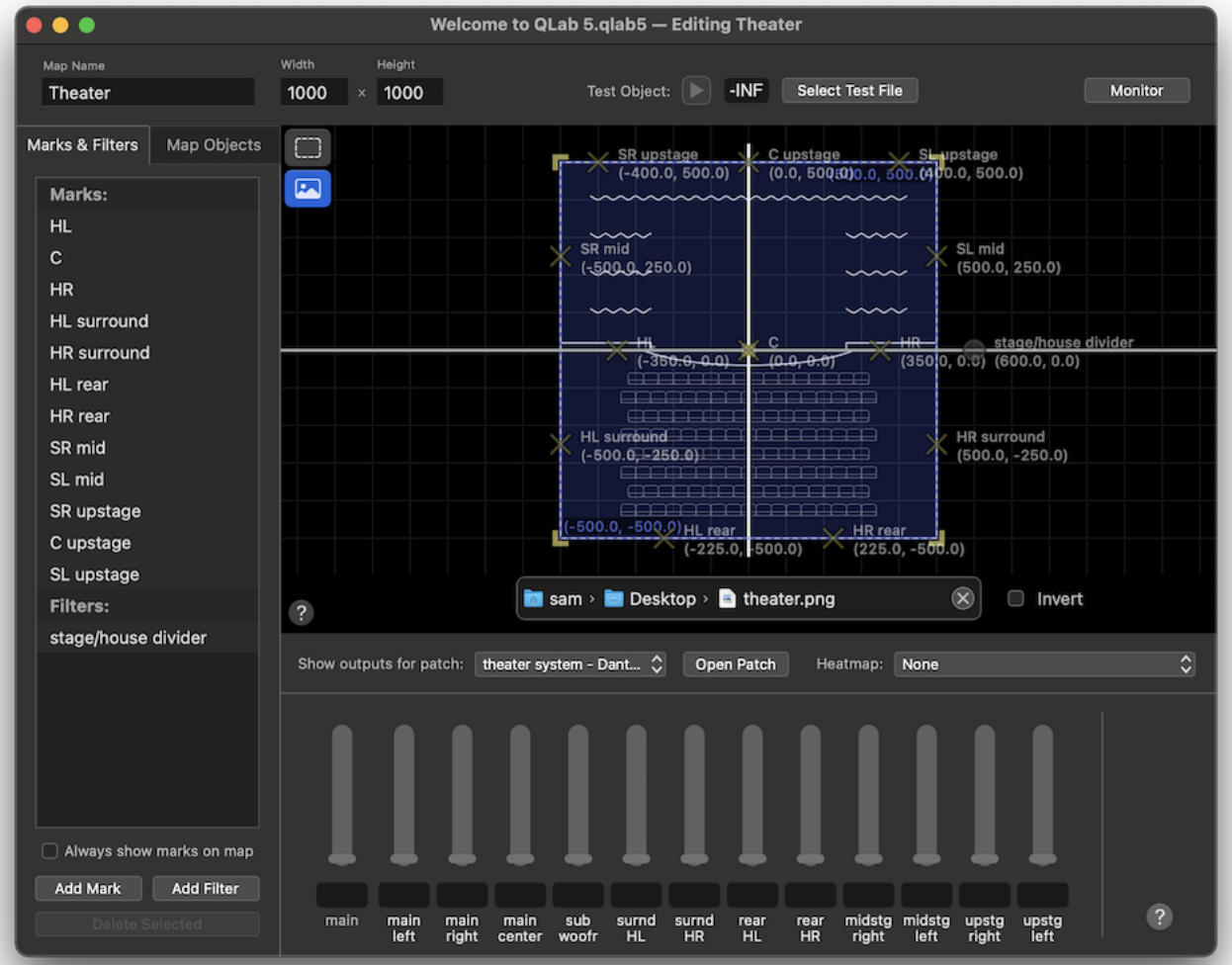
The **Add Filter** button adds a new filter to the map. You can also add a mark by right-clicking (or control-clicking or two-finger-clicking) in the canvas.

The **Delete Selected** button, which is only available when at least one mark or filter is selected, deletes the selected item(s). You can also use the ⌫ key. Deleting is undoable, of course.

The Canvas for Marks & Filters

When the Marks & Filters tab is active, marks are displayed on the canvas with their name and coordinates. Filters are displayed on the canvas, also with their name and coordinates, and with lines extending to the edges of the canvas which show their angle and width.

The two buttons in the upper left corner allow you to switch between the *select* tool, which lets you interact with marks, filters, and map objects on the map, and the *background* tool which lets you add, remove, and adjust the map's background image.



When the background tool is active, you can add a background image by dragging and dropping it into the canvas. Any image file format supported by QLab will work, but we recommend PNG or JPG images because of their small file size. QLab will offer to resize your map to the pixel dimensions of your image, which may be useful if you've made the image a specific size for this reason. You can also resize the background image by dragging on its edges or corners. If your image is generally light in color with dark lines, you may find that checking the **invert** checkbox makes it easier to see.

The ⓘ **help button** at the bottom left of the canvas displays a list of the keyboard shortcuts and modifiers that are available when working in the canvas.

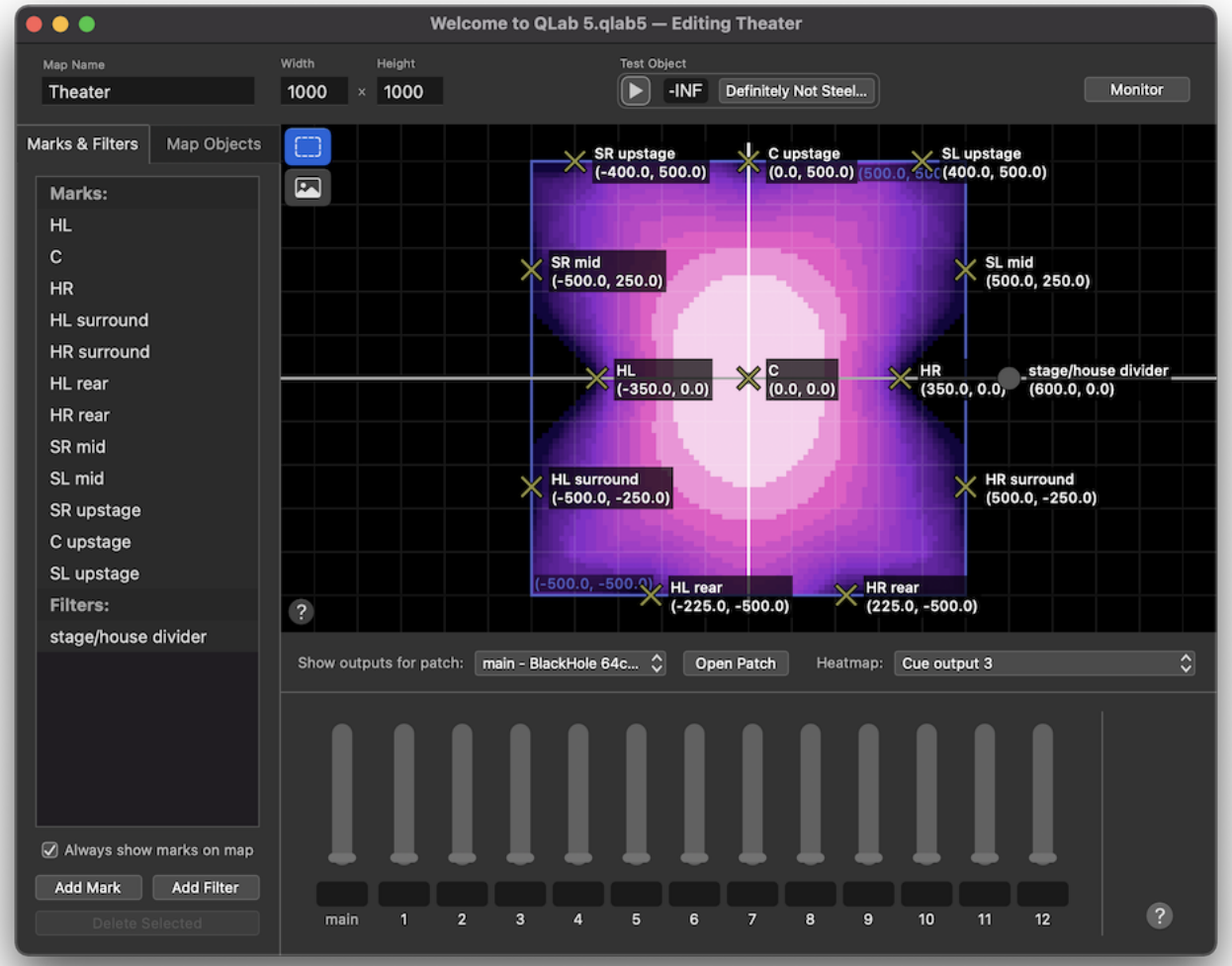
Show outputs for patch

Maps do not belong to any one audio output patch, but it can be convenient to use an audio output patch to "audition" a map as you build it. When the Marks & Filters tab is active, you can use the **Show outputs for patch** pop-up menu to select an audio output patch. When a patch is selected, the cue outputs labels of that patch will appear in the inspector, and sound from the test object will be routed through that patch. No actual connection between the map and the audio output patch is made; this is purely a testing tool.

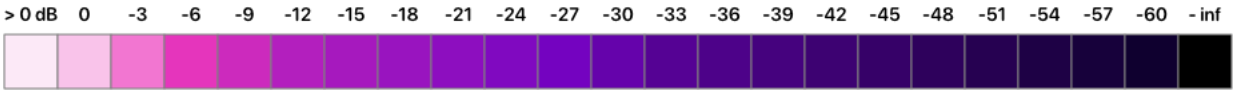
The **Open Patch** button lets you quickly open the Audio Output Patch Editor for the selected patch.

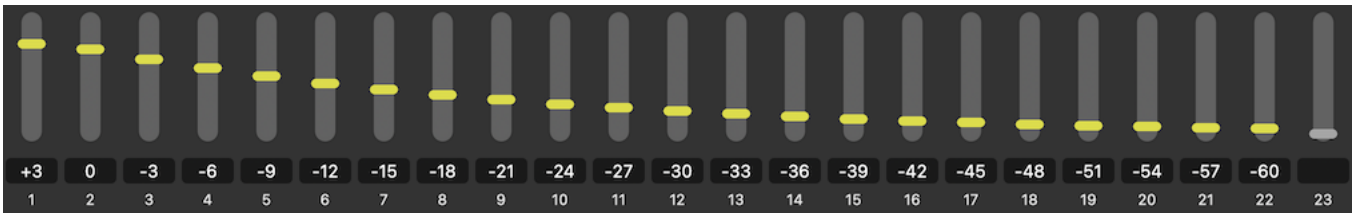
Heatmaps

The **Heatmap** pop-up menu lets you select a cue output to visualize on the canvas. The heatmap shows a representation of the level of an object in the chosen cue output at every point across the map.



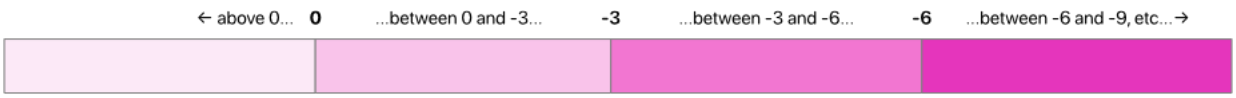
The heatmap scale is calibrated in twenty three increments, with all levels above 0 dB in the first step (nearly white), then twenty one steps from `0` (very light pink) down to `-60` (very dark purple), and `-INF` in the last step (black).





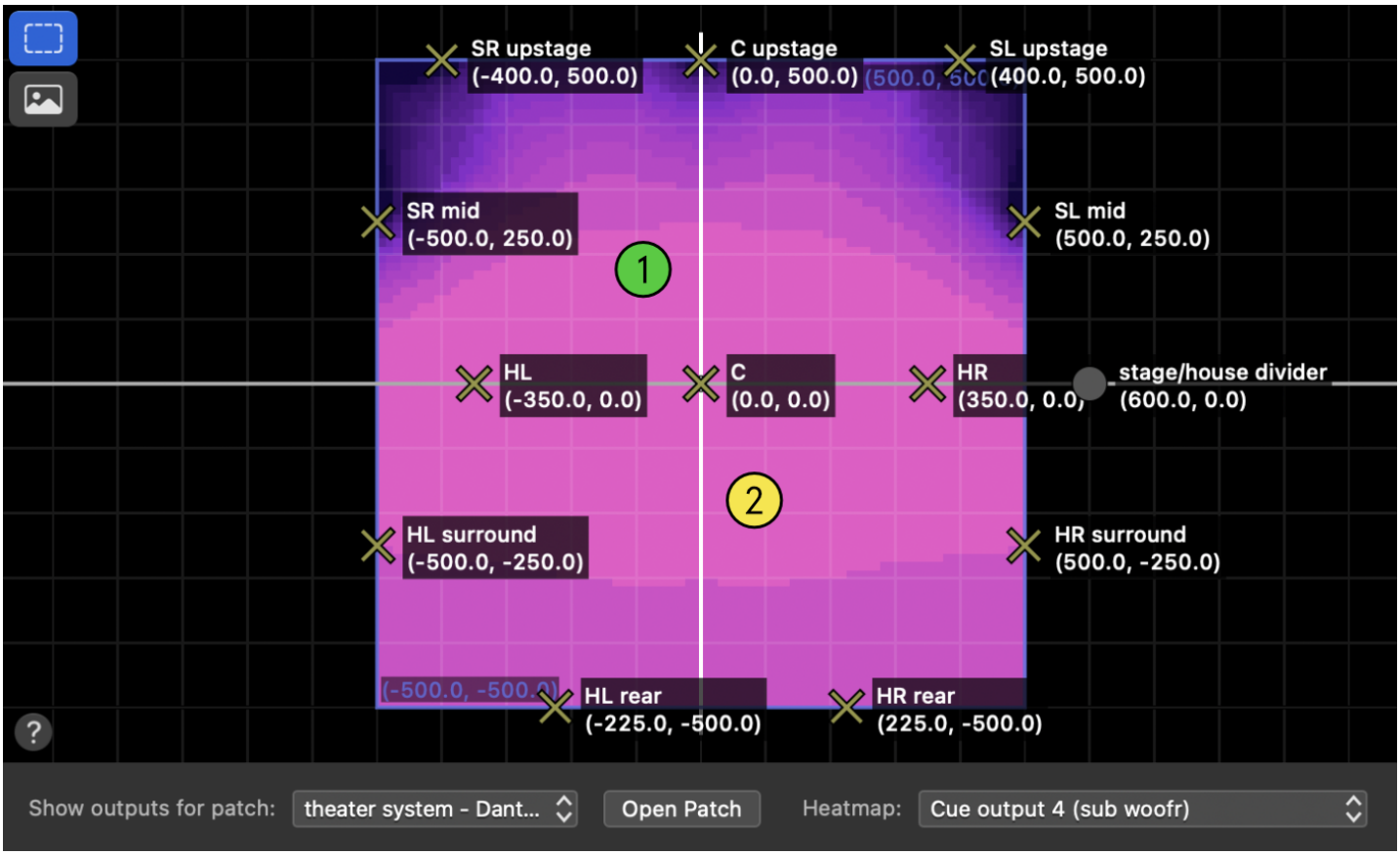
Since QLab does not (and in fact cannot) know about the physical specifics of your sound system and your venue, this map does not represent actual absolute loudness (dB SPL). It represents the relative level of an object as it is routed into a cue output.

To be exact, it's the *boundary* between each color that represents the indicated level. Each color represents a range of levels between the boundaries.



If your workspace's minimum volume limit is lower than `-60`, then the very dark purple color at the end of the scale will represent all levels between `-60` and just above `-INF`.

If you are using a map to represent real space, it can be tempting to think of the heatmap as an approximation of how things might sound to a listener in a particular spot in the theater. This temptation must be resisted! The heatmap is an approximation of how things might sound when an *object* is in a particular spot in the theater.



In the screen shot above, the heatmap is showing the level of cue output 4, which is the subwoofer output. By looking at this heatmap, you can be confident that objects placed at location (1) and (2) will be heard in the subs at approximately the same volume, certainly within `3` dB of each other. But QLab doesn't know where the subs are located, whether they are cardioid or omnidirectional, and so on. So if people go and stand or sit at location (1) or location (2), they may have very different experiences.

Option-hover

Another visualization tool that can help you as you configure your map can be used by holding down the option key (⌥) and moving your mouse cursor around the map. As you do, the levels in the inspector will update live to show output levels as though the mouse cursor were an object. This lets you slide your mouse around the map and answer the question, "how would an object be routed if it were sitting right here?"



The Inspector for Marks & Filters

Essentially, the inspector shows cue output controls, but the appearance of these controls will vary based on which audio output patch is selected in the **Show outputs for patch** pop-up menu and whether any marks or filters are selected.

If no audio output patch is selected in the pop-up menu, the inspector will display 128 cue outputs labeled with numbers. If a patch is selected, the inspector will show as many cue outputs as that patch contains, labeled however they are labeled in that patch.

Marks

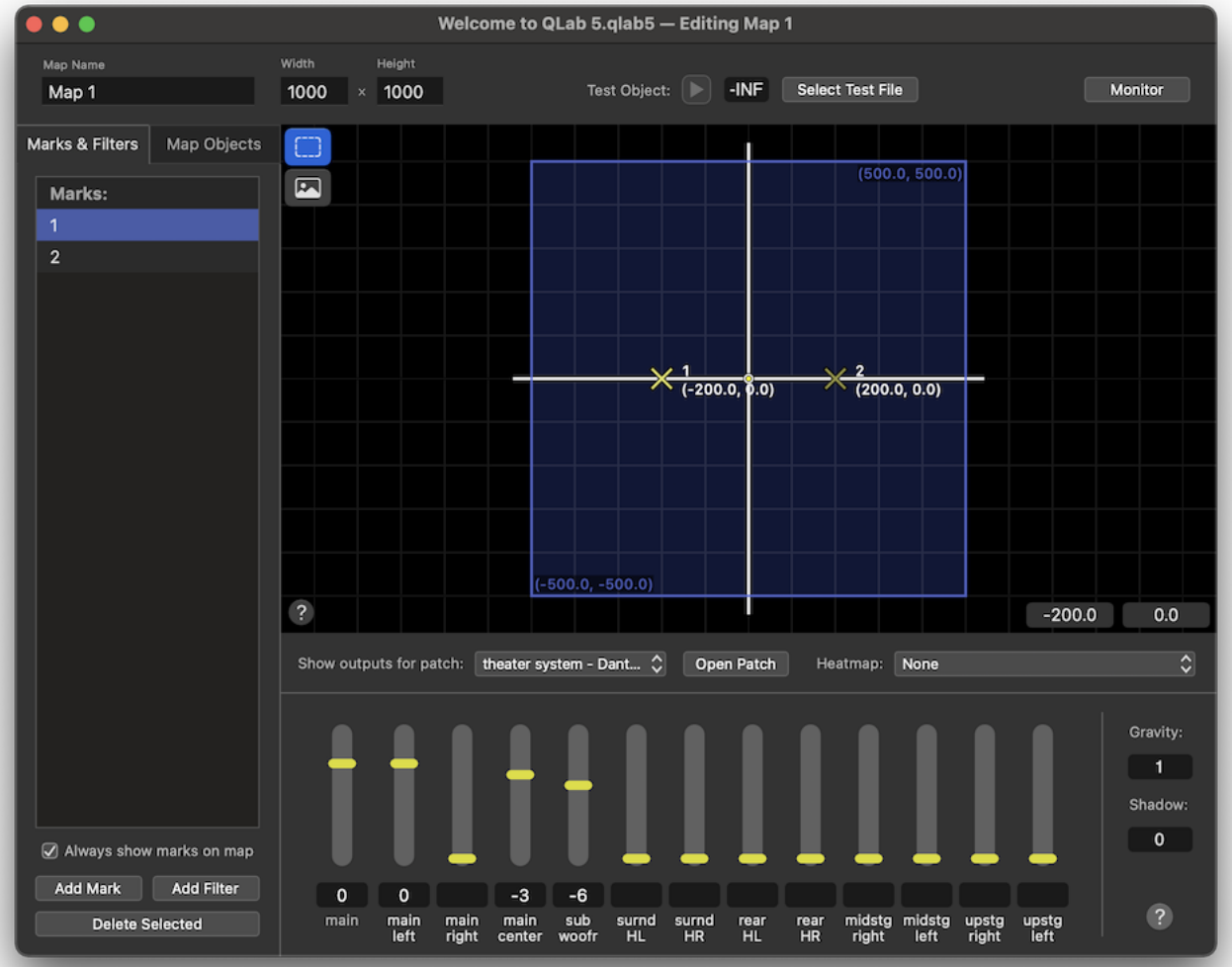
A mark defines how an object should sound at a specific location on the map. To achieve this, marks have four properties:

Position

You can drag marks around to position them, or select them and edit their position textually using the text boxes in the lower right corner of the map.

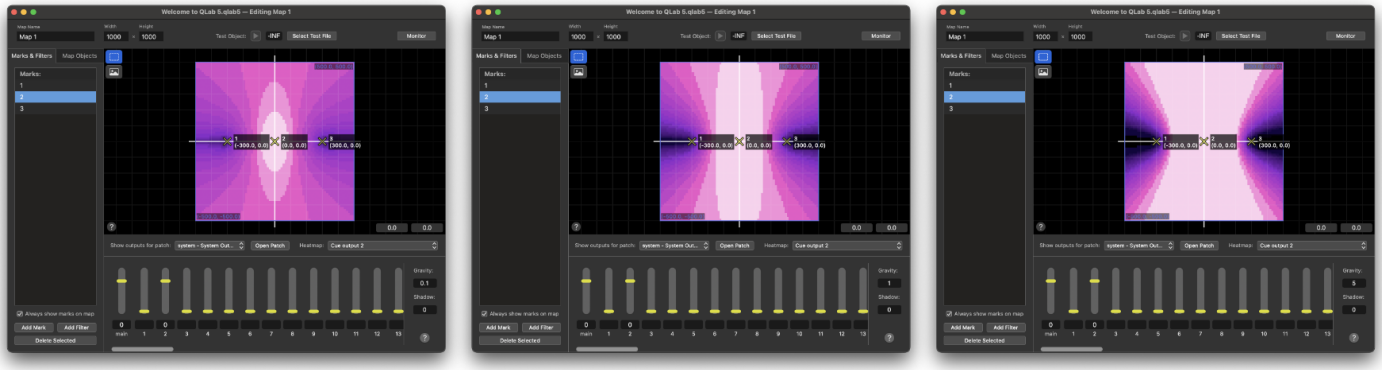
Levels

When a mark is selected, adjust the levels in the inspector exactly as though you are adjusting the levels on a cue. The level controls here work the same way as other level controls in QLab: you can drag the slider, click and drag within the text field, type in the text field, and option-click in the text field to snap from `0` to -INF.



Gravity

The magnitude of the mark's influence on the map. A mark with higher gravity will exert more influence upon the map than one with lower gravity. The default gravity of a mark is 1, the minimum possible gravity is 0.1, and the maximum is 5.

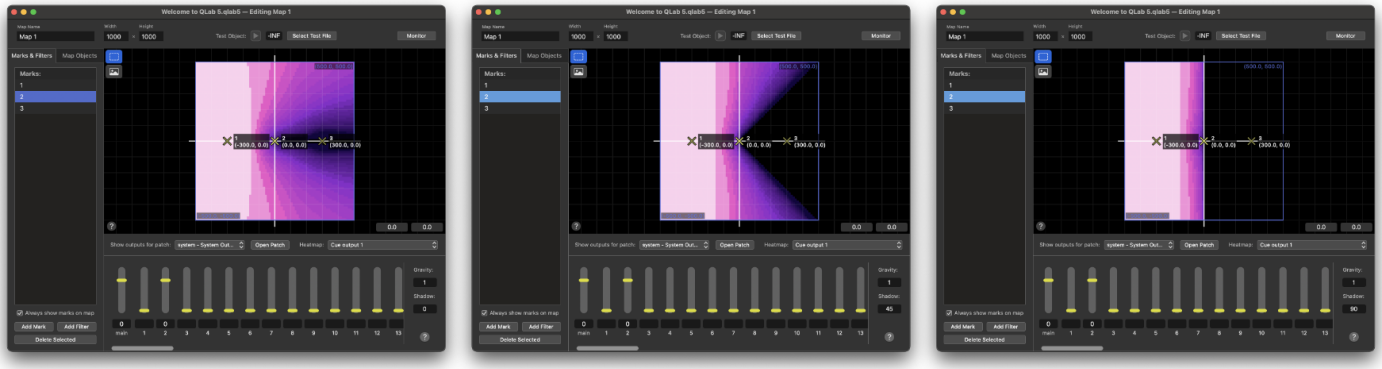


You can click on this image to enlarge it.

In these images, the gravity of the center mark is `0.1` on the left, `1` in the center, and `5` on the right, while the gravity of the other two marks remains constant at `1`. The heatmap shows how the influence of the center mark grows as the gravity increases.

Shadow

The routing of audio from an object is determined based on the relative distance of the object to each mark. Closer marks are more influential, and more distant marks are less influential. When one mark is entirely "hidden" from the object by another, its influence is extremely limited or eliminated altogether. Marks with a larger shadow will hide more objects than marks with a smaller shadow. Shadow is measure in degrees; the default is 0°, and the maximum is 90°.



You can also click on this image to enlarge it.

In these images, the shadow of the center mark is `0` on the left, `45` in the center, and `90`. The heatmap shows how the influence of the left mark is constrained by the shadow of the center mark.

Filters

Filters limit the influence of marks; objects on each side of a filter are only influenced by marks on the same side of the filter. However, filters may optionally be set to ignore specific cue outputs, allowing the behavior of those outputs to remain unaffected by the filter. Additionally, marks which sit within the width of a filter are unaffected by the filter.

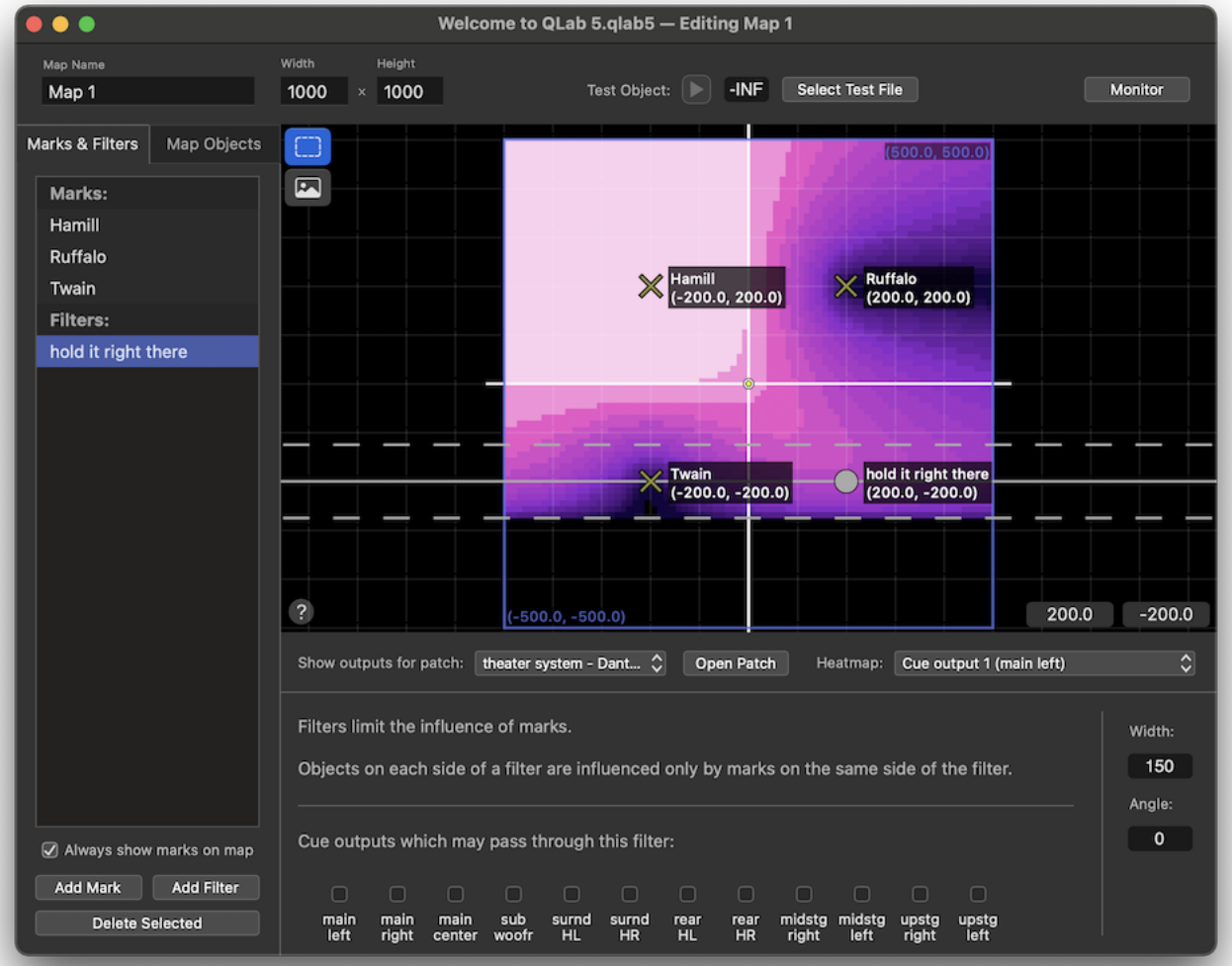
Filters have four properties:

Position

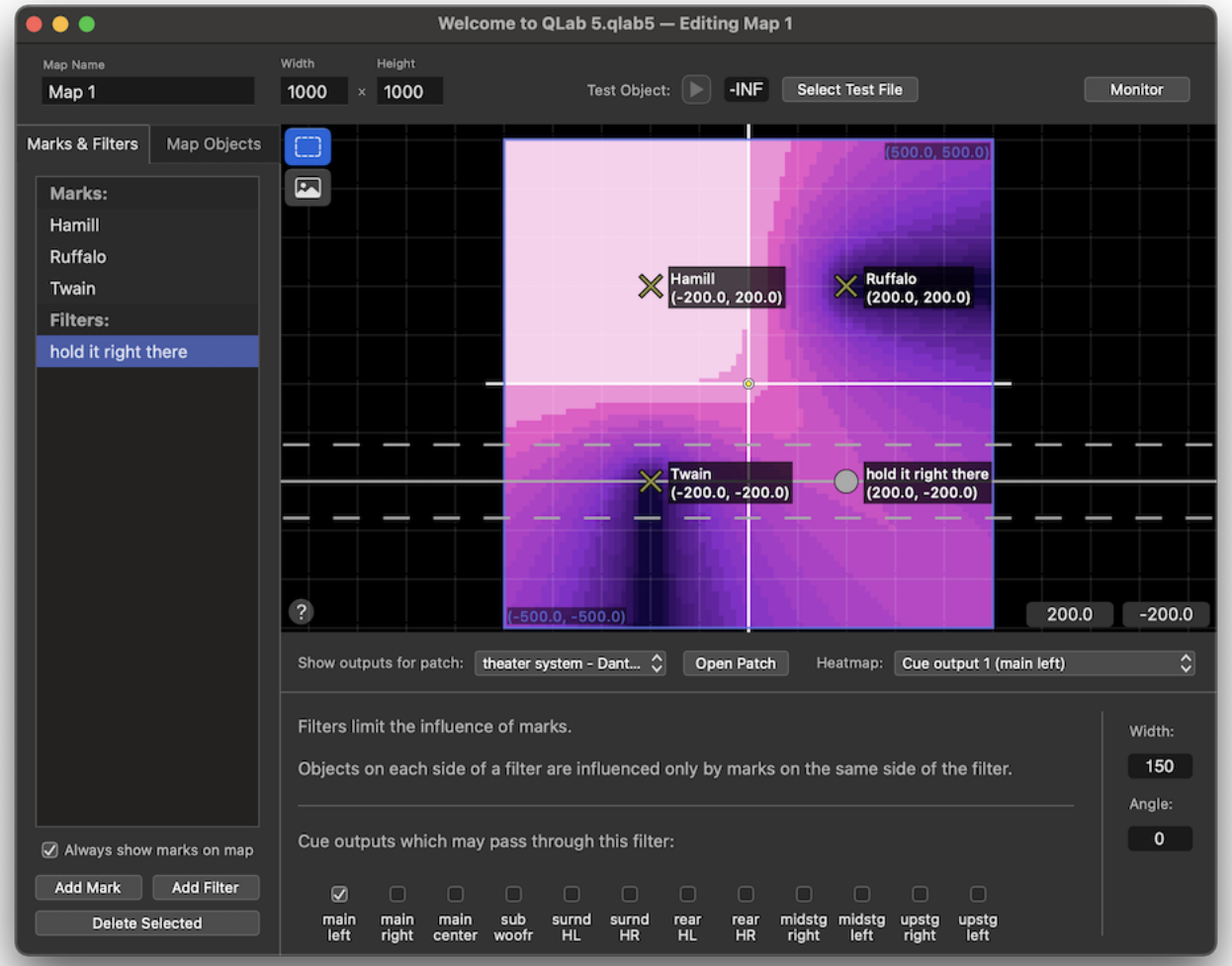
You can drag marks around to position them, or select them and edit their position textually using the text boxes in the lower right corner of the map.

Levels

By default, a filter blocks all influence. Objects on one side of the filter cannot "see" marks on the opposite side. When one or more filters are selected, the inspector displays a row of checkboxes for each cue output.



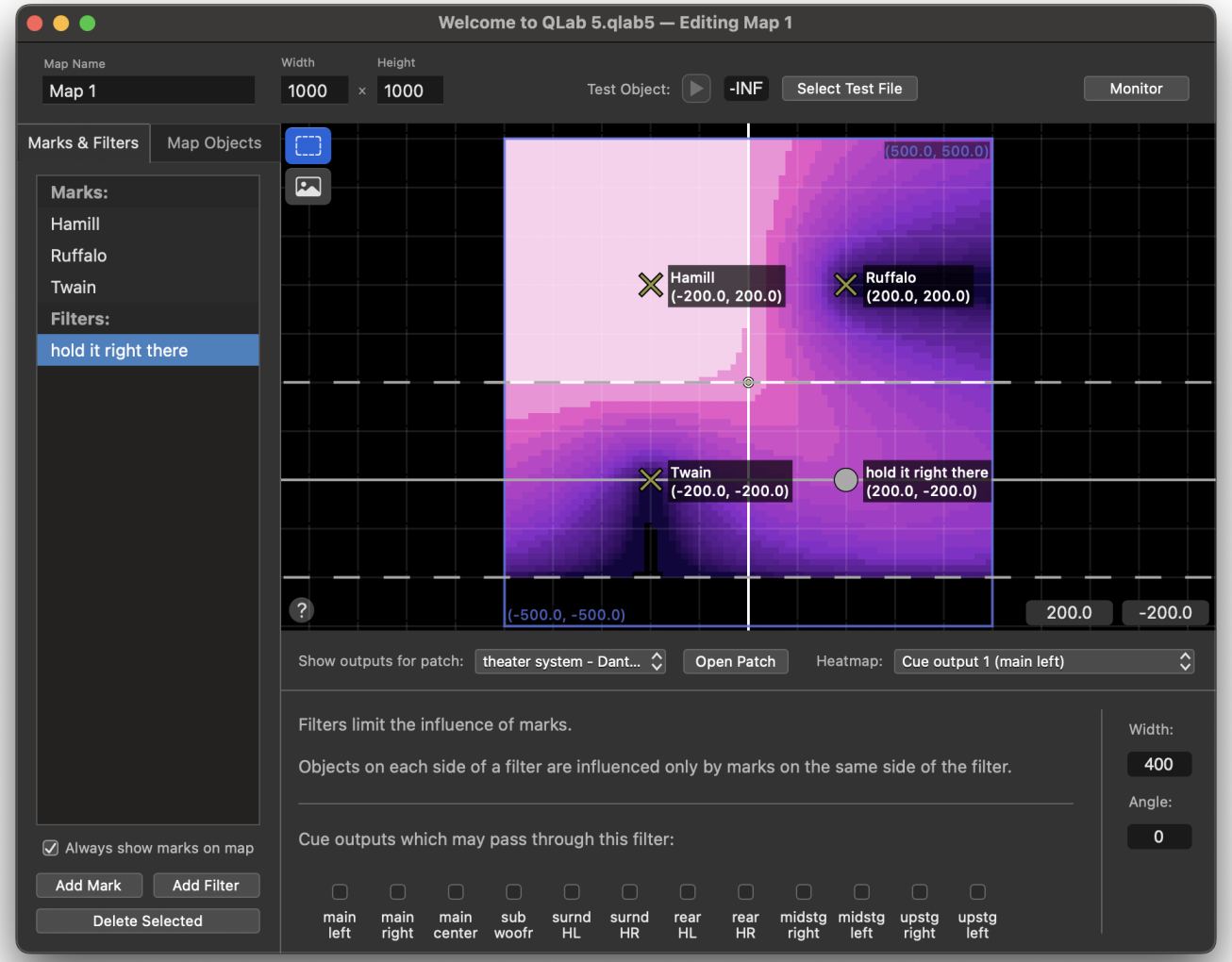
When a box is checked, that cue output passes through the filter. This means the filter no longer blocks *all* influence of all marks. Objects will “see” marks on the opposite side of a filter, but they will only “see” the marks’ influence on the cue outputs which are set to pass through.



The heatmap is showing cue output 1 (main left). Since the checkbox for main left is checked, that output is exempted from the filter. As far as cue output 1 is concerned, there is no filter.

Width

The width of a filter determines how gradually or sharply its effect is applied. You can adjust the width of a filter by clicking and dragging on either one of its two dashed lines, or you can use the text box in the lower right corner.



Angle

You can adjust the angle of a selected filter by clicking and dragging on its center line, or you can use the text box in the lower right corner.

Map Objects

The Map Objects tab displays a list of the objects that belong to the map. You do not need map objects to use object audio; cues can have their own objects too. Map objects are useful when you want to send audio from multiple cues to one object that can be positioned and controlled consistently. You might, for example, create a map object to correlate with a prop radio on stage, and send all the Audio cues for the radio through that map object. When an actor picks up the radio and moves to another spot on stage, you can use a Fade cue to move the map object accordingly, without worrying about which Audio cue is playing at the time.

Map objects are named using double letter by default, but you can rename them in the list as well as reorder them.

Below the list, the **Always show marks on map** checkbox lets you display or hide the × mark icons in cue inspectors, monitor windows, and the Map Objects tab. Only the marks' visibility is affected, not their behavior.

The **Add Map Object** button adds a new mark to the map. You can also add a map object by right-clicking (or control-clicking or two-finger-clicking) in the canvas.

The **Delete Selected** button, which is only available when at least one map object is selected, deletes the selected object(s). You can also use the ⌫ key. Deleting is undoable, of course.

The Canvas for Map Objects

When the Map Objects tab is active, map objects are displayed on the canvas with their name and coordinates. If the **Always show marks on map** checkbox is checked, marks are also displayed, though without their names and coordinates.

The ⓘ **help button** at the bottom left of the canvas displays a list of the keyboard shortcuts and modifiers that are available when working in the canvas.

The Inspector for Map Objects

Map objects have two properties.

Spread

The spread of an object is its apparent size on the map. The default spread is `0`, which means an object's position is measured as a single point on the map, even though objects are drawn much larger than than in order to make it easier to see and interact with them.

When the spread of an object is larger than 0, QLab calculates its routing as though it occupies every point within its spread. An object's level in each cue output will be as loud as the loudest point within its spread.

Color

Objects can be given a color to make them stand out in the map. There are twenty colors to choose from here, divided into eight "basic" colors and twelve "extended" colors. The color doesn't alter the behavior of the object in any way, so feel free to use color coding in whatever way is most useful to you.

Objects set to "no color" will be drawn in yellow when they are selectable, in yellow and filled when they are selected, and in grey when they are not selectable.

Reset

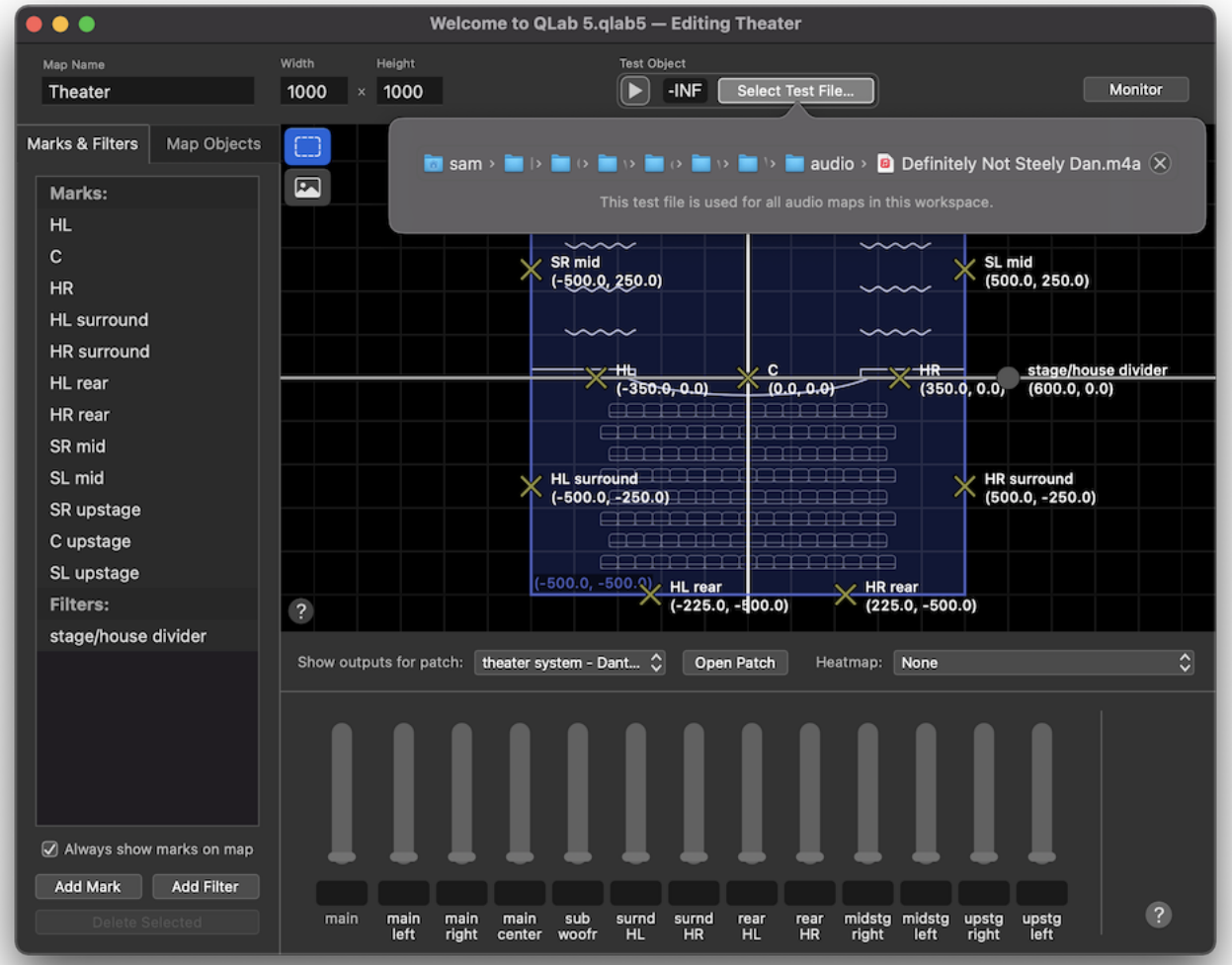
If any map objects have been moved by ↓↑ Fade cues, the ↺ Reset button becomes available. Clicking it will reset all map objects of the map to their home positions.

The Test Object

The test object makes it easy to listen to your map as you develop it, rather than jumping back and forth between a cue in your cue list and the audio map editor. The test object lets you play an audio file of your choosing and move it around the map with minimal setup.

To use the test object, first be sure to select an audio output patch in the **Show outputs for patch** pop-up menu. Without that, the map has no connection to any actual outputs, so you won't be able to hear anything.

Next, click the **Select test file...** button and choose an audio file to use as your test audio.



Please be advised that certain choices of test material may have social consequences. It is beyond the scope of this manual to advise the reader on the finer points of this topic.

Once a file is selected, you can click the ▶ button to start the test object. It will appear in the center of the map in green, and can be moved around as you please. The level of the test object starts off at -INF and can be adjusted like any other level in QLab.

Fading Audio & Audio Effects

> The Fading Audio tutorial is a hands-on exploration of the topics discussed in this section.

A Fade cue can be used to adjust the volume levels of a targeted Audio, Mic, Video, Camera cue, or an audio output patch.

Fade cues can also adjust video parameters of Video cues, Camera cues, and Text cues. When a Fade cue is selected, the inspector will only show the tabs relevant to the type of cue that the Fade cue is targeting.

The word “fade” sometimes means different things to different people. In QLab, “fade” simply means “a change in value over time.”

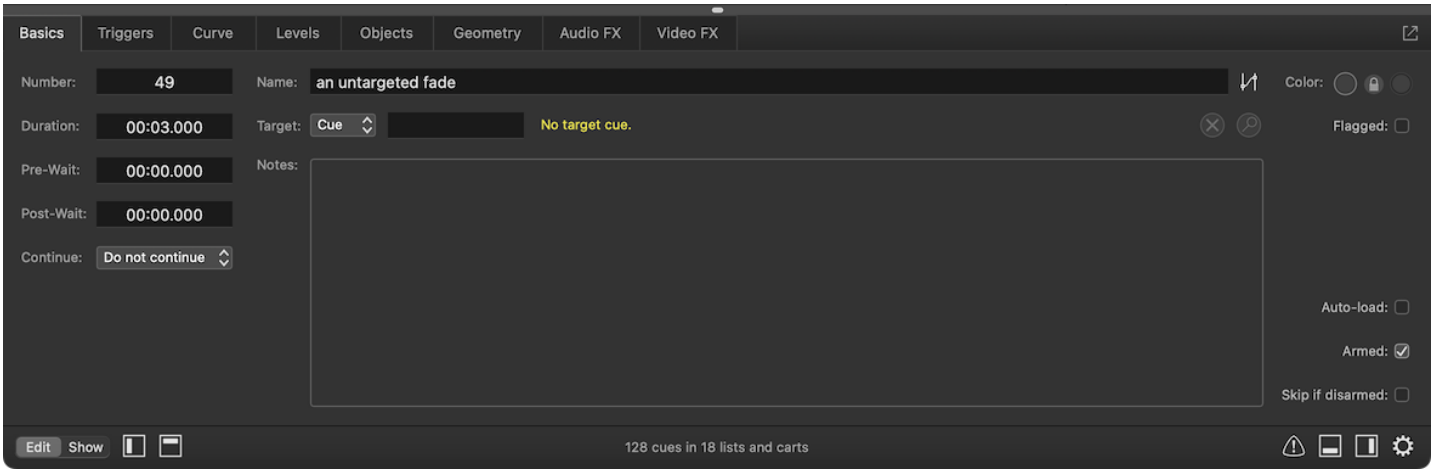
Fade cues require a cue target or a workspace settings target, have a duration, and must adjust at least one level or parameter of their cue target in order to be considered functional.

The Inspector for Fade Cues

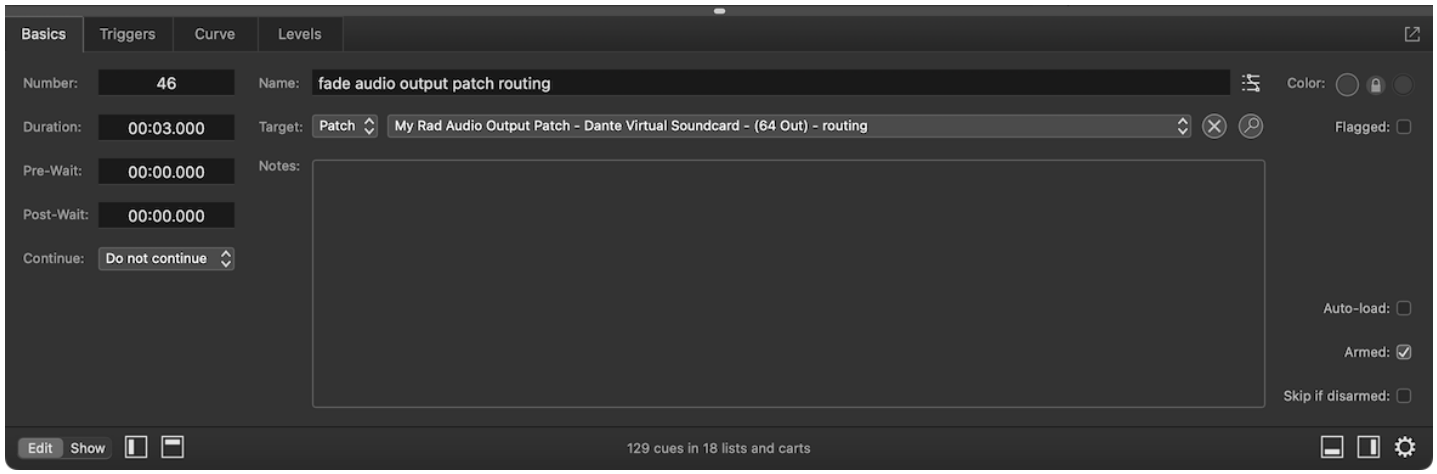
The tabs shown in the inspector when a Fade cue is selected vary based on the Fade cue’s target.

The Basics Tab

The Basics tab is always visible, but has a small but important difference from how the Basics tab usually appears.



Because Fade cues can target cues, patches, and maps, a pop-up menu appears next to the Target field to switch these different types of targets. When *Patch* or *Audio Map* are selected from this menu, a second pop-up menu appears which allows you to select the patch or map that you want to fade.



The Triggers Tab

The Triggers tab is always visible and behaves the same way that it always does.

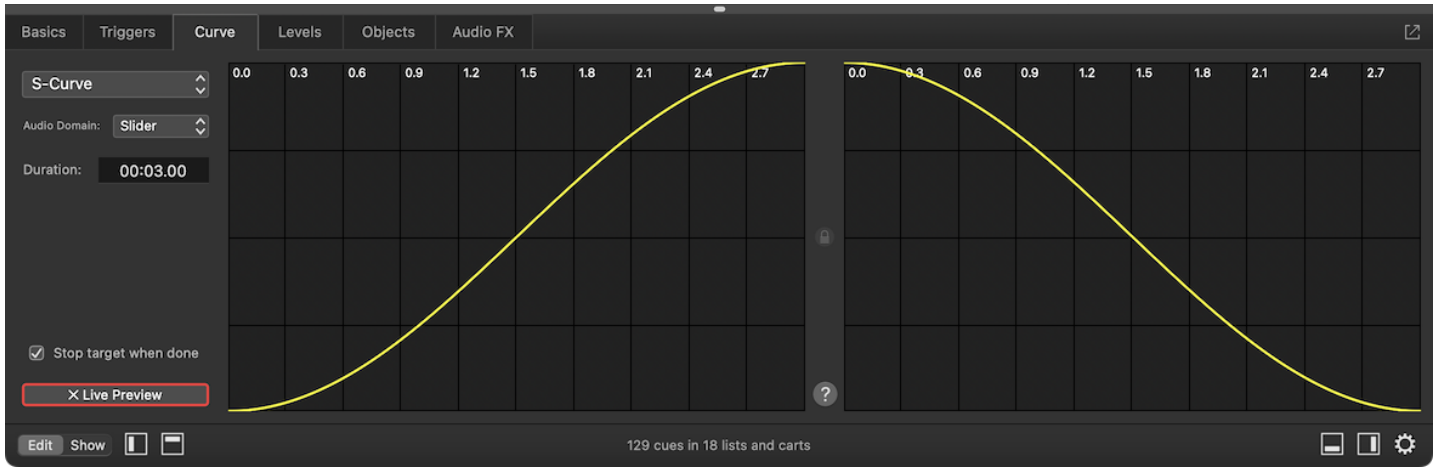
The following tabs are visible when an Audio cue, Mic cue, or audio output patch are the target of the Fade cue:

The Curve Tab

The fade curve, drawn in yellow on the right side of the tab, determines the rate of change of the parameters being faded. The curve on the left is for levels which increased by the fade, and the curve on the right is for levels which are decreased by the fade.

The horizontal axis of the curve represents time and the labels across the top will change based on the duration of the Fade cue. The vertical axis represents percentage of the total change made by the Fade cue. For the rising curve, the one of the left, the bottom left corner represents the beginning of the fade, which is to say “time = 0, completion = 0%.” The top right corner represents the end of the fade, or “time = (duration of the Fade cue), completion = 100%.” For the falling curve, the top left corner represents the beginning and the bottom right corner represents the end.

The curve shape that appears by default is set according to the Fade cue’s cue template, but you can choose another fade shape from the pop-up menu in the top left corner of the tab.



There are five options for Fade curve shapes:

- **S-Curve.** QLab's default curve shape follows an "ease-in, ease-out" envelope designed to sound natural with audio levels and look smooth with video geometry.
- **Custom Curve.** This option allows you to click anywhere along the fade curve and a create control points, which can be dragged to change the shape of the curve. Moving control points will smoothly bend the curve in the direction of the control point. To delete a control point, click on it to select it and press the **delete** key on your keyboard. To start over entirely, click *Reset to Default Shape* in the bottom left corner of the tab. The lock button between the two curves allows you to mirror the left and right curves. When the curves are locked, any change to either curve will apply to both curves.
- **Parametric Curve.** This option adds a text field labeled *Intensity* below the pop-up menu which allows you to use a mathematically precise parametric fade shape.
- **Linear Curve.** This option is similar to the *custom curve* option but instead of smoothly ending the curve, control points create a precise, sharp bend. To delete a control point, click on it to select it and press the **delete** key on your keyboard. To start over entirely, click *Reset to Default Shape* in the bottom left corner of the tab. The lock button between the two curves allows you to mirror the left and right curves. When the curves are locked, any change to either curve will apply to both curves.
- **2D Path.** This option is not yet available for Fade cues which target Audio or Mic cues.

Both the rising and the falling curve use the same fade shape type, but if you use *custom curve* or *linear curve* you can create individual curve shapes for each.

The *Audio Domain* pop-up menu lets you choose the scale that QLab uses to fade audio levels. This pop-up is only relevant to fading audio levels; it has no effect on any other parameters, and no effect on audio levels when they are not being actively faded. There are three options for the audio domain:

- **Slider domain.** The slider domain emulates the design of physical sound consoles, maximizing the useful range of audible levels and making a straight fade sound as smooth and natural as possible.
- **Decibel domain.** The decibel domain uses a logarithmic scale.
- **Linear domain.** The linear domain uses a linear scale.

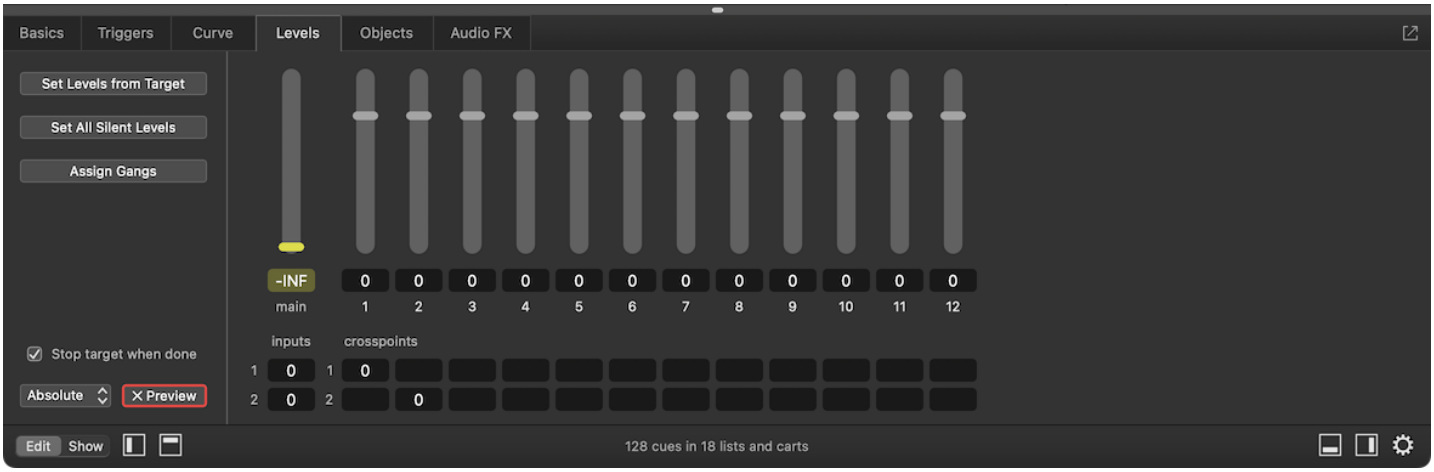
Equal Power and Equal Gain fades

To create an *equal power* fade, use a **parametric curve** with audio in the **linear domain.**

To create an *equal gain* fade, use a **linear curve** with audio in the **linear domain.**

The Levels Tab

The Levels tab allows you to specify which audio levels you wish to fade, and what their final volume will be. You can specify a different volume for each level, and one Fade cue can fade as many different levels as you like.



The left side of the Levels tab contains a few setup tools:

Set Levels from Target. Clicking this button will invoke the *paste cue properties* sheet in a special way. First, it will behave as though the target cue was selected and copied, and second it will automatically select the "Audio" set of properties to paste. You can choose other properties if you like, but if you simply hit the **enter** key, QLab will paste the levels from the target cue onto the Fade cue. This is a convenient way to get started building a fade, as it

will help you keep track of the starting point from which you will be fading. Using the keyboard shortcut ⇧⌘T will set levels from the target cue without invoking the sheet.

Set All Silent Levels. Clicking this button will return the Fade cue to a pristine state, with no levels set to change.

Assign Gangs. This button behaves the same way as it does in an Audio cue.

Stop Target When Done. If this box is checked, the target cue will stop once the fade is complete. If the box is unchecked, the target cue will continue to run after the fade is complete.

Absolute Fade. This drop-down menu lets you choose between an absolute fade, which is QLab's default, and a relative fade. Relative fades are discussed in detail below.

Preview. This button provides quick access and a visual reference to the live fade preview setting.

When the Fade cue targets a cue, the right side of the Levels tab contains a cue matrix mixer which shows the same number of input rows as the Fade's target cue with the same input names. When the Fade cue targets a patch, it shows a matrix mixer which duplicates the Patch Routing tab of the target patch.

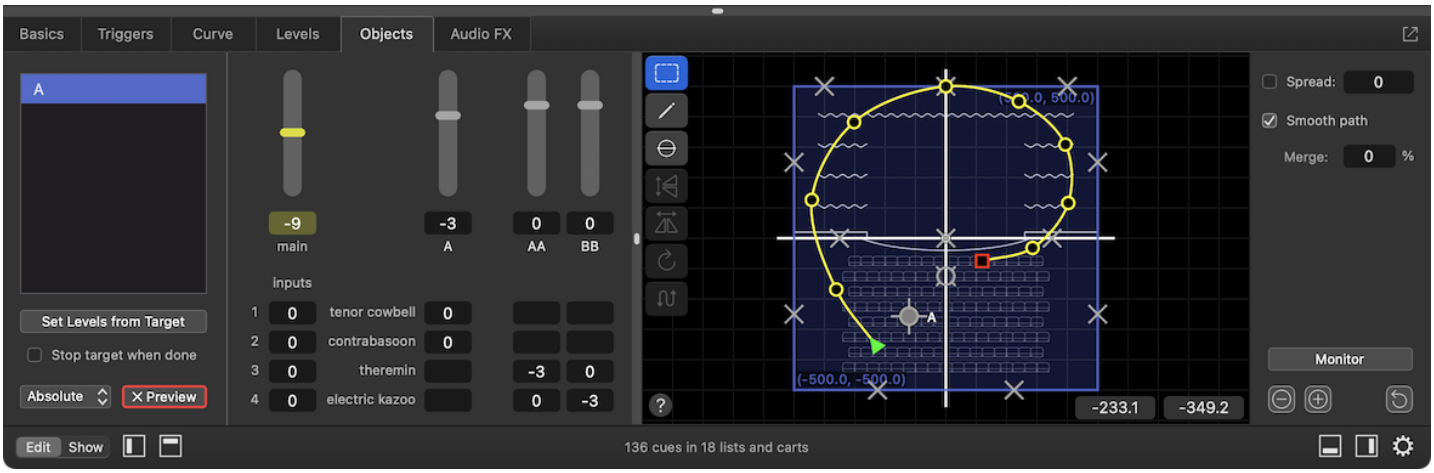
Adjusting any level in the matrix mixer will turn it yellow, which means that level is "active." When the Fade cue is run, any active levels will be faded. Inactive (grey) levels will remain untouched. You can activate or deactivate a level by clicking on it.

Audio levels are specified in decibels (dBFS). The default range of volumes in QLab is `+12 dB` to `-60 dB`. The lower volume limit is regarded as silent, and displayed as -INF. You can change the volume limits if your workspace in Workspace Settings → Audio

Entering any level at or below the minimum volume limit, or any non-numeric value, will activate that level at -INF.

The Objects Tab

If the target of the Fade cue has objects, the Objects tab allows you to adjust the levels, position, and spread of those objects.



The left side of the tab contains a list of objects in the target cue or map. To move an object or adjust its spread, it must first be selected in that list.

Fading Object levels

The middle of the tab contains a matrix mixer which is analogous to the one in the Objects tab of an Audio cue. You can adjust any levels here in exactly the same way as in the Levels tab.

The Map Canvas

The right side of the tab shows the canvas of the target's audio map. Any objects belonging to the target are shown in their current position. When an object is selected in the list on the left, the object is shown in its own assigned color. When an object is not selected, it is shown in grey.

There are two other types of indicators that may be visible on the canvas for the selected object or objects.

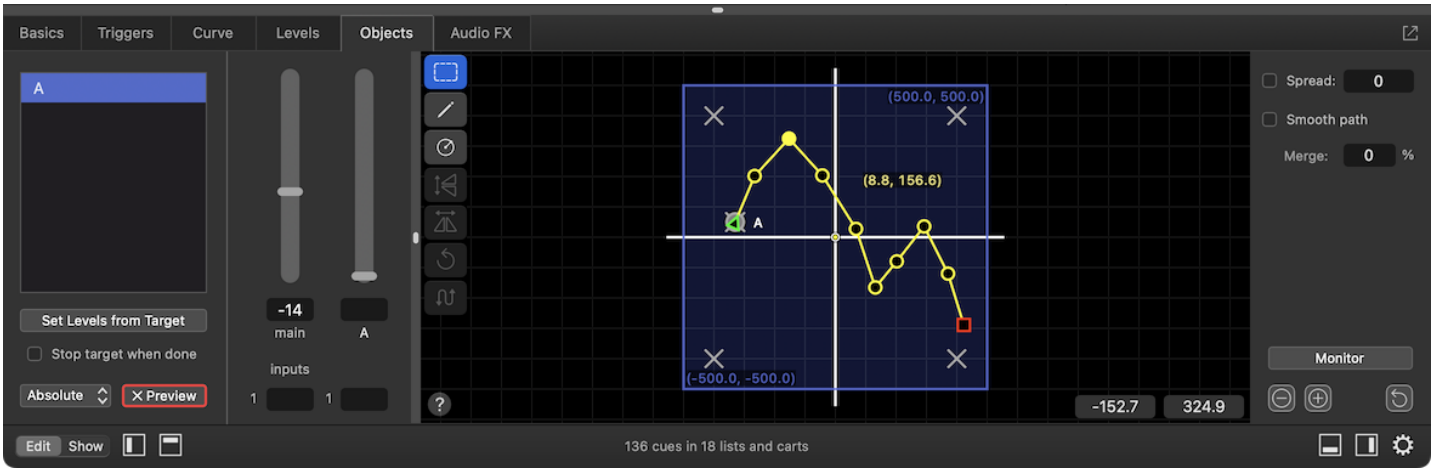
1. The ⌖ live position indicator shows the current position of the object. That position may be set in the target cue or map, or it may be the result of one or more ⇂↾ Fade cues having moved the object, or it may be the result of the object being moved by OSC messages. In any case, this indicator shows where the object is sitting in the map at this moment. If the ⇂↾ Fade cue's target is a cue, the cue must be loaded or running for the ⌖ live position indicator to appear.
2. The ⌑ home position indicator shows the saved starting position of the object. In the case of cue objects, that's the position of the object in the target cue's Objects tab. In the case of map objects, it's the position of the object in the Audio Map Editor.

To fade the position of an object, you can select it from the list on the left and then draw a fade path in the canvas. The fade path is shown as a yellow line over a blue background, with the following details:

- The starting point of the path is drawn as a green triangle.
- The ending point is drawn as a red square.
- Points along the path are drawn as yellow circles.
- If there is enough space, small directional carets pointing away from the starting point and towards the ending point will appear along the path.

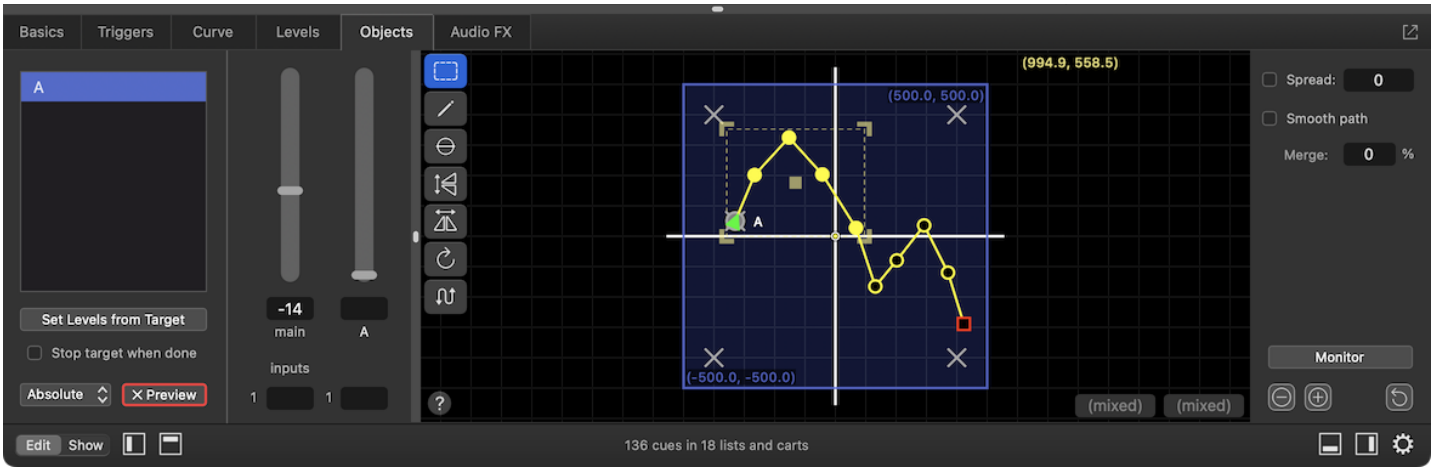
Along the left edge of the canvas is a set of tools for creating and editing paths.

The **select** tool lets you click on individual control points to select them for precise editing.



With one point selected, you can use the text fields in the lower right corner to type in a value or click and drag up and down to adjust the value.

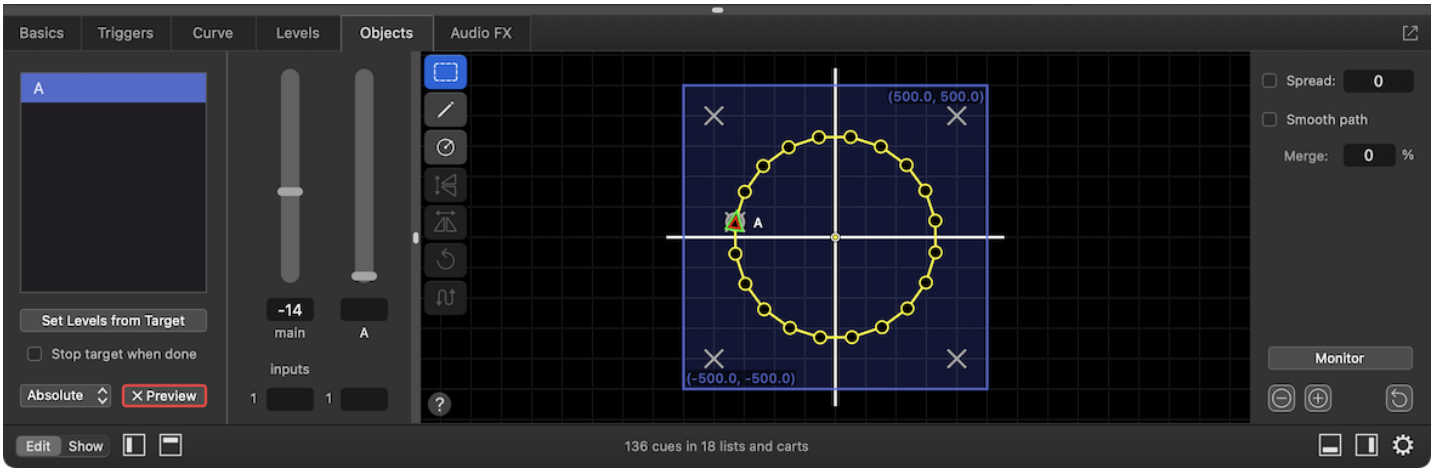
You can also click and drag a selection box around several control points:



The **draw** tool lets you click to add points to the path, or click, hold, and drag to draw a freeform line. Holding down the option key (⌥) key allows you to add a point in the middle of a path, otherwise each drawn point will be added to the end of the path. Double-clicking while using the draw tool finishes drawing and switches to the select tool.

The **circle** tool lets you draw a perfectly circular path. The first click sets the starting (and ending) point of the circle, and the second click defines the diameter of the circle. Holding down the option key (⌥) allows you to instead click once to define the center point, and then click again to define the

radius of the circle.

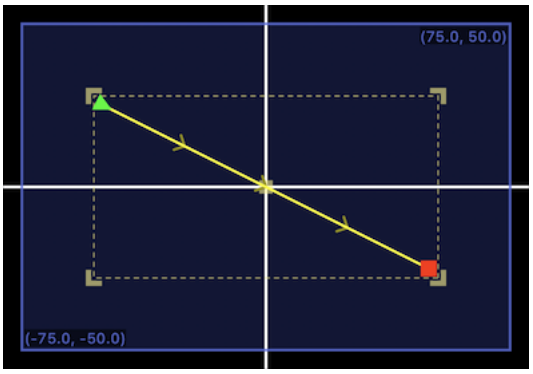


Flip vertical flips the selected group of points around the X axis.

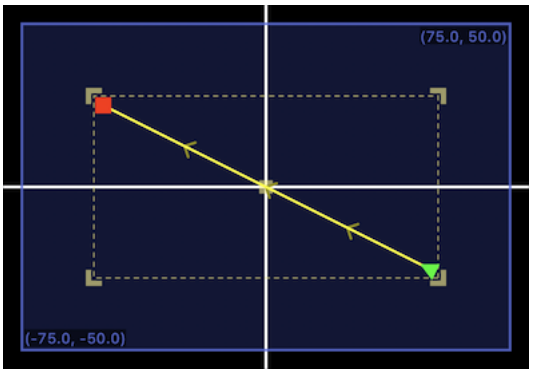
Flip horizontal flips the selected group of points around the Y axis.

Rotate spins the selected group of points in 90 degree increments around the center point of the selection.

Reverse leaves the points in the same location, but reverses their order in the path. For example, consider this simple path with two points:



After using the reverse tool, what was the start point is now the end point, and vice versa:



The (?) **help button** displays a list of the keyboard shortcuts and modifiers for working with 2D fades.

To the right side of the canvas are a few more controls:

Spread. The spread of the object can optionally be faded, regardless of whether the position of the object is changing.

Smooth path. When this box is un-checked, paths will be drawn precisely between each control point. When the box is checked, the path will be smoothed between points to create a naturally curving line.

Merge. If the object that's being moved is not in the same place as the starting point of the path, the merge control lets you determine how the object should behave when the ⇃↾ Fade cue starts. If the merge control is set to `0` %, then the object will snap from its current location to the starting point of the path instantaneously and then move along the path over the duration of the fade. If the merge control is set higher, the object will move organically towards the fade path from its current position. The merge control determines what percentage of the path will be traversed before the object meets up with the path. For example, if the merge percentage is `20` %, then the object will spend the first `20` % of the fade merging towards the path, and the remaining `80` % of the fade traveling along the path exactly.



The **Monitor** button opens the monitor window window for the map of the ⇃↾ Fade cue's target.

You can use the ⊖ and ⊕ buttons to zoom in and out on the canvas. If you're using a trackpad, you can also zoom in and out by placing the cursor over the canvas and pinching in and out. If you're using a mouse or trackball with a scroll wheel, you can zoom in and out by holding down the option key (⌥) and scrolling vertically.

While zoomed in, you can pan around using two-finger swipes on a trackpad. If you have a mouse or trackball with a middle button (oftentimes the scroll wheel can be clicked; this is usually the middle button,) you can pan around my holding that button down.

The ⏮ reset button resets the path to the default shape defined by the cue template.

If **Live Preview** is on, every adjustment you make in the Objects tab will be visible and audible live and in real time.

The Audio FX Tab

In addition to adjusting audio levels, ⇃↾ Fade cues can be used to adjust any audio effects on their target cue. When you apply an audio effect to a cue, any ⇃↾ Fade cues which target that cue will automatically recognize the effects you've applied, and those effects will be listed in the Audio FX tab of the inspector for the ⇃↾ Fade cue. By default, the checkbox next to an effect will be unchecked, meaning that the ⇃↾ Fade cue will not adjust parameters of that effect. To adjust an effect with the ⇃↾ Fade cue, just check the box next to the effect.

Unlike fading audio levels, there is no way to activate or deactivate individual parameters of an audio effect in a fade. You can only fade the entire audio effect from one "state" to another. Thus, to avoid accidentally adjusting more parameters than you intend, you can click *Set Audio FX from Target* to copy the state of the audio effect from the target cue and paste it into the ⇃↾ Fade cue. Otherwise, the levels in the ⇃↾ Fade cue will default to the built-in default state for that audio effect.

Once you've done that, or elected not to, you can click the *Edit* button for the effect to adjust the effect.

It's important to understand that when you click *Edit* to view the effect, you're not opening the same window as when you click *Edit* in the target cue. The title bar of the effect window will show the cue name and number of the cue it belongs to, but nevertheless it can be difficult to keep track of which edit window you're looking at. Take your time and be sure that you're making changes in the place you intend.

With the effect editor window open, you can make any adjustments you wish to the effect. This is a situation in which *Live fade preview* can be very helpful. It's turned on by default, but if you've turned it off, consider turning it back on when adjusting audio effects in a ⇃↾ Fade cue. That way, you can make adjustments in the ⇃↾ Fade cue while the target cue is running, so that you can hear those adjustments in real time.

When you're done, close the editor window.

Now, when you run the ⇃↾ Fade cue, the parameters you adjusted will smoothly fade from their initial values, set in the target cue, to the values you set in the ⇃↾ Fade cue.

While AudioUnits can also be placed on Cue Outputs and on Device Outputs, ⇃↾ Fade cues cannot adjust those effects. You can learn more about effects on outputs from the the Audio Output Patch Editor section of this manual.

Fading Rate

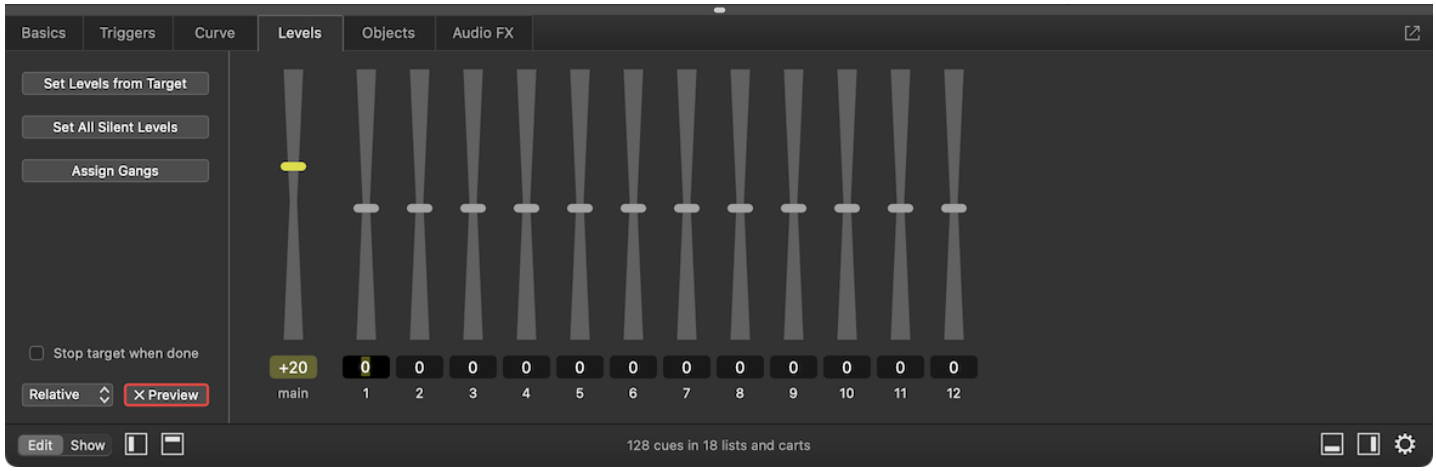
If the target of the Fade cue is an Audio cue or a Video cue, the Audio Effects tab of the inspector also lets you fade the playback rate of the target cue by checking the box labeled *Fade rate to*, and setting a target rate.

The maximum playback rate in QLab is `33`, or 33× normal speed, and the minimum is `0.03`.

Relative Fades

By default, Fade cues are *Absolute*. This means that any parameters that you adjust with a Fade cue will arrive at their final levels regardless of their status before the Fade cue runs. If you use a Fade cue to set the main level of a target Audio cue to `-12`, then the main level of that Audio cue will end up at `-12` after running the Fade no matter where that level was set.

However, using the pop-up menu on the left side of the Levels tab, you can set a Fade cue to be a *relative* fade. Instead of setting levels to a specific value, relative fades add or subtract a given amount to the active parameters, so the starting point of those parameters very much matters. When set to *relative*, the cue's icon changes to the relative form.



In this screen shot, the Fade cue raises the main level of its target cue by `20 db`. So if the main level of target cue were set at `-40`, running this Fade cue would bring the main level to `-20`. If the main level of the target cue were set at `-10`, running this Fade cue would bring the main level to `+10`.

Be very careful with relative fades that raise levels. If, for example, you were to create a relative Fade cue to raise the main level of an Audio cue from a very soft level, say `-50`, up to `-10`, you'd need a relative Fade cue with a level change of `+40`. If you accidentally ran that Fade cue twice, QLab would attempt to fade the main level of that Audio cue to `+30`, which is rather loud!

To help protect you against this, QLab has a maximum level which is set in Workspace Settings → Audio. By default, this is set to `+12`, meaning that in the example above, the Audio cue would end up at `+12`, not `+30`. Depending on the gain structure of your sound system, though, this may still be uncomfortably or even dangerously loud.

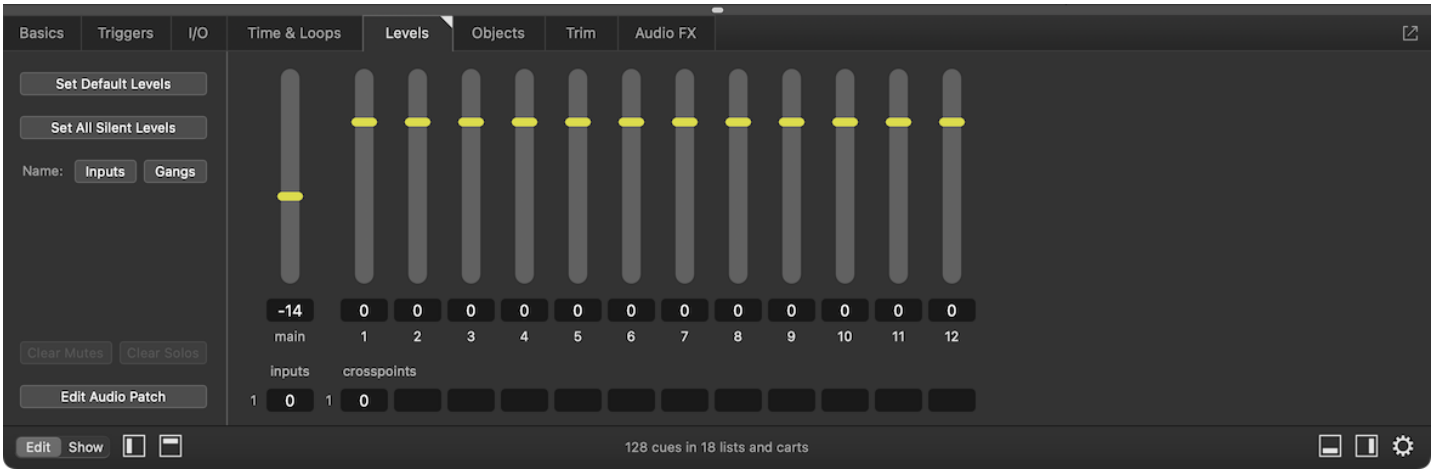
In QLab 4, using an absolute Fade cues on a parameter that had previously been adjusted by a relative Fade cue could have unpredictable results. In QLab 5, however, absolute fades supersede relative ones. In QLab 5, an absolute fade is absolutely more absolute.

Routing and Panning

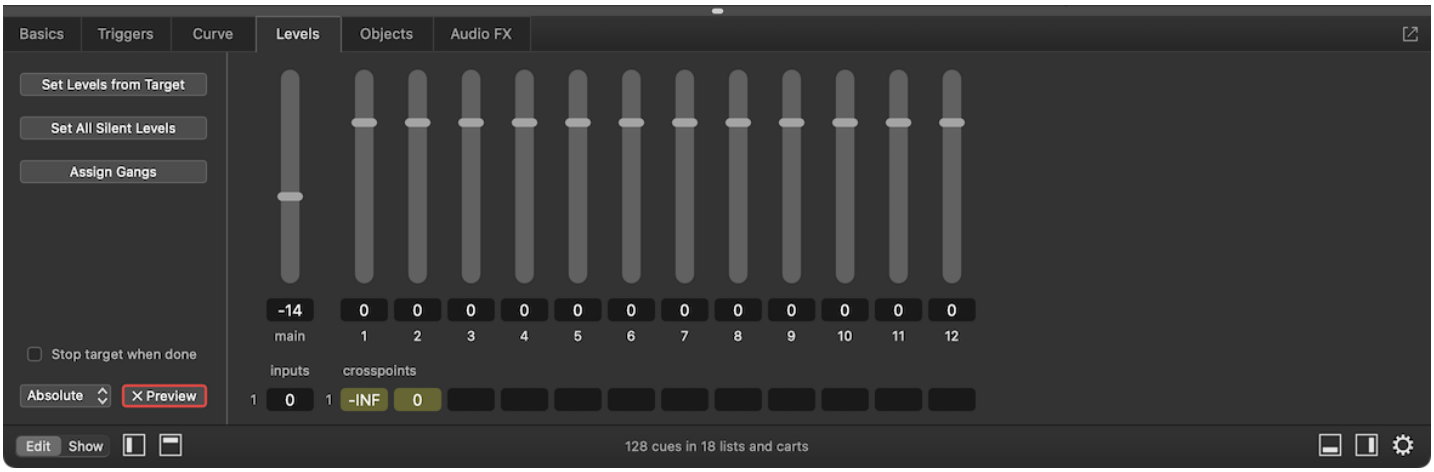
While fading in, fading out, and adjusting the overall volume of an Audio cue are probably the most common uses of a Fade cue, you can also use a Fade cue to pan an Audio cue amongst any pair or group of outputs.

The best way to understand this is to walk through a simple example. Let's say you have an Audio cue whose file target is a mono audio file, and you want to pan that audio from one speaker to another. QLab will recognize the track has one channel, and one row will appear in the Levels tab of the inspector for that Audio cue.

In the Audio cue, set the level for the cue output that's routed to the first speaker to `0`, and set the level for cue output that's routed to the second speaker to -INF. Note that QLab treats a blank field in this cue matrix mixer as -INF.



Next, create a Fade cue which targets the Audio cue, and reverse the levels: set the first output to -INF and the second output to 0.

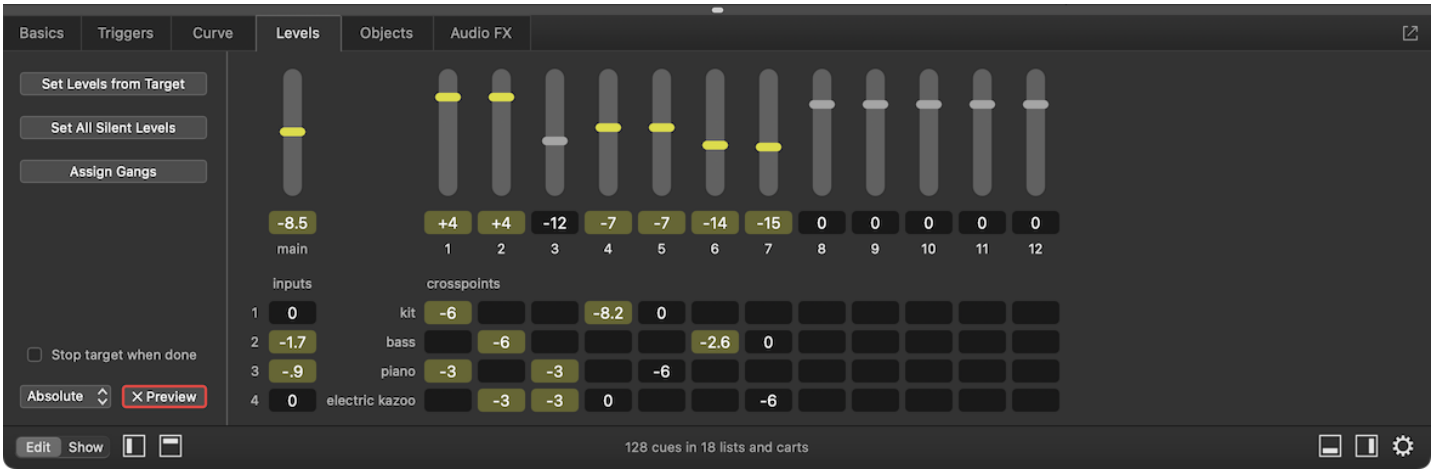


When you run the Audio cue, you will hear it exclusively through cue output 1. When you run the Fade cue, it will fade down in output 1 and fade up in output 2 over the same duration.

If you watch the Levels tab of the target Audio cue while the runs, you'll see the meters in the cue outputs (though not the controls themselves) change in realtime.



Remember too that one Fade cue can affect as many controls as you wish. If you have a track with four channels instead of just one, and you have seven speakers instead of two, you can enter values in the matrix or adjust the sliders to have different channels come up or down in different outputs.



A key concept here is that if multiple Fade cues share a target, but each Fade cue has different active controls, then those Fade cues can run simultaneously without interfering with each other. Because of this, in the scenario with a four-channel Audio cue fading amongst seven speakers, very complex fades can be achieved by using simultaneous Fade cues with different active controls, different curve shapes, and different durations.

Reverting Fades

QLab has a sort of special-case *undo* command that applies only to Fade cues, called *Revert Fade Action*. You can find this command under the **Tools** menu when a Fade cue is selected, or you can use the keyboard shortcut ⇧⌘**R**.

When *Revert Fade Action* is invoked on a Fade cue after that Fade cue has been run, QLab reverts the levels of that Fade cue's target to whatever they were before the Fade ran *except* for levels which have been otherwise changed. That is to say, the only adjustments that are reverted are the ones that the selected Fade cue caused.

Broken Fade Cues

Fade cues can become broken for the following reasons:

Missing cue target

Assign a target cue to this cue.

No parameters set to fade

Set this cue to adjust at least one parameter. You can enable or disable an audio parameter by clicking in it; active controls will be highlighted in yellow. You can enable or disable a video parameter by checking or unchecking the box next to it.

Missing cue audio effect

Either install the missing audio effect and restart QLab, or remove the missing effect from the Audio FX tab of the inspector of the target cue.

License required

An audio or video license is required to fade audio objects, playback rate, audio effects, or patches, or use Timecode triggers. A video license is required to fade video geometry or video effects. Install the appropriate type of license or adjust the cue to avoid using licensed features to clear this warning.

Chapter 6: Video

6.1 Intro to Video
6.2 Video Cues
6.3 Camera Cues
6.4 Text Cues
6.5 Video Output
6.6 Using NDI
6.7 Fading Video & Video Effects

Intro to Video

QLab 5's video system, built on Apple's Metal framework, gives you a powerful, flexible system for cueing, routing, and displaying still and moving images. The input side gives you precise positioning and timing of both prerecorded and live imagery, while the output side gives you sophisticated pixel-perfect control over how your video is displayed on monitors, televisions, projectors, LED walls, Syphon servers, and NDI feeds.

A note on style

> This manual follows the nomenclature used in professional theater in the US, where "projection" is taken to mean any form of video or film used in a live theatrical performance. Whether the imagery is moving or still, digital or analogue, displayed via a projector, a TV, an LED wall, or a projection-enabled intelligent light, it's all grouped together under the name "projection design." In QLab, Video, Camera, and Text cues are the types of cues that deal with projection. "Screen" means any physical device that displays the contents of these cues. The words "screen" and "display" are used more or less interchangeably.

Sources Of Video

Video cues play back prerecorded video or still images stored on your Mac.

Camera cues display live video from webcams, Blackmagic Design video capture devices, Syphon servers, and NDI streams.

Text cues render styled text as still images using fonts installed on your Mac.

Output Of Video

QLab's video output patches are called *stages*. Cues are assigned to stages, and then stages are rendered onto one or more actual video output devices, with a few intermediary steps in between. You may not need to use every feature that's available in every one of these steps, but taken together they afford the maximum possible flexibility.

The signal path for video goes like this:

1. **Cues** receive a source of video and display it on a **stage.**
2. **Stages** are divided into **regions**, which cover some or all of the stage.
3. Each **region** is assigned to exactly one **output route.**
4. Each **output route** represents exactly one **output device.**
5. **Output devices** are connections to physical video displays like monitors and projectors, Syphon outputs, and NDI outputs.

QLab can use any or all of the following as output devices:

- Any display that appears in the *Displays* section of *System Preferences*.
- Outputs of Blackmagic Design devices.
- Any number of Syphon outputs[1].
- Any number of NDI outputs[1].

While it's possible that USB DisplayLink displays will also work with QLab, we do not recommend nor do we support using them because of their many unpredictable variables. Likewise, AirPlay is not supported in QLab due to its variable latency.

Most Macs these days use USB-C connections for video, a thorough discussion of which can be found in the USB-C tutorial in this manual.

Sizes and Shapes

Many media servers have ties to the cinema or broadcast worlds, in which the size and shape of a video signal conforms to an exact standard. You may be familiar with terms like “standard definition”, “1080p”, or “4K”; these are all terms for video standards that have specific resolutions, aspect ratios, frame rates, and other attributes.

QLab is completely agnostic when it comes to these standards. 🎞 Video cues can play any compatible media file onto any stage, without the need to preemptively match resolution or frame rate.

When you create or edit a stage, you can set its width and height to suit your exact needs[2]. If the stage you create is larger than the size of the imagery you project onto it, the surrounding area will be filled with black pixels. If the stage is smaller than the imagery, the imagery will simply extend off the “canvas” of the stage. Individual cues can be scaled up or down to fit on any surface.

Frame rate is another question that you pretty much don’t need to worry about in QLab. Cues’ sources can have any frame rate; they do not need to match the frame rate of the output device or devices in use. You can even display multiple cues on the same stage at the same time with different frame rates. While it’s impossible to guarantee that there will never be visible artifacts if you have extreme mismatches between frame rates, generally speaking the Mac does a very good job of invisibly handling the necessary computation.

Take Your Time

As has been said in other parts of this manual, the secret to success with projection design is *time*. Give yourself time to experiment, time to troubleshoot, and time to learn the powers and limitations of your Mac.

---

1. Limited by your computer’s resources.↩
2. Up to a maximum of 16384 × 16384 pixels.↩

Video Cues

🎞 Video cues allow you to play video and still images files with precise control over timing, opacity, scale, position on screen, and 3D rotation, and with a full suite of effects and blending tools. Video cues must have a file target, which is a video or image file on your computer, and must be assigned to a stage, which connects QLab to a video output destination such as a projector, screen, LED wall, or a virtual output device such as Syphon or NDI. When a 🎞 Video cue has a still image as a target, it will run indefinitely (until they are stopped, of course) unless you give it a finite duration.

Video Files

Unlike audio files, video files have two attributes that determine whether they are compatible with QLab. The first attribute is the *codec* used by the file. Codec is short for either "compressor/decompressor" or "code/decode" and it describes the way that video data is *encoded* in the file. The second attribute is the *container format* used by the file. The container is like an envelope which *contains* the video, its audio tracks, and metadata about the media.

The following **codecs** are compatible with QLab 5 for moving images:

- MPEG-1
- MPEG-2
- MPEG-4
- H.263
- H.264
- H.265, also called HEVC
- ProRes 422 Proxy
- ProRes 422 LT
- ProRes 422
- ProRes 422 HQ
- ProRes 4444
- ProRes 4444 XQ
- Photo-JPEG
- DV/DVCPRO NTSC
- DVCPRO50 NTSC
- DV PAL
- DVCPRO PAL
- DVCPRO50 PAL
- Hap Standard
- Hap Alpha
- Hap Q

Recommended codecs

While all those codecs technically work with the underlying video frameworks that power QLab 5, the following codecs give the *best performance* in QLab for moving images without transparency (listed in preferential order):

1. ProRes 422 Proxy
2. Hap Standard
3. ProRes 422 LT
4. Hap Q
5. Photo-JPEG

If you require better picture quality, you can also use ProRes 422 or ProRes 422 HQ. These codecs provide sharper images at the cost of greater processing power, and require exceptionally good video output equipment and viewing conditions to be worthwhile.

For moving images with transparency, we recommend one of the following codecs (listed in preferential order):

1. ProRes 4444
2. Hap Alpha

If you require better picture quality with transparency, you can use ProRes 4444 XQ. The same disclaimer applies; ProRes 4444 HQ provides superlative picture quality, requires dramatically more processing power, and is not worth it unless you have absolutely top-quality equipment and viewing conditions.

Videos with transparency (alpha channel)

Videos with transparency, also called alpha channel support, must be rendered with premultiplied alpha to work properly in QLab. Premultiplying alpha is a rendering option in all video editing and compositing software, and is what allows partially transparent areas to accurately composite within QLab.

We strongly recommend against using Adobe Media Encoder, because it does not reliably produce files that play nicely with QLab. If you edit in Adobe Premiere or After Effects, use the built-in export tools from those programs. In After Effects, the composition's background color must be set to black for premultiplied alpha output to look correct.

Apple's Compressor is the best tool for transcoding pre-existing videos.

Other codecs

The following codecs generally work well, but perform especially poorly when sped up, slowed down, or when scrubbing forward or back:

1. H.265
2. H.264

All other compatible video formats are not recommended, even though they may work, or appear to work, with QLab 5.

Containers

The following **containers** are recommended for use with QLab 5 for moving images:

- MOV
- MP4

Other container formats may work, but are not recommended because they are more likely to contain incompatible codecs or other data.

Still image formats

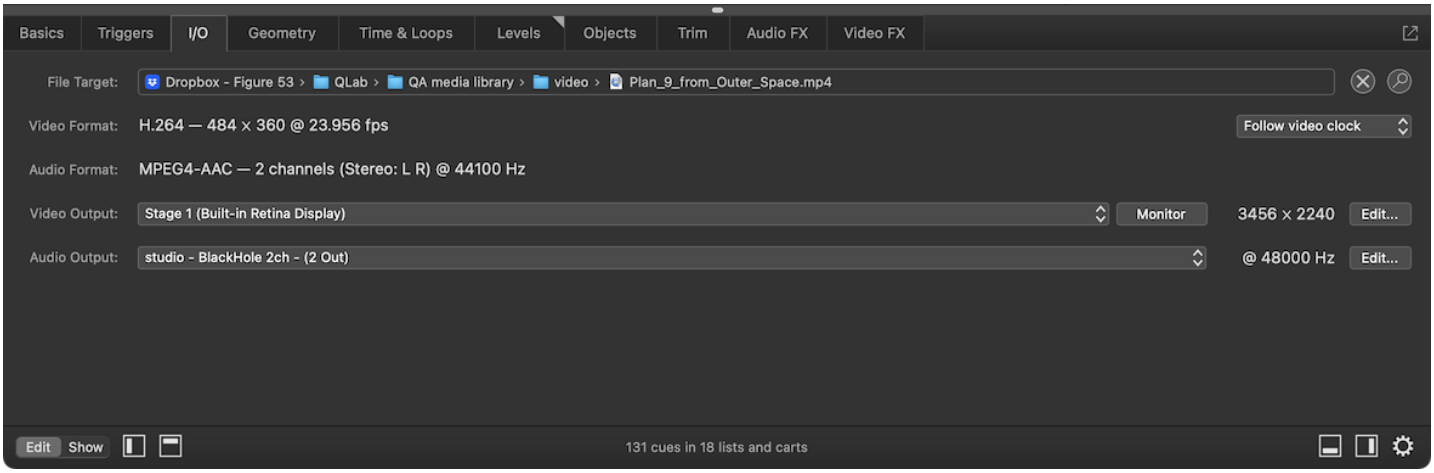
Most still image formats work with QLab 5, but we recommend PNG and JPG files. QLab 5 does not support PSD or PDF formats.

The Inspector for Video Cues

When a 🎞 Video cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as the following tabs:

The I/O Tab

The I/O tab lets you assign a target video file and video output, and view details about them as well.



File Target. If a target video file is assigned, this field shows the path to that file. You can double-click on the field to select a file target, or you can drag and drop a file in from the Finder.

Clicking the ⊗ button removes the file target from the cue (though it does not delete the file itself, of course.)

Clicking the 🔍 button reveals the file target in the Finder.

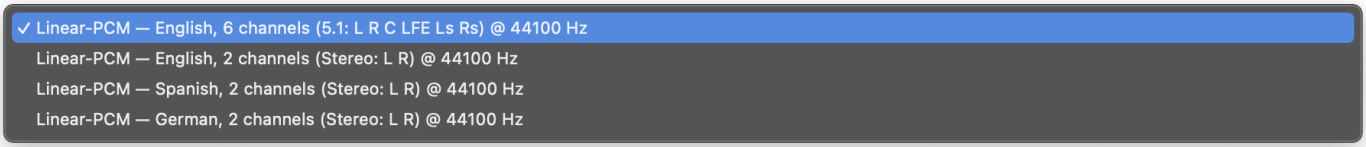
These controls are also visible in the Basics tab; you can use either one or both interchangeably.

Video Format displays the type of codec, pixel dimensions, and frame rate of the video in the file target.

Clock. This pop-up menu allows you to select the clock that will be used to set the timing for the playback of this cue.

- **Follow video clock** prioritizes the timing and smoothness of video playback by using the display clock to control playback timing.[1] This generally results in the best-looking playback, but can possibly reduce the accuracy with which this cue synchronizes with any other cues which don't use the same display clock. Notably, if this cue has an audio track and uses the video clock for timing, it is not guaranteed that it will remain in sync with other cues using the same audio output patch.
- **Follow audio clock** prioritizes the timing and smoothness of audio playback by using the clock of the audio device used by the audio output patch that the cue is using. This causes the cue to stay in perfect synchronization with other cues that use the same audio output patch, but can result in occasional visible timing imperfections which usually look like a single skipped or repeated frame.

Audio Format displays the codec, number of channels, and sample rate of the audio encoded within the video file. If you are using a video file with multiple video tracks, a pop-up menu appears here allowing you to select which audio track you'd like to use.



Each item in the menu displays the codec, number of channels, channel layout[2], and sample rate of the audio encoded in each track.

Video Output. This pop-up menu allows you to assign the cue to a **stage**, which is QLab's video output mechanism. Every 🎞 Video cue must be assigned to exactly one stage, and the way that the stage is configured defines how the imagery will ultimately be displayed. You can learn more about stages from the Video Output section of this manual. Clicking on the menu allows you to select one of the stages already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Video Settings to edit stage list...* to quickly get to Workspace Settings → Video → Video Outputs, or choose *New stage with video route* to quickly generate a new stage and select it for use.

The **Monitor** button opens the monitor window for the selected stage.

To the right of that button, QLab displays the pixel dimensions of the selected stage. QLab always displays the actual pixel dimensions, not the effective point size, so for Retina displays, these numbers may be considerably higher than the resolution reported in System Preferences.

To the right of that, the **Edit...** button opens the video stage editor for the currently selected stage.

Audio Output. This pop-up menu allows you to select an audio output patch for the cue to use. Clicking on the menu allows you to select one of the audio output patches already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Audio Settings to edit patch list...* to quickly get to Workspace Settings → Audio → Audio Outputs, or choose *New patch with audio device* to quickly generate a new audio output patch and select it for use.

To the right of that menu, QLab displays the sample rate of the audio device associated with the audio output patch.

To the right of that, the **Edit...** button opens the audio output patch editor for the currently selected audio output patch.

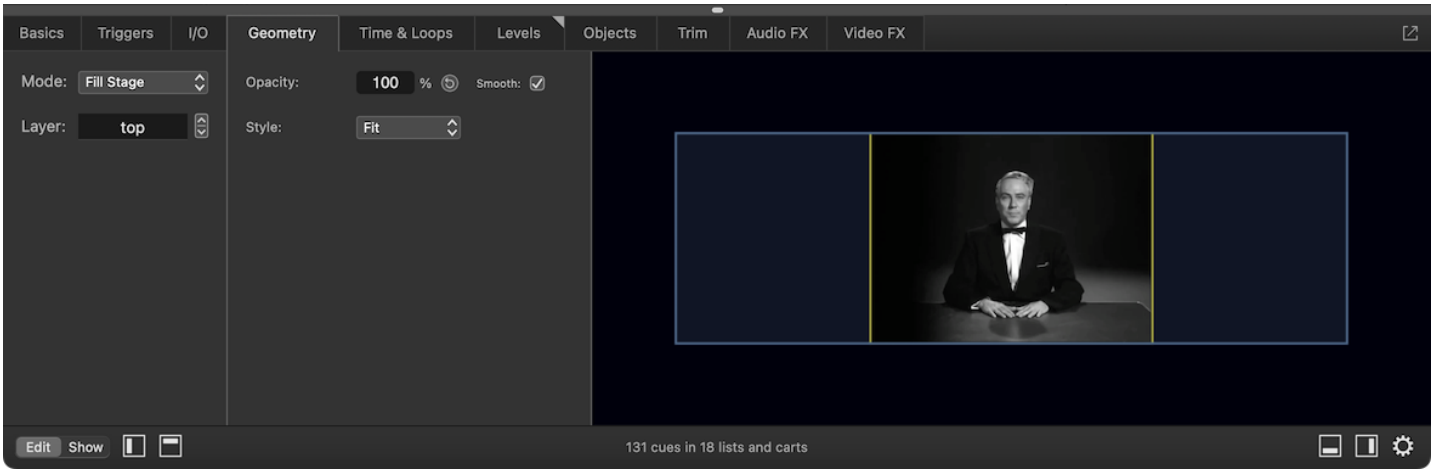
The Geometry Tab

The Geometry tab lets you adjust the basic visible parameters of the cue. There are two forms the Geometry tab can take, depending upon the *mode* of the cue.

Fill Stage mode

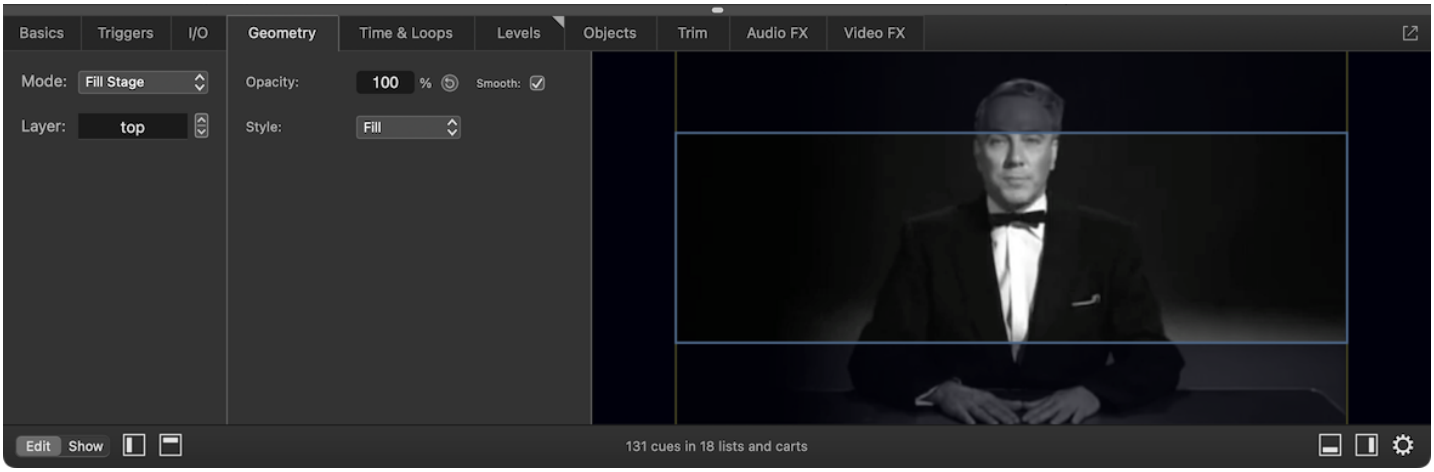
If the **Mode** pop-up menu is set to *Fill Stage*, the Geometry tab looks like this. Cues set to *Fill Stage* can be displayed in one of three styles, which can be set using the **Style** pop-up menu just to the right and below the **Mode** pop-up menu. The three styles are *Fit*, *Fill*, and *Stretch*.

Fit



A cue using the **Fit** style will be symmetrically scaled as large as possible to display the entire image without changing the aspect ratio of the cue's file target. If the aspect ratio of the cue's file target is different than the aspect ratio of the stage, there will be blank space to the sides or above and below. QLab does not fill this space with black pixels; if there is a cue on a lower layer that truly fills the stage, it will be seen in these empty spaces.

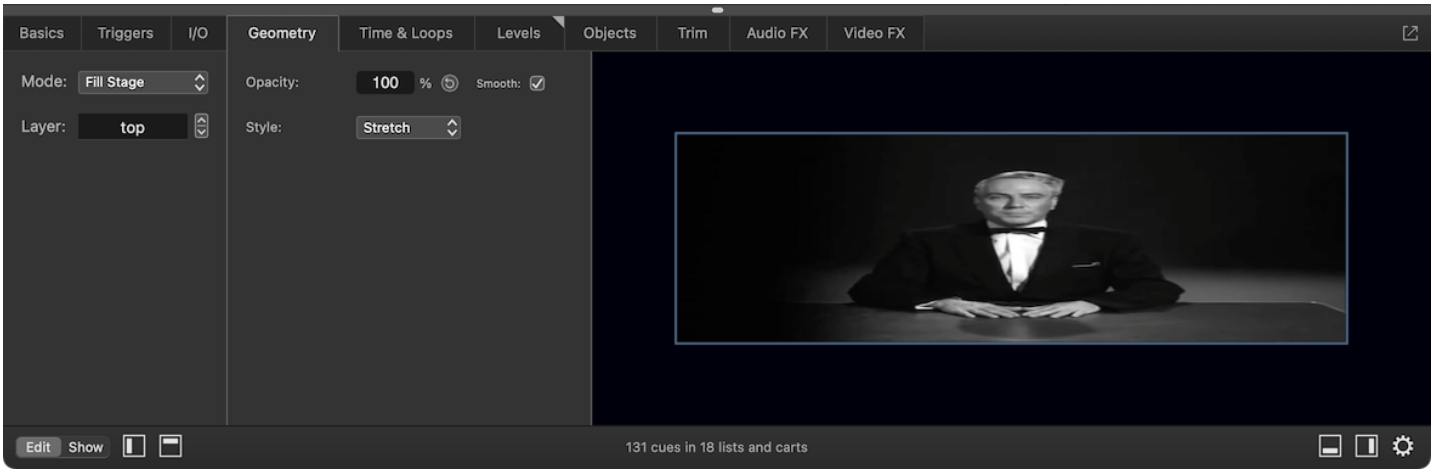
Fill



A cue using the **Fill** style will be symmetrically scaled to fill every pixel of the stage without changing the aspect ratio of the cue's file target. If the aspect ratio of the cue's file target is different than the aspect ratio of the stage, either the sides or the top and bottom of the image will fall outside the bounds

of the stage, and will therefore not be displayed.

Stretch



A cue using the **Stretch** style will be asymmetrically scaled to fill every pixel of the stage. If the aspect ratio of the cue's file target is different than the aspect ratio of the stage, the image will be stretched either horizontally or vertically.

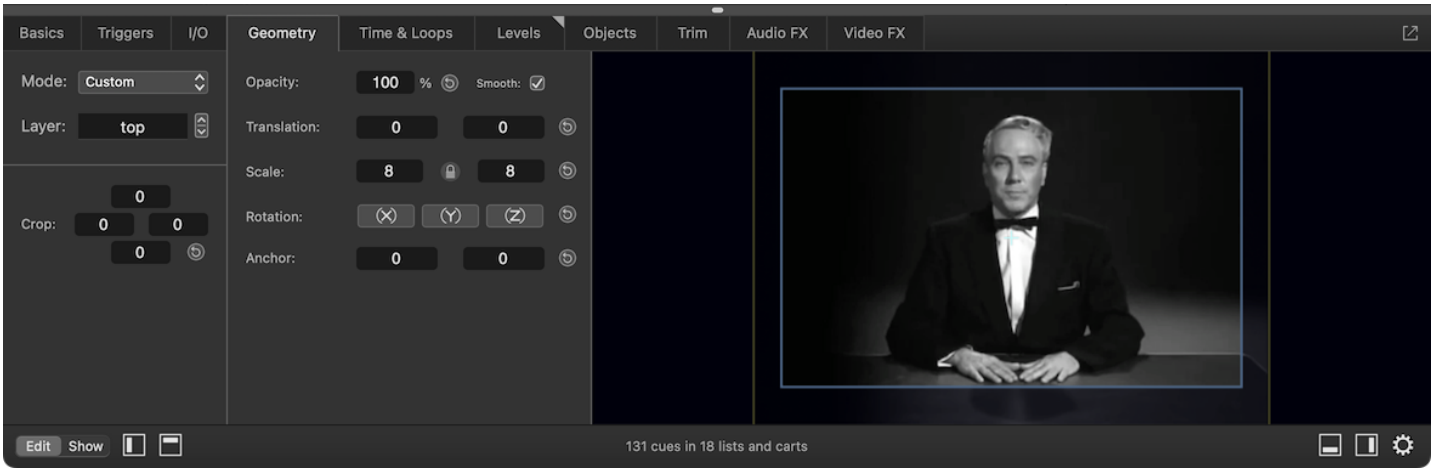
Opacity sets the opacity of the cue on a scale of 0%, meaning fully transparent, to 100%, meaning fully opaque. Transparency in the target media is not altered by this control; if a cue's target has transparent pixels, they remain transparent when the cue is set to 100% opacity. The ⟲ button resets the cue's opacity to its default value.

The **Smooth** checkbox determines whether the cue will be drawn using smooth anti-aliased scaling or nearest-neighbor scaling. Leaving the box checked will smooth out the jagged edges that can appear when images are scaled up. Unchecking the box will keep the sharp edges, which can sometimes be preferable depending upon your aesthetic goals.

Layer sets the stacking order of the cue relative to other cues assigned to the same stage.

QLab has `1001` layers: `top` is the layer closest to the viewer. Layer `999` is directly beneath, then layers continue downwards to `1`, then `bottom`. Any cues assigned to the `top` layer will play on top of all other currently running cues. Any cues assigned to the `bottom` layer will play beneath all other currently running cues. Cues assigned to the same layer as each other will stack in the order in which they are started. So, for example, if you have three cues assigned to layer `10`, whichever cue is started last will appear on top of the other two, but any cue assigned to layer `11` (or higher) will appear on top of all three cues on layer `10`, regardless of the order in which those cues are started. Similarly, any cue on layer `9` (or lower) will appear beneath all three of the cues on layer `10`.

Custom mode



If the **Mode** pop-up menu is set to *Custom*, the Geometry tab looks like this. Cues display at their natural pixel dimensions, and several more controls appear to allow you to customize the appearance of the cue. To the right, a thumbnail of the cue appears within a blue box. The blue box represents the stage that the cue is assigned to.

Crop. The four text fields of the crop parameter allow you to trim or shutter in from the four sides of the cue. Each text field represents one edge (top, bottom, left, right), and the number is the number of pixels to crop off counting from the edge towards the center of the cue. The ↺ button resets the cue's crop to its default value. *Note: the crop parameter is not fade-able. If you want to fade the crop of a cue, use the Shutter or Window video effect instead.*

Translation is the position of the cue on the stage, measured in pixels relative to the center of the surface. A translation of `(0, 0)` is centered, negative values are down and to the left, and positive values are up and to the right. You can also adjust the translation by clicking and dragging the thumbnail of the cue. The ↺ button resets the cue's translation to its default value.

Scale is the size of the cue relative to its natural pixel dimensions, expressed as a multiplier. `1.0` represents the natural size, `0.5` is half the natural size, `2.0` is double, and so on. If the 🔒 lock icon between the scale fields is closed, the aspect ratio of the video will be preserved while scaling. You can click the lock to 🔓 unlock it and adjust the height and width independently. You can also adjust scale by vertically scrolling on the thumbnail of the cue using your mouse, trackpad, or other pointing device. The ↺ button resets the cue's scale to its default value.

Rotation is the attitude of the cue in a virtual 3D space. You can click and drag on the **(X)**, **(Y)**, and **(Z)** buttons to manipulate the cue's rotation around the desired axis, or click on each and type values into the text field that appears. The ↺ button resets the cue's rotation to its default value.

> **A Word About Quaternions:** QLab uses quaternion math to handle rotation of 🎞 Video cues in 3D space. The advantage to quaternions is that when fading a cue from one position to another, QLab will always produce smooth, natural motion. The tradeoff is that there is no useful way to numerically display the current rotational position of a cue. Therefore, when you adjust the rotation of a cue, you'll see numbers in the pop-up which represent the degrees of single-axis rotation since you started this particular adjustment, not any sort of absolute measure.

Anchor is the point around which a 🎞 Video cue rotates, translates, and scales. The default anchor point is `(0, 0)`, which is the center of the cue. Adjusting the anchor will change the way the cue moves when you adjust other geometry parameters. You can also adjust the anchor by clicking and dragging the light blue cross in the thumbnail of the cue. The ↺ button resets the cue's anchor to its default value.

The Time & Loops Tab

The Time & Loops tab behaves the same as the Time & Loops tab for Audio cues, with one small but important additional control.

Underneath the waveform view, towards the right side of the window, is a checkbox labeled *Hold at end*. If this box is checked, when the 🎞 Video cue plays and reaches its final frame, that final frame will remain visible and the cue will remain active until stopped.

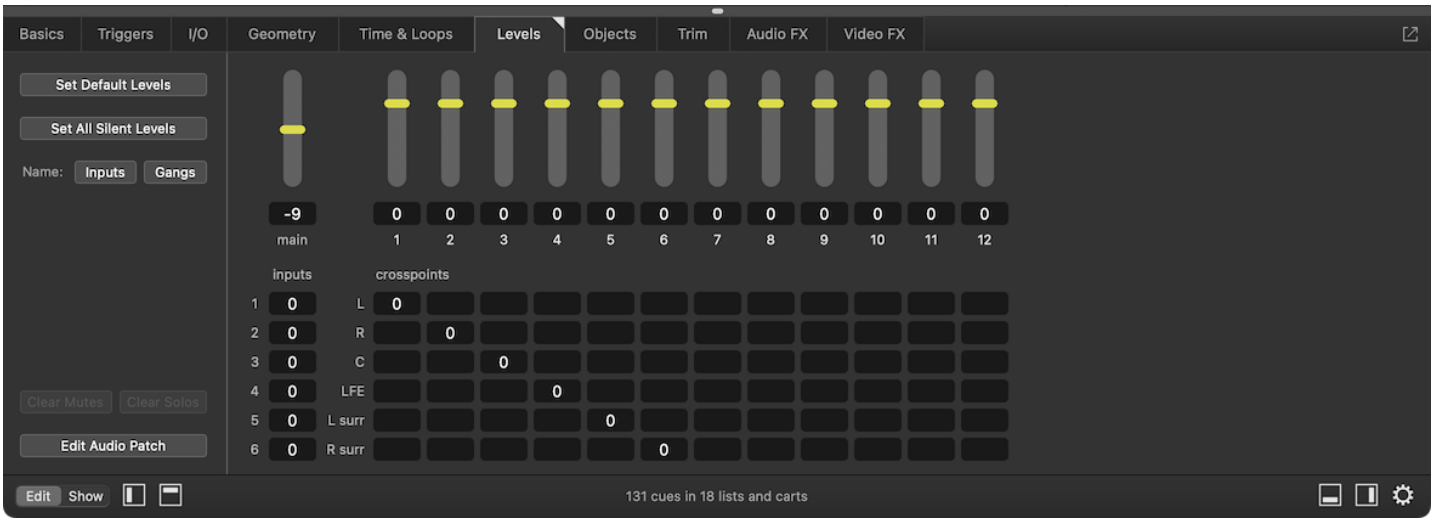
If the *Hold at end* box is not checked, the cue will stop itself after it reaches its last frame.

The Levels Tab

The Levels tab only appears if the file target of the cue contains an audio track. If so, this tab behaves the same as the Levels tab for Audio cues.

If the target video file contains channel layout metadata, that metadata will be used to name the input channels in the mixer. For example, the video file targeted by the cue in the screen shot below has a 5.1 surround track, with channels for left, right, center, low frequency element (subwoofer), left surround, and right surround.

These names can be edited using the **Inputs** button. Editing the names will only change the names inside QLab; the metadata in the video file will not be changed.



The Objects Tab

The Objects tab only appears if the file target of the cue contains an audio track. If so, this tab behaves the same as the Objects tab for Audio cues.

The Trim Tab

The Trim tab only appears if the file target of the cue contains an audio track. If so, this tab behaves the same as the Trim tab for Audio cues.

The Audio FX Tab

The Audio FX tab only appears if the file target of the cue contains an audio track. If so, this tab behaves the same as the Audio FX tab for Audio cues.

The Video FX Tab

The Video FX tab lets you set the blend mode of the cue and add live video effects to the cue.

Blend Modes

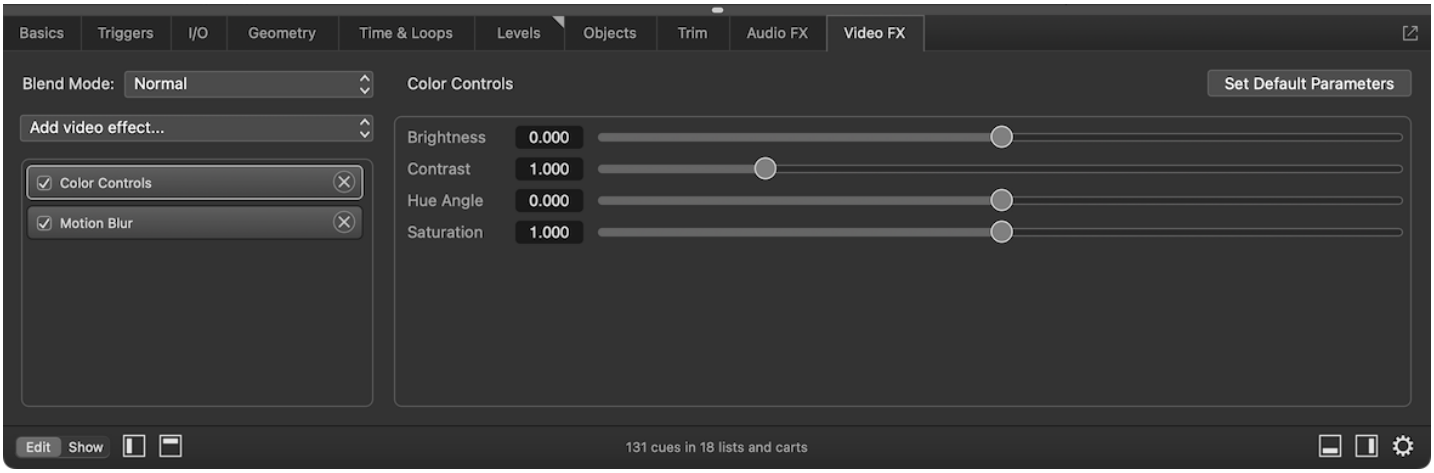
The blend mode of a 🎞 Video cue, selected using the **Blend Mode** pop-up menu, dictates the way that the cue interacts and combines with other 🎞 Video cues that occupy the same space on the stage. Blend modes are best understood as a per-pixel mathematical operation which combines the values of each channel of a given pixel in the "foreground", which is a cue on a higher layer, with the values from the pixel in the same position in the "background", or a cue on a lower layer. In this context, channel values are represented on a scale from `0.0` to `1.0` in which `0.0` is off, and `1.0` is full brightness. A pure red pixel would be represented as `(1.0, 0.0, 0,0)`. A pure white pixel would be `(1.0, 1.0, 1.0)`.

> The Blend Mode page of this manual contains descriptions and reference videos demonstrating the effects of the various blend modes available in QLab 5. The Blend Mode Demo tutorial is a downloadable, hands-on copy of the workspace which was used to create those reference videos.

Video Effects

The **Add video effect...** pop-up menu allows you to add a video effect to the cue. QLab 5 supports multiple video effects on a single cue, and effects are rendered in the order in which they appear in the list. Video effects are "live", meaning that they are rendered on the spot, in realtime.

Please note that video effects can be extremely processor-intensive and can cause visible performance problems, sometimes even on powerful Macs. Please take time to experiment with video effects and learn their idiosyncrasies before using them in a production environment. That said, the underpinnings of the QLab 5 video system affords very good video effects performance, particularly on Apple Silicon-based Macs.



When a video effect is selected, it is highlighted with a light grey border and its controls become visible on the right side of the inspector. Each video effect has its own parameters which can be adjusted either by typing into their text fields or clicking and dragging their sliders. Some video effect parameters use color swatch selectors which require a click to start editing.

Clicking **Set Default Parameters** will reset the currently visible video effect to its default state. Other video effects will not be altered.

To remove a video effect from a cue, click the ⊗ button next to its name.

A full list of available video effects and their parameters can be found in the Parameter Reference section of this manual.

Broken Video Cues

🎞 Video cues can become ❌ broken for the following reasons:

No video file selected

The cue has no target, and 🎞 Video cues require a video or image file as a target. Select an appropriate file as the target of this cue to clear this warning.

Missing video file

The cue had a file target assigned, but that file is missing. Perhaps the file was on a removable drive or network drive which is not currently connected. Re-locate the file, or assign a new file target to clear this warning.

File target in Trash

The cue's file target is in the Trash, and files in the Trash cannot be used. Either move the file target out of the Trash, or assign a new file target to clear this warning.

File target lacks read permissions

This is a bit of a bizarre error that is technically possible, but very unusual. If the file target lacks read permissions, it cannot be used. Since it cannot be used, though, it probably could not have been selected as a target in the first place. Despite this seeming paradox, it still is technically possible that this could happen, and so QLab tries to handle it. Fix the file target's permissions, or assign a new file target to clear this warning.

File target uses an incompatible codec

This is error is exceedingly unlikely, since QLab prevents you from choosing a file target which isn't compatible. Still, it is technically possible. Choose a new file target with a compatible codec to clear this warning.

File target is an incorrect size

The cue's file target has a width or height greater than 16,000 pixels. Choose a file target with both a width and height of 16,000 pixels or smaller to clear this warning.

No video output stage

The cue has no video output stage assigned. Assign the cue to a stage to clear this warning.

Incomplete video output

The cue is assigned to a stage, but the stage is incompletely configured. Visit the Video Stage Editor to complete the configuration of the stage and clear this warning.

No audio output patch

The file target has audio, but no audio output patch is assigned to the cue. Assign an audio output patch to clear this warning.

Issue with audio output patch

The cue has an audio output patch assigned, but something is wrong with it. The audio output patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the audio output patch or select a new audio output patch to clear this warning.

NDI output lacks audio channels

The cue is assigned to an audio output patch which uses an NDI device, but that NDI device has no audio channels. To add audio channels to the NDI device, go to Workspace Settings → Video → Output Devices, click the **Edit** button next to the NDI device you're using, and adjust the number of audio channels included in the NDI output. Alternately, assign a different audio output patch.

Missing cue audio effect

The cue has an audio effect assigned, but that AudioUnit is not installed. Either remove the audio effect from the cue or install the missing AudioUnit to clear this warning.

Invalid slice play counts

The cue is sliced, and all slices have a play count of `0`. This means that no part of the file target will be played, which means the cue effectively does nothing. Either set at least one slice to a play count greater than zero, or delete this cue that does nothing anyway to clear this warning.

License required

A video license is required to use custom geometry, blend modes, video effects, audio effects, or Timecode triggers. Install a video license or adjust the cue to avoid using licensed features to clear this warning.

---

1. When a Video cue is assigned to a stage that goes to only one output, QLab will use the clock associated with that output. When a Video cue is assigned to a stage that goes to more than one output, QLab follows a slightly complex heuristic: if one output has a higher refresh rate than all the others, QLab will use that output's clock. If two or more outputs are tied for highest refresh rate, QLab will prioritize clocks belonging to physical outputs (screens and projectors) over virtual outputs (NDI and Syphon.) If two or more outputs are still tied for best candidate, QLab picks one effectively at random.↩
2. The channel layout is provided by metadata within the video file, and may not always be available.↩

Camera Cues

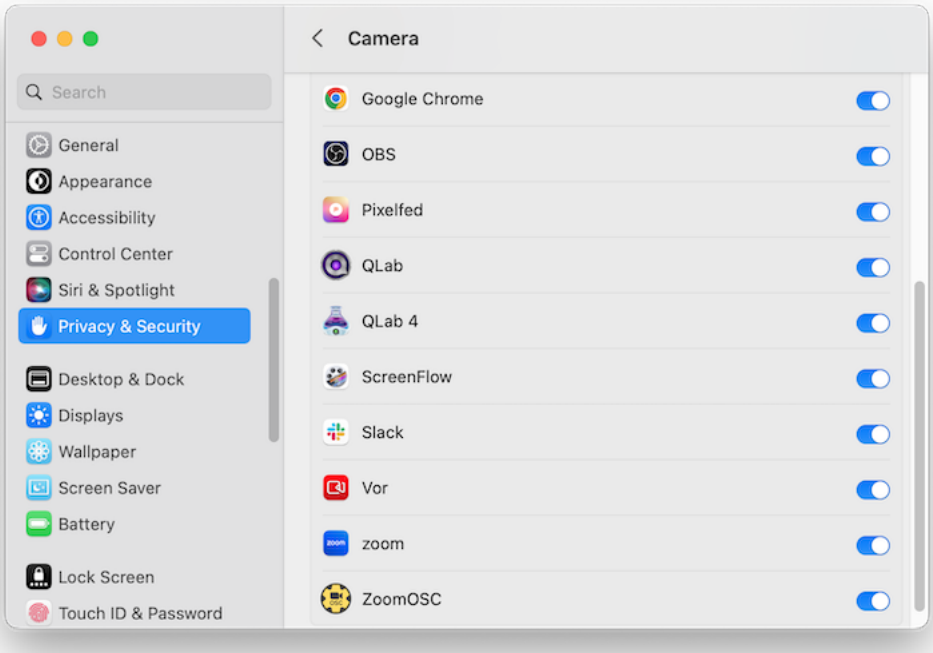
Camera cues allow you to display live video through QLab using the same cueing, routing, and video effects that are available to Video cues. New to QLab 5, every Camera cue contains an embedded Mic cue which allows you to play live audio along with the live video. The source of the live audio can either be the same device that's providing the live video, or a different one. Camera cues run indefinitely (until they are stopped, of course) unless you give them a finite duration.

Camera cues can use any of the following as sources of video:

- Any USB video device class camera or input device. This includes most webcams and any video capture device that does not require its own drivers or software to be installed.
- Any IIDC-compliant camera or input device. These are typically referred to as DV devices[^1], and are FireWire-based which means they are rather passé and hard to come by.
- Any Blackmagic Design DeckLink, UltraStudio, or Intensity device.
- Any Syphon source on your Mac.
- Any NDI source available to your Mac via its local network.

macOS Security

macOS requires you to proactively grant permission to each program that wants to use a camera or any video input device in order to limit sneakiness and make it harder for programs to spy on you. You will need to grant QLab permission for camera access in order to use Camera cues. If you're not automatically prompted to do this the first time you try, or if you need to change QLab's permission, you can do that by visiting *System Settings → Privacy & Security → Camera*, finding QLab in the list, and flipping the switch on or off.

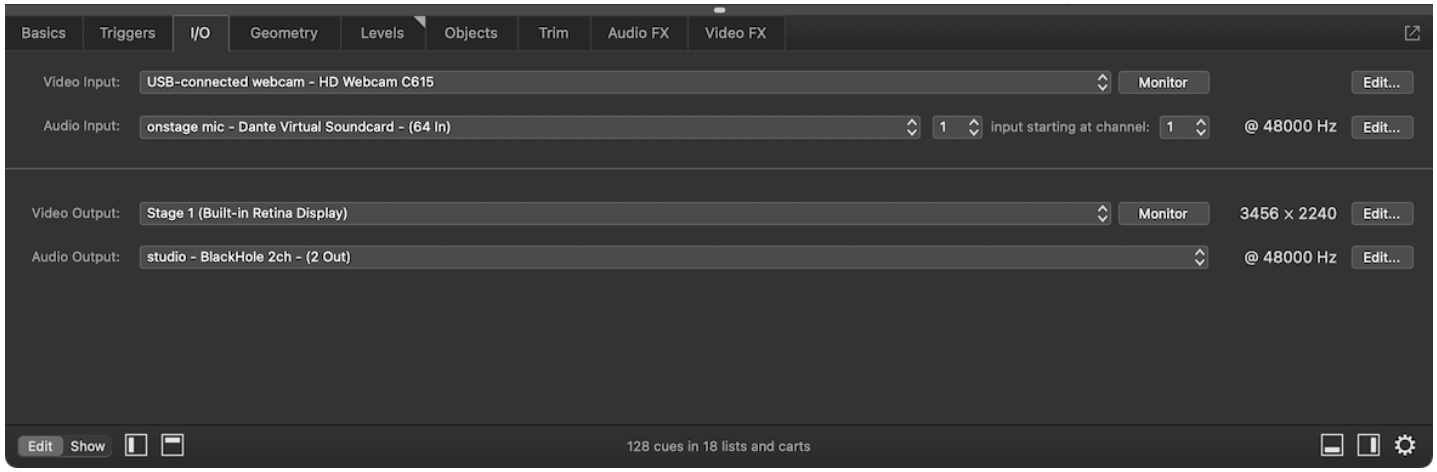


The Inspector for Camera Cues

When a Camera cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as the following tabs:

The I/O Tab

The I/O tab lets you configure the cue's source and destination for both video and audio, as well as the number of audio channels used by the cue.



Video Input. This pop-up menu allows you to select a video input patch for the cue to use as its source of video. Clicking on the menu allows you to select one of the video input patches already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Camera Settings to edit patch list...* to quickly get to Workspace Settings → Video → Video Inputs, or choose *New patch with camera device* to quickly generate a new video input patch and select it for use.

The **Monitor** button opens the monitor window for the selected video input patch, showing you an immediate live view of the input from that patch.

The **Edit...** button opens Workspace Settings → Video → Video Inputs.

Audio Input. This pop-up menu allows you to select an audio input patch for the cue to use as the source of audio in the cue. Clicking on the menu allows you to select one of the audio input patches already configured in the workspace, or *(unpatched)* if you do not want the cue to handle audio. You can also choose *Open Audio Settings to edit patch list...* to quickly get to Workspace Settings → Audio → Audio Inputs, or choose *New patch with audio device* to quickly generate a new audio input patch and select it for use.

If the selected video input patch uses an NDI source, QLab automatically uses the audio from the NDI source as the audio input for the cue. This input comes directly from NDI and does not use an audio input patch. Camera cues which use NDI sources for video cannot be configured to use other sources of audio.

***x* input(s) starting at channel *y*.** This pair of pop-up menus allow you to configure the audio input channels of the cue. The first menu sets the number of channels, which can be anywhere from 1 to the number of input channels available in the audio input patch, up to a maximum of 24. In the screen shot above, you can see that the audio input patch uses a device with only one inputs, so for this cue, the number of input channels can only be 1. The second menu sets the number of the first input that the cue should use, which can be anywhere from 1 to the number of input channels available.

To the right of those menus, QLab displays the sample rate of the audio device used by the selected audio input patch. Note that for virtual audio devices such as Existential Audio's BlackHole or the venerable Soundflower, the sample rate may not display accurately at all times, or even at all. Virtual devices are not always able to provide this information; it's not necessarily a sign of a problem.

The **Edit...** button opens Workspace Settings → Audio → Audio Inputs.

Video Output. This pop-up menu allows you to assign the cue to a **stage**, which is QLab's video output mechanism. Every Camera cue must be assigned to exactly one stage, and the way that the stage is configured defines how the imagery will ultimately be displayed. You can learn more about stages from the Video Output section of this manual. Clicking on the menu allows you to select one of the stages already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Video Settings to edit stage list...* to quickly get to Workspace Settings → Video → Video Outputs, or choose *New stage with video route* to quickly generate a new stage and select it for use.

The **Monitor** button opens the monitor window for the selected stage.

To the right of that button, QLab displays the pixel dimensions of the selected stage. QLab always displays the actual pixel dimensions, not the effective point size, so for Retina displays, these numbers may be considerably higher than the resolution reported in System Preferences.

To the right of that, the **Edit...** button opens the video stage editor for the currently selected stage.

Audio Output. This pop-up menu allows you to select an audio output patch for the cue to use. Clicking on the menu allows you to select one of the audio output patches already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Audio Settings to edit patch list...* to quickly get to Workspace Settings → Audio → Audio Outputs, or choose *New patch with audio device* to quickly generate a new audio patch and select it for use.

To the right of those menus, QLab displays the sample rate of the audio device used by the selected audio input patch. Note that for virtual audio devices such as Existential Audio's BlackHole or the venerable Soundflower, the sample rate may not display accurately at all times, or even at all. Virtual devices are not always able to provide this information; it's not necessarily a sign of a problem.

The **Edit...** button opens the audio output patch editor for the currently selected audio output patch, for quick access.

Using different devices for audio input and audio output is possible. You can read more about the technical considerations related to this in this part of the section on Mic cues in this manual.

The Geometry Tab

The Geometry tab behaves the same as the Geometry tab for Video cues.

The Levels Tab

The Levels tab only appears if the cue uses audio. It behaves the same as the Levels tab for Audio cues.

The Objects Tab

The Objects tab only appears if the cue uses audiol. It behaves the same as the Objects tab for Audio cues.

The Trim Tab

The Trim tab only appears if the cue uses audio. It behaves the same as the Trim tab for Audio cues.

The Audio FX Tab

The Audio FX tab only appears if the cue uses audio. It behaves the same as the Audio FX tab for Audio cues.

The Video FX Tab

The Video FX tab behaves the same as the Video FX tab for Video cues.

Broken Camera Cues

Camera cues can become broken for the following reasons:

No camera access

QLab has not been given permission to access camera inputs. To clear this warning, open System Preferences → Security & Privacy, then click on the Privacy tab and select *Camera* from the list on the left. Then, find QLab in the list on the right and check the box next to it. This step typically does not need to be repeated, but may need to be if you completely uninstall and then reinstall QLab or perform a major macOS version update on your Mac.

No video input patch

The cue has no video input patch assigned. Assign a video input patch to clear this warning.

Incomplete video input patch

The cue has a video input patch assigned, but something is wrong with it. The video input patch will have warnings of its won, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the video input patch or select a new video input patch to clear this warning.

No video output stage

The cue has no video output patch assigned. Assign a video input patch to clear this warning.

Incomplete video output patch

The cue has a video output patch assigned, but something is wrong with it. The video output patch will have warnings of its won, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the video output patch or select a new video output patch to clear this warning.

No microphone access

QLab has not been given permission to access audio inputs. To clear this warning, open System Preferences → Security & Privacy, then click on the Privacy tab and select *Microphone* from the list on the left. Then, find QLab in the list on the right and check the box next to it. This step typically does not need to be repeated, but may need to be if you completely uninstall and then reinstall QLab or perform a major macOS version update on your Mac.

Audio input patch does not have enough input channels

The cue has been configured to use more audio channels than the input device has available. Either select an audio input patch which uses a device with enough input channels, or reduce the number of channels used in the cue to clear this warning.

Issue with audio input patch

The cue has an audio input patch assigned, but something is wrong with it. The audio input patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the audio input patch or select a new audio input patch to clear this warning.

No audio output patch

The cue has no audio output patch assigned. Assign an audio output patch to clear this warning.

Issue with audio output patch

The cue has an audio output patch assigned, but something is wrong with it. The audio output patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the audio output patch or select a new audio output patch to clear this warning.

NDI output lacks audio channels

The cue is assigned to an audio output patch which uses an NDI device, but that NDI device has no audio channels. To add audio channels to the NDI device, go to Workspace Settings → Video → Output Devices, click the **Edit** button next to the NDI device you're using, and adjust the number of audio channels included in the NDI output. Alternately, assign a different audio output patch.

Missing cue audio effect

The cue has an audio effect assigned, but that AudioUnit is not installed. Either remove the audio effect from the cue or install the missing AudioUnit to clear this warning.

License required

A video license is required to use Camera cues. Install a video license or remove the cue to clear this warning.

Text Cues

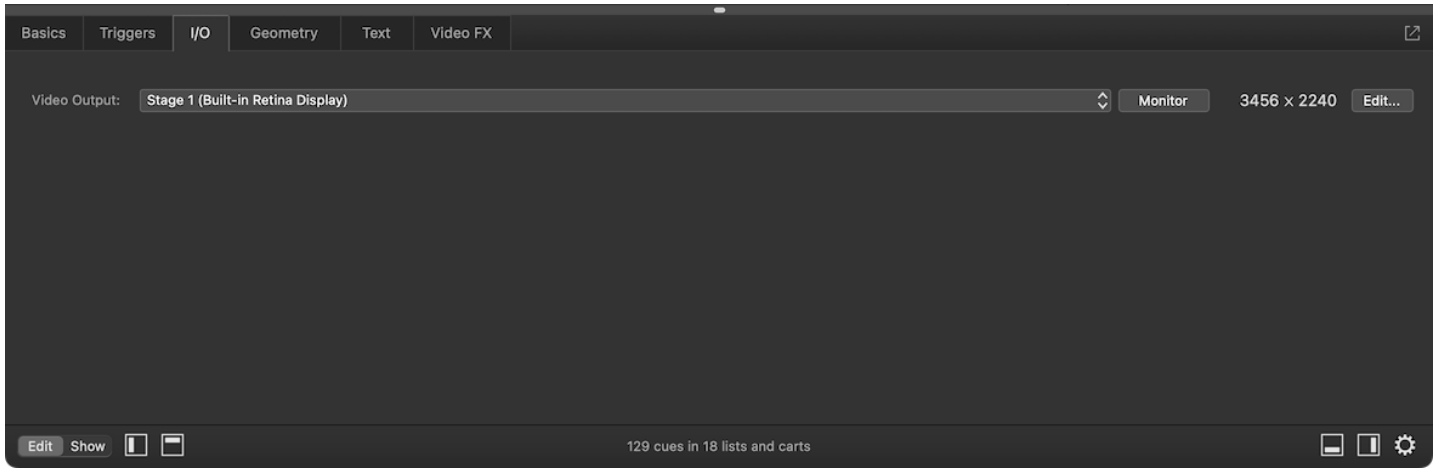
T Text cues allow you to display styled text as video output. You can adjust the font, size, style, alignment, color, and background color of the text that the T Text cue will display, and then when the cue runs your text will be rendered as video. This means that T Text cues have access to all the features of 🎞 Video cues.

The Inspector for Text Cues

When a T Text cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as the following tabs:

The I/O Tab

The I/O tab lets you assign a video output and view details about the selected output.



Video Output. This pop-up menu allows you to assign the cue to a **stage**, which is QLab's video output mechanism. Every 🎞 Video cue must be assigned to exactly one stage, and the way that the stage is configured defines how the imagery will ultimately be displayed. You can learn more about stages from the Video Output section of this manual. Clicking on the menu allows you to select one of the stages already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Video Settings to edit stage list...* to quickly get to Workspace Settings → Video → Video Outputs, or choose *New stage with video route* to quickly generate a new stage and select it for use.

The **Monitor** button opens the monitor window for the selected stage.

To the right of that button, QLab displays the pixel dimensions of the selected stage. QLab always displays the actual pixel dimensions, not the effective point size, so for Retina displays, these numbers may be considerably higher than the resolution reported in System Preferences.

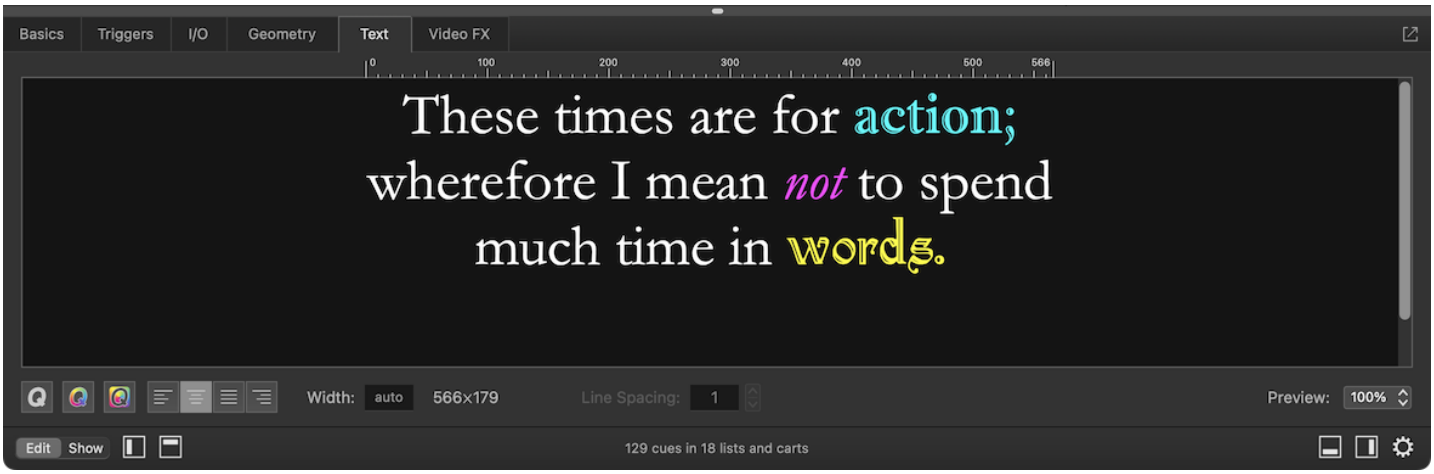
To the right of that, the **Edit...** button opens the video stage editor for the currently selected stage.

The Geometry Tab

The Geometry tab behaves the same as the Geometry tab for Video cues.

The Text Tab

The Text tab allows you to edit and style the text that will be displayed by the cue.



The text editing area behaves similarly to most text editing in macOS. Because the default background of T Text cues is transparent, running a Text cue by itself will drawn the text over black. Therefore, the background of the text editing area is drawn in black and the default color of text is white.

Below the text editing area are controls which adjust the style and presentation of the text in the cue.

Font and color

Clicking on the grey **Q** opens the font panel which lets you change the typeface, size, and style of the selected text. As you can see, one T Text cue can contain text with multiple styles at once.

Clicking on the multicolored **Q** opens a color picker which lets your change the color of the selected text.

Clicking on the grey **Q** with multicolored background opens a color picker which lets your change the color of the background of the selected text.

Alignment

To the right of the font and color buttons are four buttons that allow you set the alignment of the cue's text; left, center, justified, or right. All text in a single T Text cues shares the same alignment.

Width and spacing

When a T Text cue is started, QLab renders the text as a PNG image and displays that image. By default, the image dimensions are automatically calculated to fit the text exactly. The **Width** field allows you to set the width of the rendered image. The height of the image remains automatically set, based on the amount and size of text in the cue. You can manually increase the height of the image by adding carriage returns above and below your text.

The rendered dimensions of the cue are displayed next to the width field.

The **Line Spacing** field lets you set the vertical spacing between lines of text, which is a product of the font face and size. Smaller numbers result in less space between lines; larger ones create more space.

The **Preview** pop-up menu scales the display size of the text in the inspector to make it easier to edit at very small or very large font sizes. It does not have any effect on the actual output of the cue.

Above the text field is a ruler which shows the actual width of the cue in pixels.

The Video FX Tab

The Video FX tab behaves the same as the Video FX tab for Video cues.

Broken Text Cues

T Text cues can become ✕ broken for the following reasons:

Missing font

The font used by the Text cue is missing. Close the workspace *without saving* and install the font to clear this warning. If you save the workspace before replacing the font, the broken Text cue will adopt the default font that it was using temporarily.

Invalid size

This error usually means that through a combination of some or all of font size, scale, width, and video effects, the size of the image rendered by the cue is invalid. Adjust any of those parameters to create a valid image size and clear this warning.

No video output stage

The cue has no video output stage assigned. Assign the cue to a stage to clear this warning.

Incomplete video output

The cue is assigned to a stage, but the stage is incompletely configured. Visit the Video Stage Editor to complete the configuration of the stage and clear this warning.

License required

A video license is required to use Text cues. Install a video license or remove the cue to clear this warning.

Video Output

The video output system of QLab 5 supports a wide variety of workflows and setups, from a single TV built into a set, to multi-projector blends mapped onto complex scenery, to LED walls, to broadcast feeds. This section of the manual will take you through all the steps necessary to fully configure video output, but it will also help guide you to simple solutions when that's what you need.

All aspects of video output are configured in Workspace Settings → Video.

Video Terminology

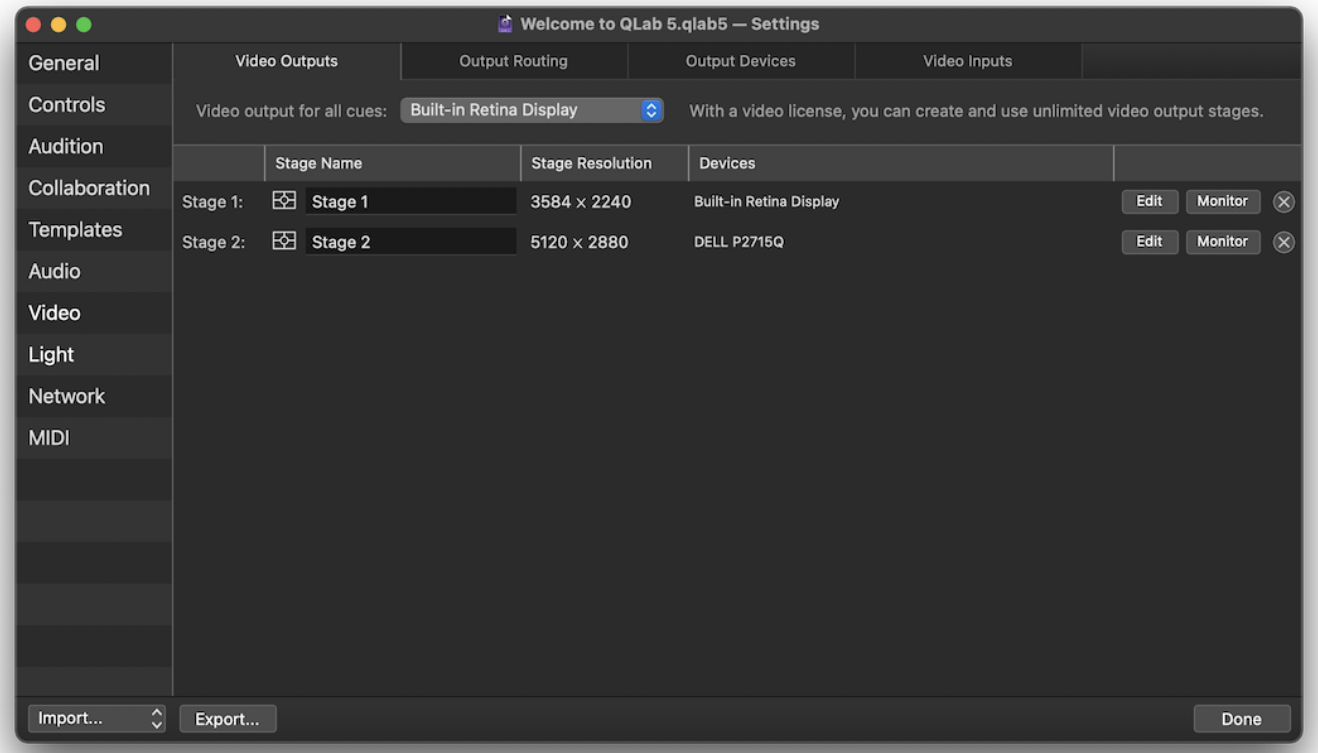
QLab uses the following terminology in the realm of video. Care has been taken to make sure these terms are approachable and not ambiguous, but nevertheless it can be difficult to keep track of. Here is a brief glossary of the essential video terms in QLab.

- A **stage** is a sort of virtual raster[1] to which Video, Camera, and Text cues are assigned. The imagery created by those cues is displayed on the stage, and the stage is displayed on screens, projectors, NDI streams, and Syphon outputs.
- A **region** is a section of a stage which will be displayed on a device. A region can cover the whole stage or just a part of it, and multiple regions can overlap or not as needed.
- An **output route** or simply **route** is the intermediary between regions and devices. Regions are assigned to routes, and the route takes care of the particulars of sending video to the assigned device. This intermediary makes it possible for cues, stages, and regions to behave consistently even if you need to trade out your devices for other hardware.
- A **device** is QLab's internal representation of an actual, physical video display device such as a computer monitor, television, video projector, or LED wall. A device can also be an NDI output or a Syphon output, since these have the same function as a physical video display.

Using Video Without A Video License

QLab can be used for video without a license installed, which allows you to address simple video needs without spending any money on QLab. You can find a list of which features are free and which require a license from the Features section of this manual.

When no video license is installed, video output from QLab is restricted to a single device.

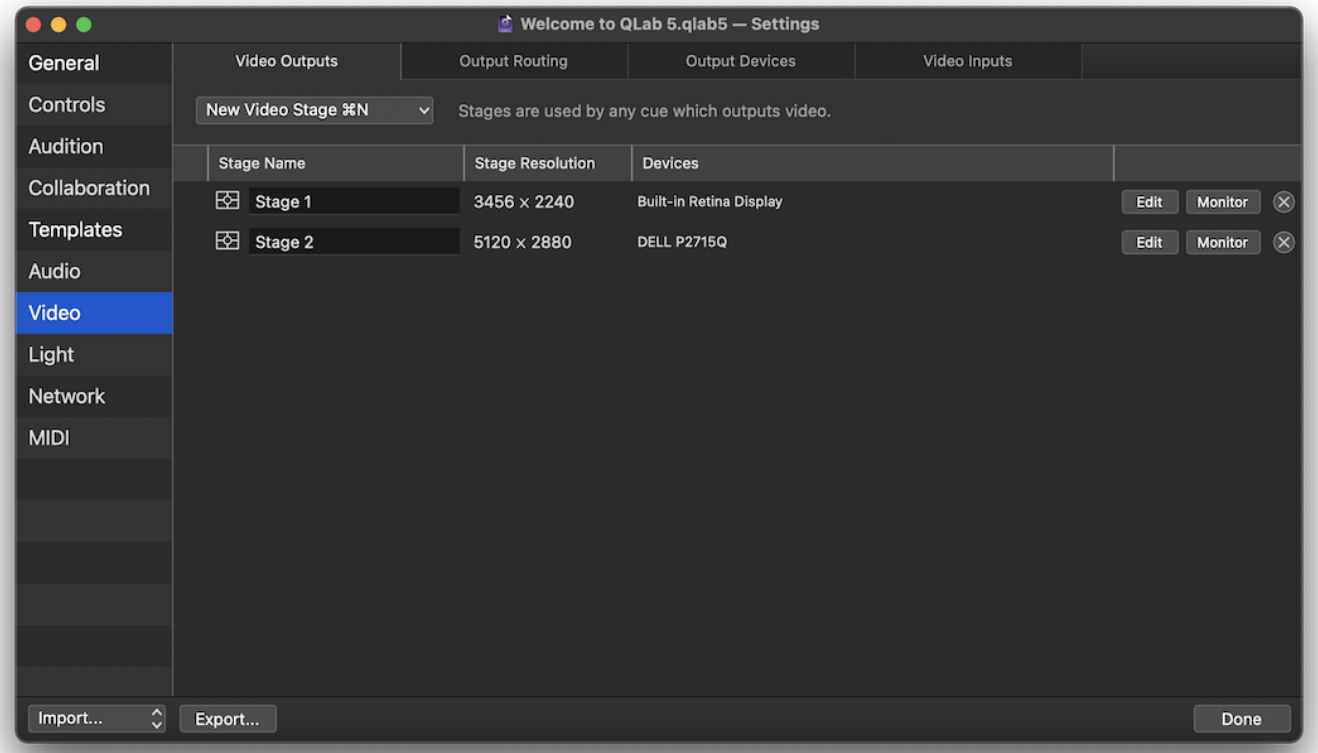


The pop-up menu at the top of the Video Outputs tab, labeled “Video output for all cues”, allows you to select a single device connected to your Mac. Any device which appears as a display in the macOS Displays preference pane will be available here. Whichever device you choose will be used by all video output from your workspace. You do not need to adjust cues if you change this setting; every cue is automatically routed to the device selected here.

The remainder of this section of this manual assumes the use of a video license. If you use QLab without a video license, you may find the rest of this section of the manual unnecessary.

Getting Started

When you create a new workspace, QLab will automatically create a stage for each display connected to your Mac. If you don’t need anything fancy, you can start programming cues as quickly as possible. If you need something more specific, you can edit these stages to suit your needs or delete them and create your own stages.



This screen shot was taken on a 2021 16" MacBook Pro with a 5120 × 2880 monitor connected to its HDMI port. When this workspace was created, QLab automatically created Stage 1 with the MacBook Pro's internal display assigned to it, and Stage 2 with the external monitor assigned to it.

The stages in the workspace are listed in a table with several columns.

- The **Stage Name** column lets you see and edit the name of the stage. Note that if you plan to use OSC to control a stage by name, you must give the stage an OSC-compatible name, i.e. one without spaces or special characters reserved by the OSC spec.
- The **Stage Resolution** column shows the pixel dimensions of the stage. The automatically created stages in this screen shot have the same dimensions as the devices that they are assigned to use, but stages can be any size up to 16384 × 16384, limited of course by the processing power and memory available on your Mac.
- The **Devices** column lists the devices in use by the stage. The automatically created stages have one device each. A stage that uses multiple devices would list them here separated by commas.
- The **Edit** button opens the video stage editor, which allows you to change the properties of the stage. The video stage editor is discussed below.
- The **Monitor** button opens a monitor window for the stage.
- The ⊗ button deletes the stage. This is an undo-able operation, by the way.

To create a new stage, click the **New Video Stage** button. When the Video Outputs tab is open and frontmost, the keyboard shortcut **⌘N** (which ordinarily creates a new workspace) is a shortcut for this button. The button opens a pop-up menu to allow you to create a new stage with some initial settings if you so choose.

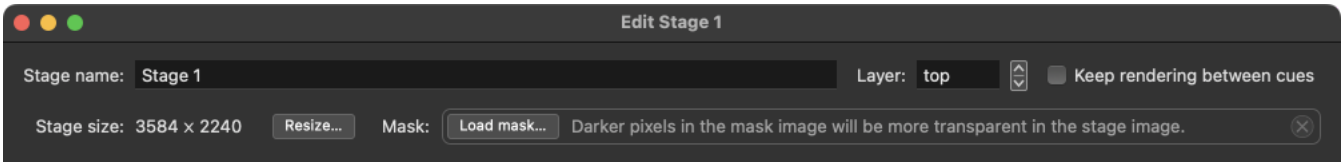
- **Empty Stage** - create a new stage with a resolution of 1920 × 1080 pixels and with no regions created, and no output routes assigned.
- **Stage with output** - listed under this heading are the output routes defined in the workspace. Choosing one will create a new stage with the same resolution as the route and configured to output to that route. This is essentially what QLab does to create the default stages in a new workspace.
- **Stage with partial output** - each output route in the workspace is listed here again, but each with its own sub-menu that allows you to create a stage that's the same size as a portion of that output only. Partial outputs are discussed below.
- **Multi-Output Stage...** will prompt you to describe a multi-projector setup, including the number of projectors, their resolution, and their physical arrangement. QLab will then create a stage that is pre-configured to work with that projector layout.
- **Stage with new NDI output...** will prompt you to create and configure a new NDI output. QLab will then create a stage using that output. NDI outputs are discussed below.
- **Stage with new Syphon output...** will prompt you to create and configure a new Syphon output. QLab will then create a stage using that output. Syphon outputs are discussed below.

The Video Stage Editor

We're not going to lie to you; the video stage editor is the most complicated part of QLab's interface. There is a lot going on here because there is a lot of power here. If you are new to configuring video in QLab, give yourself plenty of time to get acquainted with it, and don't forget that you can always write to support@figure53.com with any questions, big or small.

It is highly encouraged to work your way through this section of the manual with the video stage editor open so you can look and poke and test and try things as you read about them.

The editor window has a top area with two tabs below it, *Layout* and *Warping*, which cover the rest of the window. The top area, pictured here, contains controls which adjust the basic attributes of the stage.



Basic Stage Attributes

Stage name can be any text and is more important to humans than to software. You can name the stage however you like.

The **Layer** control for stages defines the stacking order for the stage relative to other stages that use the same device. If a workspace has two stages, "Angelica" on layer 100 and "Eliza" on layer 50, and both of these stages use the same area of the same device, then all the cues on "Angelica" will render on top of all the cues on "Eliza", regardless of the layer settings for each cue, since "Angelica" is on a higher layer than "Eliza."

Keep rendering between cues. Whenever a Video, Camera, or Text cue is running, QLab draws a black background across the whole stage that the cue is assigned to. If no cues are playing, QLab does not draw that background, which allows the macOS desktop to show through. Sometimes that might be what you want, sometimes not. If this checkbox is checked, QLab will keep the black background up until QLab quits or the workspace receives the command to panic. This not only obscures the desktop, but can help ensure a smoother transition between cues, particularly on Macs whose GPU performance is just at (or possibly just below) the level of performance required for the workspace.

Stage size shows the resolution, a.k.a. the pixel dimensions, of the stage. The **Resize...** button allows you to resize the stage, with options to ensure that the aspect ratio of the stage stays the same after the resize, and options to either resize the regions on the stage or not.

Masks

A stage **mask** is a way to completely or partially block certain pixels of a stage from displaying. Masks have several uses, including:

- Making a stage that isn't rectangular.
- Feathering the edges of a stage for a vignetting effect.
- Cutting a hole in a stage to allow another stage to show through from a lower layer.

To add a mask to a stage, click on the **Load mask...** button and select an image file.

A mask should be a greyscale image that matches the dimensions of the surface, but QLab will convert any image used as a mask to greyscale and scale it to fit the surface if necessary. Any compatible still image file type can be used as a mask, and there is no performance difference between formats when

rendering because QLab re-renders the image when importing it for use as a mask. However, if the size of the mask image is different from the size of the surface, QLab must scale it “live” and that can negatively affect performance, so try to match your surface dimensions whenever possible.

Wherever the mask image is black, the stage will be masked. Wherever the mask image is white, the stage will be visible.

Example

Stages are rectangular. A cue displayed full-stage will therefore, of course, also be rectangular.



If you wanted a stage to be oval in shape, you could create a mask that defines that shape.



With that image applied as a mask to the stage, the visible area of the stage is clipped by the mask.



It’s important to understand that the black area in this image isn’t the mask being shown, it’s the backdrop. If you place another image on another stage set to a lower layer, that image will be shown everywhere that the top layer is masked.

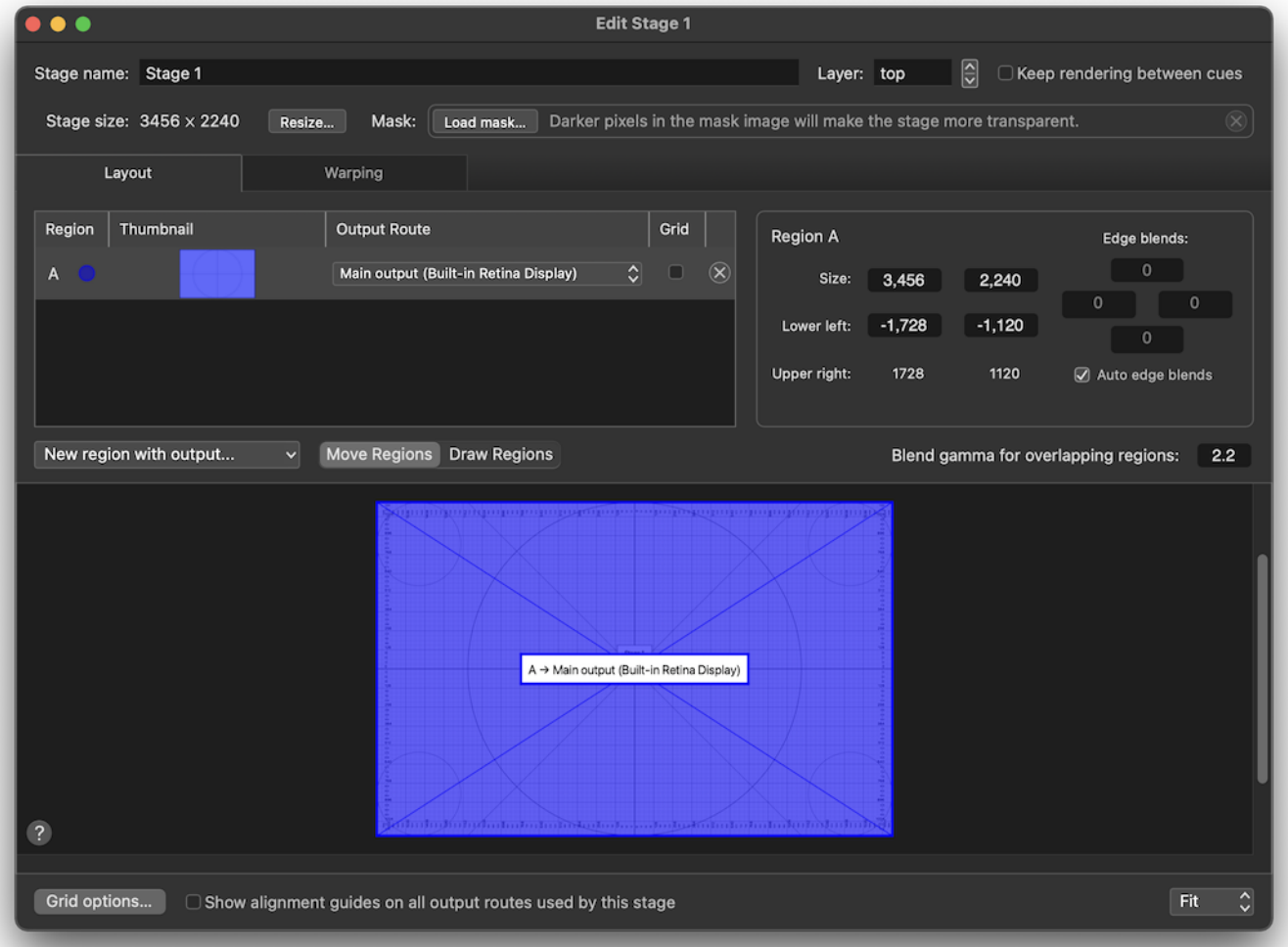


QLab does not have an integrated mask editor. Instead, it watches the mask image file to look for changes; when it sees that the file has changed, it automatically reloads it. Thus, you can use any graphics editing program to make adjustments to a mask, and see changes applied live as you go.

To remove a mask, click the ⊗ button to the right.

The Layout Tab

The Layout tab of the video stage editor allows you to define and arrange the stage's regions. The stages which QLab automatically creates use a have a single region covering the whole stage.



We'll walk thorough the controls in the Layout tab first, and then go into more detail about regions and when you'd want to use more than one of them after.

The Region List

In the upper left part of the Layout tab is a table showing the regions of the stage. The table has five columns.

- Each **Region** are automatically assigned a letter index and a color; A and dark blue, in this screen shot.
- A **thumbnail**, or miniature version of the region, can help you keep track of what's what on stages that use several regions.
- The **output route** pop-up menu lets you tell QLab which output route the region should be sent to.
- The **grid** checkbox displays a monochrome alignment grid on the region. The grid has a few configuration options which are discussed below.
- The ⊗ button deletes the region.

Underneath the list is a drop-down menu which lets you quickly create a new region using a specific output route. Next to that menu is a switch with two states. When the switch is set to *Move Regions*, you can use your cursor to move and resize existing regions in the canvas below. When the switch is set to *Draw Regions*, you can use your cursor to draw a new region in the canvas. Drawn regions will need to be assigned to an output route using the menu in the region list.

Region Parameters

The upper right part of the Layout tab shows several parameters of the region that's selected in the list.

- **Size** shows the width and height of the region, and lets you edit the size by either typing in new values or clicking and dragging the cursor up or down. Regions cannot extend outside of the stage boundaries, so these text fields are limited based on the size of the stage and the position of the region.
- **Lower left** shows the coordinates of the lower left corner of the region compared to the center of the stage. The region can be positioned precisely by typing in new values or clicking and dragging up or down. Regions cannot extend outside of the stage boundaries, so these text fields are limited based on the size of the stage and the position of the region.
- **Upper right** shows the coordinates of the upper right corner of the region compares to the center of the stage. These values are not editable and are for reference only.
- The **Edge blend** text fields and checkboxes work together to define the amount of feathering, or fall-off, of each side of the region. Edge blending is best understood in the context of a stage with multiple regions (see below), but you can also use edge blends to soften the edges of a single-region stage.

Underneath these parameters is a text box which lets you set the *gamma*, or brightness curve, of all the edge blends in the current stage.

The Canvas

The canvas view is a white rectangle with black gridlines which represents the stage. Regions are drawn as colored rectangles labeled with their name and assigned output. When the *Move Regions/Draw Regions* switch is set to *Move Regions*, you can click on drag on an edge of a region to resize it, or in the center of the region to move it. The sides, vertical centerline, and horizontal centerline of the canvas are "magnetic," so if you drag the edge of a region close by, it will snap to fit exactly. Edges of regions are also magnetic, making it easy to snap regions together and ensure that there is no accidental gap or overlap. You can hold down the command (⌘) key to disable snapping.

You can also adjust the selected region using the keyboard:

- **Arrow keys** move the selected region by one pixel.
- ⇧ **+ arrow keys** move the selected region by ten pixels.
- ⌥ **+ arrow keys** resize the selected region by one pixel.
- ⌥⇧ **+ arrow keys** resize the selected region by ten pixels.

The selected region is drawn darker than unselected regions. Regions that have no output route assigned are drawn with dotted outlines.

Regions that use edge blends will draw the blend area with a hatched background.

By default, the canvas is scaled to fit the available space in the window, so enlarging the window will let you see the canvas in more detail. You can use the pop-up menu in the lower right corner of the window to set a particular scale factor if you prefer.

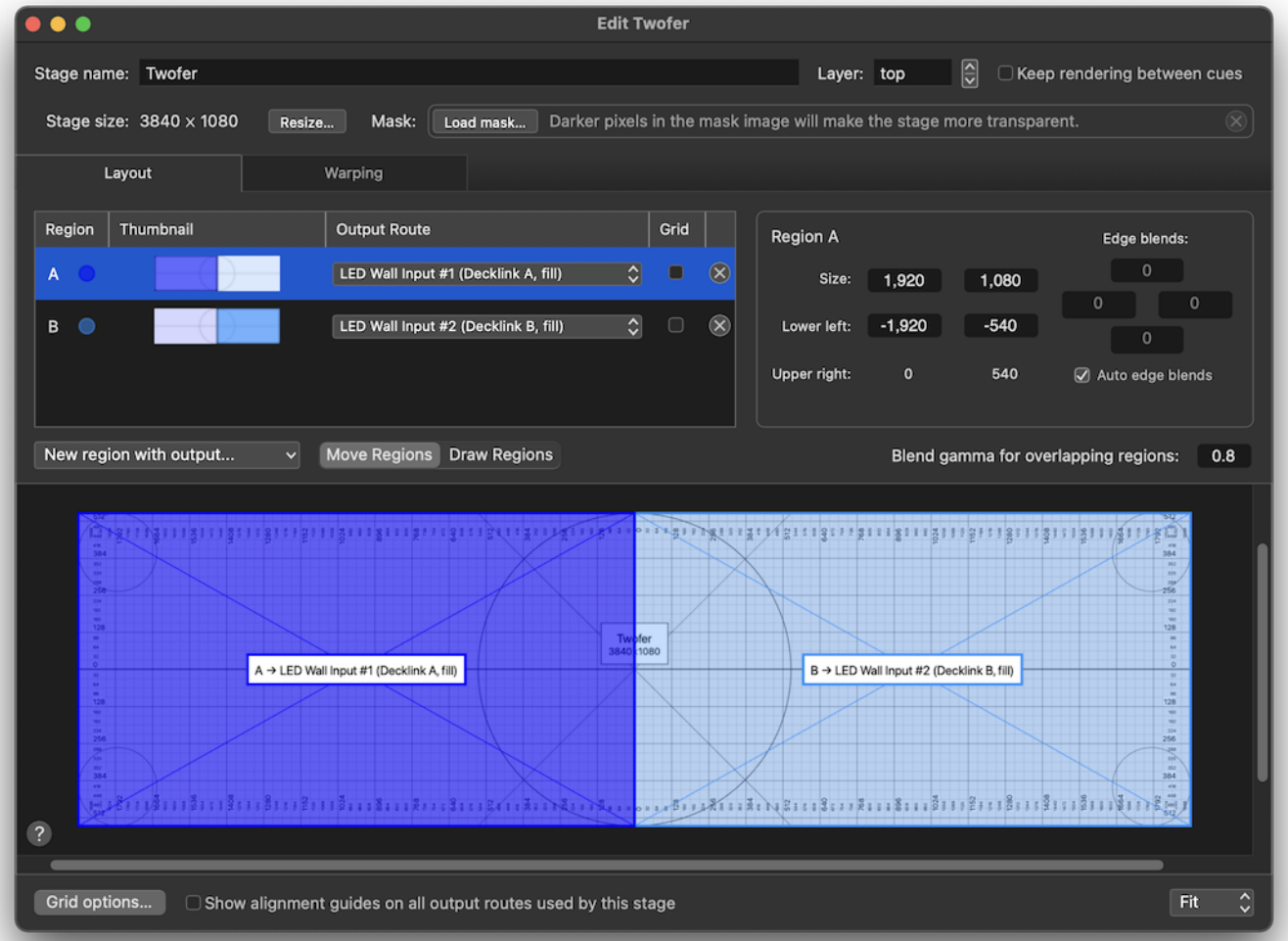
The **Grid options...** button displays a popover which allows to show all grids, hide all grids, and export a PNG image of the grid. It also allows you to switch between a white-on-black grid and a black-on-white grid, and optionally disable edge-blending between regions while the grid is showing. These options all exist in service of making it easier to configure the stage under a variety of conditions.

The checkbox labeled *Show alignment guides on all output routes used by this stage* displays a colored border, centerlines, and text labels on every output route that any region of the stage is assigned to use. Alignment guides always demonstrate the full raster of each route, without any warping, cropping, or other modification. The purpose of the alignment guides is to both positively identify each output route and its associated device, and to aid in the physical alignment and setup of the devices.

Understanding Regions

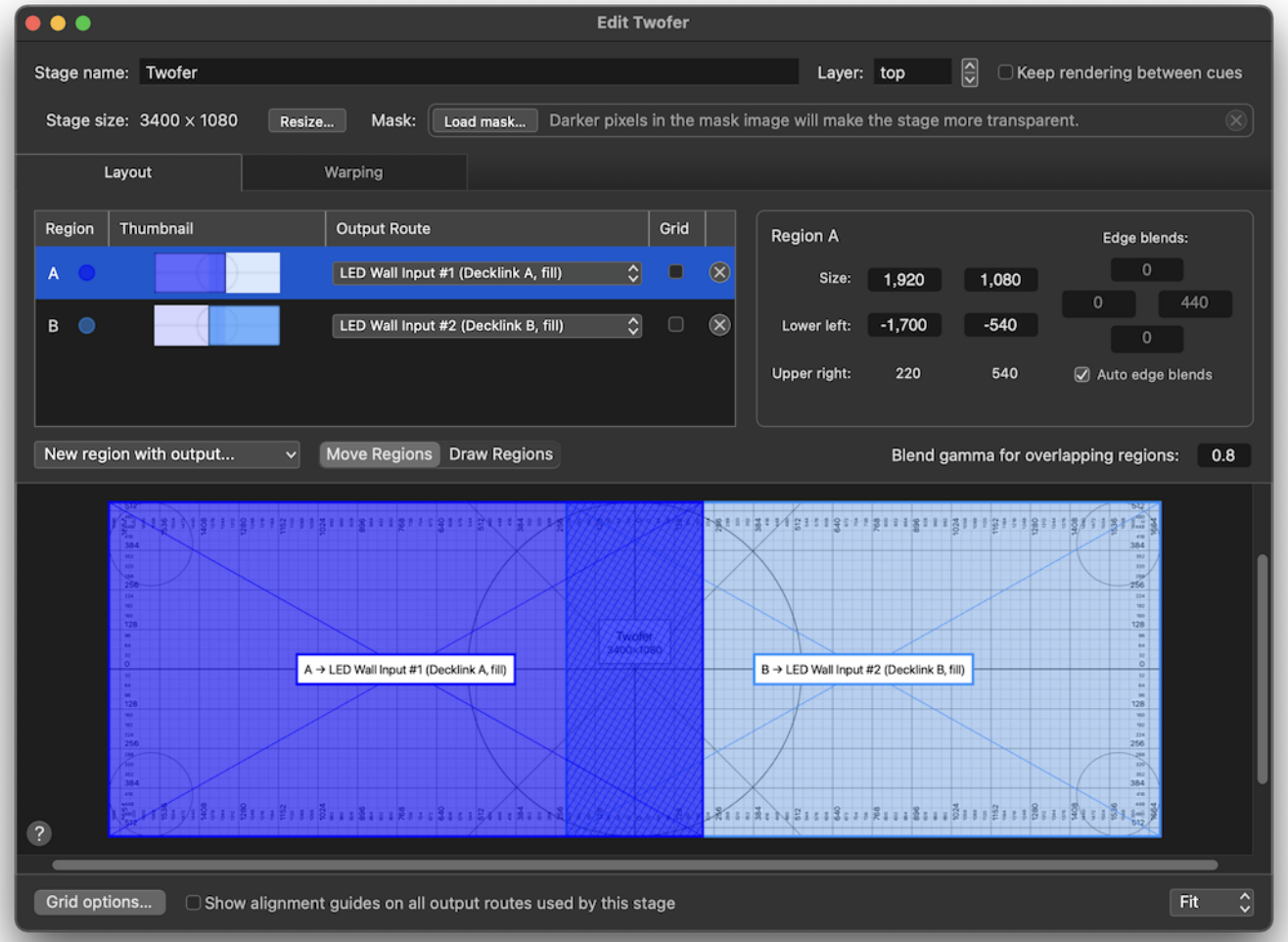
So, what are regions really and why does QLab have them?

The first and simplest answer is that regions are how a screen or projector (or NDI or Syphon output) knows which pixels of a stage to display. If you're only using a single screen, regions feel like an unnecessary step. They can be easier to understand if you look at a stage that uses multiple screens.



This stage is `3840` pixels wide by `1080` pixels tall; the size of two FHD[2] rasters next to each other. If we imagine a situation in which two FHD[2] projectors are perfectly aligned side by side and both projecting on, say, the cyclorama at the upstage edge of a traditional theater, it's easier to see what regions do for us. Region A uses the output route for the house left projector and covers the camera left half of the stage, and region B uses the output route for the house right projector and covers the camera right half of the stage.

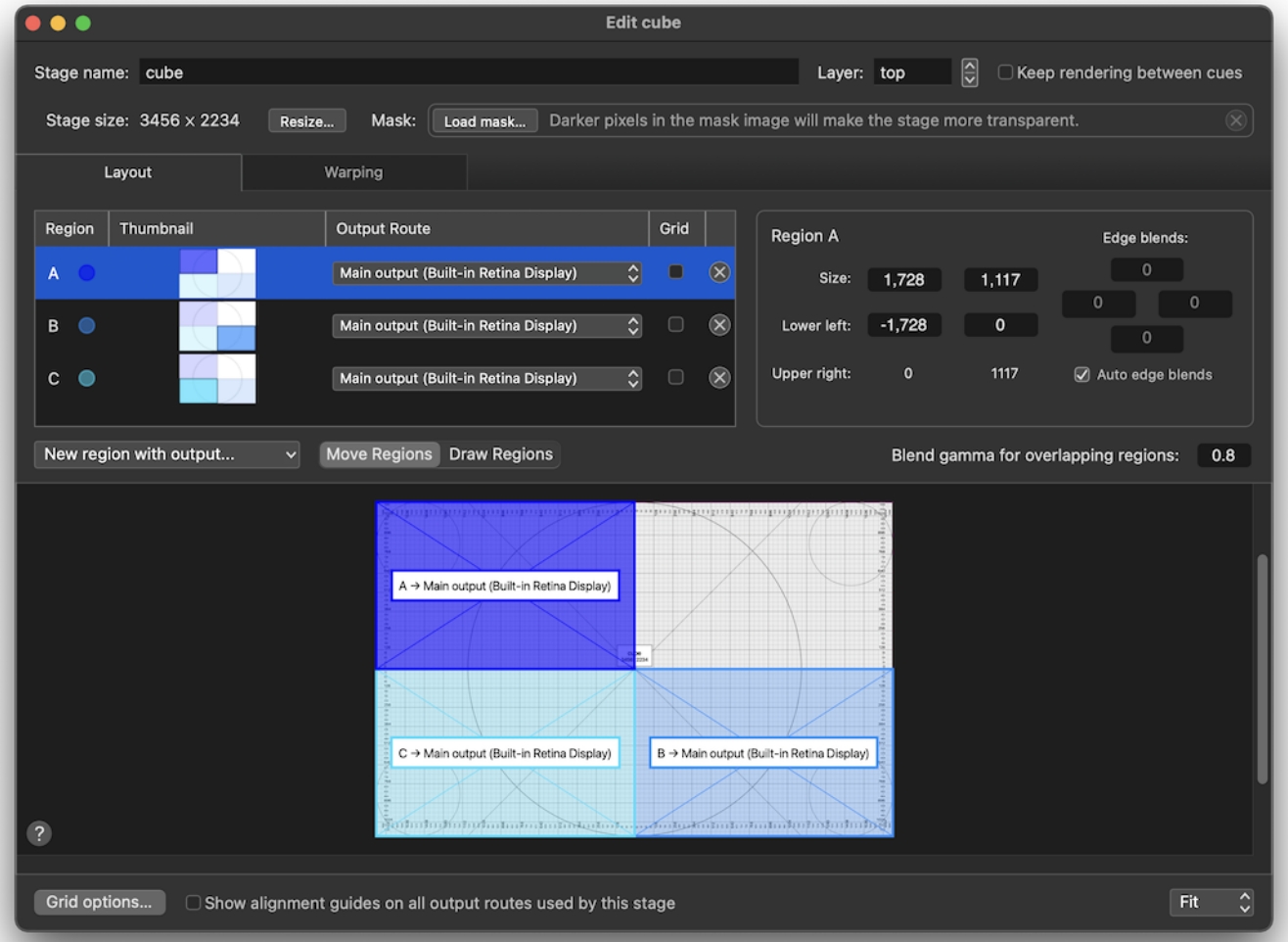
If you have two projectors laid out like this, though, the more common way to use them is to overlap their coverage a little bit at the center, and blend their edges together. This creates a very nice, seamless transition between the two projectors.



This stage is `3400` pixels wide by `1080` pixels tall. By overlapping regions A and B by the same amount as the actual projector beams are overlapped, you can create a single, seamless image across both projectors. The crosshatching in the center notates the area of each region that is blended. In this area, the same pixels from the stage are sent to both projectors, and QLab automatically calculates a soft dissolve across that area, fading the image for each projector so that the overall brightness of the stage remains constant.

You can manually adjust the size of the blend region by unchecking the *Auto edge blends* box, and you can adjust the brightness curve of the blend by using the *Blend gamma* control.

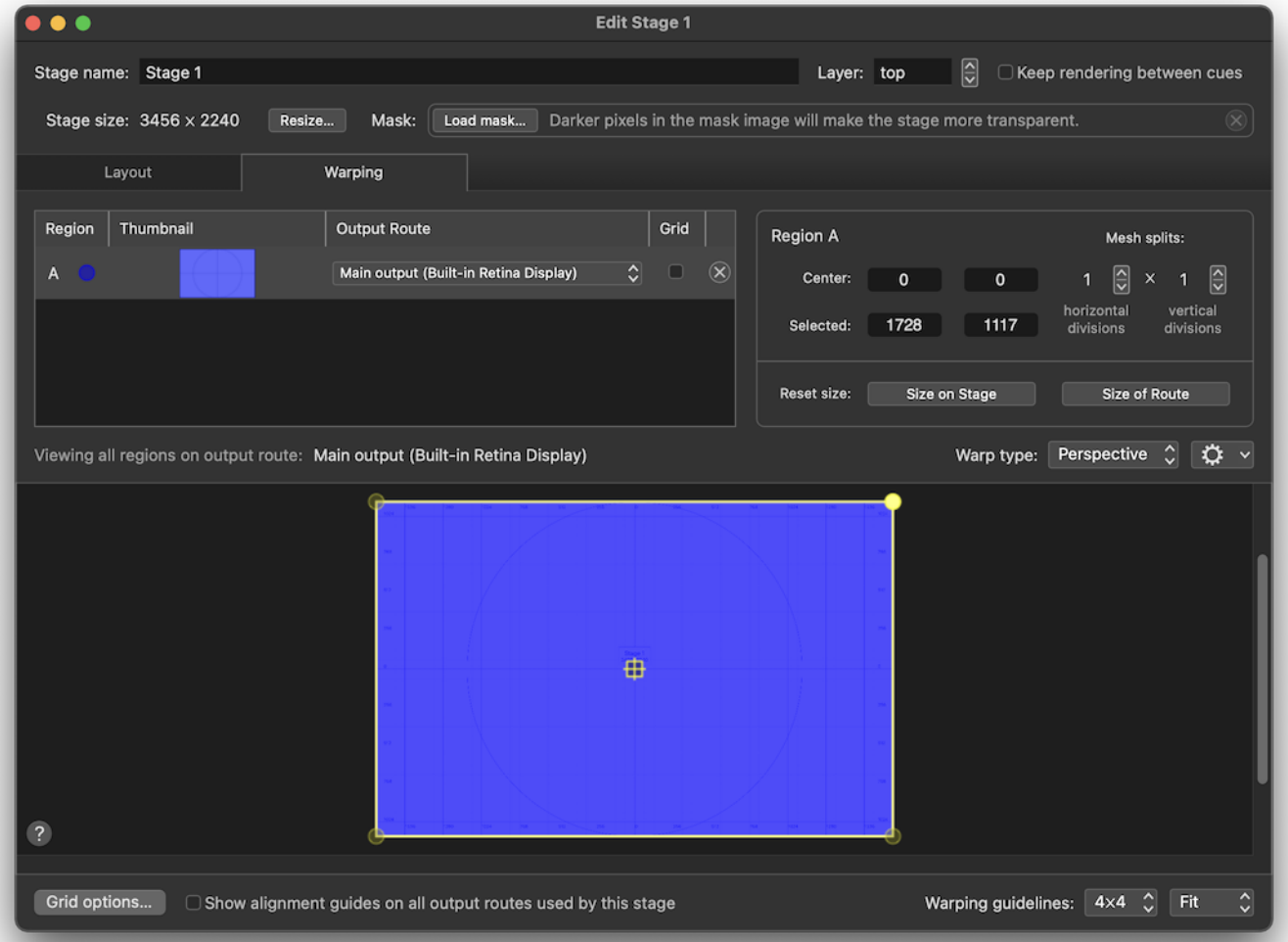
Regions can also be used to subdivide a stage into sections that need to be individually controlled.



Here, three of the quadrants of a stage are covered by three regions. Each of those regions uses the same output route, and therefore the same screen. Each of those three regions can be individually positioned within the raster of the output, and individually warped as we'll discuss in the next section.

The Warping Tab

The Warping tab has the same basic form as the Layout tab and an identical region list, but it displays different parameters of the selected region, and gives you different things to adjust in the canvas.



The purpose of the Warping Tab is to let you adjust the geometry of the stage on a per-region basis. Regions can be warped for any number of reasons, the most straightforward of which is to correct for keystoning caused by an off-axis projector. Warping can also be used to match a video projection to a non-flat projection surface, or for other creative effects.

Region Parameters

The upper right part of the Warping tab shows several other parameters of the region that's selected in the list.

- **Center** shows the coordinates of the center of the region compared to the center of the stage. When a region is warped, as will be discussed below, it can be difficult to mathematically define the true center of the region. Therefore, in this view only, "center" really means the *functional* center of the region, and not necessarily the exact actual center. You can edit these values by typing new ones in or by clicking and dragging up or down.
- **Selected** shows the coordinates of the currently selected control point below. You can edit these values by typing new ones in or by clicking and dragging up or down.
- **Mesh splits** are a pair of up/down arrow buttons which let you increase or decrease the number of controls points used to warp the selected region. Each control can be adjusted from 1 up to 32 using powers of two.

Below the editable attributes are two buttons which reset the selected region. You can use these to start over if you've tangled yourself up.

- **Size on Stage** resets both mesh splits to 1 and sets the region to match the size of region defined in the Layout tab.
- **Size of Route** resets both mesh splits to 1 and sets the region to match the size of the output route that it's assigned to use.

The Canvas

In the Warping tab, the canvas looks different depending on which region is selected. The grey rectangle with the thin, purple-ish border represents the output route that the selected region is assigned to use. Every region of the stage that's assigned to use that output route is drawn in color. The selected

region is drawn with a yellow border.

By default, the output route is drawn with a 4 × 4 grid of warping guidelines. These lines never appear in QLab's output, they're only visible here to aid with warping and alignment. You can change their layout using the pop-up menu in the lower right corner of the window next to the scale pop-up menu. The edges of the output route and the guidelines are all "magnetic", so control points snap to them.

Regions can be adjusted in a great number of ways here. Holding down the shift (⇧) key while doing any of these adjustments allows fine-tuning. Holding down the command (⌘) key while doing any of these adjustments temporarily disables snapping.

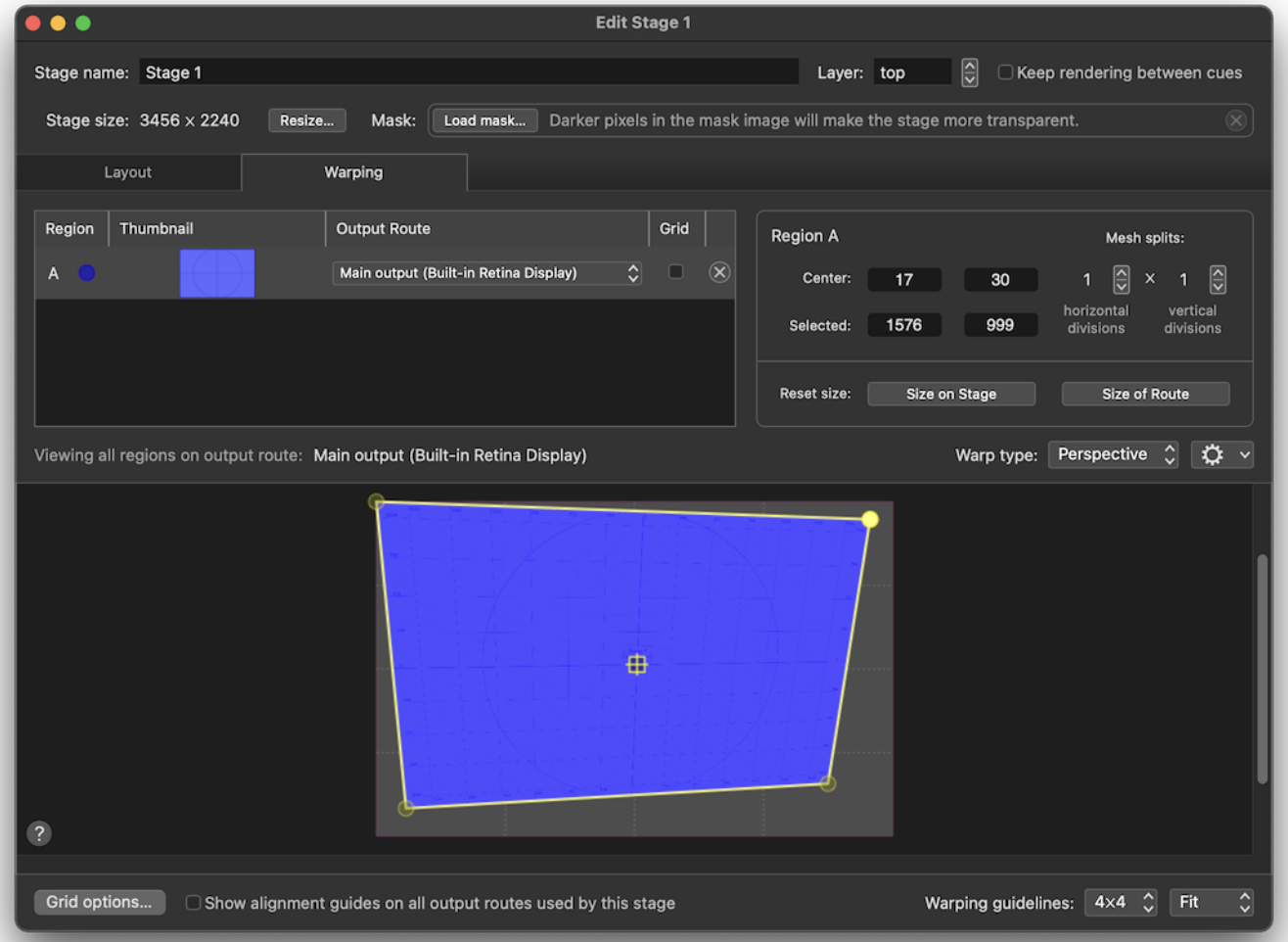
- Click and drag on a control point (which is drawn as either a circle or diamond for reasons discussed below) to warp the region using that control point.
- Click and drag on a control point while holding down the option (⌥) key to move the whole region using that control point as the reference.
- Place your cursor within the region where it displays as a hand, then click and drag to move the whole region.
- Place your cursor on an edge of the region where it displays as a line with a bidirectional arrow, then click and drag to warp the region by moving the whole edge.
- Place your cursor just inside an edge of the region where it display as a line with an inwards-pointing arrow, then click and drag to warp the region along a single axis only by moving the whole edge.

You can also use keyboard shortcuts to navigate between control points and adjust them:

- **W**, **A**, **S**, and **D** keys move the selection between control points, selecting the point that is above, to the left, below, or to the right of the currently selected point.
- **Arrow keys** move the selected control point by one pixel.
- ⇧ **+ arrow keys** move the selected control point by ten pixels.

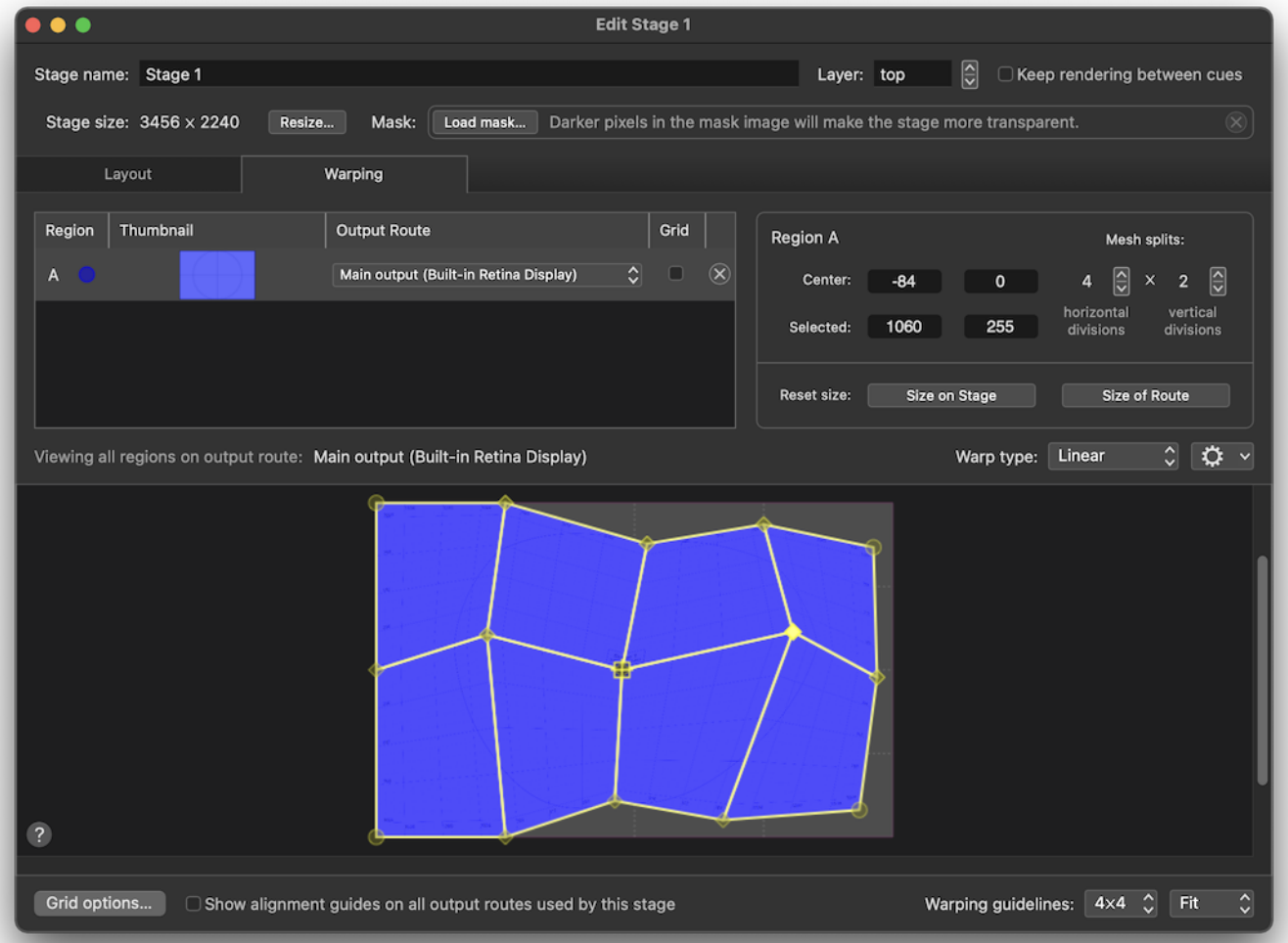
The **Warp Type** pop-up menu, which applies to all regions that share an output route, lets you choose the geometric approach that QLab uses to warp the regions on that route.

Perspective warping, which is the default type, is the most generally applicable.



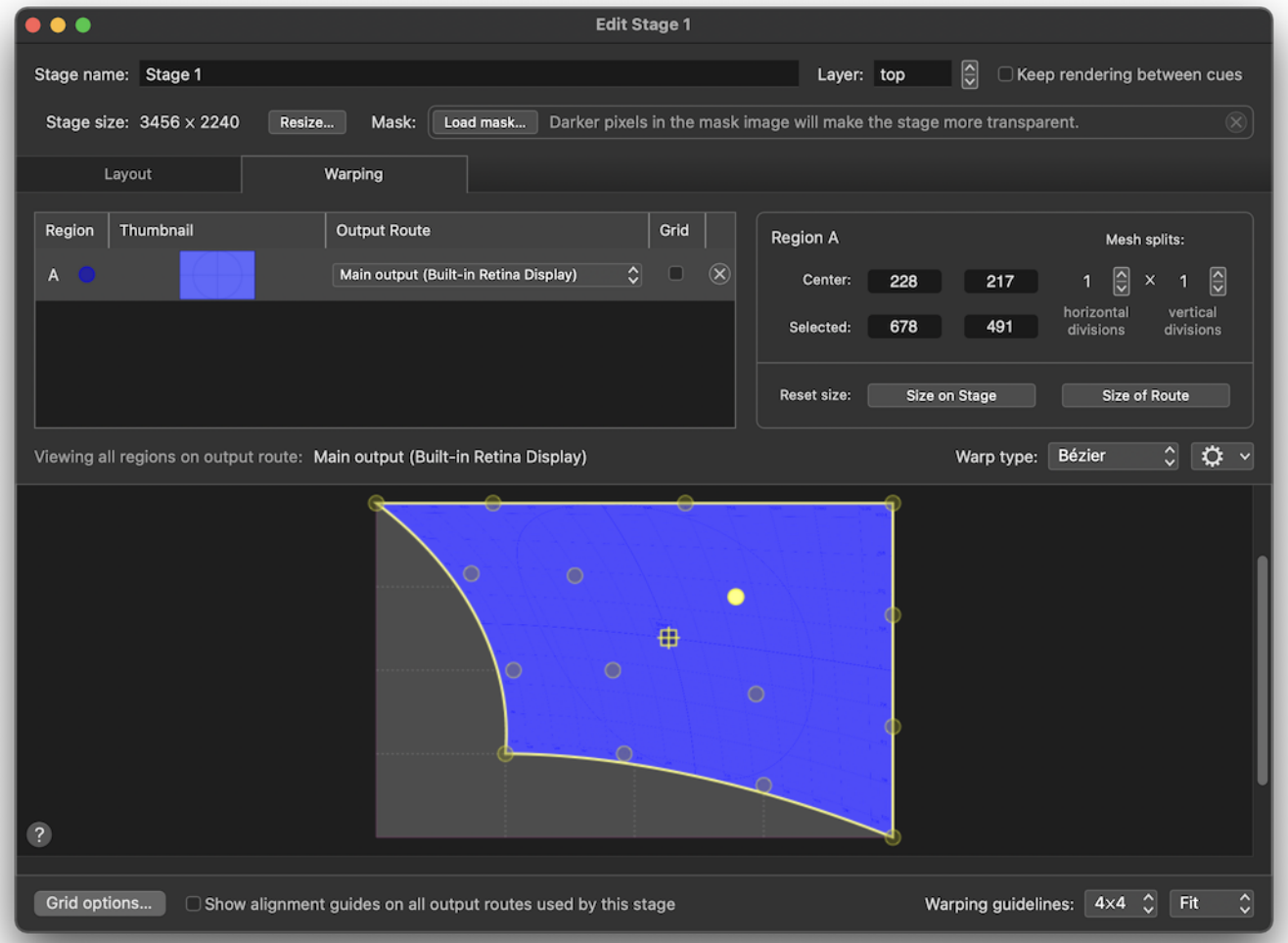
Using perspective warping, a projected object will appear the same size no matter where it's placed on the region. However, in order to allow complex perspective warped shapes, QLab uses "continuous perspective" warping which guarantees continuity of the warp, but means that extreme warps might not be truly 100% perspective-correct.

Linear warping is a more direct, mathematically straightforward warp.



This type might look better with very complex, multi-faceted warps, but will typically look odd at extreme angles.

Bézier warping is a more complex type, used to project onto curved surfaces.

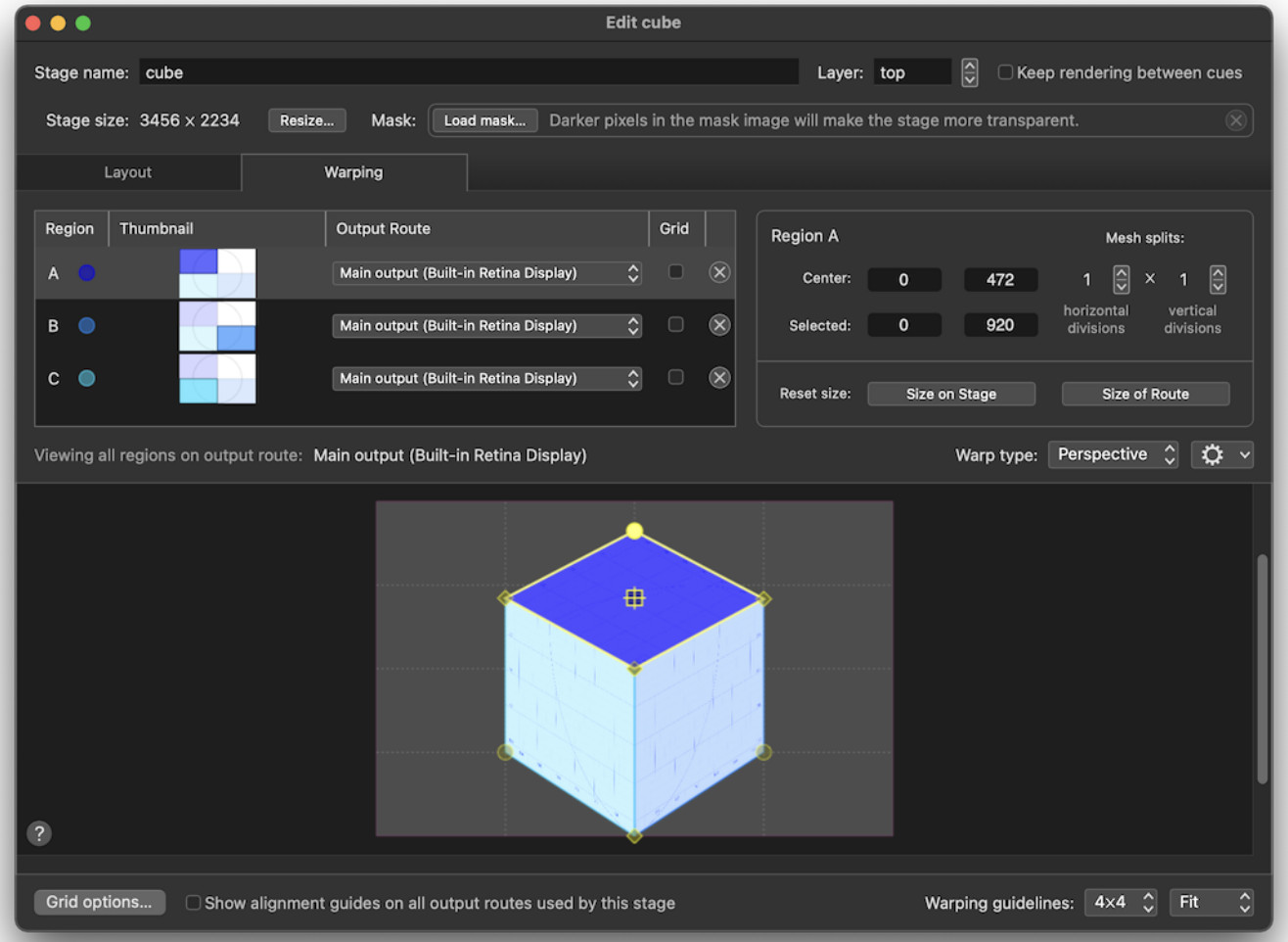


This warp type gives you sixteen control points per mesh split, and each control point bends the warp smoothly in the direction that it's pulled, as though the region is made of an elastic material.

The ⚙ menu contains two menu items which are relevant to a very specific situation: when control points of two or more regions are overlapping precisely. When they are, this menu can be used to link those control points together so that they move as one, or unlink them to move them independently.

When control points are linked, they are drawn as a diamond instead of a circle. Linked, diamond-shaped control points behave just like round ones except that moving them changes the region that each linked point belongs to simultaneously.

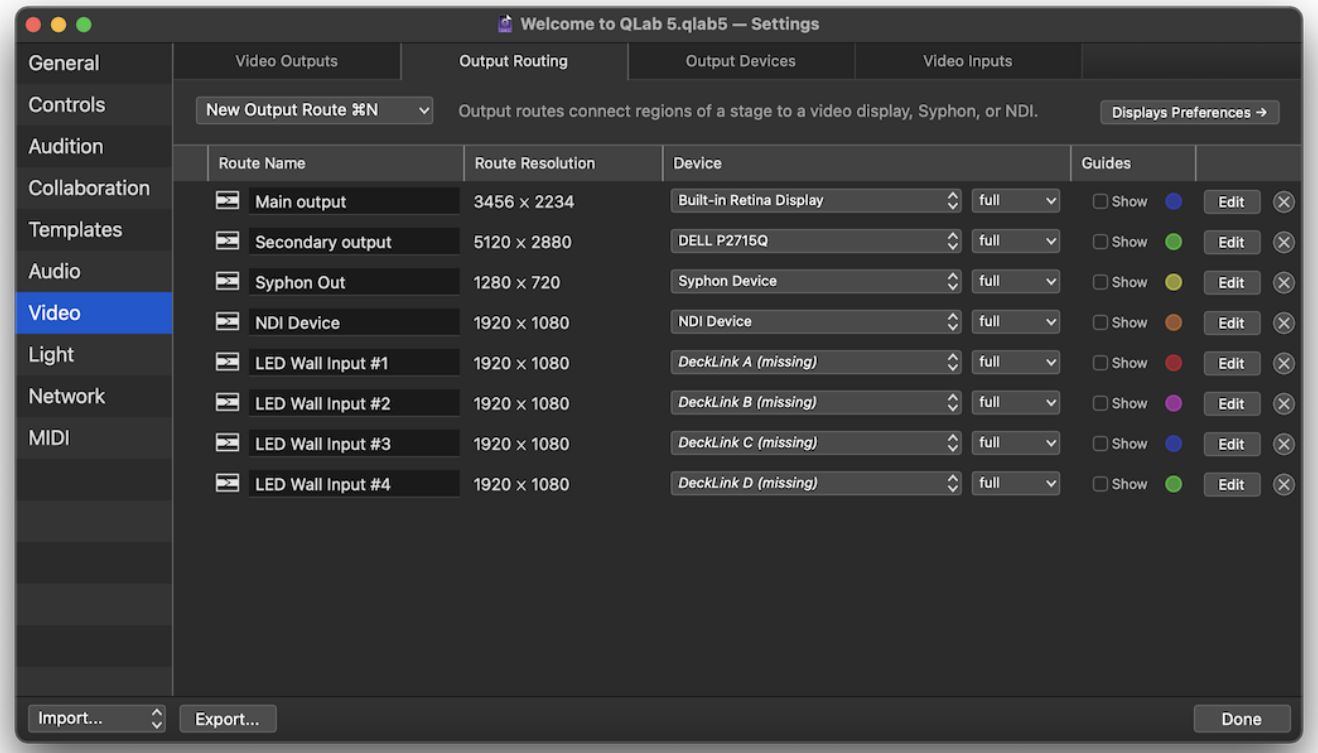
Using the region example above with three regions assigned to a stage, all using the same output route, we get a nice example of linking control points.



Imagine the output route is connected to a projector that is pointing at a cube-shaped object on stage. Here, the three regions are each warped to conform to a single face of the cube, and the control points on each region that coincide with controls points on other regions are linked. This allows you to warp the regions to fit the cube as a single object, instead of three separate objects.

Output Routes

Output routes connect QLab to video output devices.



This workspace contains eight output routes of various types. The output routes in the workspace are listed in a table with several columns.

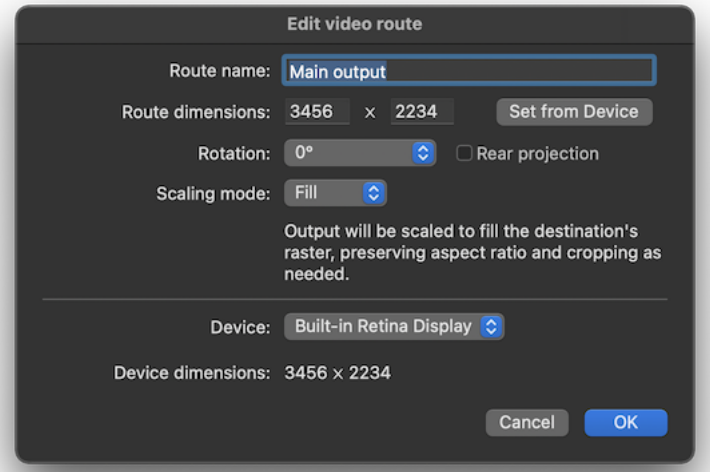
- The **Route Name** column lets you see and edit the name of the route. Note that if you plan to use OSC to control a route by name, you must give the route an OSC-compatible name, i.e. one without spaces or special characters reserved by the OSC spec.
- The **Route Resolution** column shows the pixel dimensions of the route.
- The **Device** column contains two pop-up menus which let you see and change which device the route uses, and how that device is used. These menus are discussed below. When a route was configured using a device which is not currently available, like the last four routes in the screen shot above, the device name is displayed *in italics* and appended with the word *(missing)*.
- The **Guides** column contains checkboxes which let you display a colored alignment guide for each route. When the checkbox is checked, QLab displays a colored outline, centerlines, circular corner and center registration marks, and the route name on the full raster of the output. These guides, which are not affected by region warping, can be used to identify and physically align video devices.
- The **Edit** button opens the route editor, which allows you to change the properties of the route. The route editor is discussed below.
- The ⊗ button deletes the route, which is an undo-able operation.

To create a new output route, click the **New Output Route** button. When the Output Routing tab is open and frontmost, the keyboard shortcut **⌘N** (which ordinarily creates a new workspace) is a shortcut for this button. The button opens a pop-up menu to allow you to create a new route with some initial settings if you so choose.

- **New output route with device:** - select a device to create a new output route using the selected output device, with the route configured according to the properties of that device.
- **New NDI output route...** - this option prompts you to enter the configuration details for a new NDI device, then creates that device and an output route using it.
- **New Syphon output route...** - this option prompts you to enter the configuration details for a new Syphon device, then creates that device and an output route using it.
- **New output route with partial screen device:** - this option is discussed below.

The Output Route Editor

Clicking the **Edit** button next to a route opens the output route editor.



Route name can be any name you choose, and is a name which matters more to humans than to the computer.

Route dimensions can be anywhere from `1 × 1` pixels to `30,000 × 30,000` pixels. The **Set from Device** button sets the route dimensions to match the dimensions of the assigned device.

Rotation lets you rotate the output raster in 90° increments. The **Rear projection** checkbox mirrors the output horizontally to correct for rear projection.

Scaling mode tells the route how to behave if the route dimensions and the device dimensions do not perfectly match.

- **Center** will align the centers of the route and the device, and display the route at its original pixel density. If the route is larger, the edges of the route will be cut off. If the device is larger, the route will display black around the edges.
- **Fit** will scale the route to fit the device while maintaining the original aspect ratio of the route, adding black fill where necessary.
- **Stretch** will scale the route to fit the device exactly, scaling the width and height independently.
- **Fill** will scale the route to fit the device exactly, but instead of stretching the route if it doesn't fit perfectly, it will crop either the width or height.

Device gives you another place to select the device being used by the route. The dimensions of the selected device are displayed right below.

Using scaling modes

Scaling mode mainly only comes into play if you ever need to use an existing route with a new video device of a different resolution. One example of this situation is a touring show which carries its own Mac and QLab workspace, but uses whatever video projector each venue has available.

Imagine a workspace that was created using a projector with a resolution of `1920 × 1080` and then brought on tour. Perhaps one tour stop has a `1920 × 1200` projector, another has a `1280 × 720` projector, and a third has a `3840 × 2160` projector.

When setting up at each venue, you'd connect the house projector to your Mac, open QLab, visit Workspace Settings → Video → Output Routing, select the output route for your stage, and using the menu in the **Device** column to configure the route to use the new projector.

The way your stage appears on the projector will be determined using the scaling mode of the route. Experimenting with the four modes will help you learn how each behaves and which one best matches your technical and aesthetic needs.

Partial Outputs

Display splitters such as the Matrox DualHead2Go, TripleHead2Go, and QuadHead2Go, the Datapath Fx4 and Hx4, and the AJA HA5-4K appear to macOS as a single, very large display. You then connect two, three, or four displays to the output of the splitter, and the splitter provides one half, one third, or one quarter of the pixels of its input to each output.

These devices can be configured to present as a wide display sliced into side-by-side sections, a tall display sliced into top-to-bottom sections, or a large display slice into rows and columns.

To give you the greatest amount of control when working with devices like these, you can configure an output route to use only a specific portion of an output device and behave as though that portion is a whole device. In this way, QLab can be made to behave the way the splitter is behaving.

Output Devices

Output devices are QLab's representation of the actual devices that video is displayed upon. Those devices can be:

- Monitors, televisions, screens, and projectors directly connected to your Mac.
- Any Blackmagic Design UltraStudio, Intensity, and DeckLink devices with output capabilities.
- NDI network video outputs.
- Syphon server video outputs.

Devices available to the workspace are listed in a table with several columns.

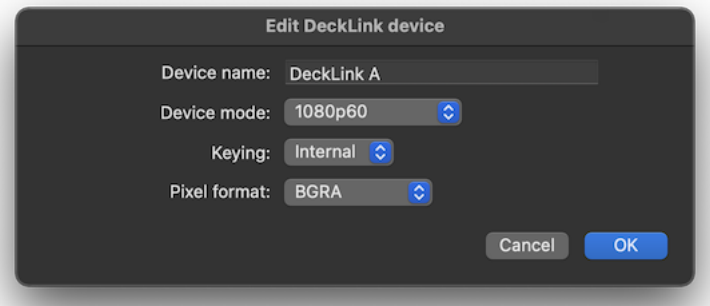
- The **name** columns shows the name of the device. For most hardware devices, this name is set by the manufacturer and not editable. NDI and Syphon device names are editable and can be set to any name of your choosing.
- The **resolution** column shows the pixel dimensions of the device.
- The **info** column shows any other available information. For many devices, this is simply the frame rate (refresh rate) of the device. For NDI devices with audio output configured, audio channel information is also shown.
- The **Edit** button next to Blackmagic Design, NDI, and Syphon devices allows you to edit the properties of those outputs.
- The ⊗ button next to NDI and Syphon devices deletes the device, which is an undo-able operation.

Blackmagic Design Outputs

Blackmagic Design makes a number of video input and output devices across a fairly wide range of prices and capabilities. Some of these devices have inputs only, and naturally those will not appear amongst the list of available output devices. Others have anywhere from one to eight outputs.

Blackmagic devices which have multiple outputs will appear a separate devices in the Output Devices list. For example, the DeckLink Duo, which is a PCIe card, is really two two-channel video interfaces built into a single card. If all four channels are configured as outputs, they will each appear as a separate device.

Clicking the **Edit...** button will allow you to edit the Blackmagic device's settings. The available settings and options vary based on the exact type of device, but they are all generally similar.



Device name

This name is read-only in QLab. You can change the device name in the *About* section in the Blackmagic Desktop Video Setup app. QLab must be restarted to recognize a name change.

Device mode

This menu lets you select the resolution and frame rate mode for output. The available modes vary based upon the device. This setting is also available in the Blackmagic Desktop Video Setup app. QLab's setting here is a sort of remote-control for the setting in that app.

Keying

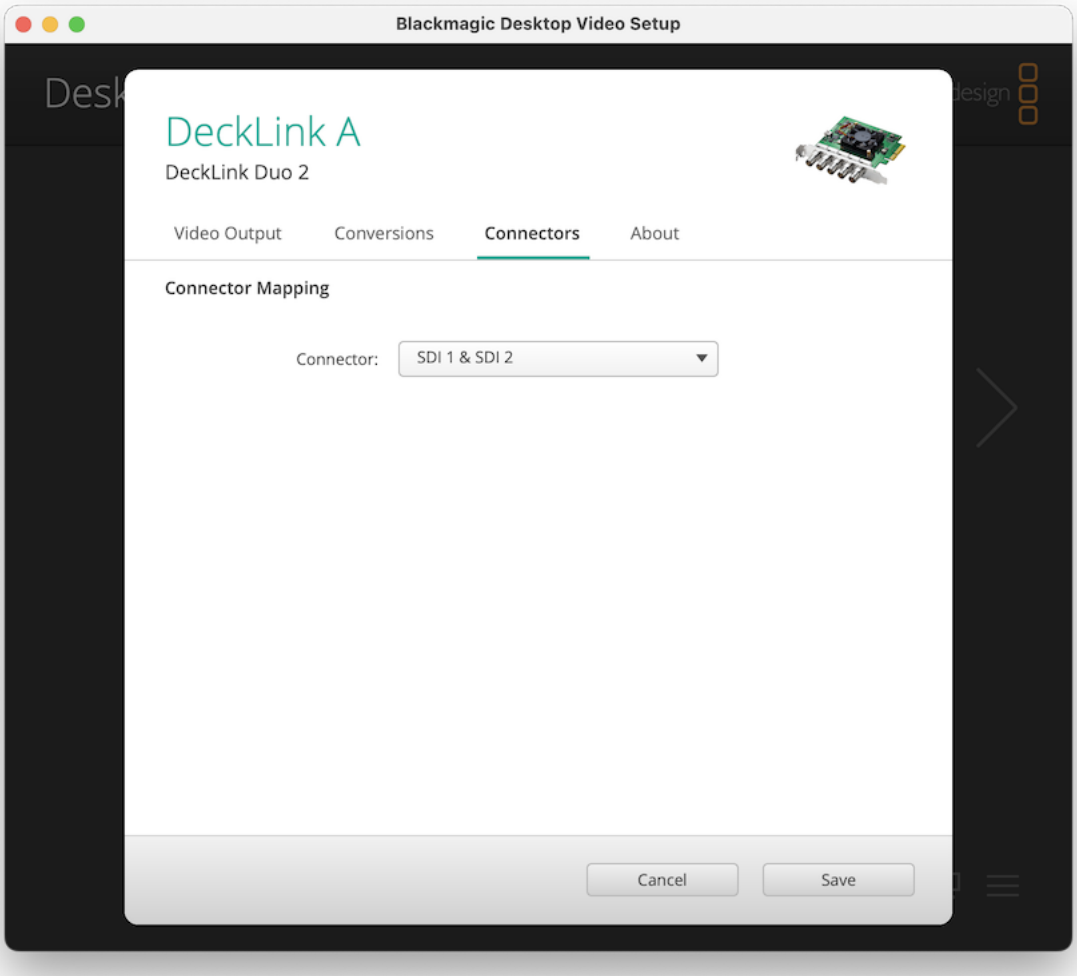
Some Blackmagic devices offer internal and external alpha keying, and QLab 5 supports keying on these devices. Alpha keying relies on transparency, or alpha, in the source image. If your source image uses a green or blue background, such as a Camera cue showing a performer in front of a green or blue screen, you can use a linear key video effect to convert the solid color background into a transparent background.

The *Keying* pop-up menu lets you choose between *None*, which is the default, *External* keying, or *Internal* keying.

Choosing Internal or External Keying

External keying uses two physical outputs on the device to output a pair of video signals called the "key" and the "fill" which can be sent to another video device such as a vision mixer. The vision mixer can then overlay the fill signal on top of another video source, using the key to "cut" out the parts of the fill that are supposed to be transparent. The Blackmagic device maintains frame-accurate synchronization between these two outputs.

If you are using a desktop or rackmount device, the two outputs in question are usually a pair of SDI connectors labeled "A" and "B". If you are using PCIe card device, you'll need to manually configure the card to use two SDI connectors for a single output. This is done in the *Connectors* section of the Blackmagic Desktop Video Setup app.



Using external keying requires hardware downstream of your QLab system to be able to take in and make use of a key/fill pair. Otherwise, the key signal will simply look like a greyscale version of the fill signal, and the fill signal will simply look like itself, but with any transparency replaced with opaque black.

Internal keying allows you to overlay the output of QLab on top of a separate live input to the Blackmagic device. This compositing is done inside the Blackmagic hardware and is extremely low-latency, so if you are using QLab to overlay graphics on top of a live video signal, you may prefer the visual results of routing the live signal through the Blackmagic device (and not through a Camera cue), and then using internal keying to superimpose your graphics.

Using internal keying requires an input signal to the Blackmagic device that does not pass through QLab. Otherwise, there's nothing to combine with and the output will look the same as it would if you set the keying mode to "none."

Many Blackmagic devices which support internal keying require all inputs and outputs involved to use the exact same resolution and frame rate. If you are having problems with internal keying, this is the best first thing to check.

Pixel format

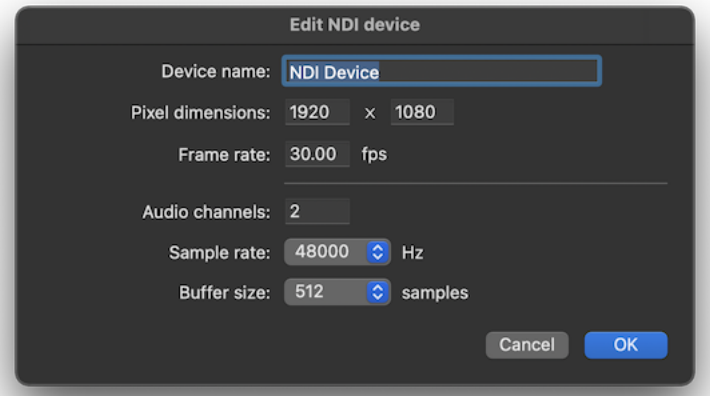
Most Blackmagic devices allow you to choose between two video output formats: BGRA and Y'CbCr. Whole books could be written about these two color models, so this manual will offer a few brief details and leave further research to the curious reader.

- **BGRA** offers the best theoretical picture quality, although its superiority may not be visible in many practical applications.
- **BGRA** must be used if you are using either of the optional keying modes detailed above.
- **Y'CbCr** may be required when using some resolutions above 1920 × 1080 or frame rates above 60 Hz.
- **Y'CbCr** may be required when connecting directly to some broadcast video equipment.

The guiding principle is this: if you are using a Blackmagic device which supports both BGRA and Y'CbCr formats, and your picture looks wrong, try changing this setting. If it looks right, it *is* right.

NDI Outputs

NDI outputs are virtual devices, and therefore do not have an inherent resolution, frame rate, or name. The **Edit** button opens the NDI device editor to allow you to configure those properties.

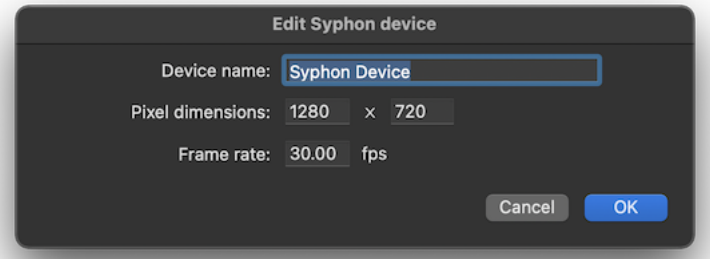


When a Video or Camera cue is assigned to a stage that uses an NDI device for output, that cue's audio output can use the audio portion of the NDI device if you like, but does not need to.

In order to remain predictable to external equipment, once a workspace starts sending video to an NDI output, QLab will keep that NDI output active until the workspace is closed, even if the workspace is panicked or hardStopped.

Syphon Outputs

Syphon outputs are also virtual devices, similarly editable.



In order to remain predictable to external equipment, once a workspace starts sending video to a Syphon output, QLab will keep that Syphon output active until the workspace is closed, even if the workspace is panicked or hardStopped.

Video Signal Flow Examples

There are several stages in QLab's video pipeline, each of which can modify an image in its own ways.

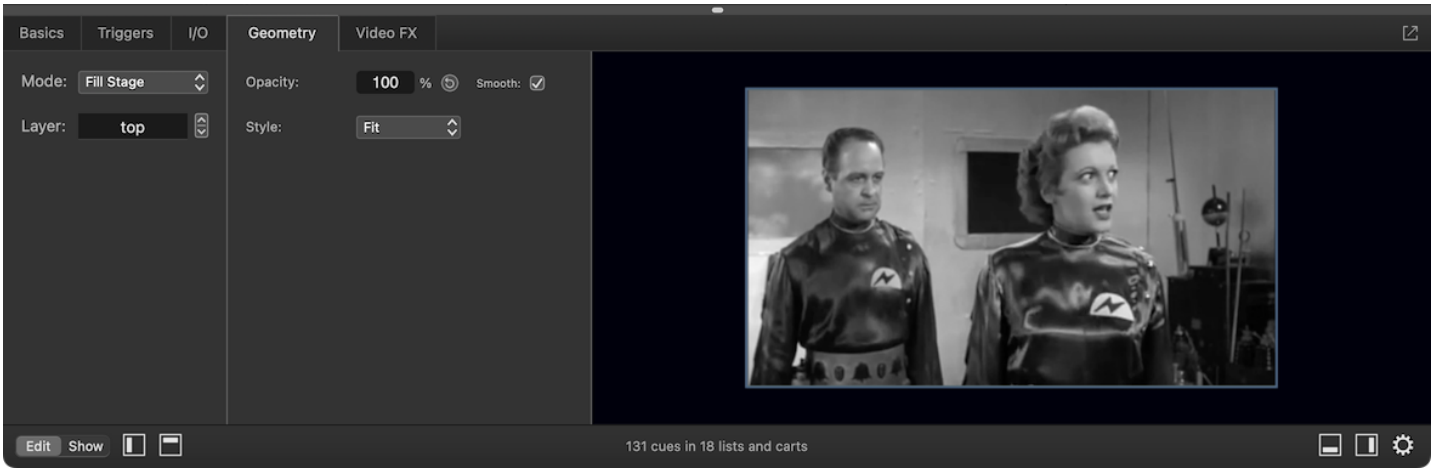
In brief, as discussed in the Introduction To Video In QLab, the video signal path in QLab does like this:

> **Source → Cue → Stage → Region → Route → Device.**

For Video cues, the source is a moving or still image file stored on your Mac. For Camera cues, the source is a video input patch which brings images in from a camera or other external source of live video. For Text cues, the source is the text entered in the Text tab of the inspector for the cue which is then rendered as a PNG image internally by QLab.

For these examples, we'll use a Video cue targeting a still image and set to fill stage mode. We could use a moving image target or another type of video-related cue and get the same results, of course, but this straightforward source cue helps make the examples clear. The cue is assigned to a stage named "Upstage Center".

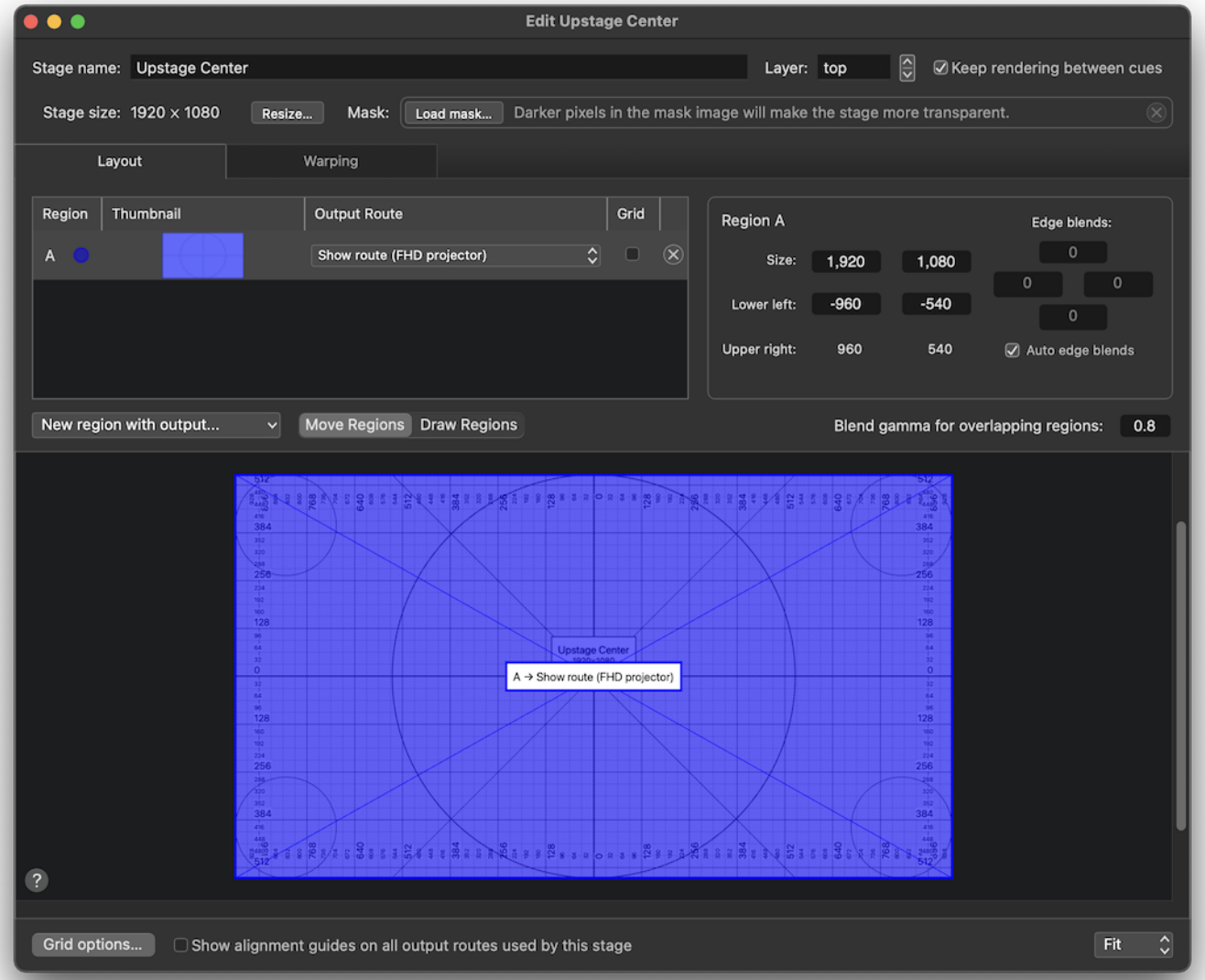
These examples also use Syphon to simulate a video projector. This lets us use the Simple Client, a very straightforward Syphon display app, to make nice screen shots of the output instead of setting up an actual projector, pointing it at an actual wall, and taking photographs.



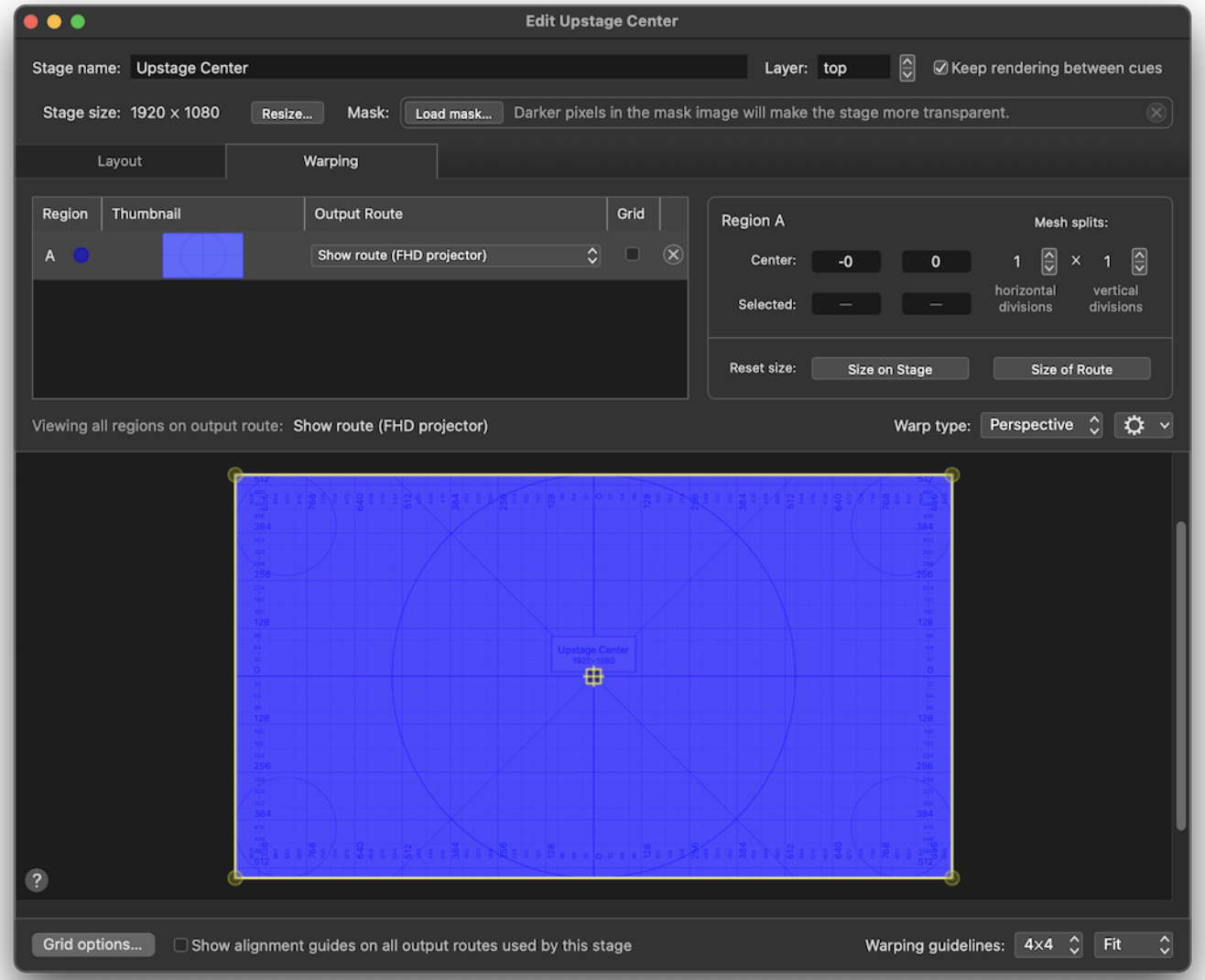
Example 1: one-to-one

If we're using a single projector for this stage, and pointing that projector at a single flat surface like a wall, screen, or cyclorama, our needs might be very simple: just display the whole cue without doing much to it.

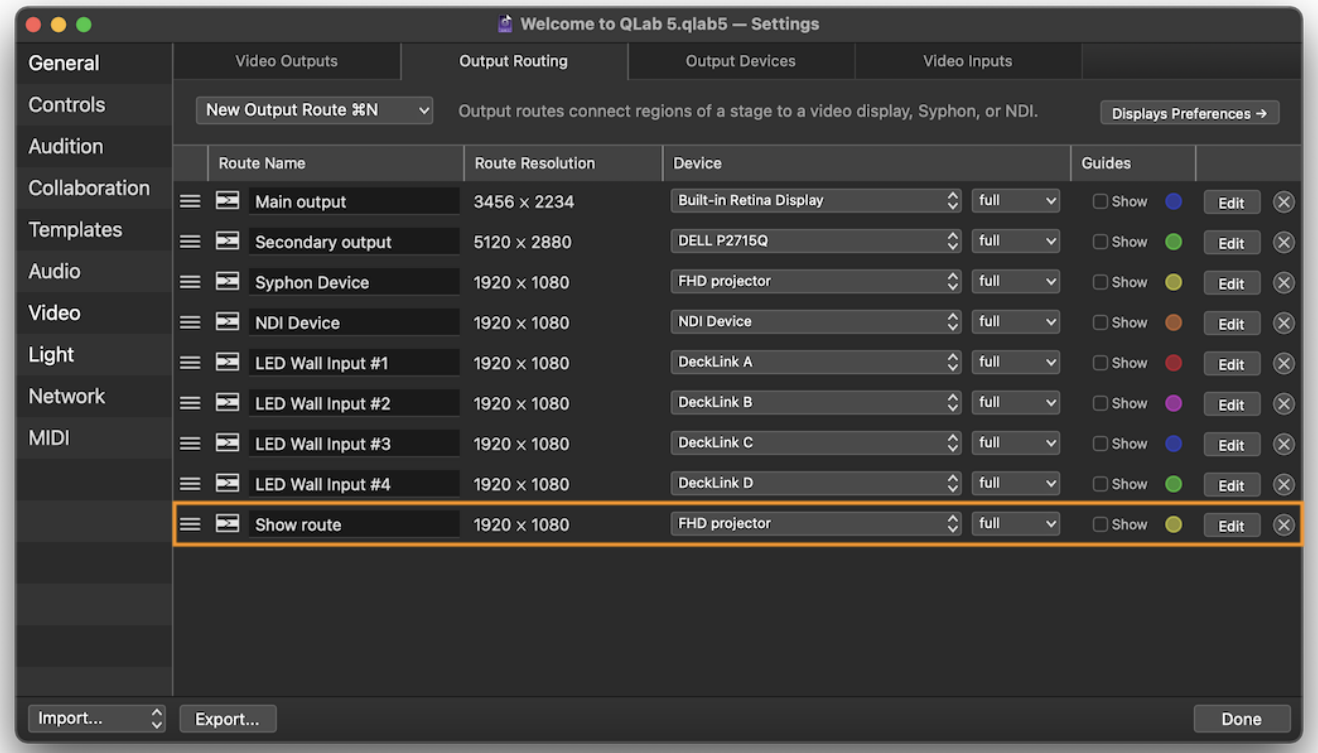
To achieve this, the stage has a single region which covers the whole stage in the Layout tab of the stage editor:



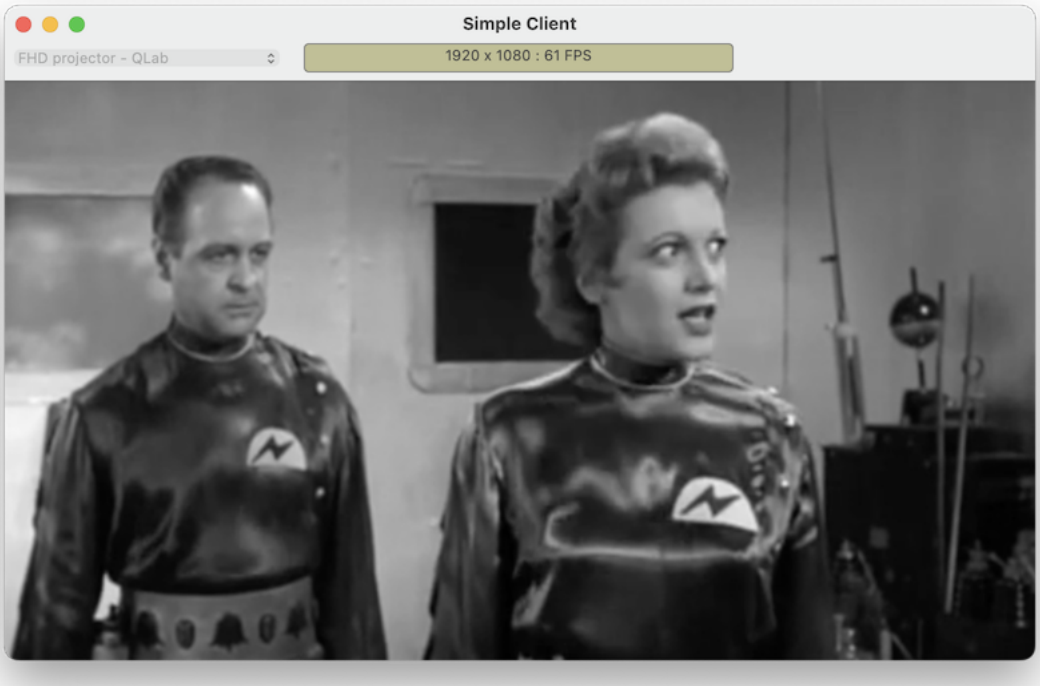
That region also fills the whole output route in the Warping tab of the stage editor:



The route is configured to use the full raster of its assigned output device:



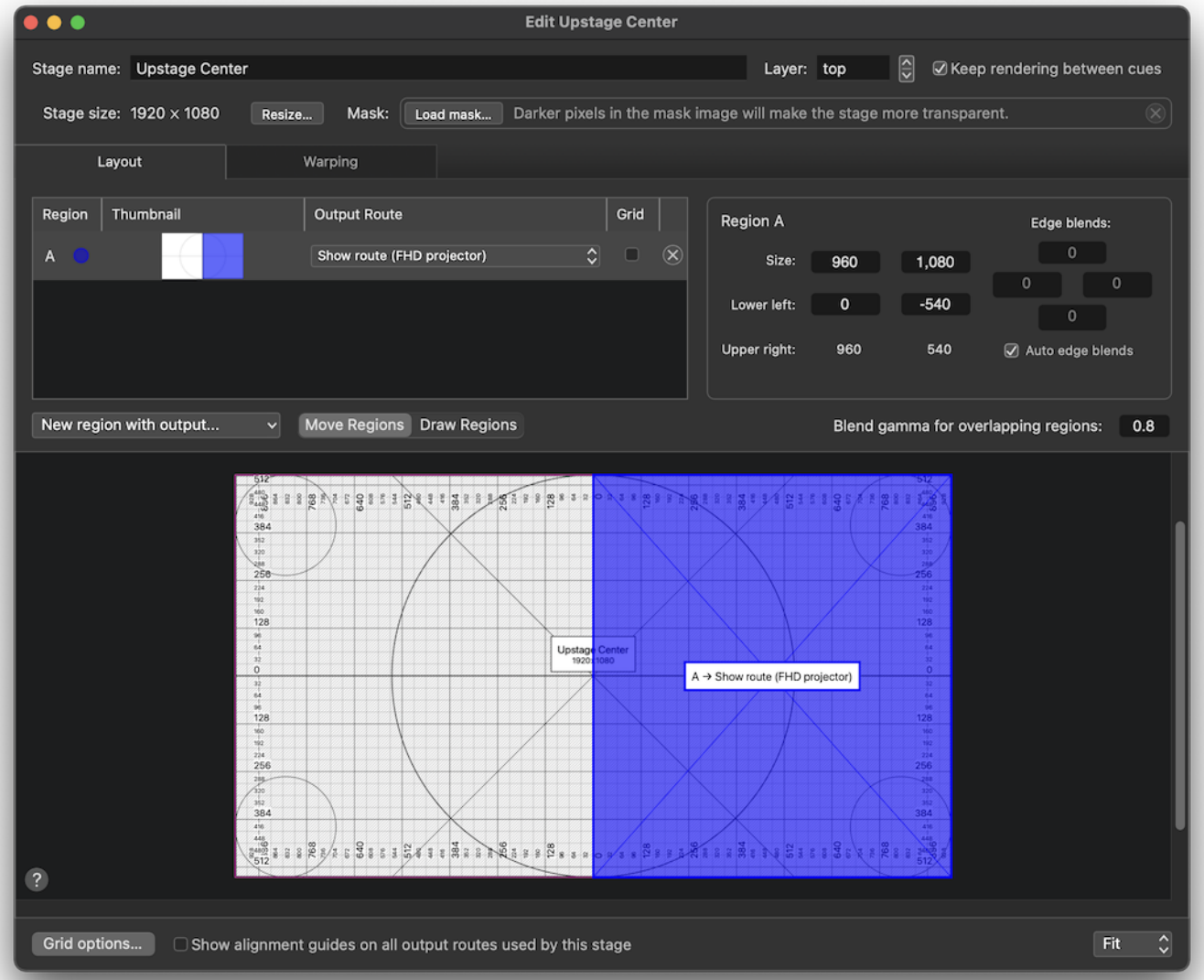
The result is that every pixel of the device is utilized to display every pixel of the cue. Since the stage, route, and device all have a resolution of 1920 × 1080, there is a straight one-to-one correspondence between the cue's pixels and the device's pixels.



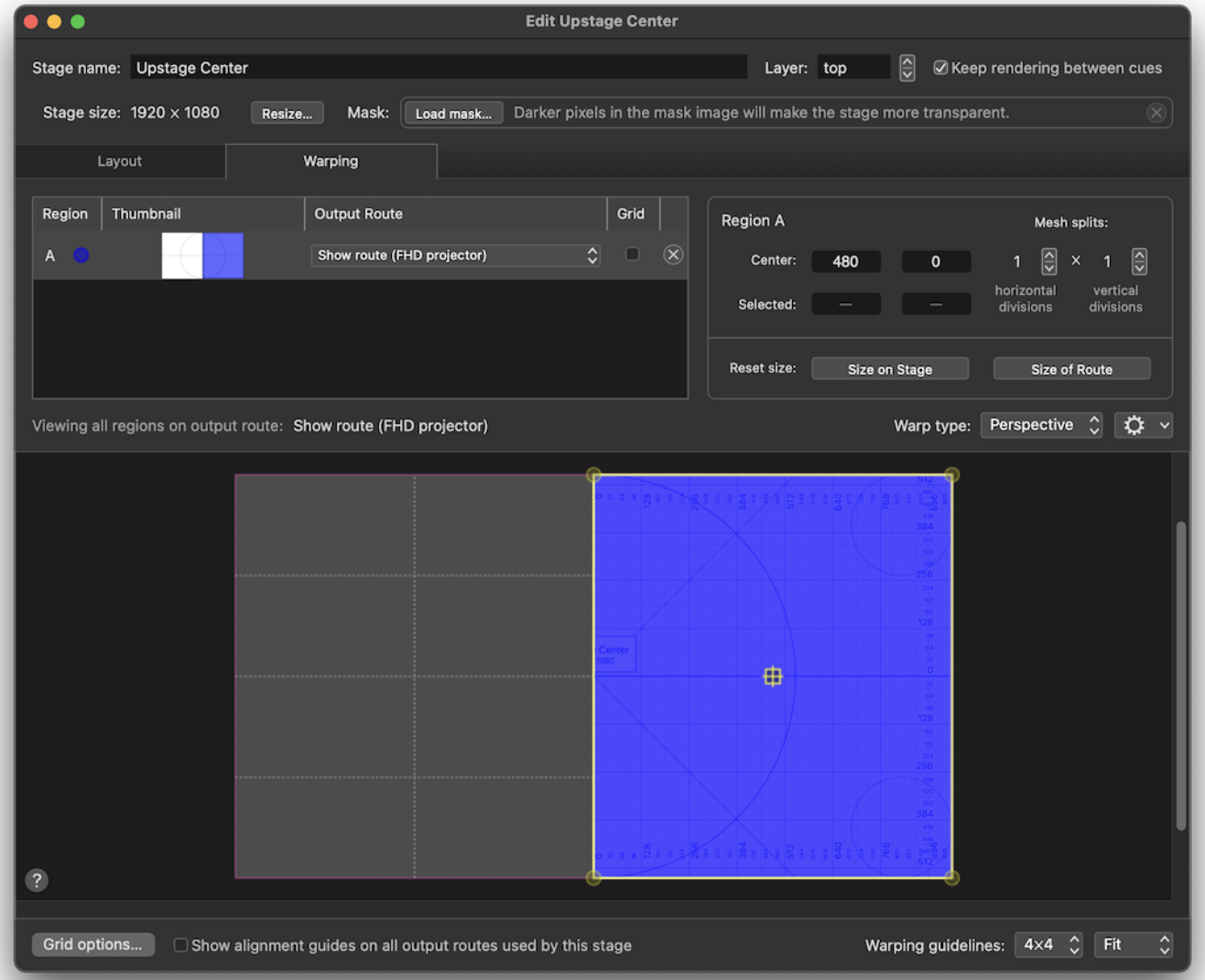
Example 2: right half

Perhaps the left half of the projector beam is hitting scenery which we don't want to project upon, but we want the right half to be left exactly as it was.

To achieve this, the region only covers the right half of the stage in the Layout tab of the stage editor:



The region similarly covers only the right half of the route in the Warping tab:



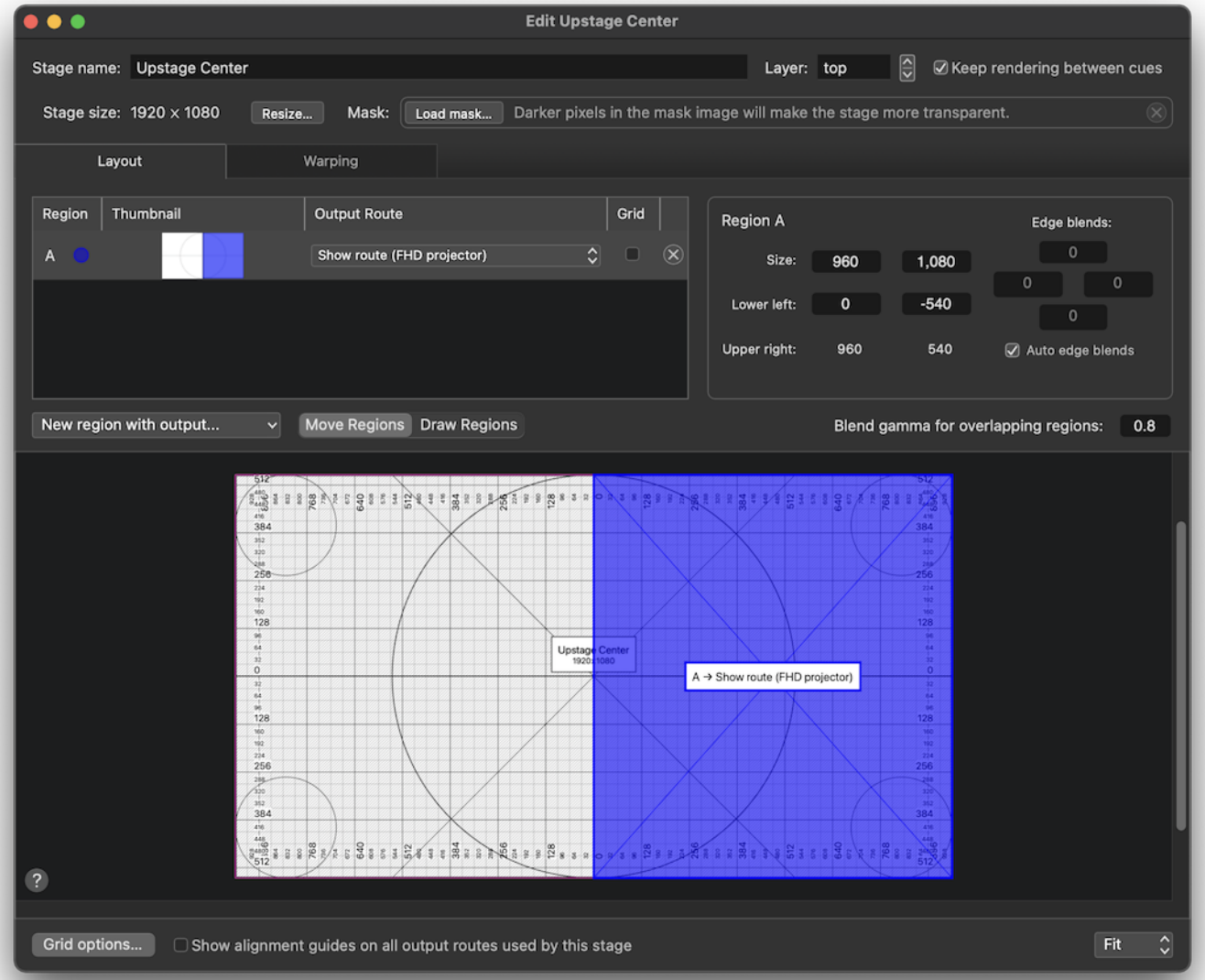
The result is that the right half of the cue is displayed using the right half of the device.



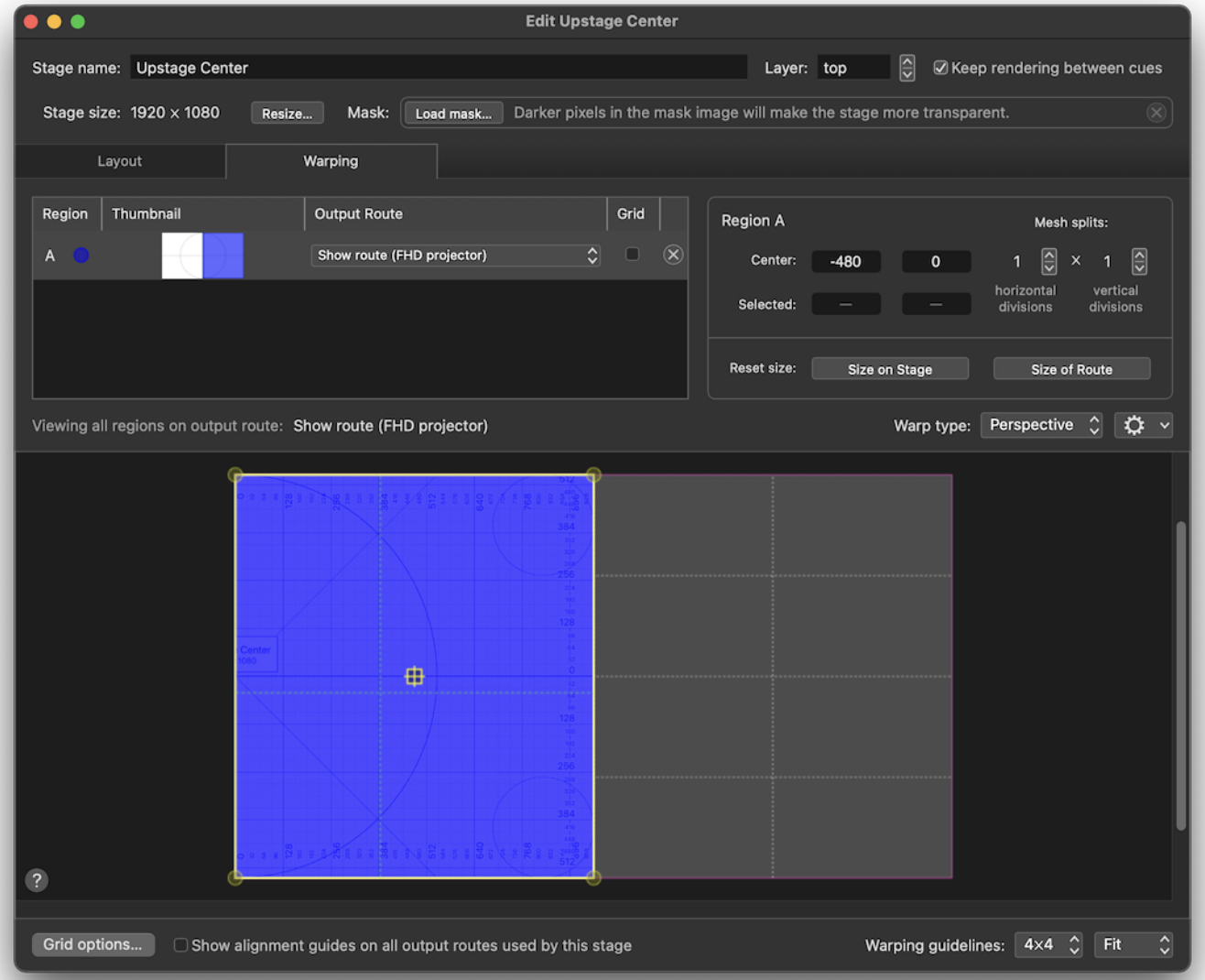
Example 3: right half reversed

What if we want to use the right half of the source image, just like in example 2, but we want to display it on the opposite side of the screen?

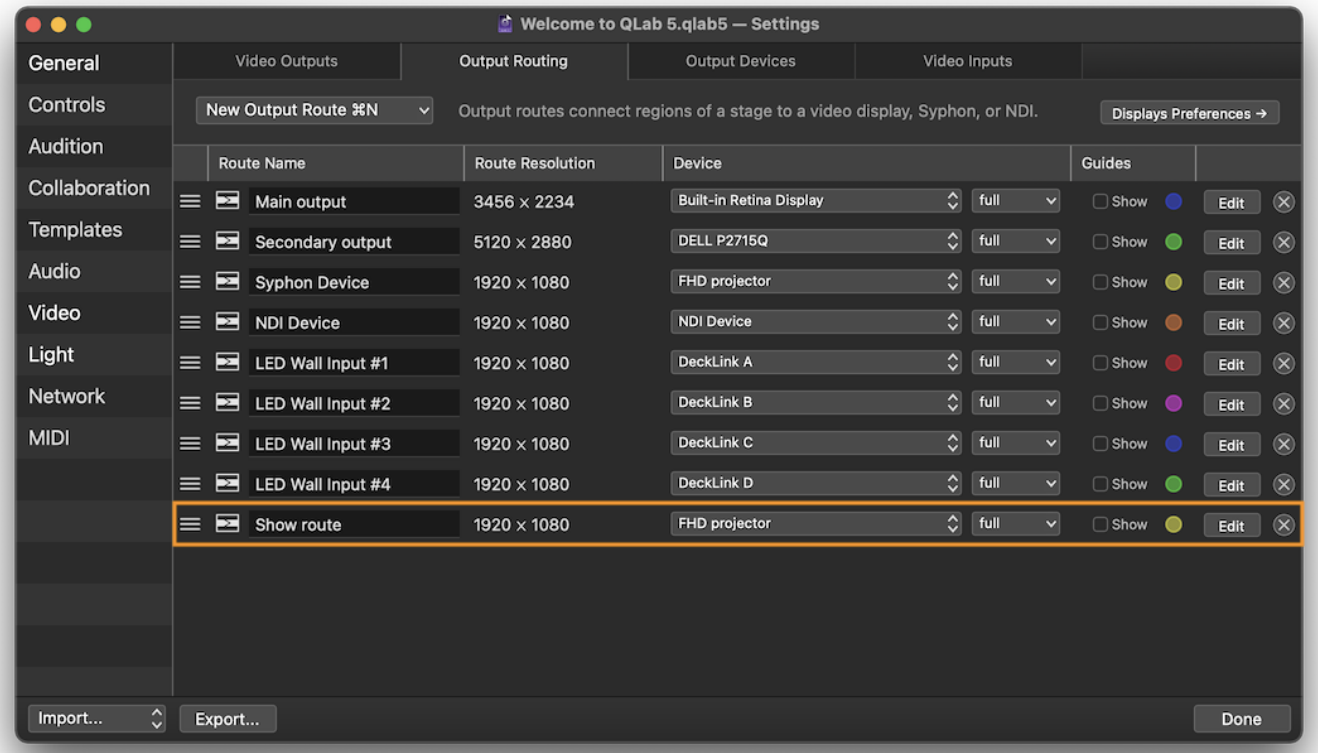
To achieve this, the Layout tab looks the same:



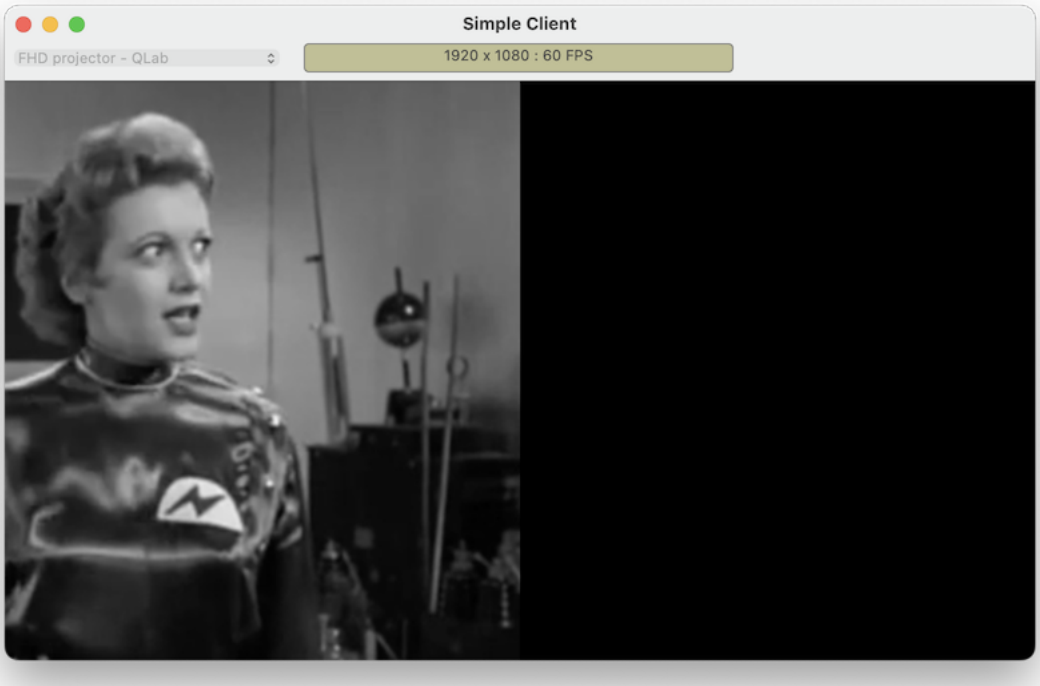
But in the Warping tab, we move the region to the left half of the route:



The route stays the same:



The result is that the right half of the cue is displayed using the *left* half of the device.



1. *Raster* originally meant the grid of dots in a television which produced a visible image. It now refers to the pattern of pixels which represent a digital image, both as encoded data within the computer and as a physical collection of pixels in a display device like a monitor or projector.↩
2. Full High Definition, or 1920 × 1080 pixels.↩

Using NDI

NDI, which stands for Network Device Interface, is a standard that allows cameras, video switchers, computers, and other devices to send and receive high-quality video over a regular IP network. NDI streams can contain video at any resolution, frame rate, aspect ratio, and format, as well as multiple channels of embedded audio, control signals, and metadata. Starting with QLab 5.0, you can use incoming NDI streams as sources for Camera cues, and publish Stages as NDI streams for other devices to receive.

QLab uses the format `DEVICE_NAME (NDI_SOURCE_NAME)` to uniquely identify NDI streams. For this reason, it's best not to have two (or more) devices on the same network which have the same name. If there are multiple devices with the same name, NDI automatically adds a numerical suffix which prevents problems, but is difficult to troubleshoot. Furthermore, there is no guarantee that the same-named devices will appear in the same order after a restart of the system, so the numerical suffixes can change.

Using NDI input with Camera Cues

To use an incoming NDI video source with a Camera cue, first go to Workspace Settings → Video → Video Input. Then, either create a new video input patch using the button on screen or the keyboard shortcut **⌘N**, or identify an existing video input patch that you want to use.

When you click on the *Device* pop-up menu next to the video input patch you've chosen, you'll see all available video input signals listed by category. NDI video sources are listed towards the bottom under the rather appropriate heading **NDI sources**. Select the source you wish to use.

Now, in the cue list, you can create a Camera cue, navigate to the I/O tab of the inspector, and select the video input patch that you just configured. If the NDI source that you selected includes audio, you can also choose how many channels of that audio you wish to use within the cue.

Camera cues using NDI sources behave just like any other Camera cue.

Setting Up Your Network for NDI

NDI has fairly low-intensity networking needs, with no esoteric hardware or stringent technical requirements. There are a few basic rules, though, and some helpful minimum specs to keep in mind.

NDI Networking Basics

- Every device that you want to use together via NDI needs to be on the same subnet, unless you're using NDI Bridge.
- The network must not block any of the ports listed below.
- The network must support multicast traffic.
- The network must have sufficient bandwidth as described below.

Network Ports used for NDI

- NDI uses mDNS to detect sources on a network. mDNS uses TCP port **5353**.
- NDI uses TCP port **5960** for messaging between servers and clients.
- Most NDI devices send their first video stream on port **5961**, their second stream on **5962**, and so on with one port for each stream. Older NDI devices made prior to 2016 use ephemeral ports for streams, with each stream using a port somewhere in the range of **49152** to **65535**. Unless you're sure which ranges of ports are used by the devices you're using, it's safest to assume that access to both ranges of ports will be necessary. QLab uses ports starting at **5961** and counting up from there.

Networking Tips from NewTek

While NDI does not require specific networking hardware, NewTek provides some guidelines which can be helpful:

- Use network switches with 1 Gbps (or higher) full duplex ports.
- Turn auto-negotiation **off** for any port used for NDI traffic.
- Turn EEE (Energy Efficient Ethernet) **off** for any port used for NDI traffic, or use switches that do not support EEE.
- Always be sure to use properly terminated ethernet cables.
- Use “basic line-cascade” network structures; avoid “pyramid” or “tree” network structures.
- QoS settings are not required, but if you use Qos, always configure NDI traffic as *real-time* and *high priority/time critical*.
- Network packet loss should be 1% or less.
- One-way latency between any two devices using NDI should be 150 ms or less.
- Jitter should be 30 ms or less.

NDI Bandwidth Examples

Examples below include 16 channels of embedded audio.

Example NDI video stream	Approximate bandwidth required
1 × SD video stream	20 Mbps
1 × 720p60 video stream	90 Mbps
1 × 1080i60 video stream	100 Mbps
1 × 1080p60 video stream	125 Mbps
1 × UHDp30 video stream	250 Mbps
1 × UHDp60 video stream	400 Mbps

The total bandwidth of your network should be high enough that all NDI traffic combined does not exceed 75% of the capacity of the network.

Troubleshooting Links

- NDI Documentation
- Recommended Network Switch Settings for NDI
- NDI Tools

Fading Video & Video Effects

A Fade cue can be used to adjust the geometry and video effect parameters of a targeted Video, Camera, or Text cue.

Fade cues can also adjust audio parameters of Video cues, Camera cues, Audio cues, Mic cues, and audio output patches. When a Fade cue is selected, the inspector will only show the tabs relevant to the type of cue that the Fade cue is targeting.

The word “fade” can often be taken to mean one thing or another, but in QLab “fade” simply means “change a value over time.”

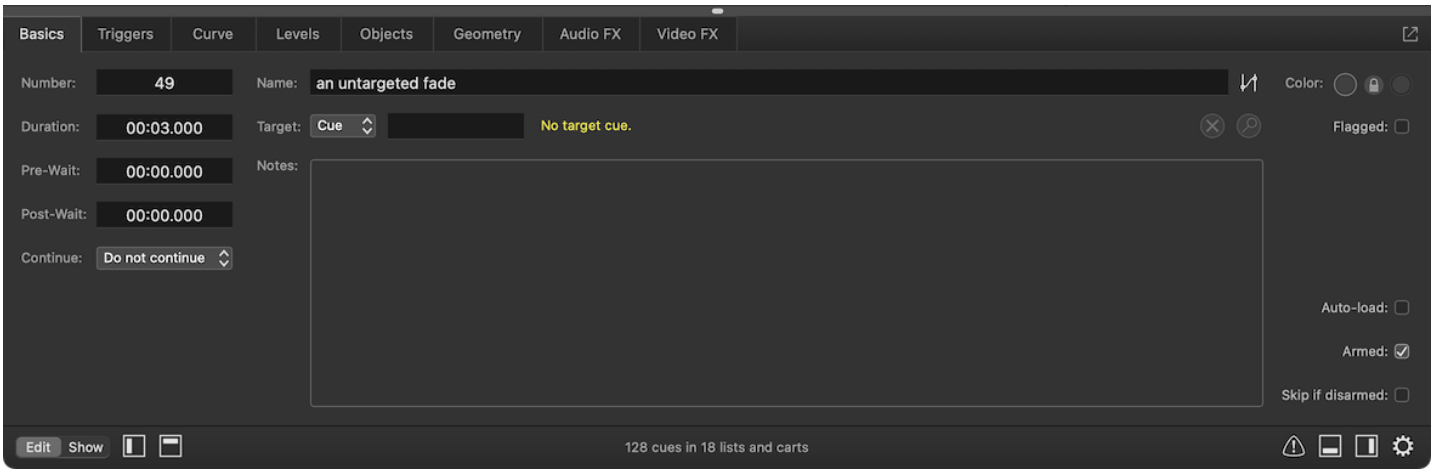
Fade cues require a cue target or a workspace settings target, have a duration, and must adjust at least one level or parameter of their cue target in order to be considered functional.

The Inspector for Fade Cues

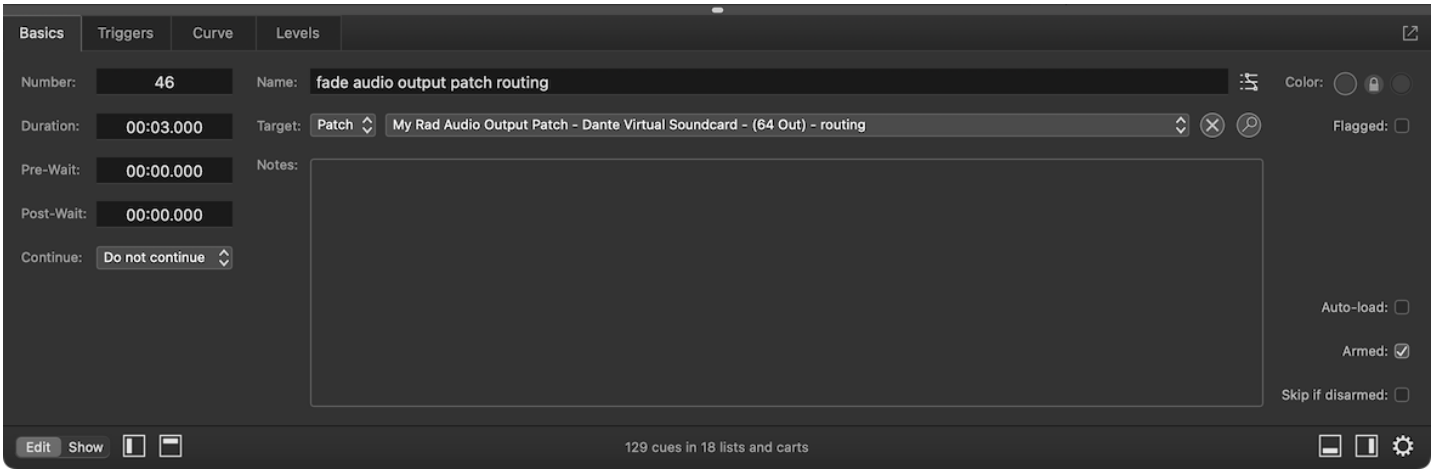
The tabs shown in the inspector when a Fade cue is selected vary based on the more of the Fade cue and its target.

The Basics Tab

The Basics tab is always visible, but has a small but important difference from how the Basics tab usually appears.



Because Fade cues can target both cues and patches, a pop-up menu appears next to the Target field to switch between cue targeting and patch targeting. When *Patch* is selected from this menu, a second pop-up menu appears which allows you to select the patch that you want to fade.



The Triggers Tab

The Triggers tab is always visible and behaves the same way that it always does.

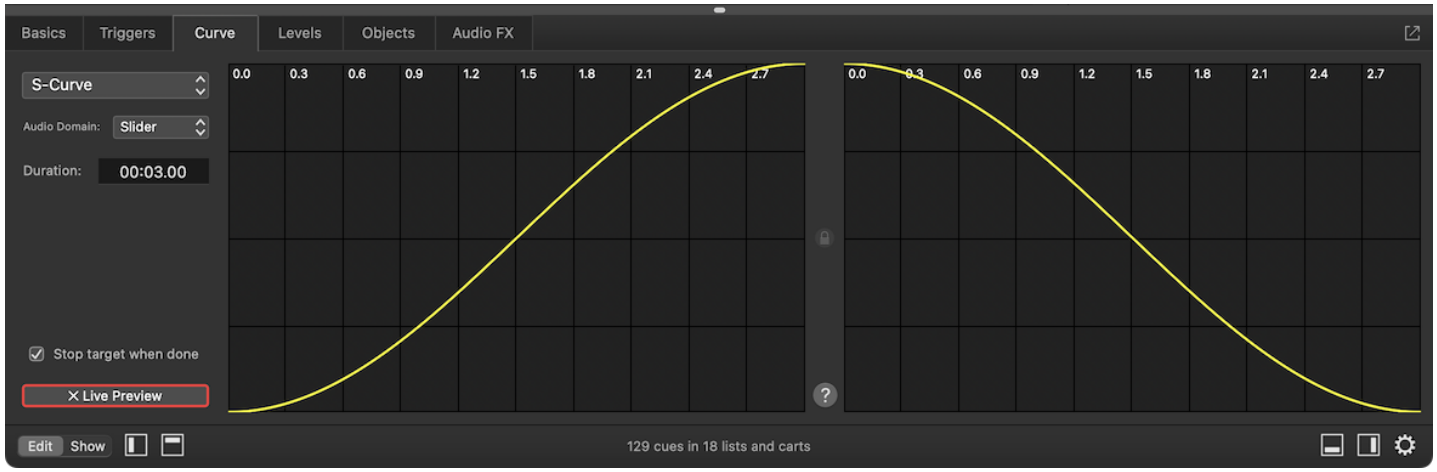
The following tabs are visible when a Video cue, Camera cue, or Text cue are the target of the Fade cue:

The Curve Tab

The fade curve, drawn in yellow on the right side of the tab, determines the rate of change of the parameters being faded. The curve on the left is for levels which increased by the fade, and the curve on the right is for levels which are decreased by the fade.

The horizontal axis of the curve represents time and the labels across the top will change based on the duration of the Fade cue. The vertical axis represents percentage of the total change made by the Fade cue. For the rising curve, the one of the left, the bottom left corner represents the beginning of the fade, which is to say "time = 0, completion = 0%." The top right corner represents the end of the fade, or "time = (duration of the Fade cue), completion = 100%." For the falling curve, the top left corner represents the beginning and the bottom right corner represents the end.

The curve shape that appears by default is set according to the Fade cue's cue template, but you can choose another fade shape from the pop-up menu in the top left corner of the tab.



There are four options for Fade curve shapes:

- **S-Curve.** QLab's default curve shape follows an "ease-in, ease-out" envelope designed to sound natural with audio levels and look smooth with video geometry.
- **Custom Curve.** This option allows you to click anywhere along the fade curve and a create control points, which can be dragged to change the shape of the curve. Moving control points will smoothly bend the curve in the direction of the control point. To delete a control point, click on it to select it and press the **delete** key on your keyboard. To start over entirely, click *Reset to Default Shape* in the bottom left corner of the tab. The lock button between the two curves allows you to mirror the left and right curves. When the curves are locked, any change to either curve will apply to both curves.
- **Parametric Curve.** This option adds a text field labeled *Intensity* below the pop-up menu which allows you to use a mathematically precise parametric fade shape.
- **Linear Curve.** This option is similar to the *custom curve* option but instead of smoothly ending the curve, control points create a precise, sharp bend. To delete a control point, click on it to select it and press the **delete** key on your keyboard. To start over entirely, click *Reset to Default Shape* in the bottom left corner of the tab. The lock button between the two curves allows you to mirror the left and right curves. When the curves are locked, any change to either curve will apply to both curves.

Both the rising and the falling curve use the same fade shape type, but if you use *custom curve* or *linear curve* you can create individual curve shapes for each.

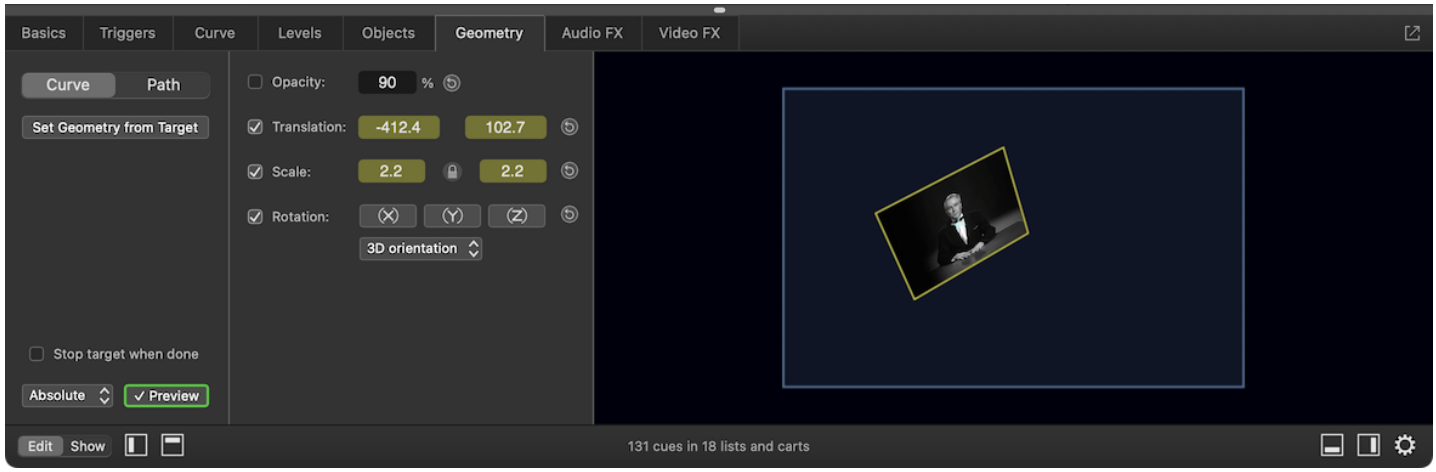
The *Audio Domain* pop-up menu is only relevant if the target cue contains audio and if anything pertaining to audio is being faded. You can learn about this control from the Fading Audio section of this manual.

The Levels Tab

The Levels tab will only appear if the target cue contains audio. It behaves the same as the Levels tab for Fade cues targeting Audio cues.

The Geometry Tab

The Geometry tab allows you to specify which parameters of the target cue you wish to fade, and what their final value will be.



The left side of the Geometry tab contains a few setup tools:

The **Curve/Path** buttons switch the tab between curve mode, which fades the selected parameters (that is, the ones whose checkboxes are checked, and which are colored yellow) according to the settings in the Curve tab, and path mode, which fades two parameters simultaneously according to a 2D path, discussed below.

Clicking the **Set Geometry from Target** button, or using the keyboard shortcut ^⌥⌘**V**, will invoke the *paste cue properties* sheet in a special way. First, it will behave as though the target cue was selected and copied, and second it will automatically select the "Video" set of properties to paste. You can choose other properties if you like, but if you simply hit the enter key, QLab will paste the geometry from the target cue onto the Fade cue. This is a convenient way to get started building a fade, as it will help you keep track of the starting point from which you will be fading.

Stop Target When Done. If this box is checked, the target cue will stop once the fade is complete. If the box is unchecked, the target cue will continue to run after the fade is complete.

Absolute Fade. This drop-down menu lets you choose between an absolute fade, which is QLab's default, and a relative fade. Relative fades are discussed in detail below.

Preview. This button provides quick access and a visual reference to the live fade preview setting.

The center section of the Geometry tab contains controls for the fade-able parameters of a Video, Camera, or Text cue. By default, the checkbox next to each parameter is unchecked which means that the Fade cue will not adjust that parameter. To "activate" a parameter, so that the Fade cue will adjust it, check the box next to that parameter. To reset a parameter to the default value for that parameter, click the :button-reset-alt-5: reset button next to that parameter.

Opacity. You can fade the opacity of the target cue to any whole-number opacity between 0% and 100%.

Translation. You can move the target cue left to right and up to down by fading translation. Change the position of the video along the X-axis and/or Y-axis by entering values in the text fields or by clicking and dragging the thumbnail image in the preview box to the desired location.

Scale. You can scale the target cue in both the X and Y axes together, or individually by clicking the icon to unlock the axes and adjust them separately. You can also adjust the scale by scrolling vertically on the thumbnail image in the preview box.

Rotation. Click and drag on each of the **(X)**, **(Y)**, and **(Z)** buttons to fade the rotation of the target cue.

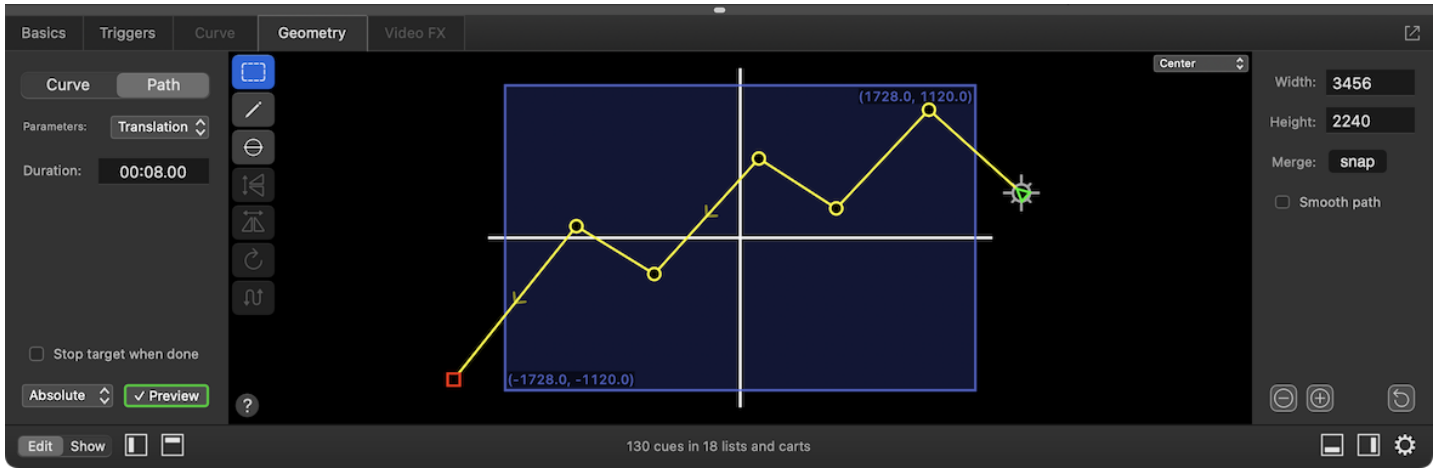
3D orientation. By default, Fade cues use three-dimensional quaternion math to rotate their target. This results in smooth, natural movements when rotating along multiple axes, but necessarily means that a fade will never pass through more than 180 degrees of movement in each axis. If you instead want to rotate the target a specific number of degrees about a single axis, you can use this pop-up menu to choose an axis, and then enter the number of degrees you wish to rotate.

The right section of the Geometry tab contains a blue preview box, representing the stage that the target cue is assigned to, and a thumbnail image of the target cue showing the finished results of the fade.

2D Paths

If the **Curve/Path** switch is set to *Path*, the other inspector tabs besides Basics and Triggers are disabled. In QLab 5.4, Fade cues can only be used to fade the translation, rotation, and scale of Video, Camera, and Text cues. Then, the Geometry tab allows you to select the parameters that will be faded by the cue, and edit the 2D path along which they will fade. The fade path is shown in the center canvas area of the tab as a yellow line over a blue background, with the following details:

- The starting point of the path is drawn as a green triangle.
- The ending point is drawn as a red square.
- Points along the path are drawn as yellow circles.
- If there is enough space, small directional carets pointing away from the starting point and towards the ending point will appear along the path.
- The current live position of the parameters being faded is indicated by the icon.
- The home position of the parameters being faded is indicated by the icon.



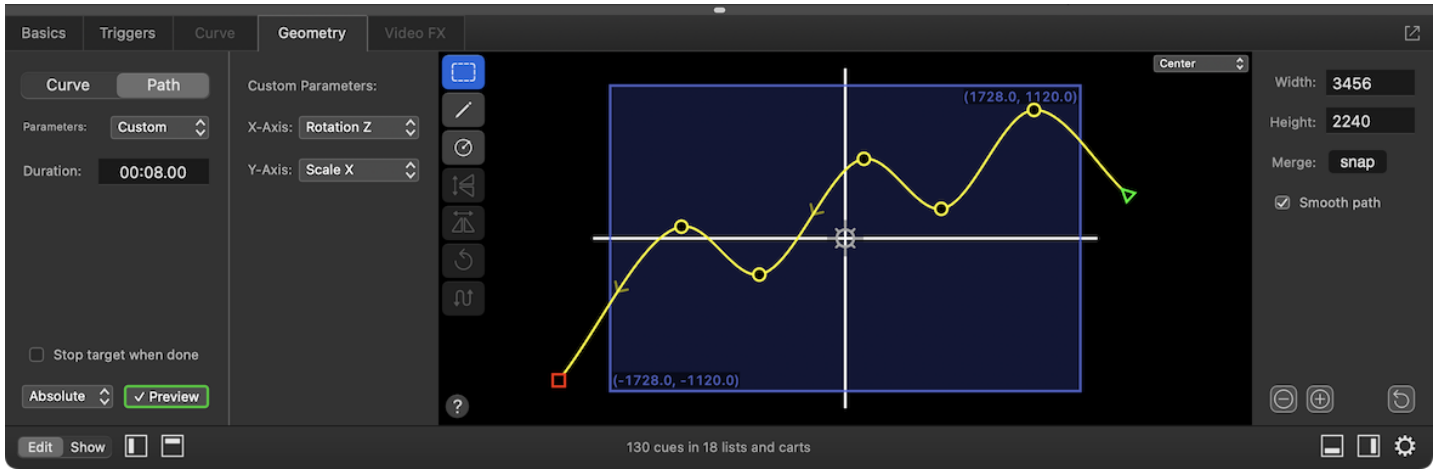
The left side of the path tab contains a few setup tools:

The **Curve/Path** button is described above.

The **Parameters** menu lets you select the parameters which are represented by each axis of the canvas view in the middle of the inspector. The default setting is *translation*, meaning that the X axis (left to right) of the Fade cue represents the X-axis translation of the target cue, and the Y axis (up to down) of the Fade cue represents the Y-axis translation of the target cue.

If you choose to use custom parameters, then each axis of the path can be assigned to any available axis of translation, rotation, or scale.

In the screen shot below, the **Parameters** menu is set to *Custom*, then the **X-Axis menu** is set to *Rotation Z* and the **Y-Axis** menu is set to *Scale X*.



With these settings, as the Fade cue runs, the target cue's Z-axis rotation will steadily go down, while the X-axis scale will oscillate up and down while trending down overall.

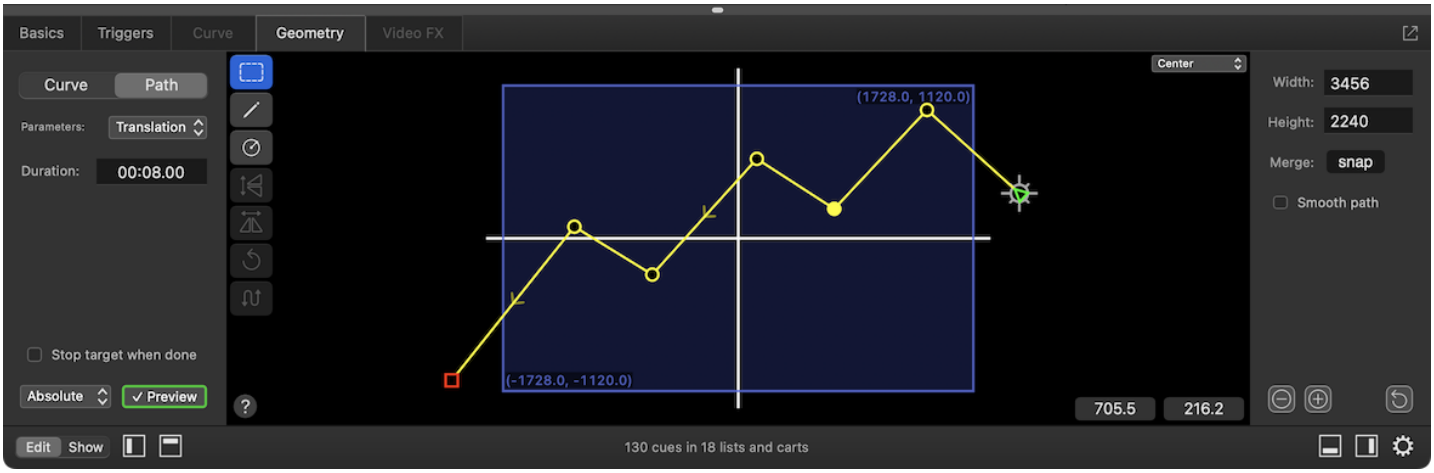
Stop Target When Done. If this box is checked, the target cue will stop once the fade is complete. If the box is unchecked, the target cue will continue to run after the fade is complete.

Absolute Fade. This drop-down menu lets you choose between an absolute fade, which is QLab's default, and a relative fade. Relative fades are discussed in detail below.

Preview. This button provides quick access and a visual reference to the live fade preview setting.

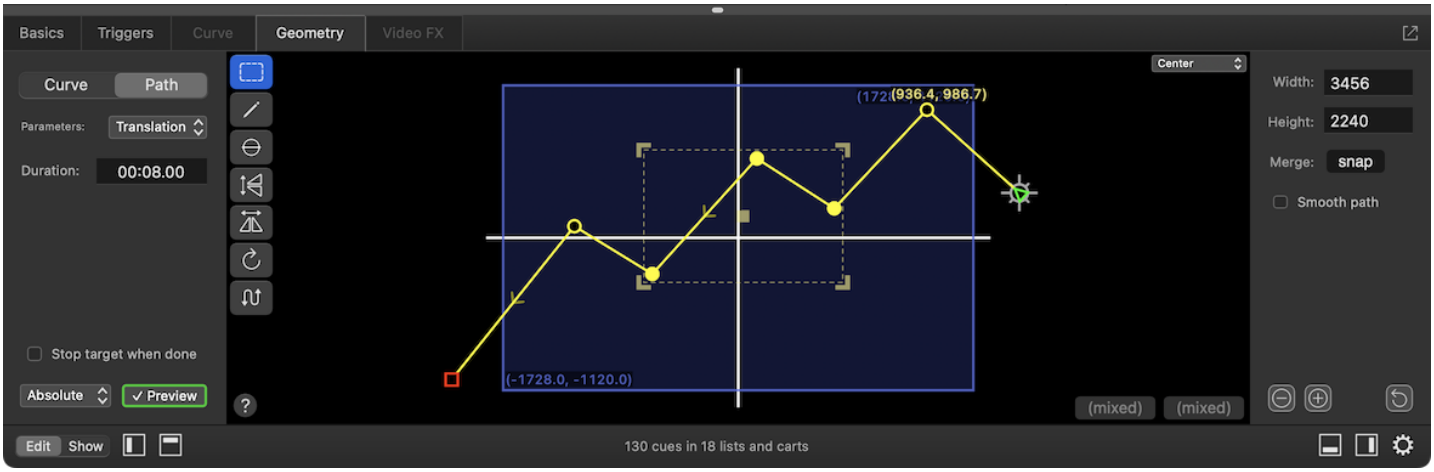
The center section of the Path tab displays the path and provides tools for you to use to create and edit the path.

The **select** tool lets you click on individual control points to select them for precise editing:



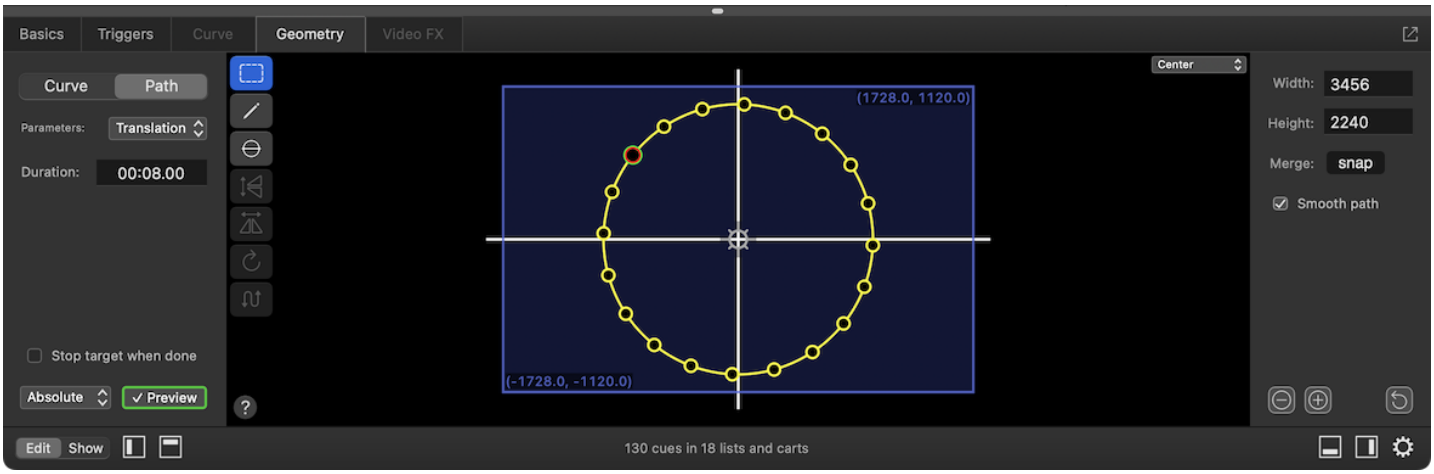
With one point selected, you can use the text fields in the lower right corner to type in a value or click and drag up and down to adjust the value.

You can also click and drag a selection box around several control points:



The **draw** tool lets you click to add points to the path, or click, hold, and drag to draw a freeform line. Holding down the option key (⌥) key allows you to add a point in the middle of a path, otherwise each drawn point will be added to the end of the path. Double-clicking while using the draw tool finishes drawing and switches to the select tool.

The **circle** tool lets you draw a perfectly circular path. The first click sets the starting (and ending) point of the circle, and the second click defines the diameter of the circle. Holding down the option key (⌥) allows you to instead click once to define the center point, and then click again to define the radius of the circle.

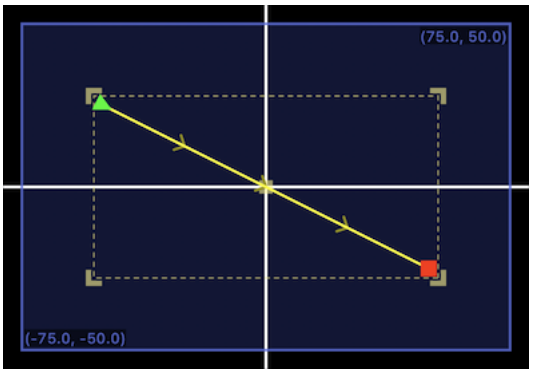


Flip vertical flips the selected group of points around the X axis.

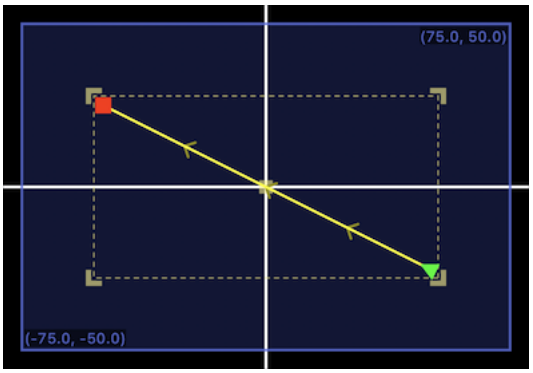
Flip horizontal flips the selected group of points around the Y axis.

Rotate spins the selected group of points in 90 degree increments around the center point of the selection.

Reverse leaves the points in the same location, but reverses their order in the path. For example, consider this simple path with two points:



After using the reverse tool, what was the start point is now the end point, and vice versa:



The ⓘ **help button** displays a list of the keyboard shortcuts and modifiers for working with 2D fades.

The **alignment** menu in the upper right corner of the center area lets you choose where the axis is placed on the canvas. By default, it is placed at the center, both horizontally and vertically, corresponding with QLab's center origin point for video geometry. However, this menu allows you to place the axis in one of the corners to help achieve your mathematical goals. It has no functional difference beyond how the canvas is displayed, and therefore what sorts of values are easiest to enter.

To the right side of the canvas are a few more controls for configuring the canvas and the overall behavior of the path:

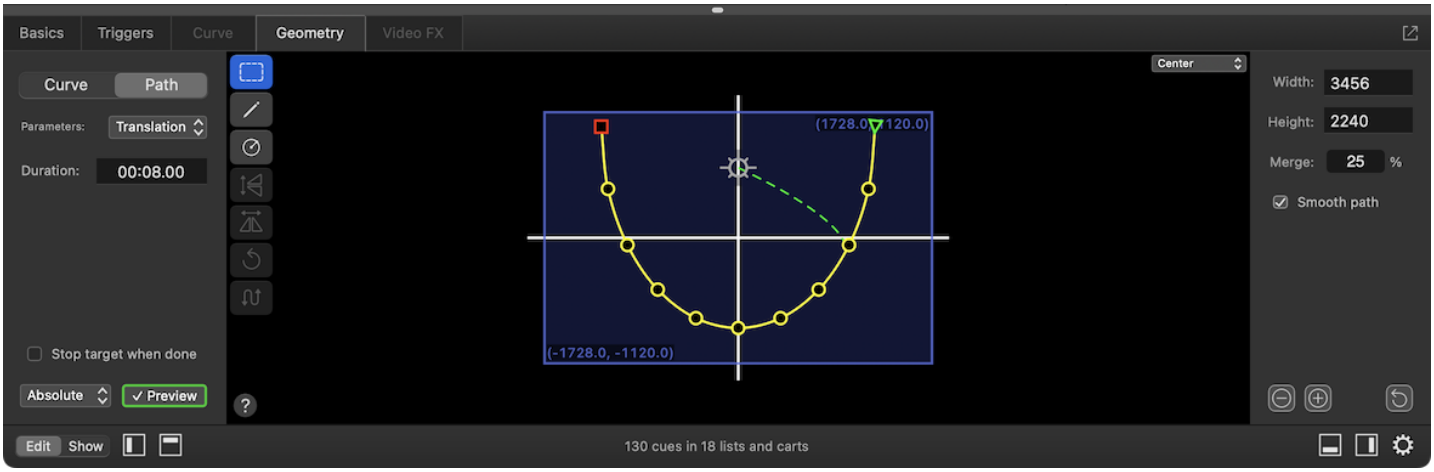
Width and **Height** allow you set the width and height of the blue rectangle. By default, QLab uses the size of the stage that the cue target is assigned to use, but you can enter any values you like. These values are not a limit; the Fade cue can send values outside the defined range. They are simply tools

to help you visualize the extent of the space within which you are fading.

If you were using this Fade cue to fade scale, for example, you may prefer to set the width and height to the maximum scale that you expect to fade to.

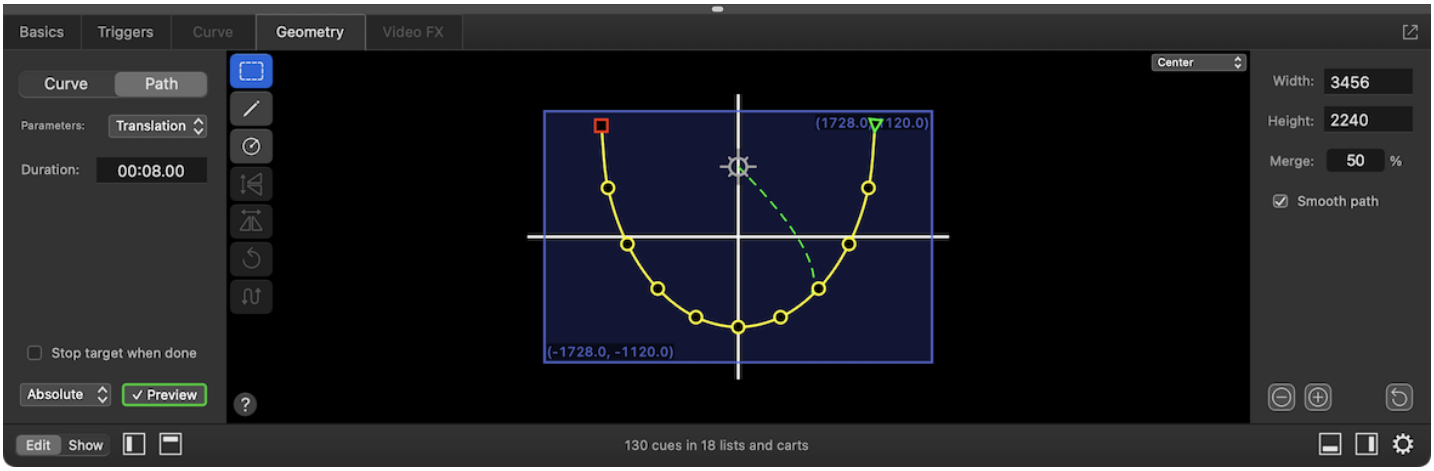
Merge percentage. If the starting point of the path does not align with the current position of the target cue, QLab can make the fade smoother by merging the current position of the target with the path. The merge percentage is the point along the path by which the target cue will "catch up" with the path. If the target is very close to the path, a merge may complete before the specified percentage, but it will never complete later.

For example, if the merge is set to 25%, when the Fade cue starts running, the target cue will start moving organically towards the values indicated by the path starting from wherever it is currently positioned. By the time the Fade cue is 25% complete, the target cue will have arrived on the path, and the remaining 75% of the fade will proceed as drawn.



The interpolated path is drawn as a dotted green line, showing you exactly where the target cue will travel from its current live position, indicated by the icon, to meet up with the path.

A higher merge percentage will result in a smoother meeting if the target cue is far away from the starting point of the path, but of course it will also result in more of the movement being off the path as drawn.



Smooth path. When this box is un-checked, paths will be drawn precisely between each control point. When the box is checked, the path will be smoothed between points to create a naturally curving line.

You can use the ⊖ and ⊕ buttons to zoom in and out on the canvas. If you're using a trackpad, you can zoom in and out by placing the cursor over the canvas and pinching in and out. If you're using a mouse or trackball with a scroll wheel, you can zoom in and out by holding down the option key (⌥) and scrolling vertically.

While zoomed in, you can pan around using two-finger swipes on a trackpad. If you have a mouse or trackball with a middle button (oftentimes the scroll wheel can be clicked; this is usually the middle button,) you can pan around my holding that button down.

The **Reset Points** button resets the path to the default shape defined by the cue template.

The Audio FX Tab

The Audio FX Tab tab will only appear if the target cue contains audio. It behaves the same as the Audio FX tab for Fade cues targeting Audio cues.

The Video FX Tab

Fade cues automatically recognize video effects added to their target cues, and list all of the parameters of all of the video effects in use. By default, the checkbox next to each effect parameter is unchecked, which means the Fade cue will not adjust that parameter. To fade a parameter, just check the box next to that parameter. Fade cues can adjust any number of parameters simultaneously.

Clicking the **Set Video FX from Target** button will set all video effects parameters to the levels that they contain in the target cue. This is a convenient way to get started building a fade, as it will help you keep track of the starting point from which you will be fading.

Checking the box labeled *Fade rate to*, and setting a target rate, allows you to fade the playback rate of the target cue. The maximum playback rate in QLab is `33`, or 33× normal speed, and the minimum is `0.03`. This control has no effect on Camera or Text cues.

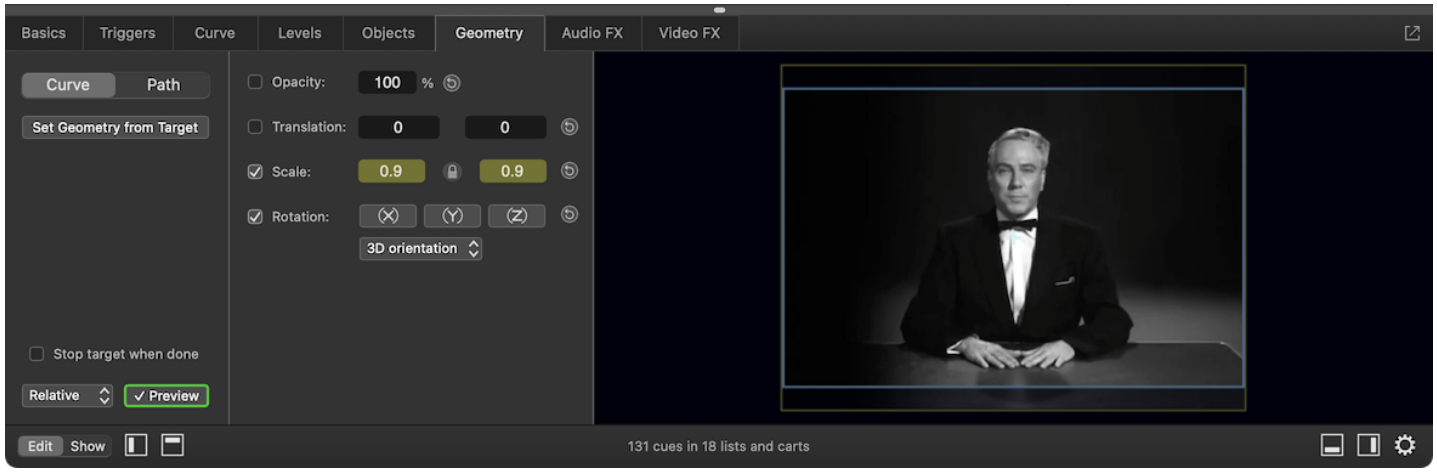
Stop Target When Done. If this box is checked, the target cue will stop once the fade is complete. If the box is unchecked, the target cue will continue to run after the fade is complete.

Preview. This button provides quick access and a visual reference to the live fade preview setting.

Relative Fades

By default, Fade cues are *Absolute*. This means that any parameters that you adjust with a Fade cue will arrive at their final levels regardless of their status before the Fade cue runs. If you use a Fade cue to set the scale of a target Video cue to `2`, then the scale of that Video cue will end up at `2` after running the Fade no matter where that level was set to begin with.

However, using the pop-up menu on the left side of the Levels tab or Geometry tab, you can set a Fade cue to be a *relative* fade. Instead of setting levels to a specific value, relative fades either add or subtract a given amount or multiply or divide a given amount from, to, or by the active parameters, so the starting point of those parameters very much matters. When set to *relative*, the cue's icon changes to the relative form.



In this screen shot, the Fade cue has its scale set to `0.9` which has the effect of reducing the size of the target to `90` % of its current size. If you ran the Fade cue three times, the cumulative result would be reducing the size of the target cue to `72.9` % of its original size, since each time the Fade cue runs, it causes a `90` % reduction of the then-current size.

Relative opacity changes are similarly multiplicative; a relative Fade cue set to an opacity of `90` % will reduce the opacity of its target by `10` % of its then-current opacity.

Relative translation and rotation changes are additive, so a relative Fade cue set to an X-axis translation of `100` will add 100 pixels to the X-axis translation of its target each time it runs.

Relative fades which adjust 3D rotation or video effects can have results which are difficult to predict, and so this manual will refrain from making blanket statements about them.

In QLab 4, using absolute Fade cues on a parameter that had previously been adjusted by relative Fade cues could have unpredictable results. In QLab 5, however, absolute fades supersede relative ones. In QLab 5, an absolute fade is absolutely more absolute.

Reverting Fades

QLab has a sort of special-case *undo* command that applies only to Fade cues, called *Revert Fade Action*. You can find this command under the **Tools** menu when a Fade cue is selected, or you can use the keyboard shortcut ⇧⌘**R**.

When *Revert Fade Action* is invoked on a Fade cue after that Fade cue has been run, QLab reverts the levels of that Fade cue's target to whatever they were before the Fade ran *except* for levels which have been otherwise changed. That is to say, the only adjustments that are reverted are the ones that the selected Fade cue caused.

Broken Fade Cues

Fade cues can become broken for the following reasons:

Missing cue target

Assign a target cue to this cue.

No parameters set to fade

Set this cue to adjust at least one parameter. You can enable or disable an audio parameter by clicking in it; active controls will be highlighted in yellow. You can enable or disable a video parameter by checking or unchecking the box next to it.

Missing cue audio effect

Either install the missing audio effect and restart QLab, or remove the missing effect from the Audio FX tab of the inspector of the target cue.

License required

An audio or video license is required to fade playback rate, fade audio effects, fade patches, or use Timecode triggers. A video license is required to fade video geometry or video effects. Install the appropriate type of license or adjust the cue to avoid using licensed features to clear this warning.

Chapter 7: Lighting

7.1 Intro to Lighting
7.2 Light Cues
7.3 The Light Dashboard
7.4 The Lighting Command Language
7.5 The Light Patch Editor
7.6 Light Library

Intro to Lighting

A note on style

> On this page, every time a new tool, interface item, or concept that we feel is particularly essential is mentioned, it will appear in **bold text.** This is meant to help you notice that you're being introduced to a new idea. Thereafter, and throughout the rest of this documentation, bold text will be used in the traditional manner, as well as to indicate a menu name (such as the **File** menu.)

How QLab Communicates With Lights

QLab 5 communicates with lighting equipment in two ways. The first is by using the **Art-Net** protocol, which uses an ethernet or WiFi network to transmit data. QLab sends Art-Net messages into the network, and those messages are received by other devices on the network which can interpret Art-Net messages.

While some lighting instruments, dimmers, and other devices are able to receive and directly interpret Art-Net messages, most lighting equipment must be controlled using the **DMX** control protocol. To connect to DMX-controlled lighting equipment, you'll need an Art-Net interface, often called a **node**, which is a device that receives Art-Net messages from a network and outputs DMX messages over a traditional DMX connection. QLab is compatible with *any* Art-net interface that uses ethernet or WiFi. QLab uses Art-net version 3, and is therefore theoretically compatible with any device that uses Art-net 4 or below, although it's possible that very old devices might not work well in a modern Art-net network.

The second way QLab can communicate with lighting equipment is by using any of the following compatible USB-DMX interfaces:

- Enttec DMX USB Pro
- Enttec DMX USB Pro Mk2
- DMXking ultraDMX Micro
- DMXking ultraDMX RDM Pro
- DMXking ultraDMX2 Pro
- DMXking ultraDMX MAX
- DMXking eDMX1 MAX
- DMXking eDMX2 MAX
- DMXking eDMX4 MAX
- Yarilo DMX PRO

These devices connect directly to your Mac over USB, and output DMX using a traditional XLR-3 or XLR-5 DMX connection.

A Very Brief Introduction To DMX

DMX, which is short for **D**igital **M**ultiple**X**, is a venerable, reliable digital communication standard for lighting equipment. DMX is generally transmitted using five-pin XLR cables, with one cable carrying a single universe of DMX. One universe contains 512 channels, and each channel is simply an address paired with a value. The status of the entire group of 512 channels is broadcast every 23 milliseconds or so; that's called one *frame* of data. You can imagine a frame of DMX like this:

Address	Level
1	50
2	75
3	08
...	...

Address	Level
512	00

(The ... represents the rows for addresses 4 through 511.)

In this frame, address 1 is set to `50`, address 2 is set to `75`, address 3 is set to `8`, and address 512 is set to `0`. If the cable that carries this DMX data is plugged into, say, a rack of dimmers that are addressed as 1 through 24, then that frame of DMX will set the level of dimmer 1 to `50`, dimmer 2 to `75`, dimmer 3 to `8` and so on up to 24. The dimmer will ignore the data for the rest of the DMX universe.

Levels in DMX range from `0` to `255`, although most equipment represents that range on a percentage scale of `0` to `100`.

You will sometimes encounter DMX-controlled devices which use RJ-45 connectors, which is the same type of connector used for ethernet[1]. It's important to keep in mind that while the connector is physically identical, the language being "spoken" is not ethernet or Art-Net. You cannot plug such a device straight into an ethernet network and expect anything to happen.

The Light Patch

Before you can use lights in Light cues, you need to tell your workspace about the lights it will be controlling: how many are they, what should they be called, what kind are they, and what DMX addresses do they use? You do this for each QLab workspace in Workspace Settings → Light → Light Patch. There, you create **instruments** in the workspace and map them to the Art-Net/DMX addresses of real fixtures or dimmers in the physical world. You can also define **light groups** which allow you to issue a single command to a collection of instruments.

Parameters

In QLab, an instrument represents a single physical DMX-controlled object in the world. Your workspace will therefore need one instrument for each lighting fixture, DMX-controlled accessory, and dimmer in your plot. Each instrument has one or more **parameters**. Conventional lights, controlled by dimmers, have a single parameter: intensity. QLab also supports instruments with more than one parameter. This could include any kind of fixture that uses more than a single DMX address, such as lights which can pan, tilt, zoom, change color, etc.

Instruments in QLab can also contain **virtual parameters,** which are combinations of parameters that have special controls. For example, color parameters on a light can be combined to let you use a graphical color picking tool, instead of manually adjusting each color parameter individually. Virtual parameters are discussed in more detail throughout the Light section of this manual.

Light Definitions

The parameters available for each instrument are specified by a **light definition**. QLab comes with a variety of light definitions for a range of lighting fixtures and dimmers, which you can find in the Light Library. You can use these definitions as-is, copy and edit them to suit your needs, and create new light definitions from scratch for any fixtures you'll be using which are not already in QLab's library. Any light definition you use in a workspace will be saved both in QLab's global library, as well as a library embedded within the workspace, so that the workspace may be safely moved from computer to computer.

If you do not wish to create your own definitions, please contact support@figure53.com and tell us the exact model name of the fixtures you want to use, ideally with a link to its manual online, and we will be happy to create definitions for you.

Light Groups

In QLab, a **light group** is a collection of instruments or other light groups. For example, you might create a light group that contains all the front light in your plot, to allow you to quickly set them all to a given level. Instruments within a group can still be controlled individually at any time; the group is just a quick way to control several instruments at once.

Instruments and lighting groups may be given any name you like using letters, numbers, and spaces. By default, instruments are given numeric names, but they are not limited to numbers. For example, if you have an instrument that lights a particular location on the stage, such as a specific chair, you could name that instrument "chair" and refer to it as such. Similarly, you can name groups to reflect their function, such as "front light" or "warms".

To add an instrument to a group, select the instrument and click the button labeled **Add to Group...**

Once you have created instruments (and, optionally, groups) for all of your lights, you are ready to start building cues.

Light Cues in QLab

In many respects, Light cues in QLab work just like cues in any other lighting console: they fade some number of instruments to specific levels over a given amount of time. An important difference, however, between QLab and most other lighting consoles is that in QLab, any instrument that is not included in a cue is given no level, rather than an assumed level of `0`. Consider a show with six conventional lights in it:

Cue	1	2	3	4	5	6
0	0	0	0	0	0	0

(Here, each row is a cue, and each column is an instrument.)

Now, let's add **cue 1** which brings all the lights to full:

Cue	1	2	3	4	5	6
0	0	0	0	0	0	0
1	100	100	100	100	100	100

Let's say that the next scene is on one half of the stage, so for **cue 2** we want to turn the other half of the stage down very low. In QLab, that might look like this:

Cue	1	2	3	4	5	6
0	0	0	0	0	0	0
1	100	100	100	100	100	100
2	15	15	15			

For instruments 4, 5, and 6, the levels from cue 1 *persist* since cue 2 does not assign any levels to those instruments. When you launch QLab and then run cue 1 followed by cue 2, the real-world levels look like this:

Cue	1	2	3	4	5	6
0	0	0	0	0	0	0
1	100	100	100	100	100	100
2	15	15	15	*100*	*100*	*100*

The italicized levels in cue 2 are actually levels from cue 1 which have persisted. Cue 1 is the **originating cue** for the levels of instruments 4, 5, and 6. Cue 2 is the originating cue for the levels of instruments 1, 2, and 3.

If you open your workspace, starting with all lights off, and then run only cue 2, you'll get this:

Cue	1	2	3	4	5	6
2	15	15	15	0	0	0

Lights 1, 2, and 3 will come on to `15%`, and lights 4, 5, and 6 will remain off. Put another way, the order in which you run Light cue matters to QLab.

QLab provides a few tools which help you make use of this difference without letting it get in your way. You'll learn more about those tools in the rest of the Lighting section of this manual.

1. To be 100% fully accurate, perhaps overly so, the connector that most folks call an "ethernet" connector is called an RJ-45 connector only when it's used in a telephone or network system. When it's used for any other purpose, it's supposed to be called an "8P8C" connector. We promise we did not make this up.↩

Light Cues

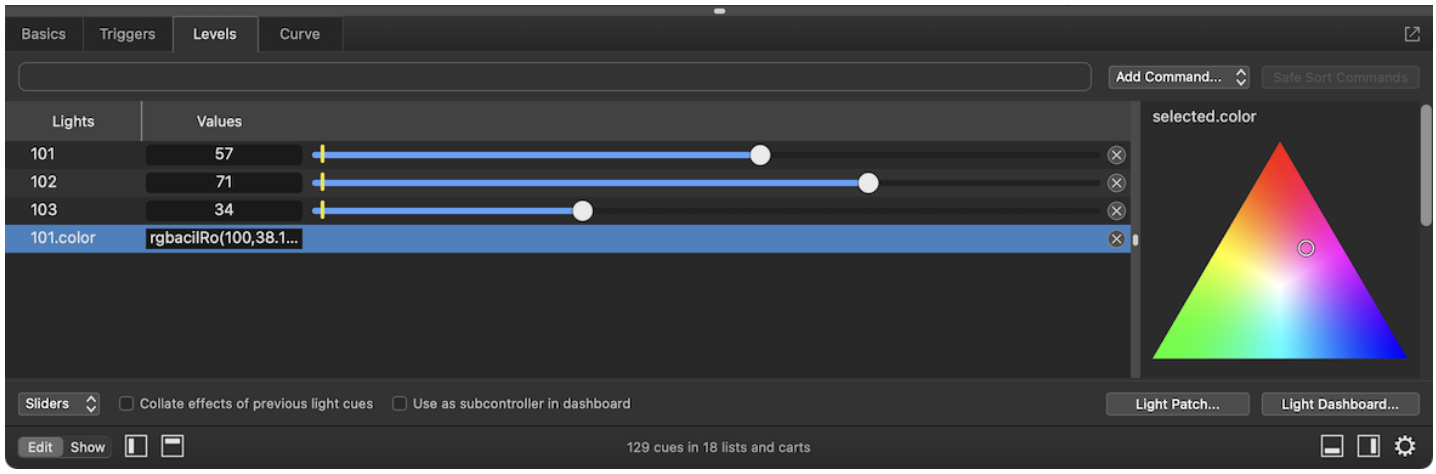
Light cues allow you to set and save lighting commands which set levels for lighting parameters. Light cues behave similarly to Fade cue in that they have a specific duration and a curve shape which define how the cue's saved levels will be applied to the live lighting state over time. Light cues do not have a target; they can act upon one lighting instrument parameter, several, or every parameter for every light in your workspace.

The Inspector for Light Cues

When a Light cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as the following tabs:

The Levels Tab

The Levels tab shows the lighting commands contained by the cue.



At the top of the Levels tab is **the lighting command line.** Lighting commands can be typed in to the command line to quickly add them to the current cue. You can learn more about lighting commands in the Lighting Command Language section of this documentation.

The **Add Command** pop-up menu contains an ordered list of all the instruments and light groups contained in the workspace. Selecting an instrument or light group will add a command for that instrument or light group to the cue. This is an alternative to the command line, and you can use whichever you prefer, or both. Instruments or light groups which already have commands in the cue will be greyed out.

The **Safe Sort Commands** button sorts the light commands in the cue alphabetically, but since commands in a Light cue are interpreted in order from top to bottom, QLab considers whether moving the command would change the end result of the cue. If moving a particular command by sorting would change the result, that command is left un-sorted.

The Lighting Commands list

The commands contained by the Light cue are listed here in the order in which they were added to the cue, with the most recent addition at the bottom. You can view the cue's lighting commands as sliders, tiles, or text by using the pop-up menu at the bottom left of the command list.

Slider View

In slider view, a yellow mark indicates the current value of each parameter. For example, in the screen shot above, instruments 101, 102, and 103 are all currently at `0`.

Additive color, subtractive color, and pan/tilt virtual parameters do not display a slider since they represent a compound value of two or more parameters. When one of these virtual parameters is selected, the appropriate control appears on the right side of the command list.

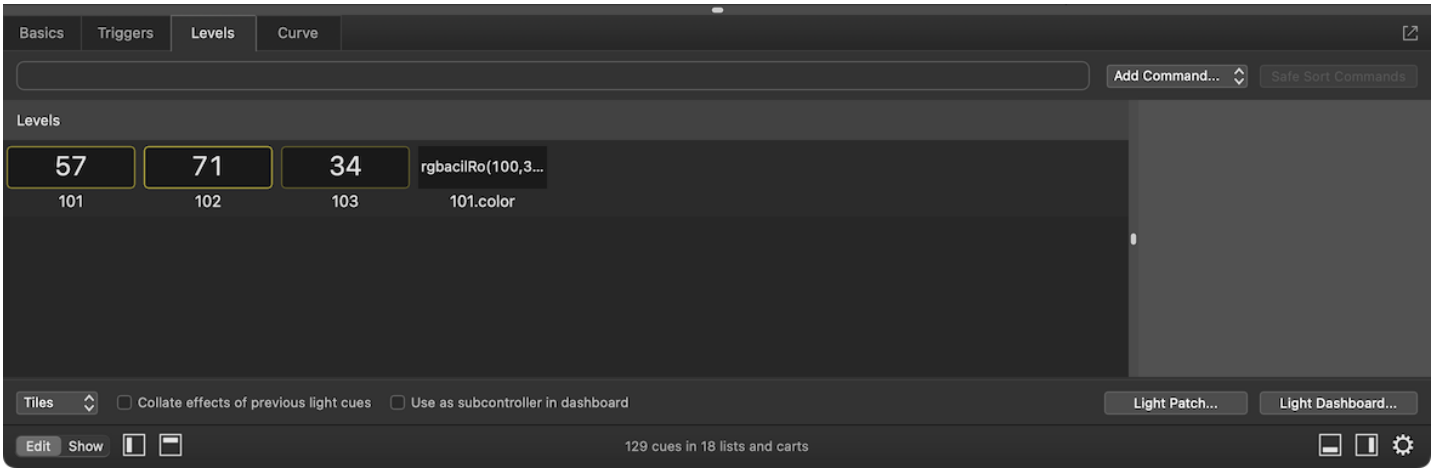
Commands can be dragged up and down in the list to re-order them. You can change a command by typing values into the text fields or dragging the slider handles. Changes made in a Light cue will not be immediately reflected on stage; you need to run the cue in order to see those changes. In this

way, editing a Light cue in the inspector is analogous to editing in “blind” on a traditional lighting console.

To edit “live”, make your changes in the Light Dashboard.

You can remove a command from the cue by clicking on the ⊗ on the right side of the command.

Tile View



In tile view, a yellow outline reflects the level for each command; brighter for higher values.

Commands cannot be reordered in tile view. You can change a command by typing values into the text fields, or by clicking and dragging up or down on a tile.

Individual tiles cannot be selected; rows of tiles can be. When a row is selected, the appropriate controls appear on the right side of the command list.

You can delete a command from the cue in tile view by hovering your mouse over the tile, and clicking on the ✕ that appears.

Text View



Text view gives you direct access to edit the light commands textually. You can delete a command from the cue in text view by selecting and deleting the text of that command, and reorder commands by copying and pasting.

Other controls

Beneath the light command list are several controls.

The pop-up menu on the left allows you to choose amongst the slider, tile, or text view for the commands in this cue. Switching between these options doesn’t change the cue itself, it simply changes the view.

When the box labeled *Collate effects of previous light cues* is checked, QLab will behave as though all prior Light cues in the same list have been run when running this Light cue. This will result in the exact same total look on stage regardless of which other cues have been run. Checking this box causes QLab’s behavior to most closely resemble the behavior of a traditional lighting console.

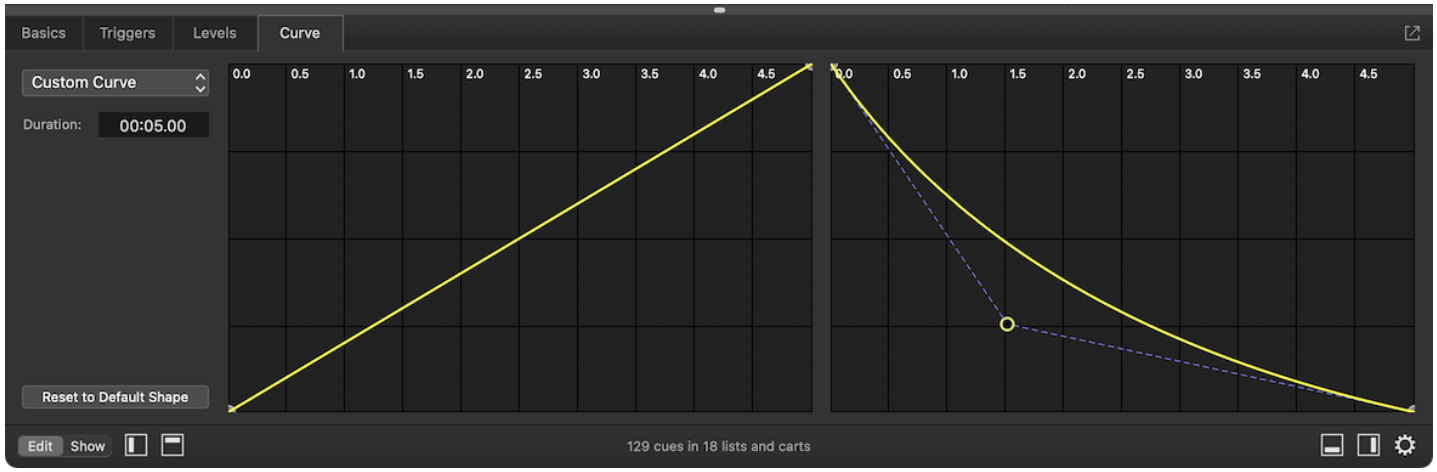
When the box labeled *Use as subcontroller in dashboard* is checked, the cue becomes available to use as a subcontroller in the Light Dashboard. You can learn more about subcontrollers from the Subcontroller section of the Light Dashboard page of this manual.

The **Light Patch...** button opens the Light Patch tab of Workspace Settings → Light.

The **Dashboard...** button opens the Light Dashboard window.

The Curve Tab

The curve shape determines the rate at which the parameter or parameters recorded into the cue are adjusted over the course of the fade. The curve is drawn in yellow, overlaid on a grey grid. The curve on the left is used for levels which are increasing, and the curve on the right is used for levels which are decreasing.



The pop-up menu at the top left of the tab allows you select one of four types of curves:

- **Custom curve**, the default, allows you to create a smooth fade curve of any shape by clicking on the fade curve to create a control point, and then dragging that control point around. The control point smoothly bends the fade curve towards the point. To delete a control point, select it and press the **delete** key on your keyboard.
- **Linear curve** allows you to precisely bend the curve using control points.
- **S-Curve** is a fixed, "ease-in, ease-out" curve which provides a relatively even fade, but prevents the beginning and end of the fade from feeling too sudden.
- **Parametric curve** allows you to create a mathematically precise curve whose shape can be adjusted using the *Intensity* text field.

The *Duration* field lets you edit the duration of the cue. This is the same as editing the duration in the Basics tab or in the Duration column in the cue list.

The **Reset to Default Shape** button resets the fade curve to its default state.

Broken Light Cues

Light cues can become broken for the following reasons:

Invalid light command

The cue contains at least one light command that does not conform to QLab's lighting command language, or perhaps includes a typo. Invalid light commands can only be viewed when the Levels tab is in text mode. Remove or correct the light command(s) in question to clear this error.

No instrument in this cue is patched

All light commands in the cue refer to instruments which are fully or partially unpatched. Add at least one light command for a patched instrument, or complete the patch for least one instrument to which the cue already refers to clear this error.

Broken light definition

The cue contains one or more light commands that refers to an instrument whose definition is broken. Either remove the command(s), correct the definition(s), or use a different definition for the light(s) in question to clear this error.

Missing USB DMX device

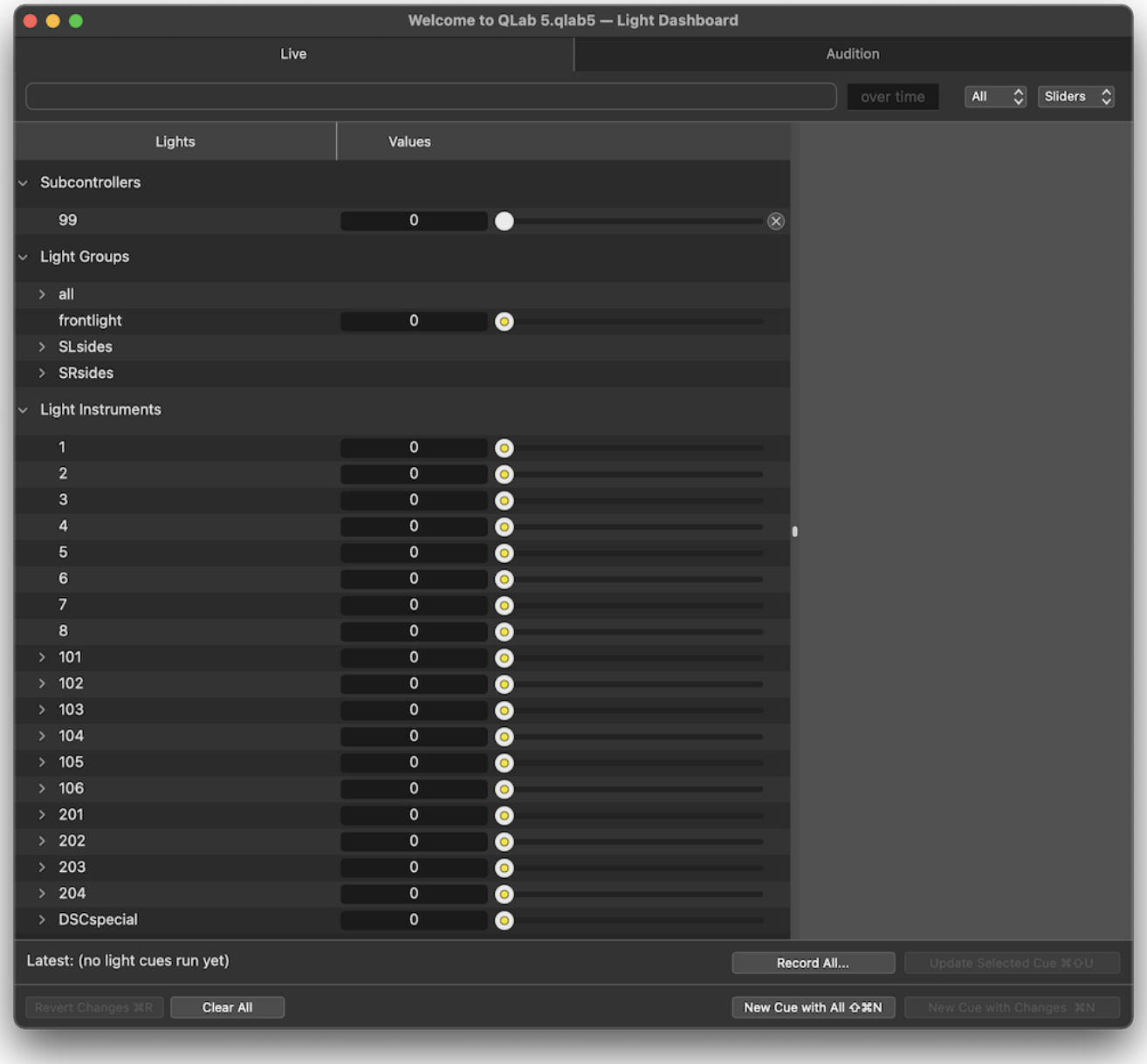
The cue contains one or more light commands for lights that are patched via a USB DMX device that is disconnected. Either repatch the light(s) or reconnect the USB DMX device to clear this error.

The Light Dashboard

The **Light Dashboard** consists of two tabs. The Live tab shows you the current, live levels of all lighting instruments in your workspace and lets you manipulate them in real time. If you change a value in the Light Dashboard, the change will be sent immediately to the lighting hardware, and you will see that change reflected in the fixtures and devices connected to your QLab lighting system. Similarly, when you run a Light cue, you'll see the changes made by that cue in the Light Dashboard as they happen.

The Audition tab shows you light levels adjusted by Light cues that have been auditioned. The Audition tab is discussed below.

You can open the Light Dashboard by choosing it from the **Window** menu or by using the keyboard shortcut ⇧⌘**D**.



Levels in the Dashboard can be modified a number of ways:

- By typing commands into the Light Command Line;
- By clicking and dragging on the sliders or tiles;
- By double clicking in individual parameters' text fields and typing values;
- By clicking and dragging on parameters' text fields.

The Light Command Line

Across the top of the Dashboard is a command line which allows you to manipulate the Dashboard quickly and powerfully. You can learn how to use it in the Lighting Command Language section of this manual.

Over Time

Typically, any commands you type into the command line will execute immediately when you hit *enter*. If you type a time into this field, however, the command will fade over that amount of time. This concept, called *sneak* on some other lighting consoles, allows you to make changes to the live state of your lights in a gentle, subtle way. The **Over Time** field resets itself to `0` after every use.

All or Used

When this pop-up menu is set to **All**, the Dashboard will display every instrument and light group defined in the workspace's patch.

When the pop-up is set to **Used**, the Dashboard will only display instruments and light groups which are currently recorded into cues plus any instruments or light groups which have been manually adjusted (i.e. any control that is drawn in yellow.)

Sliders or Tiles

The Dashboard offers two ways to look at your lighting:

Slider view shows one instrument or light group per line, sorted alphabetically, with a text field on the left side and a slider on the right. Values can be typed into the text field, or modified by clicking and dragging on the sliders. Multiple values can be adjusted at once by shift-clicking on two or more sliders and then dragging.

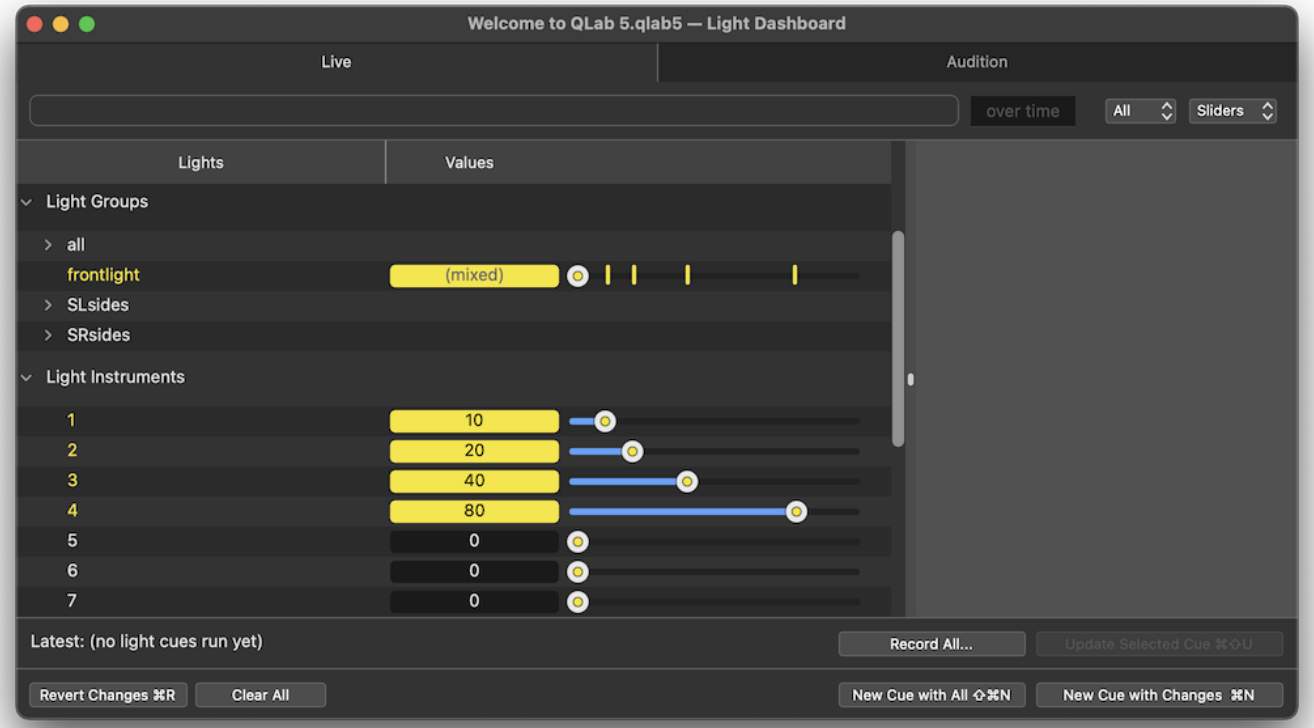
Decimal values are supported for parameters that use a percentage scale. You can only enter decimal values by typing or dragging on parameters' text fields; dragged sliders or tiles will snap to whole numbers.

When dragging on parameters' text fields, the number of decimal places shown depends on whether the parameter is 8-bit or 16-bit; more decimal places are shown for 16-bit parameters in order to give you as much precision as possible.

You can hold down the ⇧ (shift) key while dragging to be even more precise.

When a parameter has been modified in the Dashboard, that parameter's control turns yellow. The slider handle displays as a triangle pointing in the direction that the parameter last moved.

Rows which represent groups show yellow marks to indicate different levels of individual instruments within the group.

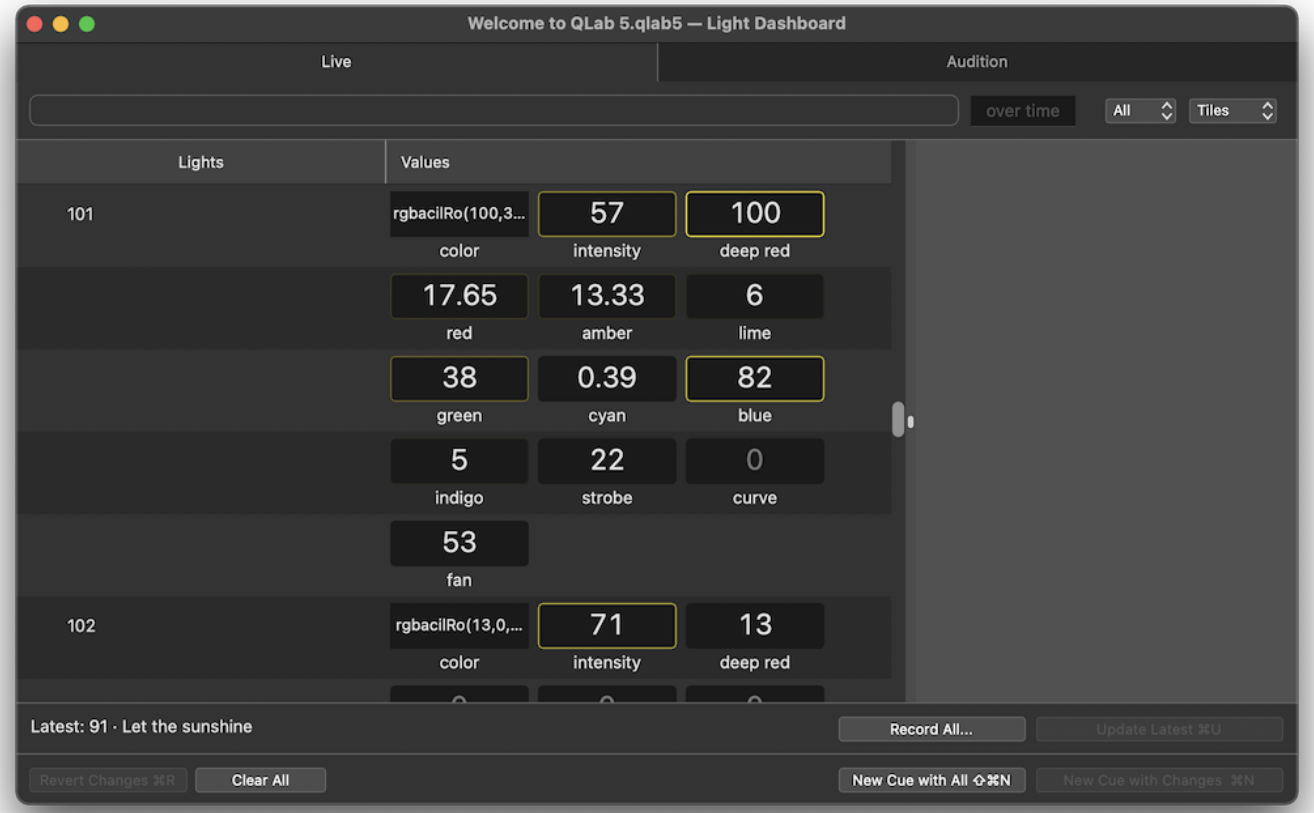


Instruments and light groups that have more than one parameter will display a disclosure triangle on the left edge of the Light Dashboard. When the triangle is pointing to the right, only the default parameter of the instrument or light group (typically the intensity parameter) will be shown. If you click the triangle, the slider will “unfold” to show all the parameters of the instrument or group.



Light groups contain all the parameters for every instrument they contain.

Tile view shows each parameter of every instrument or light group as a tile with the level of the parameter above in larger text, and the name of the instrument and parameter below in smaller text. Tiles are divided into light groups, single-parameter “simple” instruments, and multi-parameter “fancy” instruments. The brightness of the outline of each parameter control reflects the current value of that parameter.

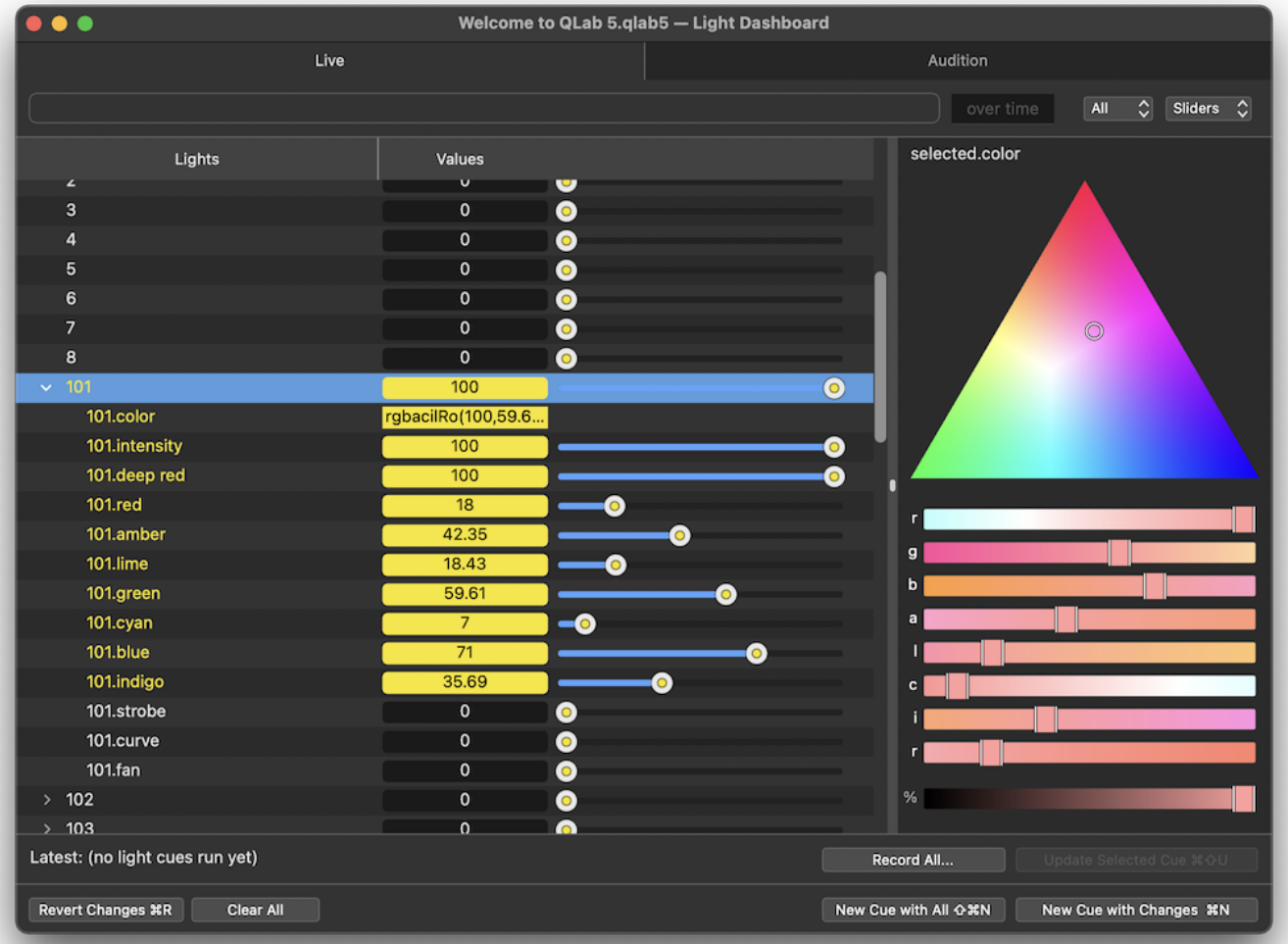


Any tile containing a value that has been manually adjusted since a Light cue was run is displayed with a yellow background.

Virtual Controls

Whenever an instrument is selected that has an additive color, subtractive color, hue/saturation/intensity, or pan/tilt virtual parameter, controls for those virtual parameters are displayed in the sidebar of the Light Dashboard.

The Additive Color Picker



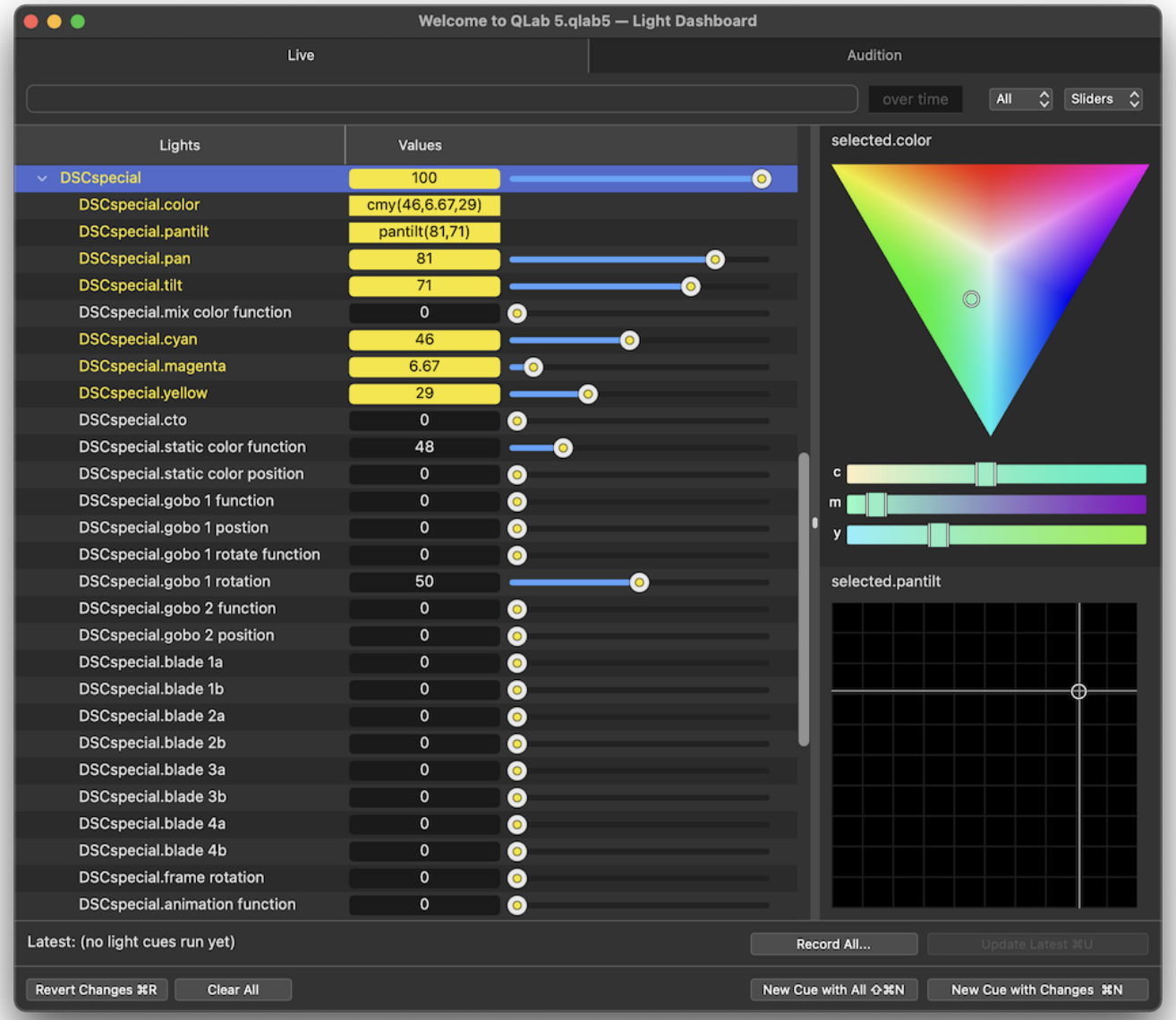
The additive color picker has at least red, green, and blue components, but can also include other components such as amber, white, lime, indigo, or others as specified by the light definition.

You can click and drag the circular pip around the triangle, or swipe with two fingers on a trackpad. Holding down the ⇧ (shift) key while swiping allows you to move more slowly and precisely.

Each component of the picker also has its own slider beneath the triangle, and these sliders update in real-time as you drag the pip. You can also drag the sliders to increase or decrease the level for individual color components. When adjusting sliders for colors other than red, green, or blue, the shading of the triangle will change to simulate the range of colors that become available in response to the adjustment.

The % slider allows you to proportionally scale all the color components at once, which is particularly helpful when using lighting instruments that have no intensity parameter.

The Subtractive Color Picker



The subtractive color picker has only cyan, magenta, and yellow components, and is shown upside-down with respect to the additive picker.

The Hue/Saturation/Intensity (HSI) Picker



The hue/saturation/intensity (HSI) picker is circular. Unlike the % slider in the additive color picker, the *intensity* slider in the HDI picker is directly connected to the intensity parameter of the instrument.

The Pan/Tilt control

The pan/tilt control shows pan in the horizontal axis and tilt in the vertical axis with light grey gridlines every ten percent.

Additional Controls

You can right-click or control-click in the sidebar to optionally display additional sliders for the default parameter and all other parameters of the selected instrument. This can be useful if you wish to keep multi-parameter instruments "folded up" in the main area of the dashboard, and use the sidebar to control all of their parameters.

Latest Light cue

Unlike many other lighting consoles, QLab is not confined to being "in" a cue, since each Light cue does not necessarily contain commands for every light in the patch. Put another way, the results of more than one cue can be reflected on stage at once, and when that's the case there's no clear cue that QLab is "in". Rather, QLab tracks cues as they are run, and lists the most recently run Light cue here.

Revert Changes & Clear All

Clicking **Revert Changes ⌘R** will restore all parameters that you've changed in the Light Dashboard to the levels that they were last set to. This restore happens over the workspace's panic duration.

The Dashboard also supports undo (and redo). Undo can be used to revert individual changes.

Clicking **Clear All** will clear any modifications made in the Dashboard and set all parameters of all lights to their home value. **Please be aware** that this generally causes a blackout. This button is the QLab equivalent of what many other consoles refer to as "Go To Cue out."

Recording and Updating Cues

In the lower right corner of the Dashboard are four buttons for creating and updating cues. Cues created using these buttons will appear in the current cue list or cart, directly after the selected cue. If no cue is selected, new cues will appear at the end of the current cue list or cart.

Record All

The **Record All...** button displays a pop-up menu with three options, each of which may or may not be available depending upon the situation:

- **Latest Cue.** Record all current light levels into the latest Light cue, overwriting any levels already in that cue. If no Light cues have been run, and no cue is displayed to the left as the latest Light cue, then this option is not available.
- **Selected Cue(s).** Record all current light levels into the currently selected Light cue or cues, overwriting any levels already in those cues. If there are no currently selected Light cues, then this option is not available.
- **New Cue.** Record all current light levels into a new Light cue. In QLab, in the context of lighting, the word *record* is used to mean "capture the current level of all parameters of all instruments in the workspace." What this means is that any parameters that do not have an explicit level in the Light Dashboard (i.e. those parameters which haven't been touched since the workspace was opened) will be recorded at their home value (typically ∅.) In this way, cues created or modified using **Record All...** function as blocking cues.

Update

The **Update...** button becomes enabled whenever the Dashboard contains modified (yellow) values. It displays a pop-up menu with three options, each of which may or may not be available depending upon the situation:

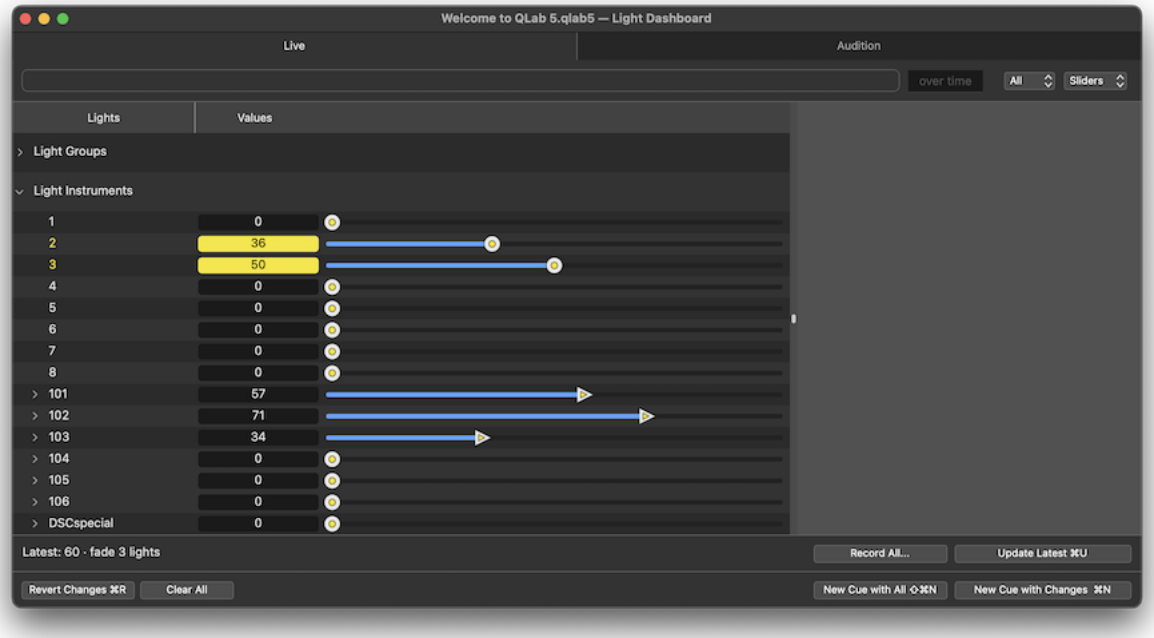
- **Latest Cue (⌘U).** Copy all modified light levels into the latest Light cue. If the modified levels belong to lights or groups that are already in the latest cue, QLab will overwrite those levels. If not, QLab will add them and leave everything else alone. If no Light cues have been run, and no cue is displayed to the left as the latest Light cue, then this option is not available.
- **Selected Cue (⇧⌘U).** Copy all modified light levels into the currently selected Light cue or cues. If the modified levels belong to lights or groups that are already in the selected cue or cues, QLab will overwrite those levels. If not, QLab will add them and leave everything else alone. If there are no currently selected Light cues, then this option is not available.
- **Originating Cue (⌥⌘U).** Copy all modified light levels into the cue or cues which *originated* their current levels. This concept is most similar to "record track" and is explained in more detail below.

New Cue with All

This button serves as a shortcut for **Record All > New Cue**. You can also use the keyboard shortcut ⇧⌘**N**.

New Cue with Changes

This button records only *changed* levels into a new Light cue. For example, consider this screen shot:



Here, a Light cue has been run which raised instrument `4` to `24` and instrument `src4` to `69`. Then, instrument `2` has been manually set to `36` and instrument `3` has been manually set to `50`

If you click **New Cue with Changes** when the Dashboard looks like that, QLab would create a new cue containing `2 = 36` and `3 = 50`. Instruments `1`, `4`, `src4`, and `viper` are not recorded into the cue because their levels have not been modified in the Dashboard.

When you run this new cue, only `2` and `3` will change, and the other lights will be left alone at whatever levels they're already set to. In this way, cues created using **New Cue with Changes** function as tracking cues.

Originating Cues Explained

Since not every cue needs to provide a value for every instrument, it is possible that, after running several cues, the Dashboard reflects the handiwork of more than one cue. **Update Originating Cue(s)** will update the value of each instrument *in the cue that originated the level for that instrument*. This is complicated, so we'll walk through that step by step. Consider a show with six instruments, and five cues:

Cue	1	2	3	4	5	6
1	100	100	100	100	100	100
2	15	15	15			
3		80				
4				25	25	
5			50			

After running all five cues, in order, the live state of the lights in this show are:

Cue	1	2	3	4	5	6
Live:	15	80	50	25	25	100

The levels for each instrument originate in different cues:

Cue	1	2	3	4	5	6

Cue	1	2	3	4	5	6
1	100	100	100	100	100	**100**
2	**15**	15	15			
3		**80**				
4				**25**	**25**	
5			**50**			

If you ran those five cues, and then adjusted all six instruments to 75, the button would read **Update 5 Originating Cues** because those instruments' levels originate in five different cues. If you then clicked the button, those originating cues would all be updated:

Cue	1	2	3	4	5	6
1	100	100	100	100	100	**75**
2	**75**	15	15			
3		**75**				
4				**75**	**75**	
5			**75**			

The DMX Status Window

The DMX Status Window, which you can find under the **Window** menu, is a diagnostic and troubleshooting tool that's not part of the Light Dashboard, but is related. The DMX Status Window shows the current value of one universe of DMX at a time, and lets you manually set the level of individual addresses. You cannot record changes you make in this window; it's intended only to help you solve patching errors, discover whether a misbehaving fixture is correctly addressed, etc.

The Tools Menu

When the Light Dashboard is the frontmost window, the contents of the **Tools** menu is re-populated with relevant menu items:

Tool	Keyboard Shortcut
New Cue with Changes	(⌘N)
New Cue with All	(⇧⌘N)
Update Latest Cue	(⌘U)
Update Selected Cues	(⇧⌘U)
Update Originating Cues	(⌥⌘U)
Record All to Latest Cue	
Record All to Selected Cues	
Clear All	
Revert Changes	(⌘R)
Park Selected Lights	
Unpark Selected Lights	

Parking

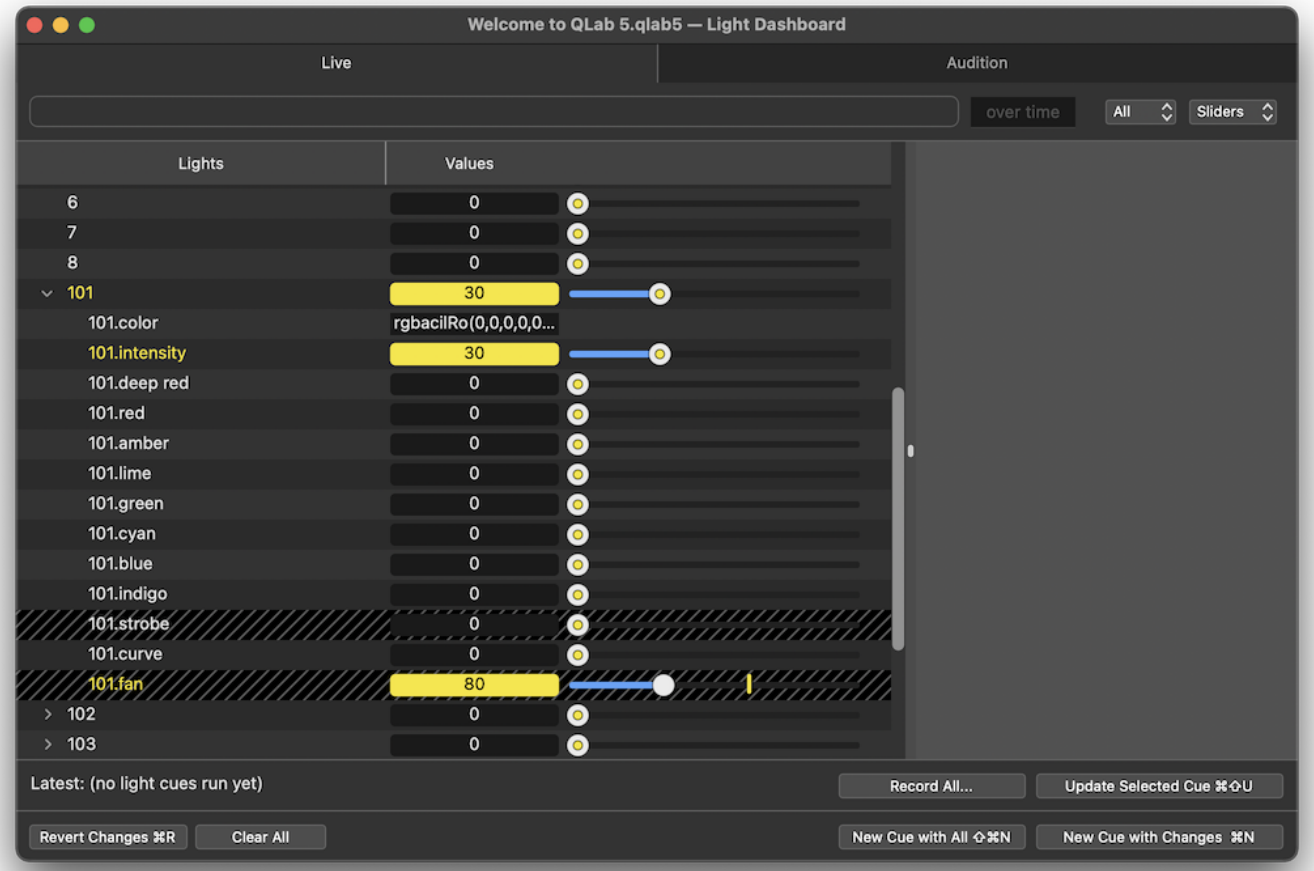
Parking an instrument freezes its current state until it is unparked. While parked, neither cues nor the Light Dashboard can alter the actual live state of the parameter, and the parked value cannot be recorded into cues. This can be helpful for things like keeping house lights at a glow during focus, or preventing a strobe light or fog machine from turning on while working on cues.

To park an instrument, set the Light Dashboard to Slider view, select the instrument's slider, and choose *Park Selected Lights* from the **Tools** menu. If the instrument has more that one parameter, you can select the top-level slider to park the whole instrument, or you can select one or more parameters' sliders to park only those parameters.

You can park multiple instruments, groups, or parameters by command-clicking on multiple sliders.

When a parameter is parked, it is drawn with a diagonal striping overlay much like a disarmed cue.

Parameters can still be edited while parked, but the actual DMX output from QLab will remain locked to whatever value was set when the parameter was parked. The parked value will be shown on the fader track using a yellow light mark.



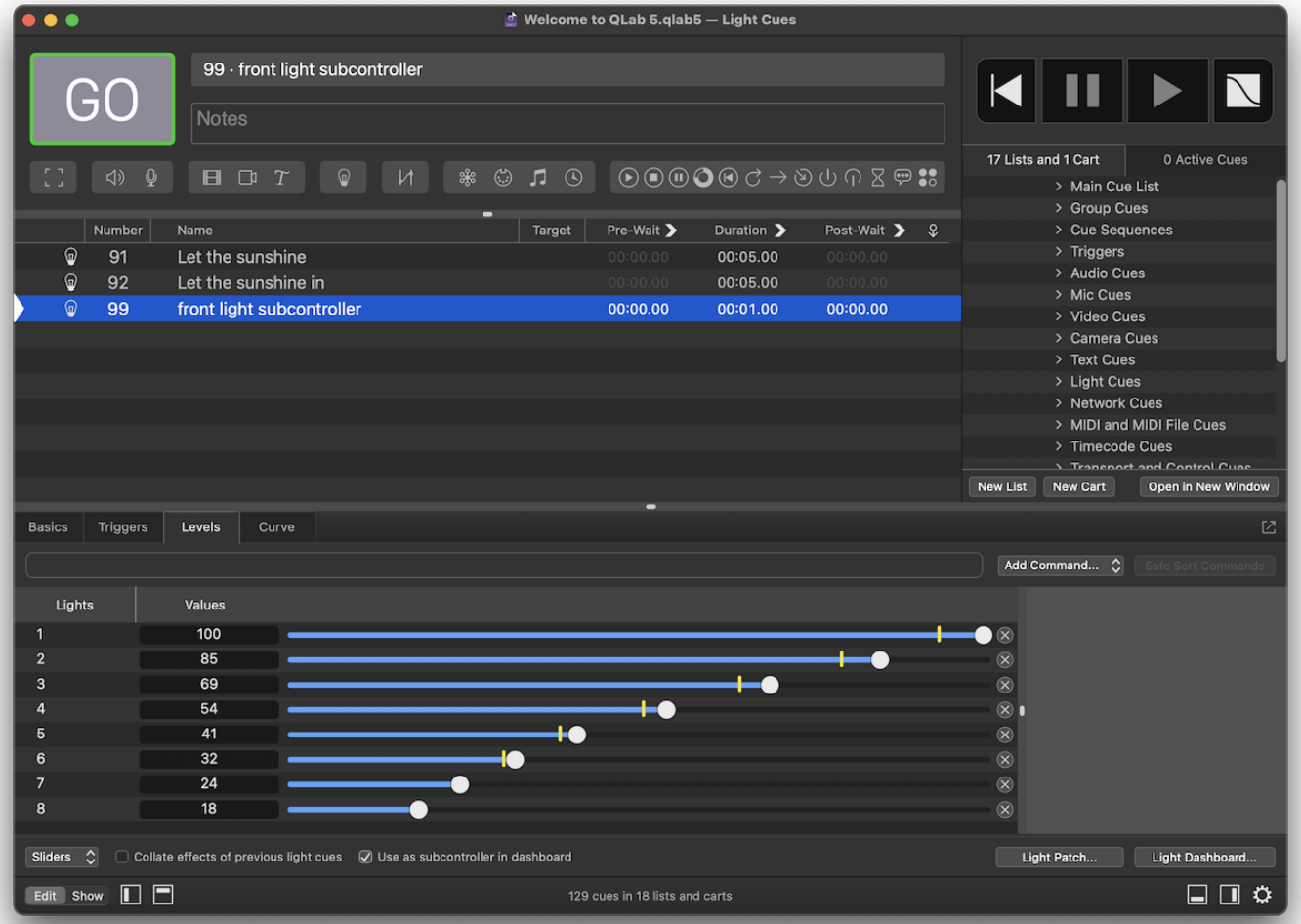
In this screen shot, two parameters of the light called `101` have been parked: the `strobe` parameter is parked at `0`, and the `fan` parameter is parked at `160`. The `fan` parameter has subsequently been set to `80`, and the light mark shows that the current live value of the `fan` parameter remains at `16`. All other parameters of the instrument remain unparked and can be used as normal.

To unpark an instrument or parameter, select it in the Light Dashboard and choose *Unpark Selected Lights* from the **Tools** menu.

When an instrument or parameter is unparked, QLab immediately updates the live value being sent to that instrument or parameter. This could cause surprising behavior if, for example, you park a group of strobe lights at `0`, then run a cue which brings them to full, and then unpark them. Because of this, QLab displays confirmation dialogue boxes for both parking and unparking.

Subcontrollers

Subcontrollers can be thought of as light groups which have proportional control over the parameters which they include. A subcontroller is defined by creating a Light cue, giving that cue a number, and checking the box labeled *Use as subcontroller in dashboard* in the Levels tab of the inspector.



Again, only Light cues with cue numbers can be used as subcontrollers.

When that checkbox is checked, a slider appears in the Light Dashboard labeled with the cue number of the cue. Dragging the slider or the level field, or typing in a level, proportionally controls all the parameters recorded into the subcontroller.



Any parameters can be included in subcontrollers, so it's important to be careful and specific when building Light cues that will be used as subcontrollers. Including parameters such as timing control channels for moving lights can have unpredictable results. Including parameters which instruct arc lamps to strike or extinguish can be potentially dangerous for the lamps.

Highest Takes Precedence

Subcontrollers interact with cues according to a rule called *highest takes precedence*, often referred to simply as *HTP*. When both a cue and a subcontroller are simultaneously setting levels for a parameter, the parameter is set to whichever level is higher. For example, if a subcontroller is holding a light at `20%` and a cue runs which brings all lights out, the one light in the subcontroller will remain at `20%`. If another cue runs which brings all lights to `100%`, the light in the subcontroller will go up to `100%` as well, since that's the higher level.

Subcontrollers do not interact with each other or with the rest of the Light Dashboard according to HTP. All controls within the Light Dashboard behave according to *latest takes precedence* or *LTP*. Under LTP rules, parameters simply respond to the most recent level they're given.

The Audition Tab

When a Light cue is auditioned or audition previewed, its commands are run in the Audition tab of the Light Dashboard instead of the Live tab. The Audition tab does not connect to actual DMX output, and as a result, auditioning a Light cue shows you its results in the Light Dashboard only, without altering the actual levels of your actual instruments. Traditional lighting consoles often refer to this type of behavior as "blind."

The buttons at the bottom of the Light Dashboard behave the same way in the Audition tab as they do in the Live tab, with one exception; the Audition tab does not have a **Clear All** button. In its place is a **Reset from Live** button which snaps all levels in the Audition tab to match the current state of the Live tab.

The Audition tab has a dark blue background to help distinguish it from the Live tab.

The Lighting Command Language

QLab 5 uses a textual light command language which you can use to control your lighting instruments. At its core, every light cue contains a plain block of text which is interpreted as this command language. The basic formats of lighting commands are as follows:

```
instrument = value
instrument.parameter = value
group = value
group.parameter = value
```

Note that spaces in a lighting command are always optional. `10=53` is the exact same thing as `10 = 53`.

When a command omits a parameter and instead takes the form of `instrument = value`, QLab fills in the default parameter as specified by the instrument's definition. Typically, this default parameter is intensity but it can be any parameter. So, if instrument 20 uses an instrument definition that sets the default parameter to intensity, `20 = 75` is the exact same thing as `20.intensity = 75`.

When a command refers to `group.parameter`, the value of that command is passed to all instruments within the group that have that parameter. So the command `group.blue = 50` would set the `blue` parameter of any instruments in that group to `50`. Instruments which do not have the given parameter are ignored.

Percentages, DMX values, and decimal numbers

QLab supports both percentage values and "raw" DMX values. Individual parameters of instruments can be set to use one or the other in their definitions. If a parameter uses percentage values, QLab will accept decimal numbers on the command line or when typing into the text field of sliders or tiles. QLab will round your entry to the nearest DMX level equivalent.

For example, if instrument `1` in your workspace is a conventional dimmer set to accept percentage values, and you enter `1 = 10.7`, QLab will round down to `10.59`, because `10.7%` is mathematically closest to DMX level `25`, and the exact representation of `25` in DMX is `10.59%`.

Pass

An instrument or group can be set to the value *pass* to explicitly prevent them from being adjusted by the current cue. Consider a show with a group called `backlight`, and three instruments in the group called `10`, `11`, and `12`. If a cue consists of the following commands:

```
backlight = 75
12 = pass
```

then instruments `10` and `11` would be set to `75`, and `12` would be left un-adjusted.

Home

The command `instrument = home` or `instrument.parameter = home` sets the instrument or parameter to its home value, as specified in the instrument definition. For example, the included "dimmer" definition has a home value of `0`, so setting an instrument that uses the dimmer definition to *home* turns it off. The included "DMX iris" definition has a home value of `100`, so setting an instrument that uses the DMX iris definition to home opens the iris all the way.

Up arrow

The **up arrow key** scrolls through the history of the command line, providing an easy way to experiment with a level. For example:

```
1 = 80 |Enter|
|Up|
70 |Enter|
```

```
|Up|
75 |Enter|
```

Instrument `1` is set to `80`, then `70`, then `75`, and reentering the `1=` is not necessary.

Ranges

A lighting command can also refer to a range of instruments:

```
1 - 3 = 50
10, 12, 14 = 75
```

The range will be expanded to its constituent commands before being added to the cue, or applied to the Dashboard. Thus, the commands above would appear in a cue as:

```
1 = 50
2 = 50
3 = 50
10 = 75
12 = 75
14 = 75
```

Ad-hoc Groups

You can alternately define an "ad-hoc" group by enclosing a range in brackets:

```
[1 - 3] = 50
[10, 12, 14] = 50
```

Bracketed commands remain as single commands in the cue, and behave just like light groups.

Pull From Cue

Parameters can be set to "pull" a level from another cue, dynamically inserting the level from that other cue at the moment the cue is run. Those familiar with the idea of palettes or presets on other lighting consoles will find this concept familiar.

To pull levels from one cue into another, instead of entering a level for a parameter, enter `cue` and the number of the cue you want to pull from:

```
10 = cue A
```

This command sets instrument `10` to whatever values it's set to in cue A. If instrument `10` is not recorded in cue A, then this command has no effect. If instrument `10` is a multi-parameter instrument, the above command will set all the parameters of instrument `10` to the values they're set to in cue A. You can also be more precise, if you like:

```
10.zoom = cue A
```

This command sets only the zoom parameter of instrument `10` to the value stored in cue A. If cue A contains light groups, you can pull the whole group or only part of the group into the new cue by making use of the fact that QLab interprets light commands sequentially. Consider a workspace where `group1` contains instruments `1`, `2`, `3`, and `4`:

```
group1 = cue A
3 = pass
4 = 75
```

This series of commands combines together to give you the following results:

```
1 = (whatever it is in cue A)
2 = (whatever it is in cue A)
3 = (left alone)
4 = 75
```

If cue A is updated, any cues which pull from it will receive the updated values the next time they run. In this way, you can think of a pull command as being “linked” to the cue that it pulls from. To “break” this link, and make the command stop pulling from the source cue, you can click the *Expand* button next to the light command in the inspector. This copies the pulled level into the current cue, and turns that command into a normal light command.

Proportional Pulls

You can scale a pulled value using the asterisk (*) character:

```
10 = cue A * .5
```

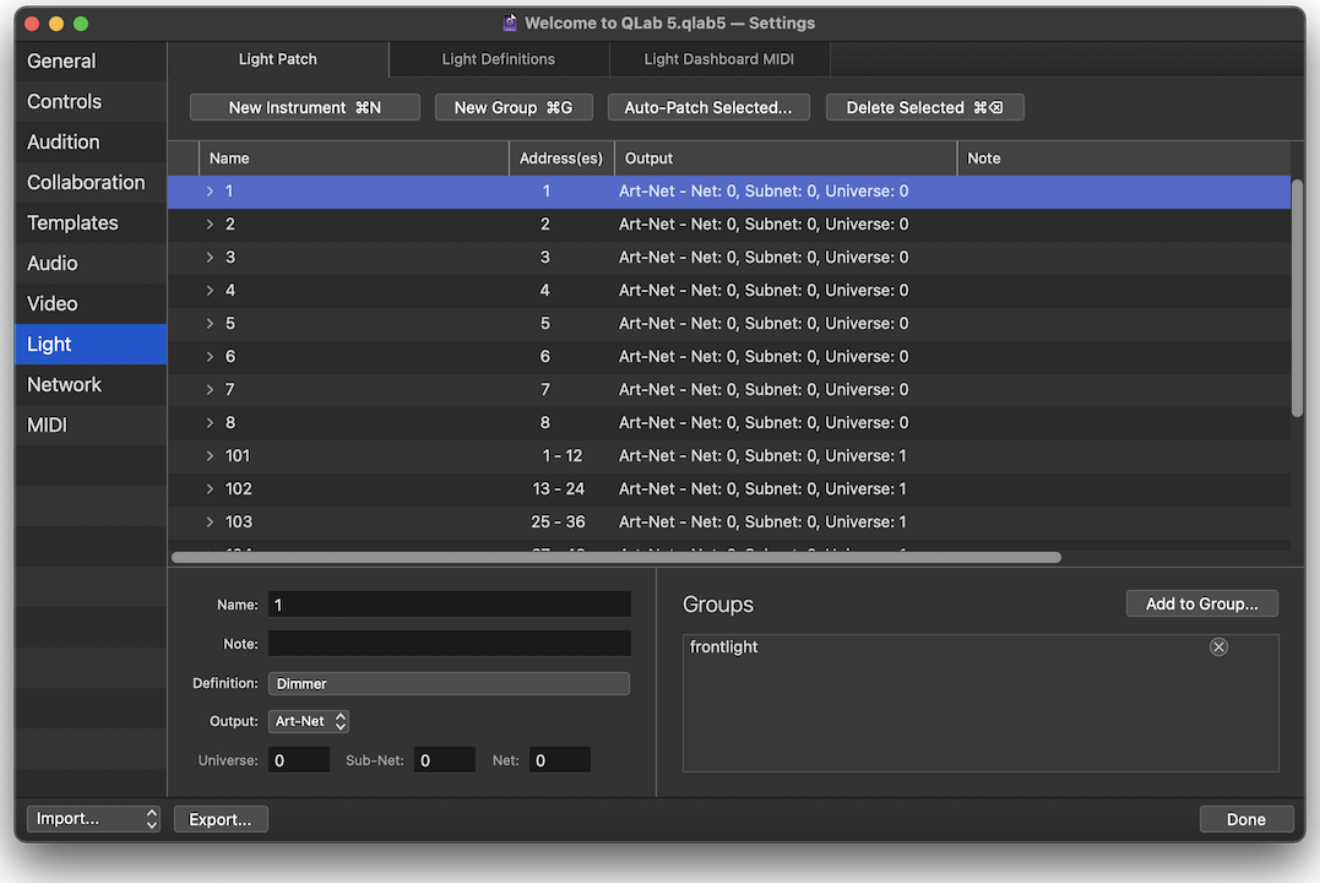
This command sets instrument `10` to half of whatever values it has in cue A. This can be an easy way to mimic a look from an earlier cue, but make it a bit dimmer or a bit brighter.

The Light Patch Editor

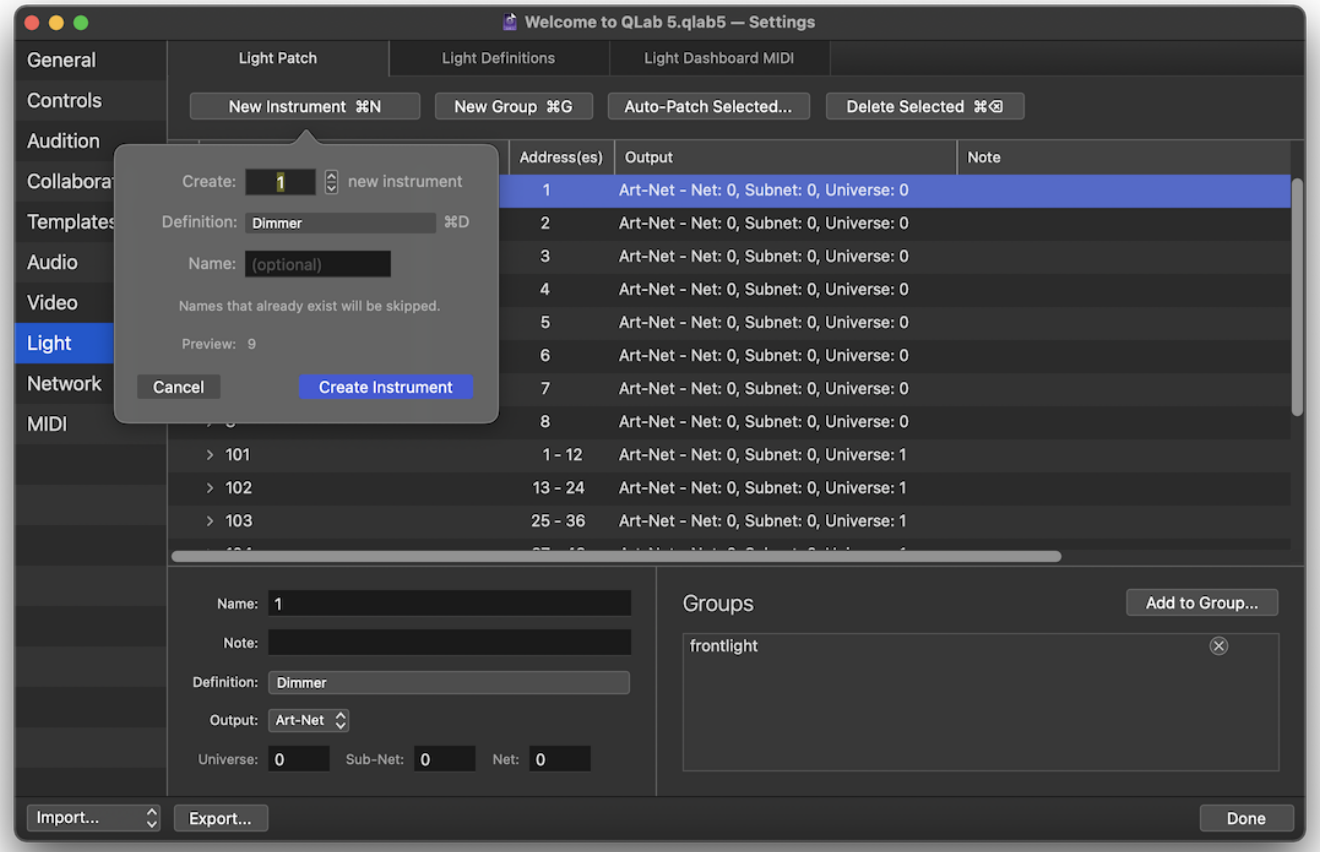
The Light Patch Editor comprises the first two tabs found in Workspace Settings → Light: the Light Patch tab and the Light Definitions tab.

The Light Patch Tab

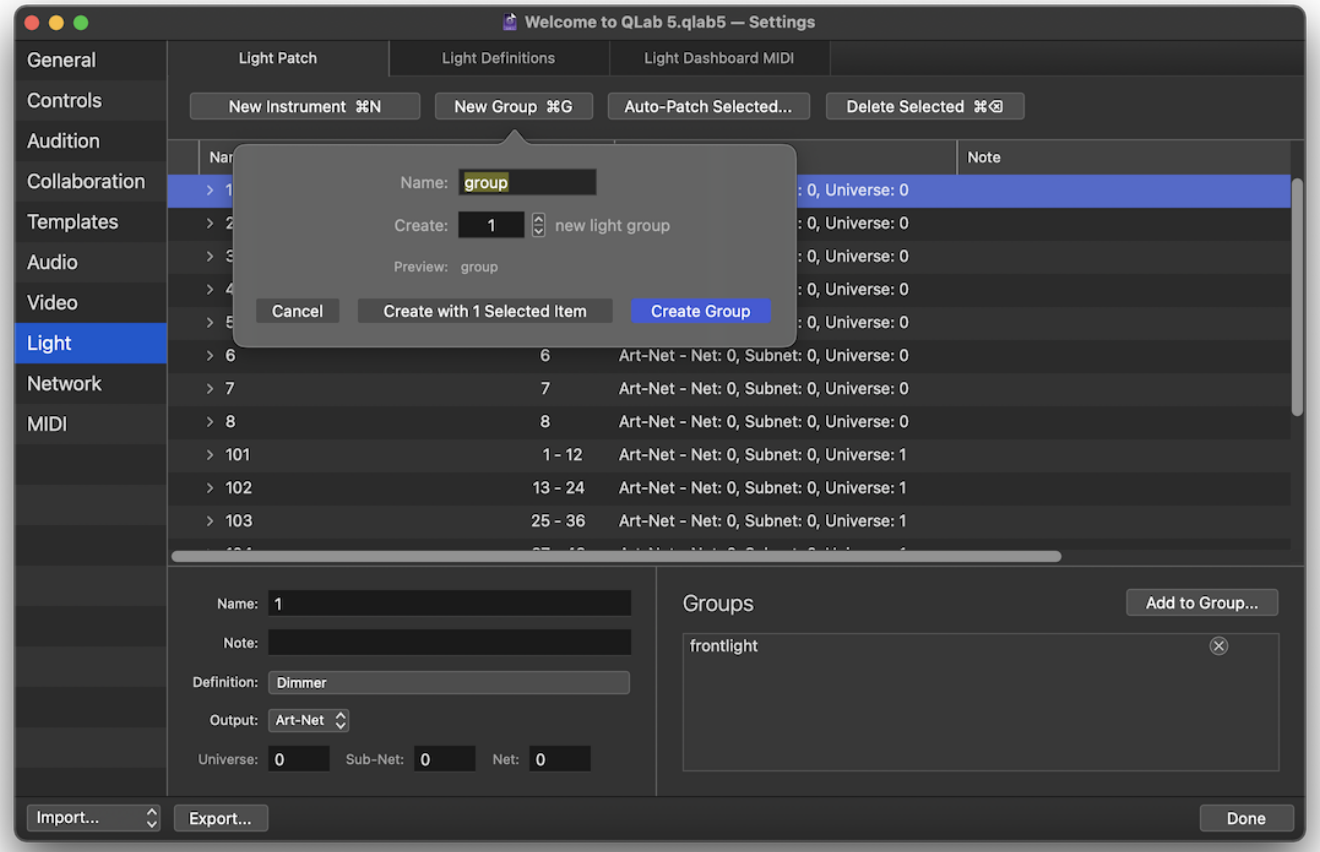
The Light Patch tab allows you to view and edit the instruments and light groups in the workspace.



The **New Instrument ⌘N** button lets you add one or more instruments to the workspace. When you click the button, a popover will appear giving you several options, including the number of instruments you want to create, the definition they should use, and so on.



The **New Group ⌘G** button lets you add one or more light groups to the workspace. When you click the button, a popover will appear giving you several options, including how many groups you want to create and whether you would like the currently selected instruments to be included in the newly created group or groups.



The **Auto-patch Selected...** button lets you automatically assign DMX addresses to the instruments currently selected in the patch list. QLab will display a dialog box which allows you to choose the DMX device you want to use, as well as the starting address. QLab will then assign addresses to the selected instruments according to the specifics you enter into the box.

The **Delete Selected ⌘⌫** button deletes the selected instrument(s) and/or light groups(s) from the workspace.

The Patch Table

All instruments and light groups in the workspace are listed here, sorted alphabetically by name. You can click on the right-pointing disclosure triangle to reveal the parameters for each instrument, or the instruments contained in each light group.

Status. The status column displays a ╳ icon for a broken instrument, or a ∅ icon for an instrument without an address assigned. Otherwise, this column displays nothing.

Name. Names must be unique within a workspace, and can contain letters, numbers, and spaces only.

Address(es). This column displays the DMX address or addresses for the instrument. You can double click on the address to edit it. If the instrument requires multiple addresses, enter only the starting address. Although multiple instruments cannot be assigned to the same address, QLab will allow you to enter an address which is already used. Instruments with conflicting addresses will be marked as ╳ broken and will not work until the conflict is solved. Without an applicable license installed, any instruments using more than the freely allowed 16 addresses will be marked as ╳ broken.

Addresses must be unique *per universe, per device.*

Output. This column displays the output device, Art-net or USB, that the instrument is set to use. If the output device supports more than one universe of DMX, that information is shown as well.

For multi-universe devices, the first universe is numbered `0`. Art-net additionally allows 128 *nets*, each containing 16 *sub-nets*. Each sub-net contains 16 *universes.* So, the first 512 addresses in a lighting device are all part of universe 0, which is part of sub-net 0, which is part of net 0. Since sixteen

universes contain 8,192 addresses, you likely will not need to worry about net or sub-net unless you have a *very* complex show.

Note. For your notational pleasure.

You can right-click (or control-click or two-finger-click) on the column headings to show two more columns which are hidden by default:

Definition. Shows the instrument definition for the instrument. Click on the definition icon to choose a different instrument definition.

Groups. Lists the light groups that the instrument belongs to, not including the default "all" light group, which all instruments always belong to.

You can also hide any of the default columns as you prefer.

Beneath the patch table is a sort of mini-inspector that shows details of the selected instrument or light group.

Name, Note, and Definition are all duplicate, alternative views for the same information listed in the patch table. You can view and change each of these properties in both places.

Output. QLab can use both Art-net and certain USB DMX devices to communicate with lighting equipment. Each individual instrument can be assigned to either the Art-net network or to an individual USB DMX device.

If you assign the instrument to Art-net, QLab will display **Universe**, **Sub-Net**, and **Net** fields for you to fill in. These fields should correspond with the configuration of the Art-net node that the instrument is connected to.

If you assign the instrument to a USB device, QLab will display a **Device** pop-up menu of available USB DMX devices. If you select a USB device has more than one universe, a text field will also be displayed allowing you to select which universe the selected instrument should use. Note that the first universe of a device is always universe `0`.

If you select "None", the instrument will remain unpatched. Unpatched instruments are not displayed in the Dashboard.

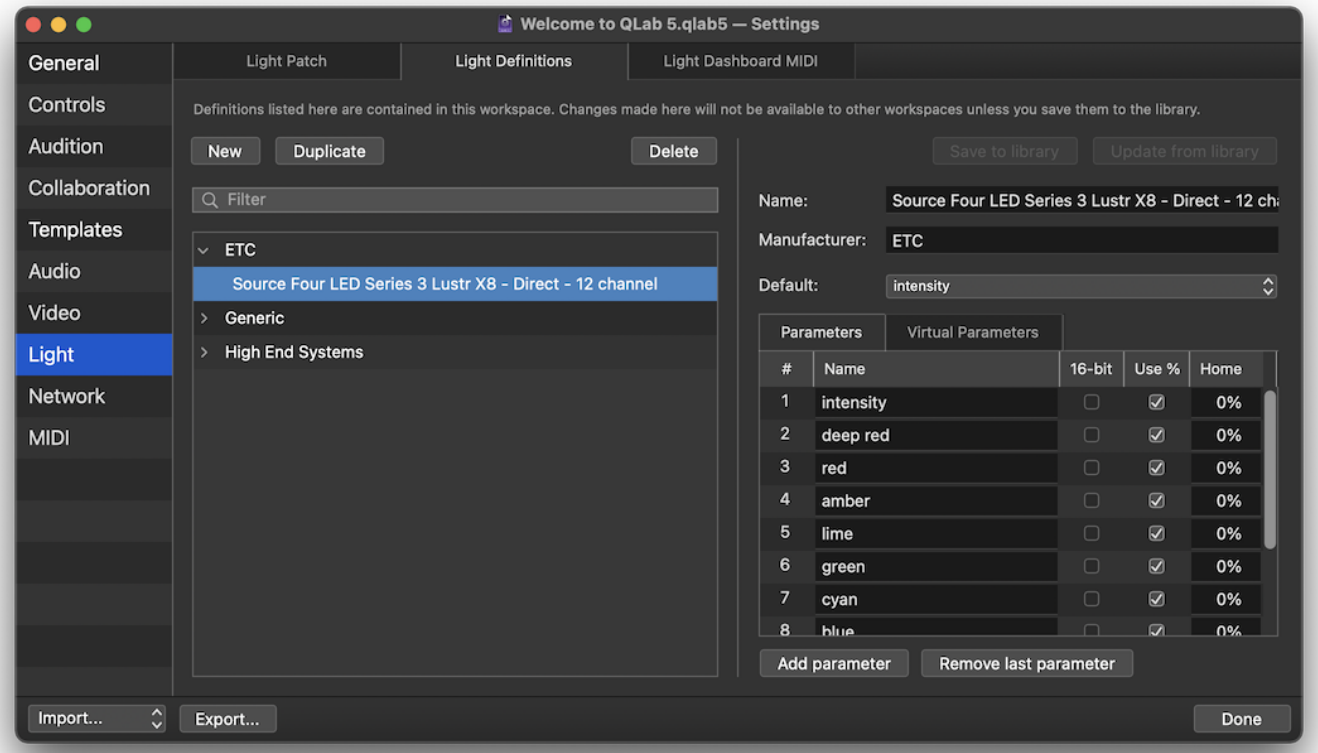
It's important to remember that all Art-net nodes on a network share one set of Universes, Sub-Nets, and Nets, whereas all USB DMX devices have their own, unique sets of Universes. If you have two single-universe Art-net nodes both set to universe 1, sub-net 2, net 3, then the address space is shared on both devices, and you have a total of 512 addresses available to use. If you have two single-universe USB DMX devices, they each refer to themselves as universe 0, but they are in fact separate address spaces and you have a total of 1024 addresses available to use.

The **Add to Group...** button displays a pop-up of all light groups in the workspace, and allows you add the selected instrument(s) and/or light group(s) to a light group. Light groups can indeed contain other light groups.

Below the button is a list of the light groups that the selected instrument(s) and/or light group(s) belong to.

The Light Definitions Tab

The Light Definitions tab lists every light definition used by at least one instrument in the workspace. Definitions are copied into the workspace from the Light Library when that definition is selected for an instrument in the light patch. If all the instruments using a particular definition are deleted from the workspace, however, the definition remains in the workspace until it's manually removed.



There are five buttons across the top of the Light Definitions tab:

- The **New** button creates a new instrument definition in the workspace.
- The **Duplicate** button creates a copy of the selected definition within in the workspace.
- The **Delete** button deletes the selected definition from the workspace. QLab will not permit you to delete a definition that is currently being used. Note that deleting a definition from a workspace does not delete that definitions from the Light Library.
- The **Save to library** is enabled when a definition is selected that exists in the workspace, but not in the Light Library. Clicking this button saves the definition to the Light Library, which allows it to be used in any workspace.
- The **Update from library** button is enabled if the definition in the workspace has the same name as a definition in the global Light Library, but different parameters. Clicking this button replaces the workspace's definition with the global one.

Filter. Type here to temporarily filter the list of definitions by name, to make it easier to find what you're looking for.

Definition List. Definitions are hierarchically sorted by manufacturer, and listed in alphabetical order. Selecting a definition allows you to view and edit its details to the right of the window.

Editing Instrument Definitions

Instrument definitions must include a unique *name*, a *manufacturer*, and at least one *parameter*. Definitions can optionally also include *virtual parameters* which enable the use of complex controls like color pickers.

Default. A definition can optionally have a default parameter, which is the parameter that will respond to commands that do not specify a parameter. In most cases, the default parameter is `intensity`, so that you can use commands like `1 = 100` or `1 = 50` and control the brightness of the light. If you want to set a different default parameter, or set no default parameter, use this drop-down menu.

To add a parameter to the currently selected instrument definitions, click the **Add parameter** button below the Parameters table. To delete the last parameter from the current instrument definition, click **Remove last parameter**.

The Parameters tab five columns: **#.** The order in which parameters are listed is defined by the manufacturer of the lighting instrument. If you are creating your own definitions, be sure to add parameters in the correct order according to the manufacturer's specifications.

Name. Parameter names must be unique per instrument, and can contain only letters, numbers, and spaces.

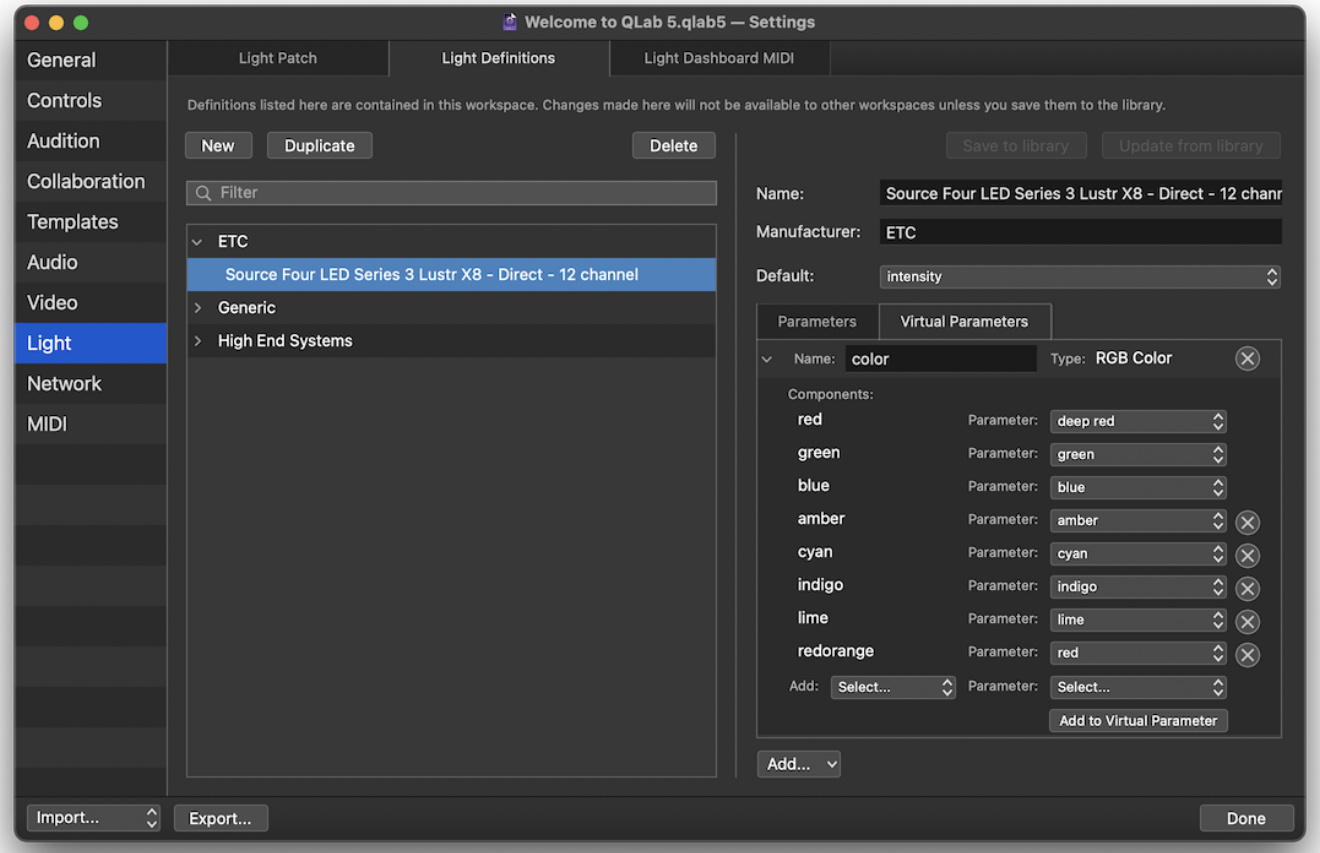
16-bit. Some lighting instruments and accessories use two DMX addresses for a single parameter to allow much more precise control; two DMX addresses allows for a range of 0 to 65,535. Check this box to assign the parameter to use two adjacent addresses together as a single 16-bit address.

Use %. DMX values range from 0 to 255. Check this box to have QLab express the parameter in terms of a percentage scale. This does not alter the behavior of the parameter at all; it only alters how the parameter is displayed in the Light Dashboard and in cues.

Home. Set the home position for the parameter. For dimmers, that's usually 0%, meaning off. For irises it's typically 100%, meaning open. For moving mirrors it's usually 50%, meaning center.

Virtual Parameters

Virtual parameters are a way of grouping together several related parameters of a light. These parameters can then be controlled together with a single light command, and with a corresponding control tool in the Light Dashboard.



There are four types of virtual parameters, each of which behaves in a specific way. To create a new virtual parameter, navigate to the Virtual Parameters tab of the Parameters table, click on the **Add...** button at the bottom, and select from the four options:

- **RGB Color.** This virtual parameter is used to combine color parameters on lights which use additive color mixing. Lights with this virtual parameter display an additive color picker control in the Light Dashboard and in the Light cue inspector. A light needs at least one `red`, one `green`, and one `blue` parameter to include in an *RGB Color* virtual parameter, and if the light has additional colors, such as `white`, `amber`, or `lime`, those can be added as well. If your light contains multiple cells or zones with their own color controls, you can create multiple *RGB Color* virtual parameters; one for each cell or zone.
- **CMY Color.** This virtual parameter is used to combine `cyan`, `magenta`, and `yellow` color parameters on lights which use subtractive color mixing (or simulated subtractive color mixing.) Lights with this virtual parameter display a subtractive color picker control in the Light Dashboard and in the Light cue inspector.
- **HSI Color.** This virtual parameter is used to combine `hue`, `saturation`, and `intensity` (or `value`) parameters on lights which use HSI or HSV color mixing. Lights with this virtual parameter display an HSI color picker control in the Light Dashboard and in the Light cue inspector.
- **Pan/Tilt.** This virtual parameter is used to combine `pan` and `tilt` parameters. Lights with this virtual parameter display a pan/tilt control in the Light Dashboard and in the Light cue inspector.
- **One-to-Many.** This virtual parameter is used to create a single control that simultaneously commands multiple individual parameters. For example, if you have a multi-cell strip light that does not have an overall intensity channel, you could create a `One-to-Many` virtual parameter that includes the intensity parameter for each cell. This virtual parameter will then function as an overall intensity control for the whole fixture. `One-to-Many` virtual parameters can include both regular parameters and other virtual parameters.

The Light Dashboard MIDI Tab

The Light Dashboard MIDI tab allows you to map incoming MIDI messages to light instrument parameters in the Light Dashboard. This lets you, for example, use MIDI control surfaces to directly control lights in QLab. You can learn more about the Light Dashboard MIDI tab in the Workspace Settings → Light section of this manual.

The Patch Menu

When the Light Patch Editor is active, the Tools menu becomes the **Patch** menu, containing tools for working with the light patch:

Unpatch All. Remove the DMX address assignment for all instruments in the workspace. Needless to say, the instruments will need to have DMX addresses reassigned before they can be used. Unpatched instruments are not displayed in the Dashboard.

Unpatch Selected. Remove the DMX address assignment for the instruments currently selected in the patch list.

Auto-patch All... Automatically assign DMX addresses to all instruments in the patch list. QLab will display a dialog box which allows you to choose the device you want to use, as well as the starting DMX address. QLab will then assign addresses to all instruments according to the specifics you enter into the box, moving sequentially through to the last parameter of the last instrument. This will overwrite any existing DMX assignments, but will not alter instrument names, definitions, light group assignments, or any other attributes.

Auto-patch Selected... Automatically assign DMX addresses to the instruments currently selected in the patch list. QLab will display a dialog box which allows you to choose the device you want to use, as well as the starting DMX address. QLab will then assign addresses to the selected instruments according to the specifics you enter into the box, moving sequentially through to the last parameter of the last instrument. This will overwrite any existing DMX assignments, but will not alter instrument names, definitions, light group assignments, or any other attributes.

Select all instruments selects every instrument in the light patch.

Select all unpatched instruments selects only instruments with absent or incomplete patch information.

Light Library

QLab's global Light Library can be found by choosing *Light Library* from the **Window** menu. The Light Library works exactly like the Definitions tab of the Light Patch Editor except that it lets you view and edit the global collection of lighting definitions on your Mac, rather than the collection of lighting definitions within the workspace.

Light definitions (other than the stock definitions shipped with QLab) are stored in `~/Library/Application Support/QLab/LightLibrary`; each definition is a JSON file with the file extension `.qlablight`.

If you are only just getting started with lighting in QLab, and have not done much work with light definitions, this folder may not yet exist since QLab only creates it to store definitions that you add or edit. If you go looking for this folder and cannot find it, you can create it yourself by hand. Just navigate to `~/Library/Application Support/QLab/` and create a new folder called `LightLibrary`.

QLab ships with definitions for the following fixtures. These definitions do not appear in the `LightLibrary` folder; they reside in the QLab application itself. The inclusion of a particular fixture in QLab's library should not be construed as an endorsement of that fixture, and omission of a fixture should not necessarily be construed as an admonishment. There are *quite a lot of lights* out there, and we cannot possibly assess each one individually. We try to include definitions for as many lights as possible that we think are likely to be used with QLab. If you would like us to create definitions for a specific fixture, or a whole range of fixtures, please contact us at support@figure53.com and tell us about it. We'll be happy to create the instrument definitions for you, and we may also add them to a subsequent release of QLab.

Generic

- Dimmer
- DMX Iris, 8-bit and 16-bit
- RGB fixture, with and without intensity channel, 8-bit and 16-bit
- RGBW fixture, with and without intensity channel, 8-bit and 16-bit
- RGBAW fixture, with and without intensity channel, 8-bit and 16-bit
- Scroller
- Variable white fixture with intensity channel, 8-bit and 16-bit

Acme

- Aeco 5
- Aeco 10
- Aurora
- Beluga
- Cetus
- Chiron
- Coolie 14FC
- Coolie 545
- Diabolo
- Ellipsoidal 10
- Ellipsoidal 40
- Ellipsoidal 300
- Energy Hybrid
- Energy Spot Pro
- Flamenco CW
- Flamenco WW
- Geist Beam
- Geist BSWF
- Geist Framing
- Glamour 350Z III
- Glamour 700Z
- Glory Framing 5000
- Glory Framing N
- Glory Profile T
- Glory Wash 5000

- Icarus 320
- Icarus 620
- Leo
- Libra
- Nebula CW
- Nebula WW
- Ocellaris
- Orion
- Oxozone
- Oxygen
- Ozone
- Pageant 300ZR II
- Pageant 600Z II
- Ray 500
- Rayzor Hybrid
- Scorpius
- Solar Flare II
- Solar Impulse
- Solaris
- Sparkling
- Spartan Hybrid
- Stage Blinder IP CW
- Stage Blinder IP WW
- Stage Blinder IP VW
- Stage Blinder IP RGBW
- Stage PAR 100 IP CW
- Stage PAR 100 IP WW
- Stage PAR 100 IP Colour
- Stage PAR 400 Zoom IP
- Stage PAR 600 Zoom IP
- Stage PAR CCT
- Stage PAR CCT D15
- Stage PAR Colour
- Stage PAR Colour D15
- Stage PAR Tungsten
- Stage PAR UV
- Sunrise
- Super Geist Framing
- Theatre Spot 150
- Theatre Spot 300
- Thunder
- Thunderbolt
- Thunderstruck
- Thunder Breaker
- Thunder Roar
- Tour Pixel Bar 1010 II
- TV Light Panel 1000 CW
- TV Light Panel 1000 WW
- TV Light Panel 1000 VW
- TV Light Panel 2000 CW
- TV Light Panel 2000 WW
- TV Light Panel 2000 VW
- TV Panel 100
- TV Panel 200
- Willow 500

Altman

- AP-150 RGBW LED Par
- Spectra Series fixtures (all models)
- PHX Series fixtures (all models)

American DJ

- 5P Hex
- 5PX Hex
- 12P Hex
- 18P Hex
- 7P Hex IP
- 12P Hex IP
- 18P Hex IP
- 15 Hex Bar IP
- 32 Hex Panel IP
- 3 Sixty 2R
- 64B LED Pro
- 7PZ IP
- Asteroid 1200
- Chameleon QBar Pro
- COB Cannon LP200ST
- COB Cannon Wash
- COB Cannon Wash DW
- COB Cannon Wash ST
- COB Cannon Wash ST DW
- Dotz Matrix
- Dotz Panel 2.4
- Dotz Par
- Dotz Par 100
- Eco UV Bar DMX
- Eliminator Stealth Beam
- Eliminator Stealth Craze
- Eliminator Stealth Spot
- Eliminator Stealth Wash Zoom
- Eliminator Stinger Spot 30
- Eliminator Stryker Beam
- Eliminator Stryker Wash
- Encore Burst RGBW IP
- Encore Burst UV IP
- Entour Faze
- Encore FR Pro Color
- Encore FR50Z
- Encore FR150Z
- Encore LP5IP
- Encore LP7IP
- Encore LP12IP
- Encore LP18IP
- Encore Profile 1000 Color
- Encore Profile 1000 WW
- Encore Profile Mini Color
- Encore Profile Pro Color
- Encore Profile Mini WW
- Encore Profile Pro WW
- Encore Z7LP
- Entour Haze Pro
- Entour Ice
- Entour Snow
- Entour Venue
- Entourage
- Event Bar Pro
- Flat Par QA5XS
- Flat Par TRI7XS
- Flat Par TRI18XS
- Flat Par TW5
- Flat Par TW12
- Focus Flex
- Focus Flex L7
- Focus Hybrid
- Focus Profile
- Focus Spot One
- Focus Spot Two
- Focus Spot Three Z
- Focus Spot 2X
- Focus Spot 4Z
- Focus Spot 5Z

- Focus Spot 6Z
- Focus Wash 400
- Fog Fury 3000
- Fog Fury Jett
- Fog Fury Jett Pro
- Hydro Beam X1
- Hydro Beam X2
- Hydro Beam X12
- Hydro Profile
- Hydro Spot 1
- Hydro Spot 2
- Hydro Wash X7
- Hydro Wash X19
- Inno Color Beam Z7
- Inno Color Beam Z19
- Inno Pocket Roll
- Inno Pocket Scan
- Inno Pocket Spot LZR
- Inno Pocket Spot Twins
- Inno Pocket Wash
- Inno Pocket Z4
- Inno Spot Pro
- Jelly PAR Profile
- Lightning COB Cannon
- Mega 64 Profile Plus
- Mega Bar 50RGB
- Mega Bar 50RGB RC
- Mega Bar RGBA
- Mega Flash DMX
- Mega HEX Par
- Mega PAR Profile Plus
- Mega TriPar Profile
- Mega TriPar Profile Plus
- MOD HEX100
- MOD QA60
- MOD QW100
- MOD STQ
- MOD TW100
- Ninja 5RX
- PAR Z Move
- PAR Z Move RGBW
- PAR Z100 3k
- PAR Z100 5k
- PAR ZP100 3k
- PAR Z120 RGBW
- PAR ZP120 RGBW
- Pocket Pro
- Profile Panel RGB
- Profile Panel RGBA
- Saber Bar 6
- Saber Spot RGBW
- Starship
- Stinger Spot
- Sweeper Beam Quad LED
- UB 6H
- UB 9H
- UB 12H
- UBL6H
- Ultra Bar 6
- Ultra Bar 9
- Ultra Bar 12
- Ultra Hex Par 3
- Ultra Hex Bar 6
- Ultra Hex Bar 12
- UV 72IP
- UV COB Cannon
- UV LED Bar20 IR
- Vizi Beam 5RX
- Vizi Beam 12RX
- Vizi Beam CMY
- Vizi Beam RXONE
- Vizi BSW 300
- Vizi CMY300
- Vizi Hex Wash7
- Vizi Hybrid 16RX
- Vizi Q Wash7
- Vizi Roller Beam 2R
- Vizi Wash Z19
- Vizi Wash Z37
- WiFly Bar QA5
- WiFly EZR HEX5 IP
- XS 600

AO Lighting

- Falcon 4000 LED-Video
- Falcon 4000 white
- Falcon Beam color 3000W
- Falcon Beam color 7000W
- Falcon Beam 2 Arc colour 3000W
- Falcon Beam 2 Arc colour 7000W
- Falcon Beam 2 Arc white 3000W
- Falcon Beam 2 Arc white 7000W
- Falcon Flower colour 3000W
- Falcon Static Arc colour 3000W
- Falcon Static Arc colour 7000W
- Falcon Static Arc white 3000W
- Falcon Static Arc white 7000W
- Mite Beam
- Sky Falcon Arc colour 1200W

ARRI

- SkyPanel series
- L-series DT
- L-series TT
- L-series C

Astera

Note: all Astera instruments use the same set of definitions, although not all definitions are available for use with every instrument.

- AX1 Pixeltube
- AX2-50 Pixelbar
- AX2-100 Pixelbar
- AX3 Lightdrop
- AX5 TriplePAR
- AX7 Spotlite
- AX9 PowerPAR
- AX10 Spotmax
- FP1 Titan Tube
- FP2 Helios Tube
- FP3 Hyperion Tube
- FP5 NYX Bulb
- FP6 HydraPanel
- PB15 Pixelbrick

Ayrton

- AlienPix-RS
- Bora
- CosmoPix-R
- Diablo
- Domino
- DreamPanel Shift
- DreamPanel Twin
- Eurus
- Ghibli
- Huracán-X
- IntilliPix-XT
- Karif-LT
- Khamsin
- Levanta
- MagicBlade-FX
- MagicBurst
- MagicDot-R
- MagicDot-SX
- MagicDot-XT
- MagicPanel-FX
- MiniBurst
- MiniPanel-FX
- Mistral
- NandoBeam S3
- NandoBeam S6
- NandoBeam S9
- Perseo
- VersaPix-RS
- WildSun K25-TC
- WildSun S25

Barco

- DMX Adapter
- F70
- F80
- HDQ
- HDX
- UDM
- UDX
- XDL

BeamZ

- 440Z
- BAC503
- BAC504
- BAC509
- BPP210
- BPP220

Big Dipper

- LM70
- LM70S
- LM70N
- LM70SN

Blizzard

- Aria Profile RGBW
- Aria Profile WW
- Colorise Zoom RGBAW
- Cyc Out
- G-Max 200
- HotBox 5
- Kryo Mix CMY
- LB Par Hex
- Oberon Profile NZ
- Oberon Profile WZ
- Oberon Fresnel
- Oberon Fresnel Zoom
- PixelStorm 240
- The Puck Fab5 Skywire

Boomtone DJ

- SilentPAR 7x10

BriTeQ

- Illusion 1
- Illusion 1 Mk II
- Panther 25

- Snowblind
- Stiletto GLO19
- Stiletto Ray-Z

- BT-Orbit
- BT-Profile HD
- BT-Profile WW
- BT-Shark
- BT-Theatre 100MZ
- BT-Theatre 150EZ Mk2
- BT-Theatre 250EZ Mk2
- BT-Theatre 200TW
- BT-Theatre 400TW
- BT-Theatre 261FC
- BT-Theatre 50WW
- BT-Theatre 60FCL
- BT-Theatre HD2
- BT-Theatre Venuelite4
- BT-Tracker
- BT-TVPanel TW
- BTI-Blizzard BSW2
- BTI-Cyclo
- BTX-Blizzard Wash1
- BTX-Saturn
- BTX-Supreme Hybrid
- BTX-Titan
- Beam Fury-1
- Pro Beamer Zoom Indoor Mk3
- Stage Beamer

Cameo

- Auro Bar 100
- Auro Beam 150
- Auro Beam 200 DC
- Auro Matrix 500
- Auro Spot 100
- Auro Spot 200
- Auro Spot 300
- Auto Spot Z300
- Auro Spot 400
- Azor B1
- CL 200
- Evos S3
- Evos W3
- Evos W7
- F2 D
- F2 FC
- F2 T
- F4 D
- F4 FC
- F4 T
- Flat 1 RGB 10 IR
- Flat 1 TRI 3W IR
- Flat 1 TW

- Flat Moon
- Flat PAR RGBW IR
- Flat Pro 7
- Flat Pro 7 IP
- Flat Pro 7 G2
- Flat Pro 7Spot
- Flat Pro 7XS
- Flat Pro 12
- Flat Pro 12 IP
- Flat Pro 12 G2
- Flat Pro 18
- Flat Pro 18 IP
- Flat Pro 18 G2
- Flat Pro Flood 600 IP65
- Flat Pro Flood IP65 TRI
- Flat Star
- Flat Storm
- Flat UV
- Flood 600
- Hydrabeam 1000 RGBW
- Instant Fog 1700 Pro
- Instant Hazer Pro 1400
- Instant Hazer Pro 1500T
- IODA 400 RGY
- IODA 600 RGB
- IODA 1000 RGB
- LUKE 400 RGY
- LUKE 700 RGB
- LUKE 1000 RGB
- Movo Beam 100
- Movo Beam Z 100
- Nanobeam 300
- Nanobeam 600
- Nanospot 120
- Nanospot 300
- Opus H5
- Opus S5
- Opus SP5
- Opus SP5 FC
- Opus X
- Otos H5
- Outdoor PAR TRI 12 IP
- PAR 56 Q 8W
- PAR 56 RGB 10
- PAR 56 RGB 5
- PAR 56 TRI 3W
- PAR 64 Q 8W
- PAR 64 RGB 10
- PAR 64 RGB 10
- PAR 64 RGB 3W
- PAR 64 RGBA 10
- PAR 64 RGBWAU 10W
- PAR 64 TRI 3W
- Phantom H2
- Phantom F5
- Pixbar 200 Pro
- Pixbar 300 Pro
- Pixbar 400 Pro
- Pixbar 450 Pro
- Pixbar 500 Pro
- Pixbar 600 Pro
- Pixbar 650 pro
- Pixbar DTW Pro
- Q-Spot 15 RGBW
- Q-Spot 15W
- Q-Spot 40 CW
- Q-Spot 40 RGBW
- Q-Spot 40 TW
- Q-Spot 40 WW
- Q-Spot 40i
- ROOT PAR 4
- ROOT PAR 6
- ROOT PAR TW
- Steam Wizard 1000
- Steam Wizard 2000
- Studio Mini COB 30W
- Studio Mini Q 4W W
- Studio Mini Q 8W
- Studio Mini TRI 3W
- Studio PAR 64 Q 8W
- Studio PAR 64 RGBA Q 8W
- Studio PAR 64 TRI 3W
- Studio PAR DTW
- Studio PAR 64 RGBWA+UV 12W
- SuperFly XS
- Thunder Wash 100 RGB
- Thunder Wash 100 W
- Thunder Wash 600 RGB

- Thunder Wash 600 RGBW
- Thunder Wash 600 UV
- Thunder Wash 600 W
- Tribar IR series
- TS 40 WW
- TS 60 W RGBW
- TS 100 WW
- TS 200 FC
- TS 200 WW
- Wookie 150 G
- Wookie 200 R
- Wookie 200 RGY
- Wookie 400 RGB
- Wookie 600 B
- Zenit B60
- Zenit B200
- Zenit P40
- Zenit P130
- Zenit P100 DTW
- Zenit P200 DTW
- Zenit W300
- Zenit W600
- Zenit W600-D
- Zenit Z120
- Zenit Z120 G2

Chauvet DJ

- 4Bar Flex Q ILS
- 4Bar LTBT ILS
- 4Bar Tri
- 4Bar Quad
- Abyss 2
- Abyss LED 3.0
- COLORband H9 USB
- COLORband PiX
- COLORband PiX-M USB
- COLORband Q4 IP
- COLORband T3 BT
- COREpar 40 USB
- COREpar 80 USB
- COREpar Q60 ILS
- COREpar Q120 ILS
- COREpar UV USB
- COREpar UV 120 ILS
- Cosmos HP
- Cubix 2.0
- Cumulus
- EVE P-130 RGB
- EZBeam Q3 ILS
- EZLink Par Q6BT ILS
- EZpar 64 RGBA
- EZpar T6 USB
- Festoon 2
- Freedom Gobo
- Freedom Par Q9
- Freedom Par Tri-6
- FXpar 3
- FXpar 9
- Freedom H1
- Freedom Q1N
- Freedom Stick
- Geyser P7
- Geyser RGB
- GigBAR 2
- GigBar Move ILS
- Hurricane Haze 2D
- Intimidator Barrel 305 IRC
- Intimidator Free Spot 60 ILS
- Intimidator Spot 355Z IRC
- Intimidator Spot 375Z IRC
- Intimidator Spot LED 150
- Intimidator Wash Zoom 450
- Kinta KX
- LED Followspot 120ST
- Mini Kinta
- Obsession
- Obsession HP
- Rotosphere HP
- Shocker Panel FX
- SlimBANK T18 USB
- SlimPANEL TRI 24 IP
- SlimPAR 38
- SlimPAR 64
- SlimPAR 64 RGBA
- SlimPAR H6 USB
- SlimPAR PRO H USB
- SlimPAR Q12 USB
- SlimPAR T12BT
- Wash FX 2
- Wash FX Hex
- Wedge Tri

Chauvet Professional

- AmHaze ECO
- AmHaze Stadium
- AmHaze Whisper
- Color STRIKE M
- COLORado 1 Quad
- COLORado 1 Quad Zoom
- COLORado 1 Solo
- COLORado 1 Tri IP
- COLORado 1 Tri Tour
- COLORado 1QS
- COLORado 2 Quad Zoom
- COLORado 2 Solo
- COLORado 3 Solo
- COLORado Batten 72
- COLORado Batten 72X
- COLORado Batten Q15
- COLORado M Solo
- COLORado Panel Q40
- COLORado Solo Batten
- COLORado Solo Batten 4
- COLORdash Accent 3
- COLORdash Accent Quad
- COLORdash Batten-Quad 12
- COLORdash Batten-Quad 6
- COLORdash Par H7X
- COLORdash Par H12X
- COLORdash Par H18X
- COLORdash Par Hex 12
- COLORdash Par Hex 7
- COLORdash Par Q12 IP
- COLORdash Par Quad 7
- COLORdash S-Par 1
- Cloud 9
- Legend 230SR
- Maverick Force 1 Spot
- Maverick Force 2 Profile
- Maverick Force 2 BeamWash
- Maverick Force 2 SoloWash
- Maverick Force 3 Profile
- Maverick Force S Profile
- Maverick Force S Spot
- Maverick MK Pyxis
- Maverick MK1 Hybrid
- Maverick MK1 Spot
- Maverick MK2 Profile
- Maverick MK2 Spot
- Maverick MK2 Wash
- Maverick MK3 Profile
- Maverick MK3 Profile CX
- Maverick MK3 Spot
- Maverick MK3 Wash
- Maverick Silens 1X Profile
- Maverick Silens 2 Profile
- Maverick Silens 2X Profile
- Maverick Storm 1 Beam
- Maverick Storm 1 Hybrid
- Maverick Storm 1 Spot
- Maverick Storm 1 Wash
- Maverick Storm 2 BeamWash
- Maverick Storm 2 Profile
- Maverick Storm 3 BeamWash (single-pixel modes)
- Maverick Storm 4 Profile
- Maverick Storm 4 SoloWash
- onAir Flex Drive 2
- onAir IP Panel Min
- onAir IP Panel 1
- onAir IP Panel 2
- onAir Panel 3 IP
- Ovation B-1965FC
- Ovation B-565FC
- Ovation CYC 1 FC
- Ovation CYC 3 FC
- Ovation E-2 FC
- Ovation E-4WW IP
- Ovation E-120WW
- Ovation E-120WW IP
- Ovation E-160WW
- Ovation E-260CW
- Ovation E-260WW
- Ovation E-260WW IP
- Ovation E-910FC

- Ovation E-930VW
- Ovation ED-190WW
- Ovation ED-200WW
- Ovation F-55FC
- Ovation F-55WW
- Ovation F-145WW
- Ovation F-165WW
- Ovation F-265WW
- Ovation F-415FC
- Ovation F-415VW
- Ovation F-915FC
- Ovation F-915VW
- Ovation FD-105WW
- Ovation FD-165WW
- Ovation FD-205WW
- Ovation H-265WW
- Ovation H-55FC
- Ovation H-55WW
- Ovation P-56FC
- Ovation P-56UV
- Ovation P-56VW
- Ovation P-56WW
- Ovation Rêve E-3
- Ovation Rêve F-3 IP
- Rogue Outcast 1 Beam
- Rogue Outcast 1 BeamWash
- Rogue Outcast 1L Beam
- Rogue Outcast 1M Beam
- Rogue Outcast 1 Hybrid
- Rogue Outcast 2 Beam
- Rogue Outcast 2 Hybrid
- Rogue Outcast 2W Wash
- Rogue Outcast 3 Spot
- Rogue R1 Beam
- Rogue R1 BeamWash
- Rogue R1 Spot
- Rogue R1 Wash
- Rogue R1E Spot
- Rogue R1X Spot
- Rogue R1X Wash
- Rogue R2 Beam
- Rogue R2 Spot
- Rogue R2 Wash
- Rogue R2E Spot
- Rogue R2X Beam
- Rogue R2X Spot
- Rogue R2X Wash
- Rogue R2X Wash VW
- Rogue R3 Beam
- Rogue R3 Spot
- Rogue R3 Wash
- Rogue R3E Spot
- Rogue R3X Wash
- Rogue RH1 Hybrid
- Strike 1
- Strike 4
- Strike Array 2
- Strike Array 4
- Strike Bolt 1C
- Strike P38
- Strike Saber
- Vesuvio II
- Well Gobo
- Well Panel

Christie

- Pandora's Box

Chroma-Q (Philips)

- Color Force 12
- Color Force 48
- Color Force 72
- Color One 100
- Color One 100X

CITC

- Maniac II

City Theatrical

- AutoYoke
- QolorPIX Tape Controller

Clay Paky

- A.leda B-EYE K10
- A.leda B-EYE K10 CC
- A.leda B-EYE K10 Easy
- A.leda B-EYE K20
- A.leda B-EYE K20 CC
- A.leda Wash K10
- A.leda Wash K10 CC
- A.leda Wash K10 TW
- A.leda Wash K10 W
- A.leda Wash K20
- A.leda Wash K20 CC
- A.leda Wash K20 TW
- A.leda Wash K20 W
- Actoris ParLED
- Actoris Profile FC
- Alpha Beam 700
- Alpha Beam 1500
- Alpha Profile 700
- Alpha Profile 800 ST
- Alpha Profile 1500
- Alpha Spot 300
- Alpha Spot HPE 300
- Alpha Spot HPE 700
- Alpha Spot QWO 800
- Alpha Spot HPE 1500
- Alpha Wash 300
- Alpha Wash 700
- Alpha Wash 1500
- Arolla Aqua
- Arolla Profile HP
- Arolla Profile MP
- Arolla Spot MP
- Axcor Beam 300
- Axcor Profile 400
- Axcor Profile 400 HC
- Axcor Profile 600
- Axcor Profile 900
- Axcor Spot 300
- Axcor Spot 400
- Axcor Spot 400 HC
- Axcor Wash 300
- Axcor Wash 600

- Axcor Wash 600 HC
- Hepikos
- HY B-EYE K15
- HY B-EYE K25
- K-EYE K10 HCR
- K-EYE K20 HCR
- K-EYE S10 HCR
- K-EYE S20 HCR
- Klemantis AS line
- Midi-B
- Midi-B FX
- Mini Xtylos
- Mini Xtylos HPE
- Mini-B
- Mini-B Aqua
- Mini-B ParLED Aqua
- Mini-B Spot
- Mythos
- Mythos 2
- Orkis Fresnel-Pebble M
- Orkis Profile M
- Panify
- ReflectXion
- Scenius Profile
- Scenius Spot
- Scenius Unico
- SharBar
- Sharpy
- Sharpy Plus
- Sharpy Plus Aqua
- Sharpy Wash
- Sharpy Wash 330 PC
- Sharpy X Frame
- Sharpy X Frame FD
- Sharpy X Spot
- Show Batten 100
- Show Batten 100 AS
- Sinfonya Profile 600
- Skylos
- Spheriscan
- Stormy
- Supersharpy
- Supersharpy 2
- Tambora Batten
- Tambora Flash
- Tambora Linear 100
- Volero Batten Aqua
- Volero Wave
- Xtylos

CLF Lighting

- Aorun
- Apollo
- Apollo XS
- Apollo XL
- Ares
- Ares XS
- Beam 6
- Color PAR 12
- Conan
- Dynamic White PAR
- EF Smoke 1500
- EF Smoke 3100
- Haze I
- Haze II
- Hera
- Hercules
- Juno
- LEDbar Pro
- LED Fan
- LED Fan XL
- LED Wash CW-WW
- LED Wash RGBW
- LEDWash XL
- Odin
- Orion
- Poseidon Beam
- Poseidon Beam Ultra
- Poseidon Hybrid
- Poseidon Mirror
- Poseidon Wash XS
- Poseidon Wash M
- Poseidon Wash XL
- Quadcolor Mini PAR
- Sera
- Sera Batt
- Serius
- SoftLED 100 VW
- SoftLED 200 VW
- Spectrum P1
- Spectrum P2
- Stinger
- Tricolor Mini PAR
- Xena
- Yara

Coemar

- Risalto LED S
- Risalto LED M

Color Kinetics (Philips)

- ColorBlast
- ColorBlast RGBA
- ColorBlast RGBW
- ColorBlaze 48
- ColorBlaze 72
- ColorBlaze RGBA 48
- ColorBlaze RGBA 72
- ColorBlaze RGBW 48
- ColorBlaze RGBW 72

CrowdGlow

- CrowdGlow

disguise

- State Listen or Command Mode
- Timeline Listen
- Variable Video Module

Draco

- Dracast Fresnet Studio Series 1000B

Elation

- ACL 360 BAR
- ACL 360i
- Artiste DaVinci
- Artiste Monet
- Artiste Van Gogh
- Chorus Line 8
- Chorus Line 16
- Color Chorus 12
- Color Chorus 24
- Color Chorus 48
- Color Chorus 72
- Colour 5 Profile
- CW Profile HP
- CW Profile HP IP
- DARTZ 360
- DLED Strinp RGBAW
- DTW PAR 300
- DW Chorus 12
- DW Chorus 24
- DW Chorus 48
- DW Chorus 72
- DW Fresnel
- DW PAR Z19 IP

- DW Profile
- E Spot III
- Emotion
- Fuze Par Z60
- Fuze Par Z120 IP
- Fuze Par Z175 IP
- Fuze Profile
- Fuze Profile CW
- Fuze Spot
- Fuze Wash 575
- Fuze Wash Z120
- Fuze Wash Z350
- KL Fresnel 4
- KL Fresnel 6
- KL Fresnel 8
- Opti Tri Par
- Paladin
- Platinum Beam 5R
- Platinum Beam 5R Extreme
- Platinum FLX
- Platinum HFX
- Platinum Seven
- Platinum Spot 15r
- Platinum Spot III
- Platinum Spot LED II
- Proteus Beam
- Proteus Hybrid
- Proteus Maximus
- Proteus Smarty Hybrid
- Proteus Smarty Max
- Proteus Rayzor 760
- Protron 3K
- Protron 3K Color
- Rayzor 360Z
- Rayzor 760
- Rayzor Beam 2R
- Rayzor Q7
- Rayzor Q12
- Satura Profile
- SEVEN Batten 14
- SEVEN Batten 42
- SEVEN Batten 72
- SEVEN PAR 7IP
- SEVEN PAR 19IP
- SixBar 1000
- SIX PAR Z19 IP
- SIXPAR 100
- SIXPAR 100IP
- SIXPAR 200
- SIXPAR 200IP
- SIXPAR 200WMG
- SIXPAR 200WMG HW
- SIXPAR 300
- SIXPAR 300IP
- SIXPAR 300WMG
- SIXPAR 300WMG HW
- TVL Cyc RGBW
- TVL Panel DW
- TVL Softlight DW
- WW Profile
- WW Profile HP IP
- ZCL 360i

Electronic Theater Controls (ETC)

- ColorSource Cyc
- ColorSource Linear
- ColorSource PAR
- ColorSource Fresnel V
- ColorSource Spot
- ColorSource Spot jr
- ColorSource Spot V
- Desire Fresnel
- fos/4 Fresnel
- fos/4 Panel
- Relevé
- Selador Desire D40 series
- Selador Desire D60 series
- Selador Vivid 11 (Lustr, Paletta, and R)
- Selador Vivid 21 (Lustr, Paletta, and R)
- Selador Vivid 42 (Lustr, Paletta, and R)
- Selador Vivid 63 (Lustr, Paletta, and R)
- Source 4WRD Color II
- Source Four LED Series 1 (Lustr+ and Studio HD)
- Source Four LED Series 2 (Lustr, Daylight HD, and Tungsten HD)
- Source Four LED Series 3 (Daylight HDR and Lustr X8)

Epson

- EB-PU2100 series
- EB-PU2200 series
- Pro L1405U
- Pro L1500U
- Pro L1505U

Equinox

- Fusion 120 Zoom MkII
- Micro Batt
- SpectraPix Lithium Batten
- Water

Eurolite

- LED Bar 6
- LED Bar 12
- LED PARty TCL Spot
- LED PFE-100 RGBW Profile Spot
- LED PLL-480 QCL Panel
- LED STP-14 Sunbar
- LED THA-100F (MK1, MK2, and MK3)
- LED THA-250F COB
- Stage Panel 16 HCL LED.

Froggy's Fog

- Boreas C6
- Fobbles F4
- Hyperion D4
- Hyperion D6
- Hyperion SJ8
- Poseidon Aqua 2
- Poseidon Aqua 4
- Poseidon A6
- Titan Hazer H2
- Titan Hazer H4
- Titan Hazer HT6

Gantom

- Gantom 7
- Gantom DMX Flood
- Gantom DMX Spot
- Gantom IQX
- Precision

German Light Products (GLP)

- Agilio
- Force 120
- Fusion EXO Beam 10
- Fusion EXO Spot 30
- Fusion EXO Wash 30
- Fusion MBL 20
- Fusion MBL 40
- Fusion Stick FS10
- Fusion Stick FS16 Z
- Fusion Stick FS20
- Fusion X-PAR 12Z
- GT-1
- Highlander
- Impression 120RZ

- Impression 240XL
- Impression 90 RGB
- Impression 90 RGB Static
- Impression 90 WhiteAmber
- Impression E350
- Impression FR1
- Impression FR10 Bar
- Impression S350
- Impression S350 Wash
- Impression SpotOne
- Impression WashOne
- Impression X1
- Impression X4
- Impression X4 L
- Impression X4 S
- Impression X4 S Tunable White
- Impression X4 Tunable White
- Impression X4 XL
- Impression X4 Bar 10
- Impression X4 Bar 20
- Impression X5
- Impression X5 Bar 1000
- Impression X5 Compact
- JDC1
- JDC Line 500
- JDC Line 1000
- Volkslicht RGB
- Volkslicht RGB Spot
- Volkslicht Spot
- X4 Atom PSU-6
- X4 Atom PSU-12

High End Systems (ETC)

- Cyberlight LED
- Halcyon Gold
- Halcyon Platinum
- Halcyon Titanum
- Hyperstar
- Lonestar
- Ministar
- Quad
- SolaFrame 750
- SolaFrame 1000
- SolaFrame 1500
- SolaFrame 2000
- SolaFrame 3000
- SolaFrame Studio
- SolaFrame Theatre
- SolaHyBeam 1000
- SolaHyBeam 2000
- SolaHyBeam 3000
- SolaPix 7
- SolaPix 19
- SolaPix 37
- SolaSpot 1000
- SolaSpot 2000
- SolaSpot 3000
- SolaSpot Pro CMY
- SolaWash 1000
- SolaWash 2000
- SolaWash 3000
- Studio Color 575
- Talen
- Technobeam
- TurboRay

ibiza-light

- LED Bar24-RC

Infinity (Highlite)

- B401 Beam
- Furion S201 Profile
- Furion S401 Spot
- Furion S601 Profile

- iB-2R
- iB-5R
- iB-16R
- iFX-615
- iM-2515
- iS-100
- iS-200
- iS-250
- iW-720 RDM
- iW-740 RDM
- iW-741 RDM
- iW-1240 RDM
- iW-1915 Pixel
- iW-1941 RDM
- TCYC-7 Cyclorama
- TF-300 Fresnel
- TF-300 Profile
- TFLD-7 Floodlight
- TS-260C7 Fresnel
- TS-260C7 Profile

JB Lighting

- A8
- A12
- A8 Tunable White
- A12 Tunable White
- P7 Spot
- P12 Spot
- P12 Profile
- P18 Profile mk2
- P12 Wash
- P12 Wash mk2
- Sparx 7
- Sparx 10
- Sparx 18
- Sparx 30

K9

- Bulldog
- Bulldog Pro
- Labrador
- Pup

Martin

- Atomic 3000
- Atomic 3000 LED
- ELP CL
- ELP WW
- ELP PAR
- ERA 150 Wash
- ERA 300 Profile
- ERA 400 Performance CLD
- ERA 400 Performance WRM
- ERA 500 Hybrid IP
- ERA 600 Performance
- ERA 600 Profile
- ERA 700 Performance IP
- ERA 800 Performance
- ERA 800 Profile
- MAC 101
- MAC 101 CT
- MAC 101 CLD
- MAC 101 WRM
- MAC 250
- MAC 250+
- MAC 250 Krypton
- MAC 250 Entour
- MAC 250 Wash
- MAC 300
- MAC 301 Wash
- MAC 350 Entour
- MAC 500
- MAC 575 Krypton
- MAC 600
- MAC 600 NT
- MAC 700 Profile
- MAC 700 Wash
- MAC 1200
- MAC 2000 Performance
- MAC 2000 Performance II
- MAC 2000 Wash XB
- MAC Allure Profile
- MAC Allure Wash PC
- MAC Aura
- MAC Aura PXL
- MAC Aura XB
- MAC Auda XIP
- MAC Axiom Hybrid
- MAC Encore Performance CLD
- MAC Encore Performance WRM
- MAC Encore Wash CLD
- MAC Encore Wash WRM
- MAC One
- MAC Quantum Profile
- MAC Quantum Wash
- MAC TW1
- MAC Ultra Performance
- MAC Ultra Wash
- MAC Viper AirFX
- MAC Viper AirFX Quadray
- MAC Viper Performance
- MAC Viper Profile
- MAC Viper Wash
- MAC Viper Wash DX
- MAC Viper XIP
- RUSH Blinder 1 WW
- RUSH CS
- RUSH FiberSource 1
- RUSH MH 1 Profile Plus
- RUSH MH 2 Wash
- RUSH MH 3 Beam

- RUSH MH 4 Beam
- RUSH MH 5 Profile
- RUSH MH 6 Wash
- RUSH MH 6 Wash CT
- RUSH MH 7 Hybrid
- RUSH MH 8
- RUSH MH 10 Beam FX
- RUSH MH 11 Beam
- RUSH PAR 1 RGBW
- RUSH PAR 2 CT Zoom
- RUSH PAR 2 RGBW Zoom
- RUSH PAR 3 RGB
- RUSH PAR 4 UV
- RUSH Scanner 1
- THRILL Compact Par 64 LED
- VDO Atomic Bold
- VDO Atomic Dot CLD
- VDO Atomic Dot WRM

Marq

- Gesture Spot 300
- Gesture Spot 500

Mega-Lite (MEGA Systems)

- Axis Grid
- Axis Mini Grid
- Baby Color H84
- Baby Color Q70
- Baby Color VW
- Circa Scoop
- Circa Scoop XS
- Circa Scoop XL
- Color PAC Q900
- Color PAC Spot W300
- Colorlite Q120
- Drama FS-800
- Drama FS-900
- Drama W50
- Drama W120
- Drama Profile C3
- Drama Profile Q2
- Drama Profile VW2
- Drama Profile W3
- Drama Z50 W
- Drama Z50 CW
- Drama Z50 WW
- Drama Strip P840
- MB1
- MS1
- MW1
- N-E Color FX18
- Nova-Lite CW150
- Nova-Lite VW150
- Nova-Lite WW150
- Nova-Lite Q200
- Nova-Lite UV200
- O Blinder 700
- Obra Flood N180
- Outshine Q500
- Outshine Strip Q80
- Piccolo Blinder 120
- Spotbot
- Tiltbot Trio
- Tuff Baby P84
- Tuff Baby Q60
- Tuff Baby UV35
- Tuff Baby VW42
- Washbot
- XS LED RGB
- XL LED RGB
- XS LED W
- XL LED W

Minute Une

- IVL Carré
- IVL Pyramide
- IVL Square

Monoprice (Stage Right)

- 3-Color LED Light Bar
- 3-Color LED Moving Head Light
- 3-Color LED PAR-64
- Ellipsoidal 60-watt COB LED
- Ellipsoidal 180-watt COB RGBW LED
- Ellipsoidal 200-watt COB LED
- PAR 8-watt x7 RGBW LED
- PAR 10-watt x6 RGBW LED IP65
- PAR 10-watt x9 RGBW LED
- PAR 12-watt x7 RGBAW-UV LED
- PAR 15-watt x12 RGBAW LED
- PAR 18 Watt x18 RGBWA-UV LED
- PAR 18-watt x18 RGBWA-UV LED with zoom
- Stage Beam 30-watt LED Moving Head
- Stage Wash 10-watt x7 RGBW LED Moving Head
- Stage Wash 10-watt x36 LED RGBW Moving Head with zoom
- Stage Wash 12-watt x7 LED Moving Head RGBW with zoom
- Truss Wash 3-watt x3 Uplight

Ocean Thin Films

- SeaChanger Color Engine

Panasonic

- PT-DS20K
- PT-DS20K2
- PT-DW17
- PT-DW17K2
- PT-DW750
- PT-DW830E
- PT-DX100E
- PT-DX820
- PT-DZ16K2
- PT-DZ21K
- PT-DZ21K2
- PT-DZ780
- PT-DZ870E
- PT-FRZ50
- PT-FRZ60
- PT-MZ10K
- PT-MZ13K
- PT-MZ16K
- PT-RCQ10
- PT-RCQ80
- PT-RQ13K

- PT-RQ22K
- PT-RQ32K
- PT-RQ35K
- PT-RS20K
- PT-RS30K
- PT-RW620
- PT-RW730
- PT-RW930
- PT-RX110
- PT-RZ16K
- PT-RZ21K
- PT-RZ31K
- PT-RZ120
- PT-RZ570
- PT-RZ575
- PT-RZ660
- PT-RZ670
- PT-RZ690
- PT-RZ770
- PT-RZ790
- PT-RZ870
- PT-RZ890
- PT-RZ970
- PT-RZ990

PR Lighting

- XR220 BWS

Prolights

- Air 5Fan
- ArenaCob 4FC
- ArenaCob 4Halo
- Aria 700Profile
- Astra Beam260IP
- Astra Wash7Pix
- Astra Wash19Pix
- Diamond 7
- Diamond 19
- Diamond 19CC
- Diamond 19TW
- DisplayCob DY
- DisplayCob FC
- DisplayCob WW
- EclCyclorama 50
- EclCyclorama 100
- EclFresnel DY or PDY
- EclFresnel TU or PTU
- EclFresnel TW
- EclFresnel Jr DY or PDY
- EclFresnel Jr TU or PTU
- EclFresnel Jr TW
- EclFresnel Mini DY or PDY
- EclFresnel Mini TU or PTU
- EclFresnel Mini TW
- EclFresnel 2K DY or PDY
- EclFresnel 2K TU or PTU
- EclFresnel 2K TW
- EclPanel TWC
- EclPanel Jr TWC
- EclPar DY, TU, or UV
- EclPar FC
- EclProfile CT+
- EclProfile FC
- EclProfile FS
- EclProfile FW
- EclProfile HDTWC
- EclProfile HD2
- EclProfile JrZIP
- Jade
- Jet Beam1
- Jet Beam2
- Jet Spot1
- Jet Spot2
- Jet Spot3
- Jet Spot4Z
- Luma 1500SH
- Luma 1500SP
- LumiPar 7IP
- LumiPar 12IP
- LumiPar 12UAW
- LumiPar 12UH
- LumiPar UQPro
- LumiPix 8H
- LumiPix 12UQ
- LumiPix 16H
- LumiPix 15IP
- PanoramaIP AirBeam
- PanoramaIP Spot
- PanoramaIP WBX
- Pin SpotD
- Pixie Beam
- Pixie Spot
- Pixie Wash
- Pixie WashXB
- Pixie Zoom
- Pixie ZoomXB
- Ra 2000Profile
- Ra 2000ProfileHB
- Ra 3000Profile
- Razor 440
- Ruby
- Ruby FCX

- Solar 27Q
- Solar 48Q
- Stark 400
- Stark 1000
- Stark Bar1000
- Stark Blade8
- StudioCob DY
- StudioCob FC
- StudioCob UV
- StudioCob WW
- StudioCob Plus DY2
- StudioCob Plus FC
- StudioCob Plus TW
- StudioCob Plus UV
- StudioCob Plus WW
- Sunbar 2000FC
- Sunbar 2500Max
- Sunblast 3000FC
- Sunblast 3500Max
- Sunrise 2IP
- Sunrise 2L
- Sunrise 4
- VersaPar
- Mini VersaPar

Rayzr

- MC series

Robe

- BMFL Blade
- BMFL Followspot
- BMFL Followspot LT
- BMFL Spot
- BMFL Wash
- BMFL Wash XF
- BMFL WashBeam
- BMFL WashBeam EV
- Colormix 240
- ColorStrobe
- Cuete
- CycBar 15X
- CycFX 4
- CycFX 8
- Cyclone
- DigitalSpot 3500 DT
- Divine 60 UV
- Divine 160 RGBW
- Divine 160 SC
- Divine 160 SW
- DL4F Wash
- DL4S Profile
- DL4X Spot
- DL7F Wash
- DL7S Profile
- Dominator 1200 XT
- Esprite
- Esprite FS
- Esprite Fresnel
- Esprite PC
- Footsie 1 RGBW
- Footsie 1 TW
- Footsie 1 WW
- Footsie 2 RGBW
- Footsie 2 TW
- Footsie 2 WW
- Forte
- Forte FS
- iBar 15
- iBeam 350
- iBolt
- iForte
- iPainte
- iParfect 150 FW
- iParfect 150 RGBA
- iParfect 150 FW RGBA
- iPointe
- iT12 Profile
- iT12 Fresnel
- iT12 PC
- iTetra 2
- iSpiider
- LEDBeam 100
- LEDBeam 100 DayLight
- LEDBeam 100 SmartWhite
- LEDBeam 150
- LEDBeam 150 FW
- LEDBeam 150 RGBA
- LEDBeam 150 FW RGBA
- LEDBeam 350
- LEDBeam 350 FW
- LEDWash 300
- LEDWash 300 SmartWhite
- LEDWash 300+
- LEDWash 300X
- LEDWash 600
- LEDWash 600+
- LEDWash 600X
- LEDWash 800
- LEDWash 800 DayLight
- LEDWash 800 SmartWhite
- LEDWash 800X
- LEDBeam 1000

- LEDWash 1200
- MegaPointe
- MiniMe
- MiniMe DV
- MiniPointe
- MMX Blade
- MMX Spot
- MMX WashBeam
- Painte
- Painte Spot
- ParFect 100
- ParFect 100 DL
- ParFect 100 SW
- ParFect 150
- ParFect 150 FW
- ParFect 150 RGBA
- ParFect S1
- PATT Driver (suitable for the onePATT, picklePATT, pixelPATT, and PATT 2017)
- Pointe
- ProMotion
- RoboSpot Motion Camera
- SilverScan
- Spiider
- Spikie
- Spote
- SuperSpikie
- Strobe IP
- T1 Fresnel
- T1 PC
- T1 Profile
- T1 Profile FS
- T2 Fresnel
- T2 PC
- T2 Profile
- T2 Profile FS
- T11 Fresnel
- T11 PC
- T11 Profile
- TX1 PosiProfile
- T15 Fresnel
- T15 PC
- T32 CYC
- T32 CYC Slim
- Tarrantula
- Tetra 1
- Tetra 2
- TetraX
- Viva
- Viva CMY

Robert Juliat

- Alice
- Charles 961 SX
- Charles 963 SX
- Charles 964 SX
- Dalis 860
- Dalis 861
- Dalis 862
- Dalis 862S
- Dalis 863
- Dalis 864
- Dalis 864S
- Oz
- Roxie2
- RJ-LED
- RJ-NET
- Tibo 533
- Tibo 535
- ZEP2 340 LF2
- ZEP2 360 LF2
- ZEP2 640 SX2
- ZEP2 660 SX2

Rockville

- Motion Strip
- Rock Par 50
- Rock Spot 260w
- RockOn-7
- RockPAR TRI
- Rockstrip 252
- Stage Matrix 36 RGB
- Stage Panel 864

Rosco

- I-Cue
- RevoPRO

Selecon (Philips)

- PLCYC1 mkII
- PLFRESNEL1 mkII
- PLPROFILE1 mkII
- PLPROFILE4 mkII

SGM

- G-1 Beam
- G-1 Wash
- G-4 Wash Motorized Barndoors
- G-4 Wash
- G-4 Washbeam
- G-4 Wash White
- G-4 Washbeam White
- G-7 Spot
- G-Profile
- G-Profile Turbo
- G-Spot
- G-Spot Turbo
- G-Wash
- i-2
- i-5
- P-1
- P-2
- P-5
- P-6
- P-10
- Q-2
- Q-7
- Q-10
- S-4
- SixPack

Showline (Philips)

- SL Band 300 RGBW
- SL Band 300 TW
- SL Bar 510
- SL Bar 510N
- SL Bar 520
- SL Bar 620
- SL Bar 640
- SL Bar 660
- SL Bar 720ZT
- SL Beam 100
- SL ePar 180

- SL eStrip 10 RGBW
- SL eSTROBE 130
- SL Hydrus 350
- SL LEDSpot 300
- SL Nitro 510
- SL Nitro 510C
- SL PAR 155 Zoom RGBW
- SL ParBlazer 100 UV
- SL Punchlite 220
- SL Strip 10 IP
- SL Wash 350

ShowPro

- EX35 LED Flood
- Hex 16 Strip
- LEDFRE117

Showtec (Highlite)

- ACT Flood 80 RGBW
- ACT PC 60 RGBW
- ACT Fresnel 50 WW
- ACT Profile 50 WW
- Compact Par 7 Tri MKII

- Compact Par 7 Q4
- Compact Par 7/14 Q4
- Dreamwave
- Followspot LED 75W
- Galactic RGB-300
- Kanjo Spot 10
- Kanjo Spot 60
- Par65 LED 100z 3200
- Performer Fresnel 1000 LED MkII
- Performer Fresnel 1500 Daylight
- Performer Fresnel 1500 Q6
- Performer Fresnel 1500 Tungsten
- Performer Fresnel 2000 DDT MkII
- Performer Fresnel 2000 LED
- Performer Fresnel 2000 RGBAL
- Performer Fresnel 2500 Daylight
- Performer Fresnel 2500 Q6
- Performer Fresnel 2500 Tungsten
- Performer Pendant Q6
- Performer Profile 600 DDT
- Performer Profile 600 LED MkII
- Performer Profile 600 LED MkIII
- Performer Profile 600 Q4
- Performer Profile 600 Q5
- Performer Profile IP 3200K
- Performer Profile IP Q4
- Phantom 3R Beam
- Phantom 3R Hybrid
- Phantom 60 LED Bar
- Phantom 65 Spot
- Phantom 100 Spot
- Phantom 130 Spot
- Phantom 180 Wash
- Phantom 280 Hybrid
- Phantom 1220 Zoombar
- Phantom Matrix FX
- Polar 100 Beam
- Polar 300 Hybrid
- Polar 340 WashFX
- Saber
- Shark Beam FX One
- Shark Combi Spot One
- Shark Spot One
- Shark Wash One
- Shark Zoom Wash One
- Spectral PC 600Z IP
- Spectral PC 1800Z IP
- Sunstrip Active

Stairville

- Beam Moving Head B5R
- CLB2.4 Compact LED Par System
- HL-x18 QCL RGBQ Floor 18x8W
- Led Bar 240/8
- LED Vintage Bowl
- MH-X25
- MH-X50+ LED Spot
- Octagon Theater CW WW 36x1W
- Quad Par Profile 5x8W RGBW
- RevueLED 150
- xBrick Quad 16x8W

Strand Lighting (Philips)

- Aurora LED Strip 4 Cell
- Aurora LED Strip 12 Cell
- Canata LED Fresnel Full Color
- Canata LED Fresnel Tunable Cold White
- Canata LED Fresnel Tunable Warm White
- Coda LED Cyc
- Leko LED Profile Full Color
- Leko LED Profile Tunable Cold White
- Leko LED Profile Tunable Warm White

Ultratec

- Radiance

Vari*Lite (Philips)

- Fresnel FC
- Fresnel TCW
- Fresnel TWW
- IP Image Projecting Profile FC
- IP Image Projecting Profile CW
- Profile Light Engine FC
- Profile Light Engine TCW
- Profile Light Engine TWW
- VL5LED
- VL10 BeamWash
- VL440 Spot
- VL500 Wash
- VL550 Wash
- VL600 Acclaim Cyc
- VL600 Acclaim Fresnel
- VL600 Acclaim Fresnel Plus
- VL600 Acclaim Zoomspot
- VL770 Spot
- VL800 BeamLine
- VL800 EVENTPAR RGBA
- VL800 EVENTPAR WW
- VL800 EVENTPROFILE

- VL800 EVENTWASH
- VL800 PROPAR
- VL880 Spot
- VL1000 (TI, AI, TS, AS)
- VL1100 (TI, AI, TS, AS)
- VL1100 LED
- VL1100 LED HP
- VL1600 Profile
- VL2000 Spot
- VL2000 Wash
- VL2500 Spot
- VL2500 Wash
- VL2600 Profile
- VL2600 Spot
- VL2600 Wash
- VL3000 Spot
- VL3000 Wash
- VL3000Q Spot
- VL3000Q Wash
- VL3015 Spot
- VL3015LT Spot
- VL3500 Spot
- VL3500 Wash
- VL3500 Wash FX
- VL3515 Spot
- VL3600 LT Profile IP
- VL3600 Profile IP
- VL4000 BeamWash
- VL4000 Spot
- VL6000 Beam
- VL6500 Wash
- VLX Wash
- VLX3 Wash
- VLZ Profile
- VLZ Spot
- VLZ Wash

Wybron

- Coloram
- CXI
- Forerunner

Yellow River

- P1012-H

Yorkville

- LP-LED/x Bar series

zactrack

- zactrack

Chapter 8: Networking, MIDI, and Show Control

8.1 Collaboration
8.2 QLab Remote
8.3 Stream Deck
8.4 Using OSC
8.5 Using MIDI & MSC
8.6 Using Timecode
8.7 Show Control Broadcast
8.8 Network Cues
8.9 MIDI Cues
8.10 MIDI File Cues
8.11 Timecode Cues

Collaboration

QLab Collaboration allows multiple people using multiple Macs to collaborate on a single QLab 5 workspace over a local network.

Terminology

The Mac that hosts the QLab workspace which others will connect to is called the **primary** Mac. The workspace on that Mac, therefore, is called the **primary workspace.**

A Mac which connects to the primary is called a **remote client**, or sometimes either a **remote** *or* a **client**. The workspace as it appears on a remote mac is called a **remote workspace.**

The types of access that a remote has while collaborating are called the remote's **access permissions.** There are three types of permissions:

- **Connect & View** access, also called just **view** access, allows the remote to connect and view the primary workspace but does not allow starting, stopping, pausing, or resuming cues; moving the playhead; or editing anything in the workspace other than flagging or un-flagging cues and editing cue notes. A remote without view access will not be allowed to connect to the primary.
- **Edit** access allows the remote to edit all parameters of all cues and workspace settings, but does not allow starting, stopping, pausing, or resuming cues or moving the playhead.
- **Control** access allows the remote to start, stop, pause, and resume cues and set the position of the playhead, but does not allow editing cues or settings.

Requirements

Any Mac that can run QLab 5 can be a collaboration primary. The license or licenses installed on the Mac dictate the available behavior.

With no license installed on the primary Mac:

- Any number of Macs with licensed copies of QLab can connect with any level of access. While connected, they behave as though they have no license installed.
- Any number of Macs with un-licensed copies of QLab can connect with *connect & view* access only.

When one or more licenses installed on the primary Mac:

- Each license installed on the primary allows one Mac with an un-licensed copy of QLab to connect with any level of access.
- Any number of Macs with licensed copies of QLab can connect with any level of access.
- Any number of Macs with un-licensed copies of QLab can connect with *connect & view* access only.

Collaboration sessions are always subject to the license installed on the primary. Every Mac collaborating on a workspace will behave according to the license or licenses installed on the primary Mac.

Example

The primary Mac, a Mac Studio in the tech booth, has an audio license and a video license installed. This means:

- Two Macs with un-licensed copies of QLab can both connect with any level of access permission. Both of them will behave as though they have audio and video licenses installed while they are connected to the primary.
- Additional Macs can also connect with any level of access permission as long as those Macs have at least one license of any kind installed. No matter what license they have, though, they will behave as though they have audio and video licenses installed while they are connected to the primary.
- Additional Macs with un-licensed copies of QLab can connect with view-only access. This makes it easy for a stage manager or backstage crew members to follow along with QLab visually as long as they already have access to a Mac.

Physically, all Macs involved in collaboration must be connected to the same local area network via ethernet or WiFi, and must use compatible IP addressing schemes. Collaboration also works over virtual private networks (VPNs) so with a little careful planning, an off-site Mac can be included in your collaboration setup. In short, if two Macs are able to connect via file sharing or screen sharing, they will be able to connect via QLab Collaboration as well.

> The Basic Computer Networking for the Theater tutorial provides an overview of the basics of setting up a network geared towards theater practitioners. If all this networking stuff is making your head spin a little, this tutorial can help!

QLab Collaboration uses Bonjour to make connections between Macs, which means less configuration work for you.

Setting Up A Primary

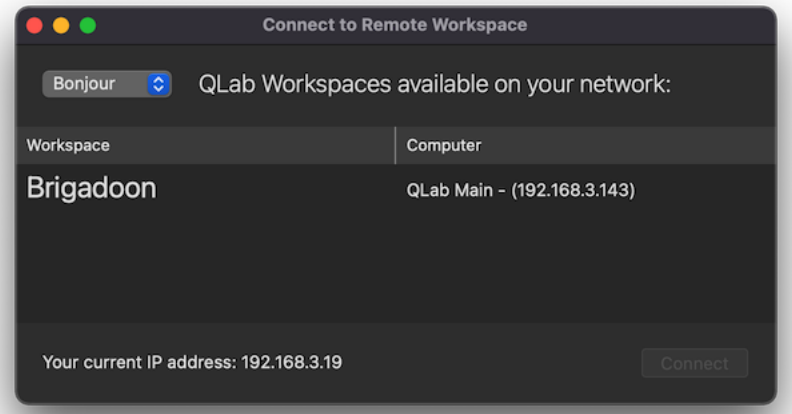
This topic is covered in the Workspace Settings → Collaboration section of this manual.

Connecting As A Remote

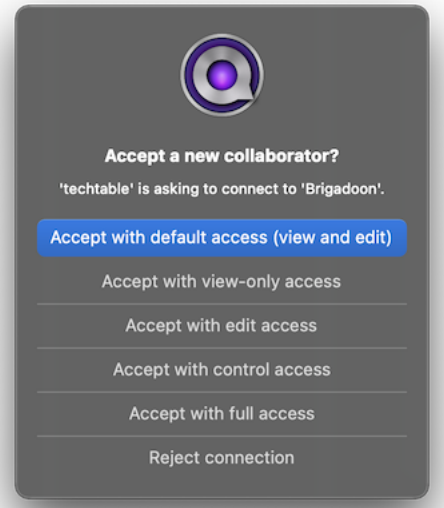
There are three ways to connect as a remote to a primary Mac:

- From the Launcher Window, click the **Connect to Workspace** button
- Choose *Connect to Workspace...* from the **File** menu
- Use the keyboard shortcut ⇧⌘**K**

Any of these three options will open a window listing the QLab workspaces available to connect to.



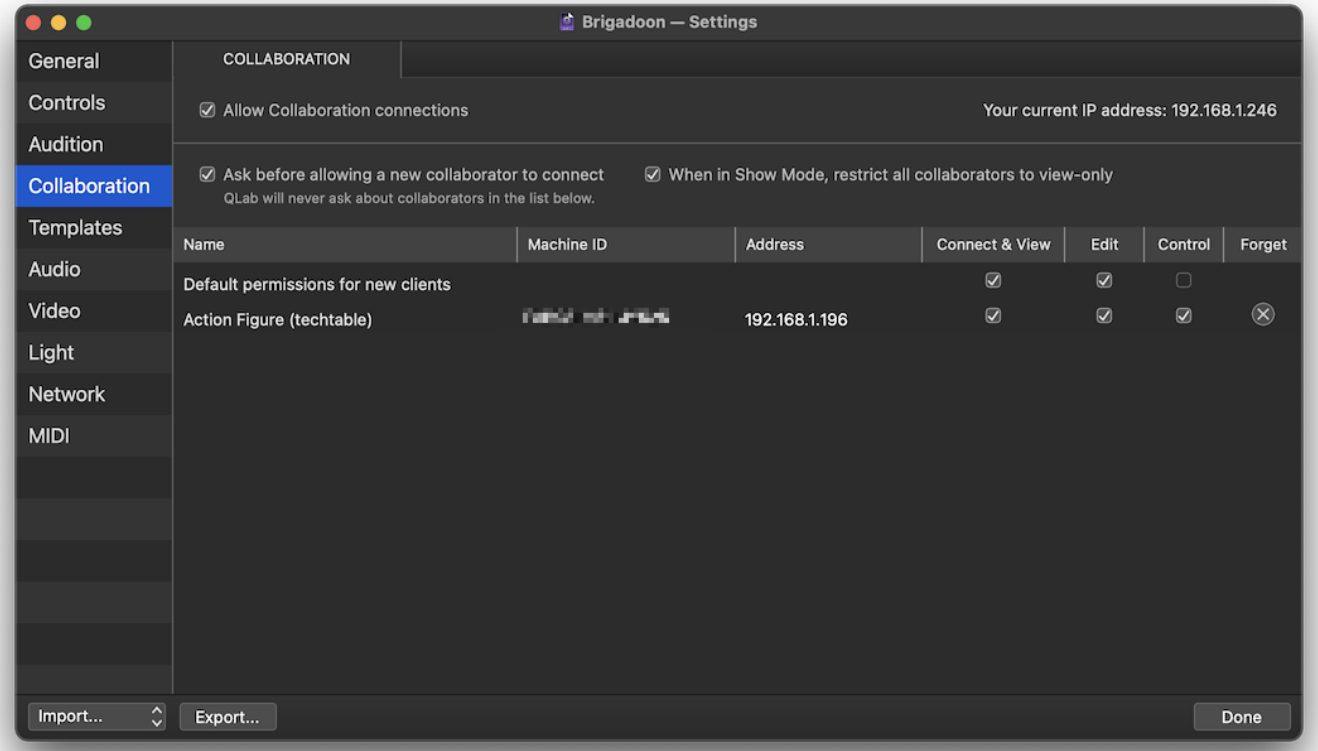
If the primary is set to ask before allowing new collaborators to connect, and this client has never connected to this primary before, attempting to connect will display a permission request on the primary:



Clicking the blue button or pressing the return or enter key will allow the connection and assign the default access permissions set in Workspace Settings → Collaboration. Clicking any of the other "accept" buttons will allow the connection and assign the access permissions indicated by the button. Clicking **Reject connection** will, of course, reject the connection and not add the remote to the list of collaborators.

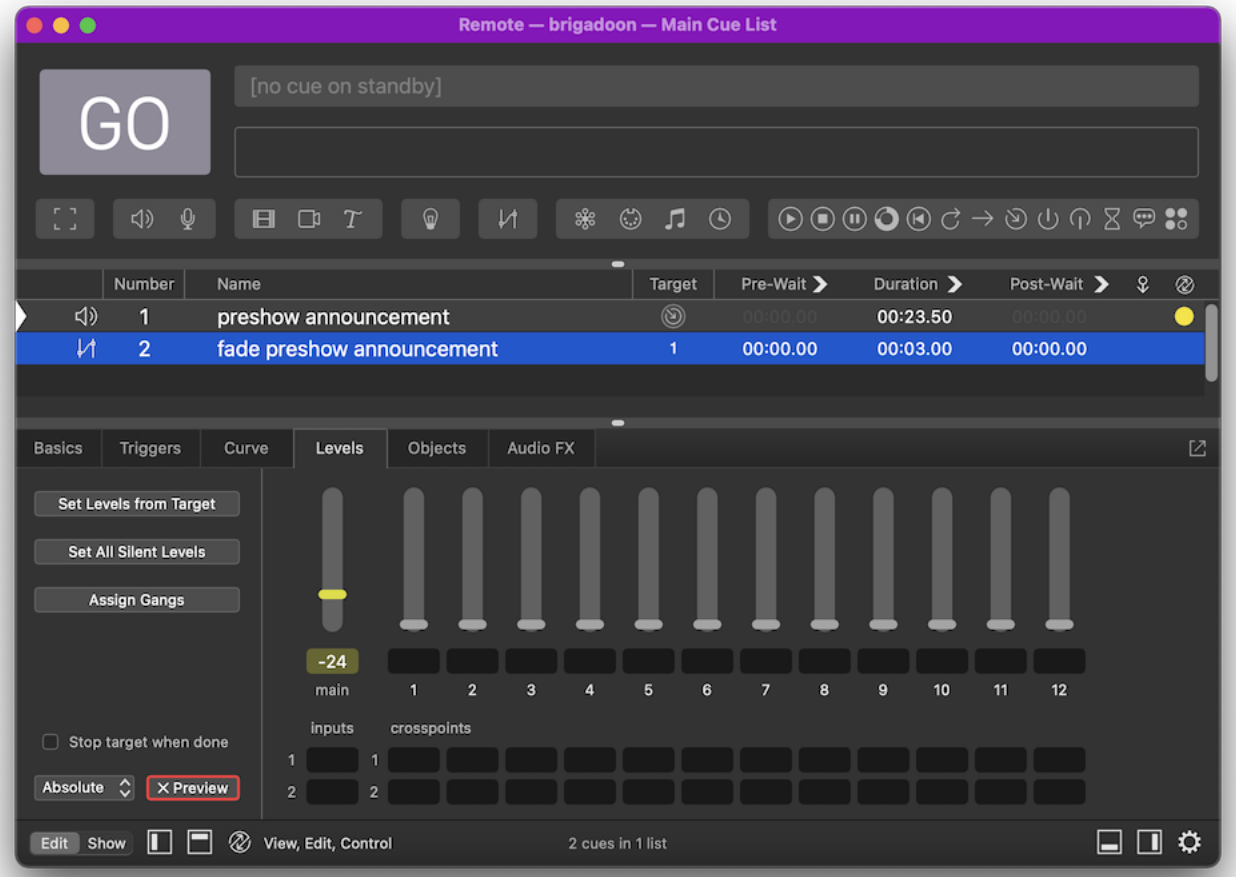
After a remote connects and is accepted, future re-connections will happen without approval.

Remotes can have their access permissions changed and can be removed from the list of approved collaborators in Workspace Settings → Collaboration.



Using A Remote

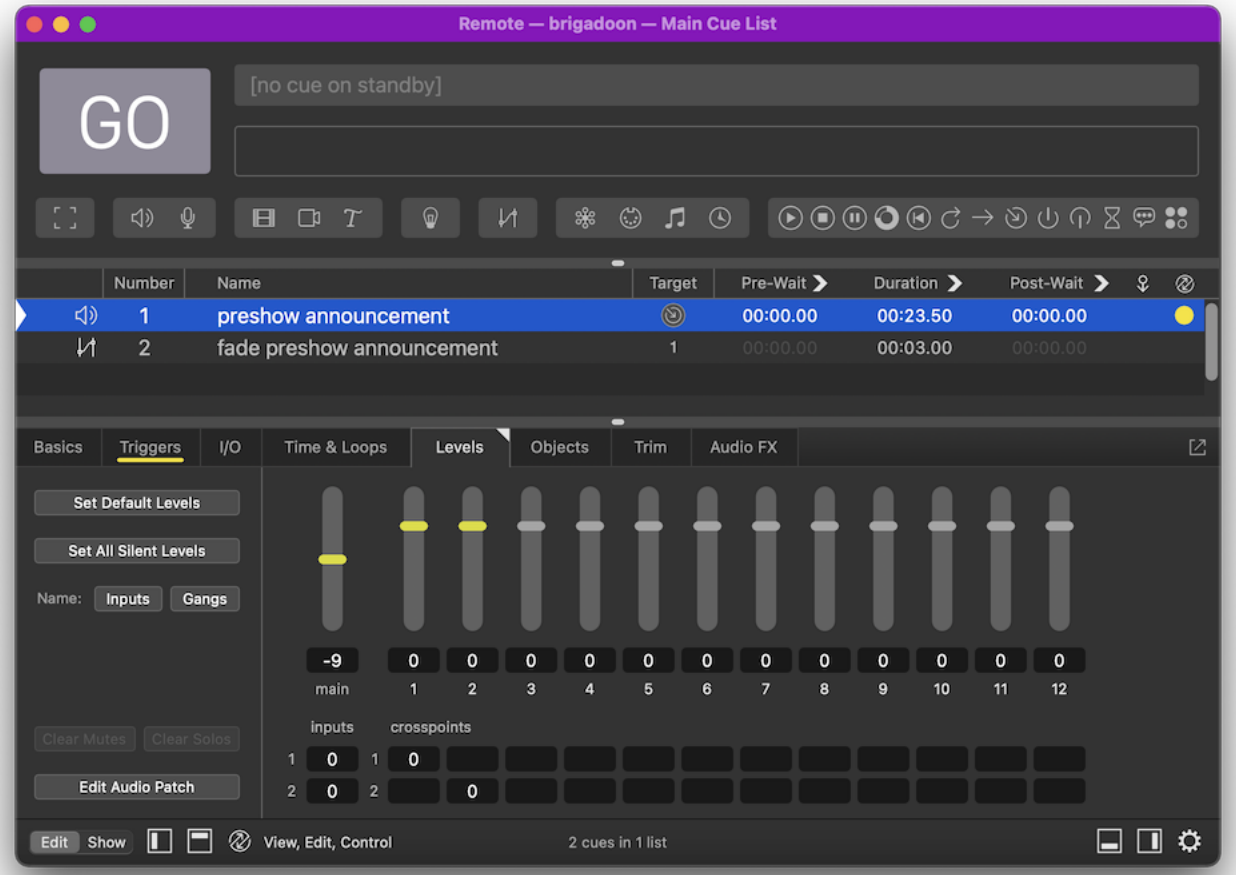
A remote workspace is visually distinct from a normal, local workspace in several ways:



- The title bar is drawn in purple, and the word “Remote” appears before the workspace name.
- The GO button is drawn with a purple border.
- The footer displays the collaboration icon with the current access permissions listed beside it. You can click on the icon to display the IP address and other details of the primary.

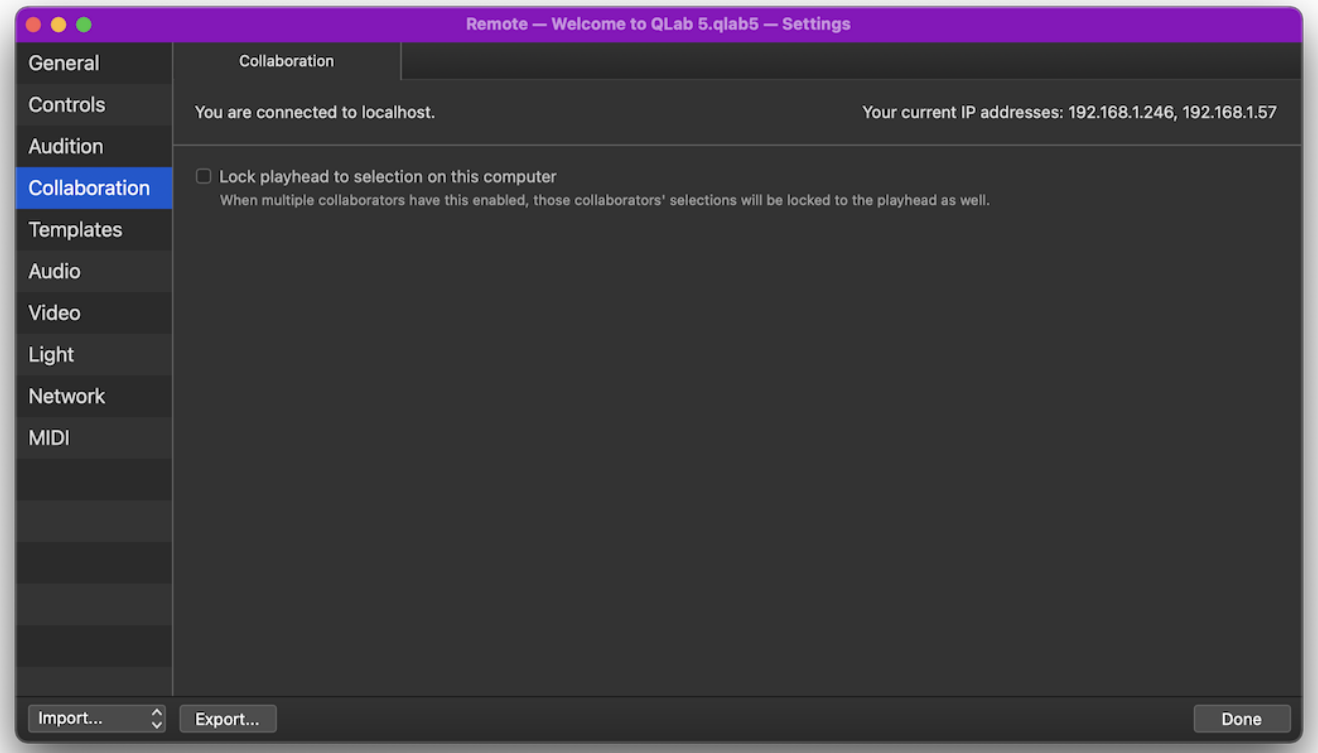
Both remote and primary workspaces (while collaborating) display an additional column on the right side of the cue list view; the collaboration column. A colored dot here indicates a cue that is currently selected by a collaborator on another Mac. If multiple collaborators are connected, one dot appears for each collaborator.

When two or more collaborators have the same cue selected, the inspector shows a colored indicator on the inspector tab that each other collaborator is viewing.



By default, the selection on all remote workspaces is unlocked from the playhead, so each collaborator can independently select separate cues. The playhead position, meanwhile, is necessarily a state shared across all collaborators, because it is a property of each cue list on the primary. Any collaborator that has control permission can move the playhead to a cue by clicking in the status column for that cue. Any collaborator with control permissions can also quickly jump the playhead to their locally selected cue by pressing the keyboard shortcut for load, which is **L** by default.

Additionally, you can optionally set any collaboration remote with control permission to lock the playhead to its selection by visiting Workspace Settings → Collaboration on that remote.



If this box is checked, the playhead will be locked to the selection on this remote. If multiple collaborators have this box checked, they will all be able to move the playhead by moving their selection. This can make some workflows much simpler, but can complicate others.

Working With Cues

All edits made while collaborating follow a last-edit-wins policy. Because of network latency, however, it can sometimes be surprising which edit was in fact last. For this reason, collaborators are encouraged to devise their own person to person guidelines about how to work together smoothly without confusing each other.

Undo and Redo

Each collaborator maintains their own, unique undo history for edits made in cue lists and carts. This allows you to focus on your own work, and not worry about keeping track of what your collaborators are doing.

The Light Dashboard has a single undo history shared across all collaborators.

Each settings area (General, Controls, Audition, etc...) has its own single undo history shared across all collaborators.

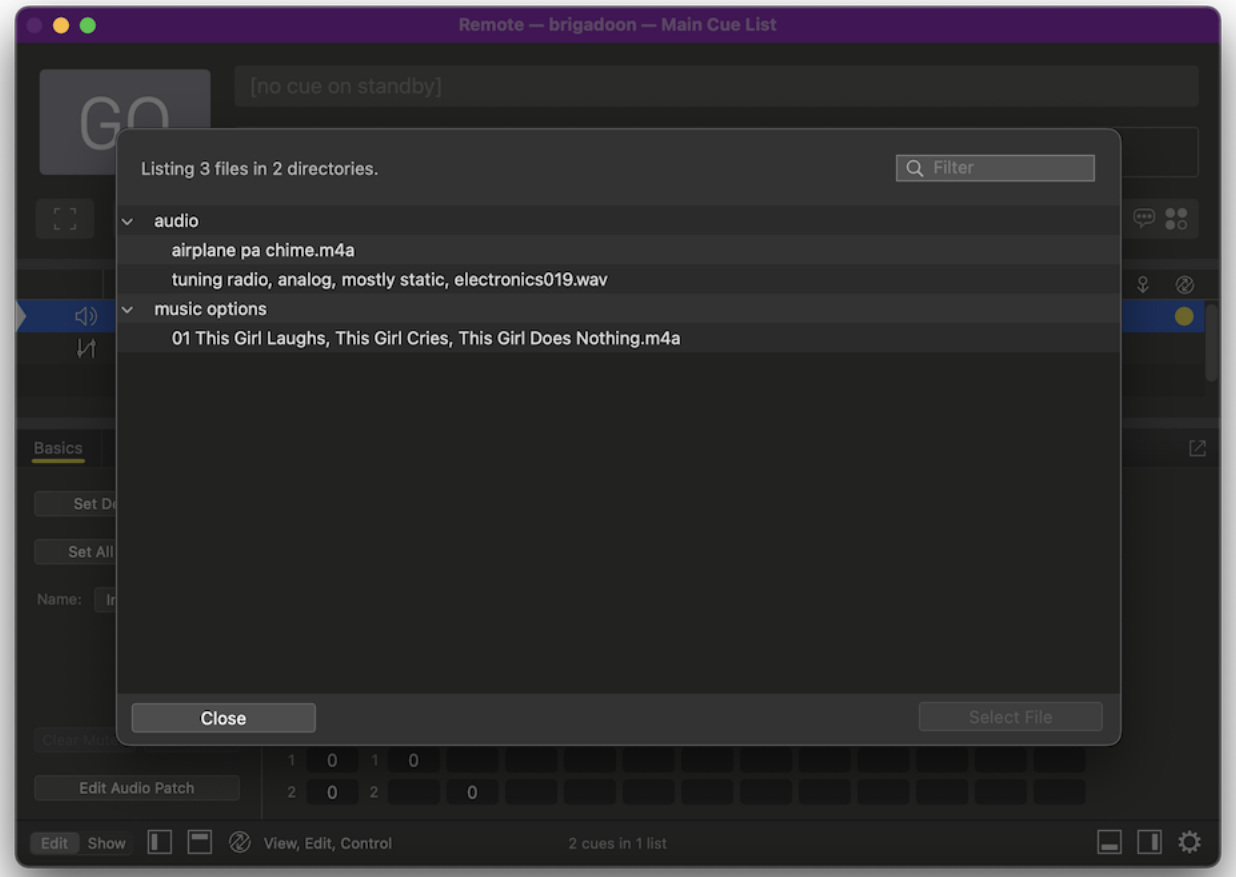
Creating, Deleting, and Moving Cues

While collaborating, creating a cue, deleting a cue, and moving a cue cannot be undone. This blocking of undo allows QLab to avoid having more complex rules which you need to follow while collaborating, and prevents the workspace from entering an un-synchronizable state. Because this is different from how QLab normally behaves, QLab will display an explanatory message when you add, delete, or move a cue while collaborating.

Assigning File Targets

When assigning file targets on a remote, QLab displays a collaboration-specific file browser which shows files available in the folder within which the workspace is saved on the primary. This means that the primary must be saved at least once on the primary before a remote can assign file targets to

cues. It also means that remotes can only “see” files in the same folder as the primary workspace, or in folders within that folder.



QLab only shows folders which contain files that are valid targets for the selected type of cue. In the screen shot above, an Audio cue is selected. The “video” folder on the primary is not visible because it does not contain any files which are valid targets for an Audio cue.

Running Cues

Cues always run on the primary. When a remote with control permission tells a cue to GO or preview, the same thing happens as though the primary told that cue to GO or preview.

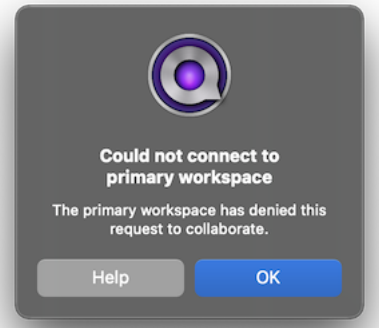
Editing Audio Effects

As of QLab 5.4, you can edit audio effects via collaboration. To do this, the audio effect needs to be installed on both the primary and the remote computer. If a workspace uses an audio effect that a remote does not have installed, that remote will be able to see that the effect is in use but will not be able to adjust its parameters.

Collaboration Error Messages

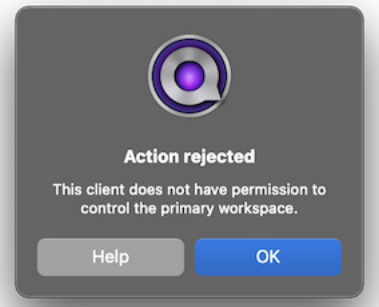
The following error messages may appear on a remote while collaborating.

Access Denied



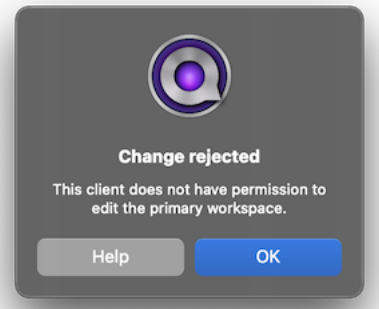
If a new remote attempts to connect to the primary, and the person at the primary clicks **Reject connection**, the remote will see this message. If the rejection was made in error, simply try to connect again. If the rejection was deliberate, please understand that *many* people read for this part and we thank you for your time.

Action Rejected



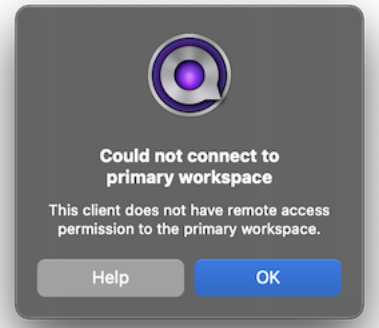
If a remote does not have control permission, this error message will appear if the remote attempts to move the playhead, or start, stop, pause, or resume a cue.

Change Rejected



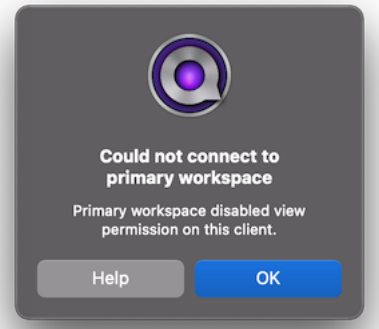
If a remote does not have edit permission, this error message will appear if the remote attempts to make any edit other than 🚩 flagging or unflagging a cue, or editing the notes of a cue.

Connection Rejected



If a remote attempts to connect to a primary which has been configured to refuse access, this error message appears.

View Permission Disabled



If a remote attempts to connect to a primary that has that remote in its collaborators list, but the *Connect & view* checkbox is unchecked, this error message appears.

QLab Remote

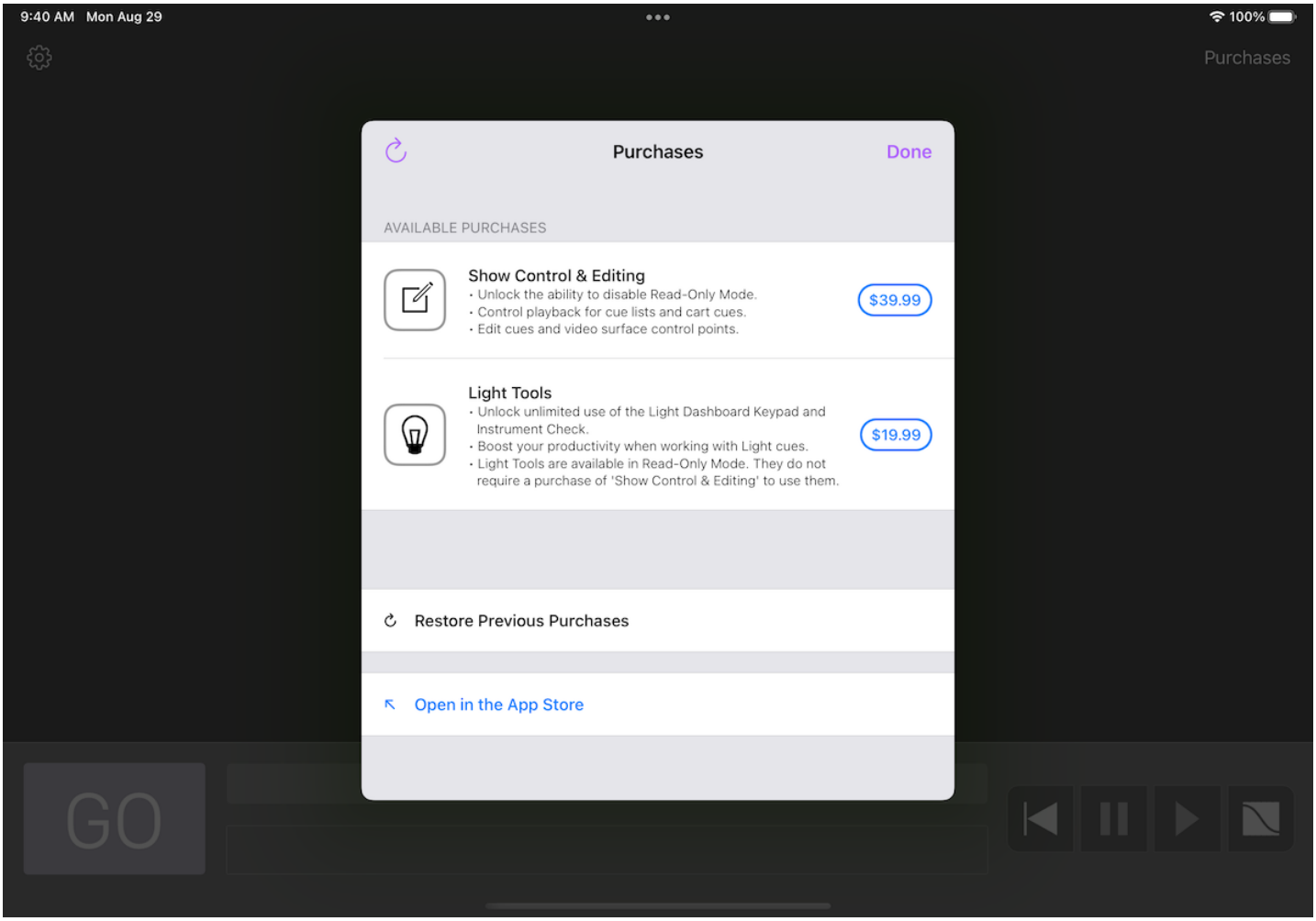
QLab Remote is an iOS app, available on the App Store, that can connect to QLab to remotely view, edit, and run cues. QLab Remote requires iOS 15 or later and is compatible with QLab 3.0 or later[1]. QLab Remote can run on any iPhone, iPad, or iPod touch that can run iOS 15.

QLab Remote adapts to different screen sizes and orientations as best as it can, and it supports Split View and multiple windows on iPad. Some features will only be available when screen space permits, which means some features are only available on iPad.

Features

QLab Remote is a free app with two optional available in-app purchases.

Tapping the *Purchases...* button in the upper right corner of QLab Remote's initial screen lets you view and make in-app purchases.



If you purchased an older version of QLab Remote, your Apple ID should automatically enable the **Show Control & Editing** purchase. If it does not, you can tap *Restore Previous Purchases* to correct the situation.

Read-only Mode (free)

When QLab Remote is in Read-Only mode, you cannot use it to start or stop cues, move the playhead, or change any attributes of any cues except flags and notes.

Read-Only mode is intended to be used, for example, by a designer during a run-through or preview performance. In this situation, it can be desirable to edit notes and flag or unflag cues, but disallow any other editing.

Read-Only mode is also great for a remote cue list monitor for a stage manager or any other member of the team who needs to see what QLab is up to, but does not need to control QLab.

Show Control & Editing

This in-app purchase unlocks QLab Remote's abilities to send commands to QLab. You can run cues, move the playhead, edit parameters of cues (with some limitations), and adjust video stage geometry.

Light Tools

This in-app purchase unlocks QLab Remote's Light Keypad remote, which gives remote access to the Light Dashboard and lets you quickly and easily enter light commands, change levels, and record and update cues.

It also unlocks the Light Instrument Check tool which lets you quickly run a channel check on your lighting system.

You do *not* need to purchase **Show Control & Editing** to use **Light Tools**.

Backwards Compatibility

We make every effort to keep QLab Remote compatible with all releases of QLab 3, QLab 4, and QLab 5. Some features of QLab Remote can only work when connecting to newer versions of QLab, however. For example, QLab Remote's Light Keypad requires QLab 4.2 or newer. Anything relating to lighting requires at least QLab 4, since QLab 3 does not control lighting at all. Encrypted network communications require QLab 5. This manual is written with the assumption that you are using QLab 5.

Using the most recent version of QLab and the most recent version of QLab Remote guarantees access to the most complete set of features.

You can see a complete chart showing which features require which version of QLab at the bottom of this page.

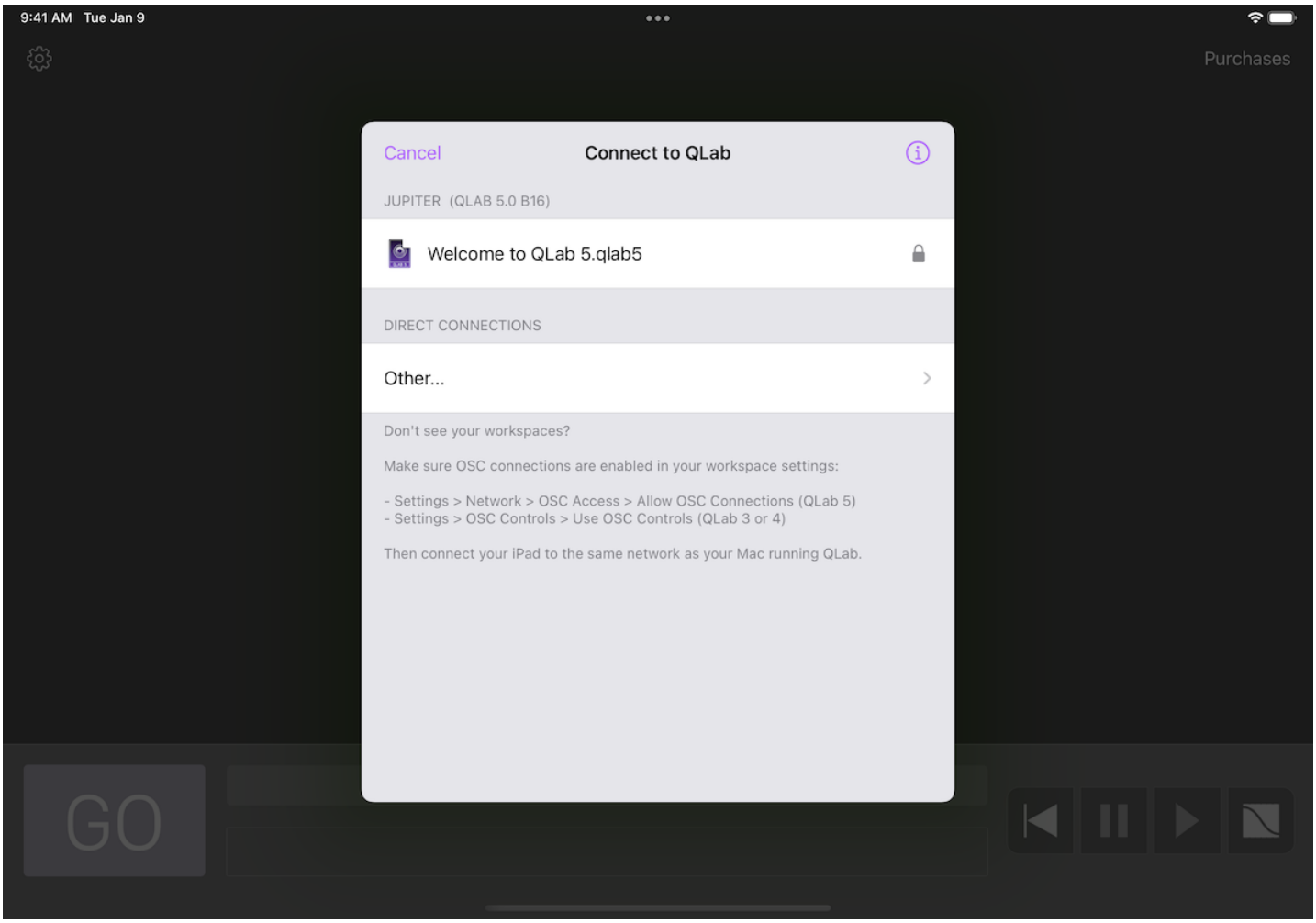
Getting Started

To begin, the iOS device running QLab Remote and the Mac running QLab must be on the same local network, and must be able to "see" each other. If you have a firewall on the network, it must be configured to permit traffic on ports 53000 and 53001. If you are connecting to a workspace that uses a customized port for OSC traffic, that port also needs to be accessible.

Alternately, you can connect your iOS device to the Mac running QLab using a Lightning-to-USB cable.

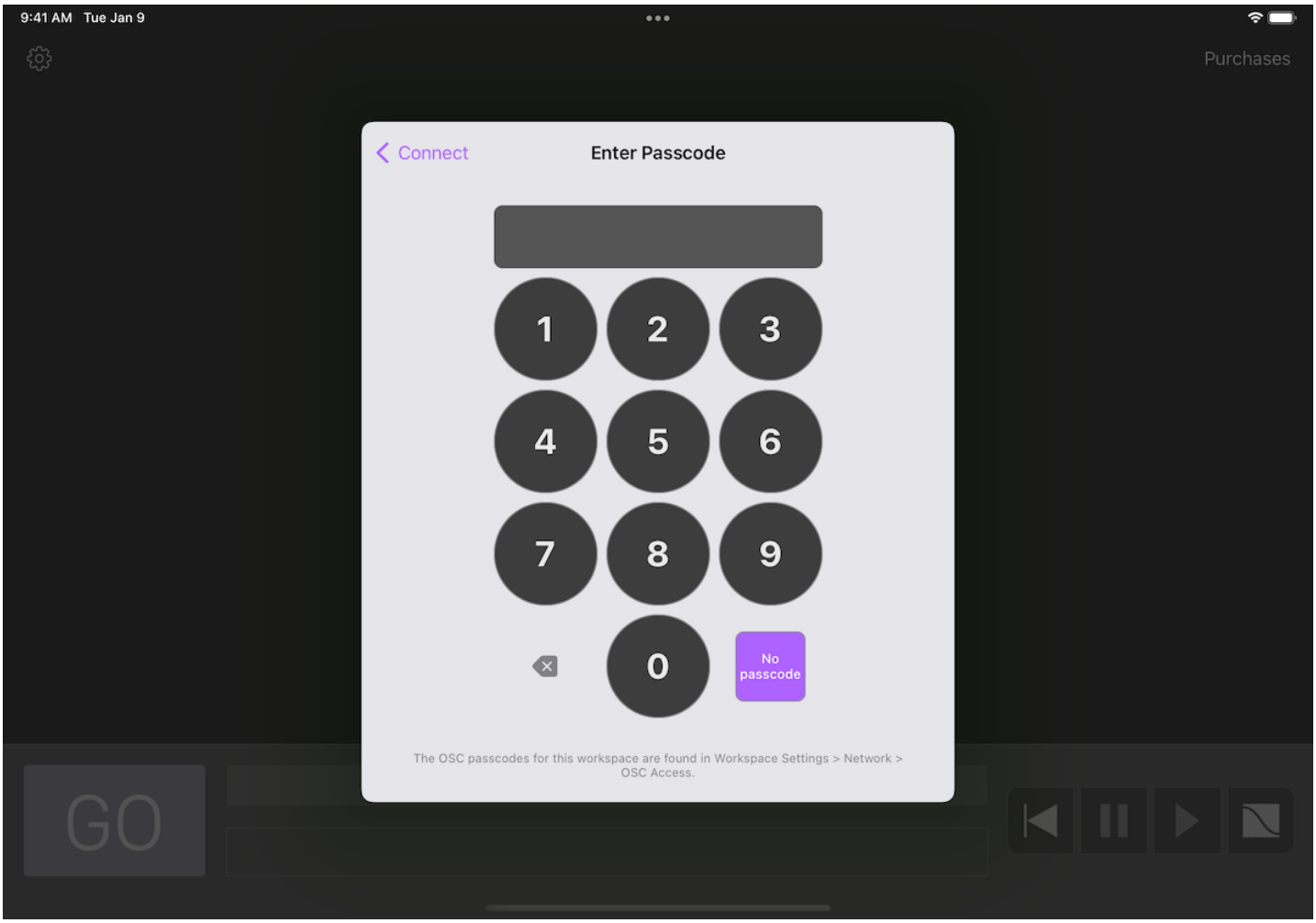
Whether you're connecting via a network or via USB tethering, your QLab workspace must also be set to use OSC controls, which can be set in Workspace Settings → Network → OSC Access. OSC controls are turned on by default in all QLab workspaces.

With your devices connected physically, and your workspace open on your Mac, launching QLab Remote will give you this screen:



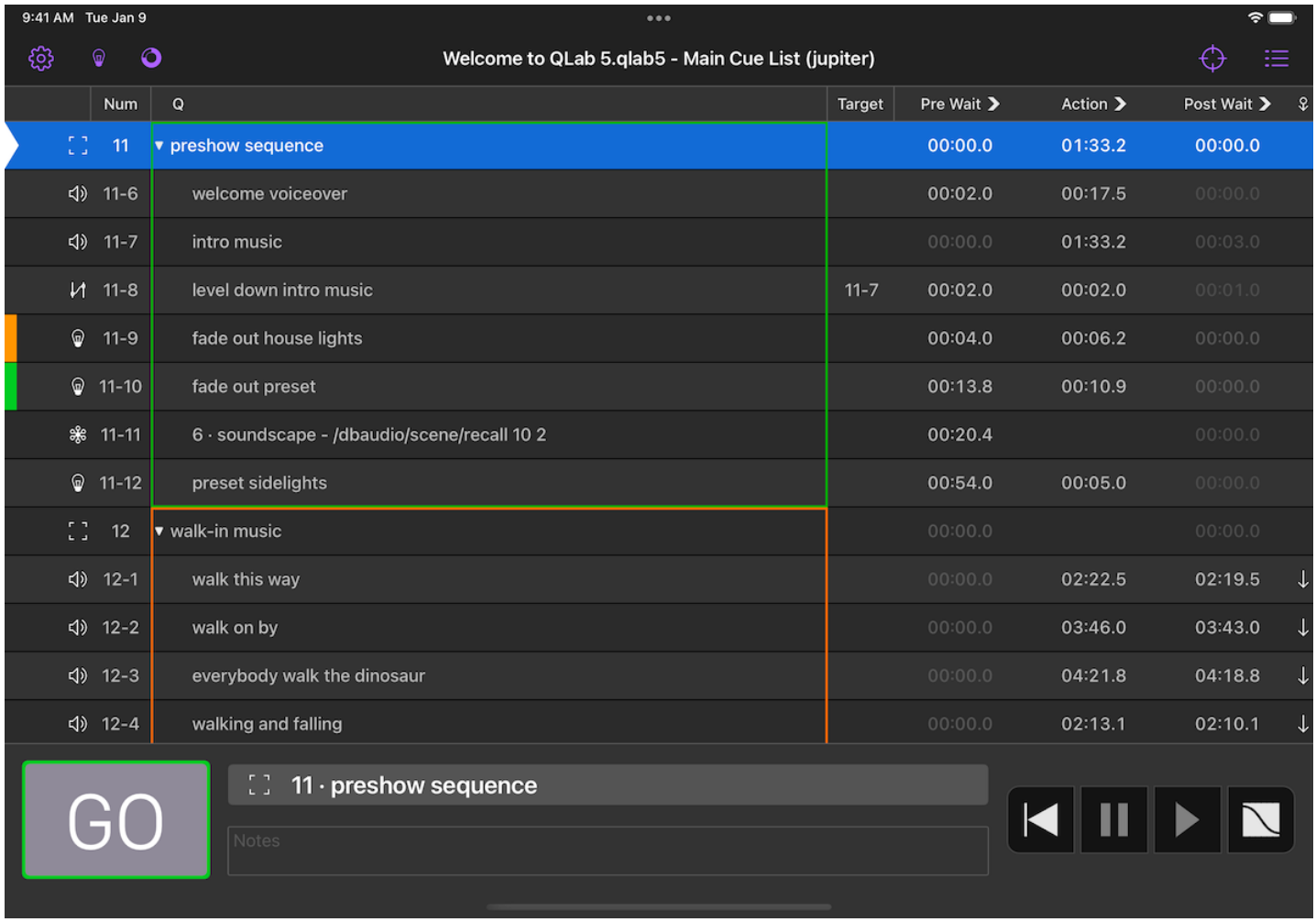
QLab Remote will automatically "see" any workspaces open on your Mac via Bonjour, and you can simply tap to connect. If you have an unusual network setup, such as a network which blocks the Bonjour protocol, you can tap "Other" to directly enter the IP address of the Mac running QLab.

QLab 3 and QLab 4 workspaces which use a passcode will display with a padlock icon on the right, to indicate that you'll be prompted for a passcode before connecting. All QLab 5 workspaces are displayed with a padlock because the QLab 5 access system allows for multiple passcodes, so QLab Remote can't know what the passcode situation is for a given workspace until after it connects. Therefore, whenever you connect to a QLab 5 workspace, you'll be presented with the passcode keypad.



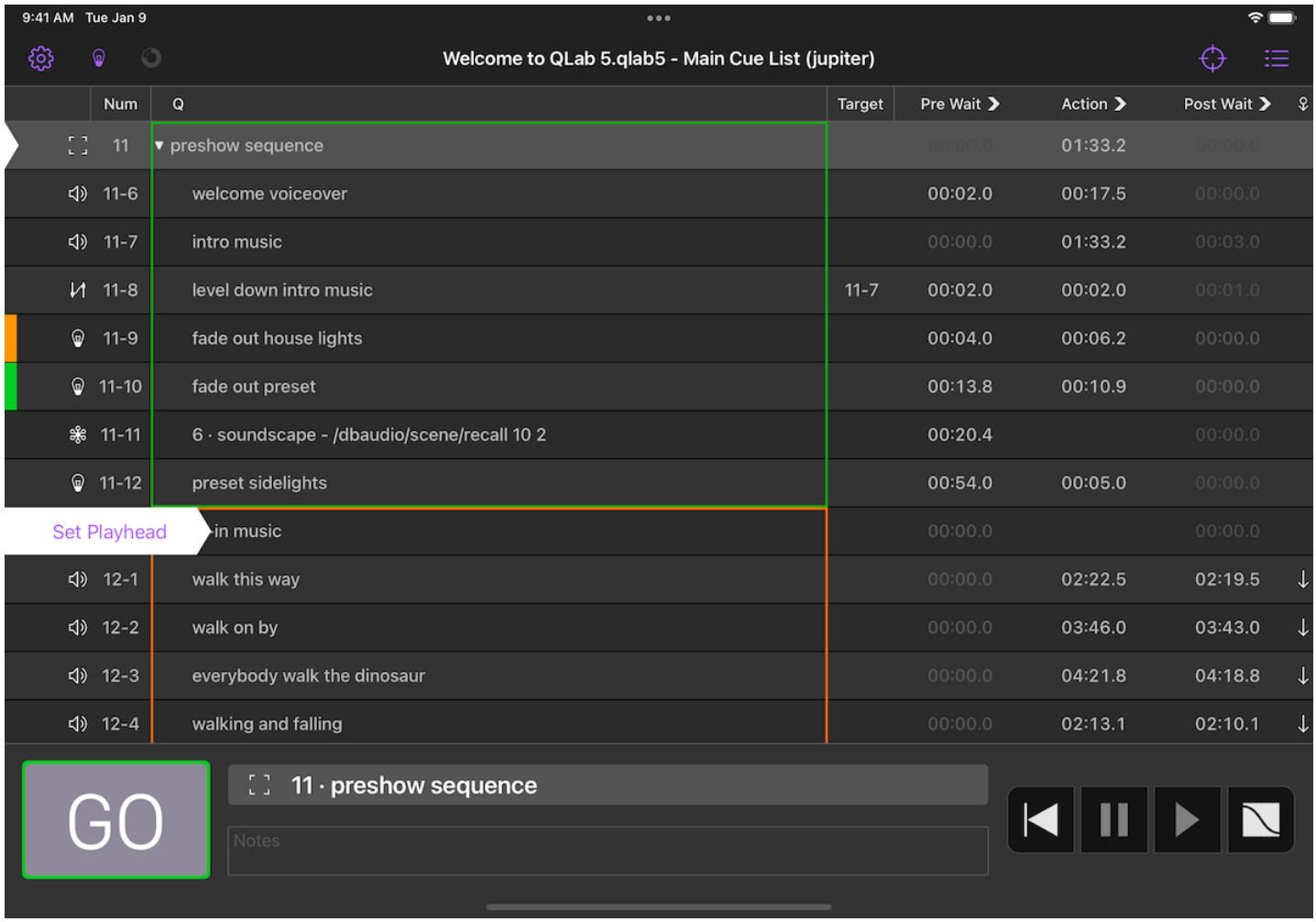
If the workspace allows no-passcode connections, you can simply tap *No passcode* to connect.

A Tour Of QLab Remote

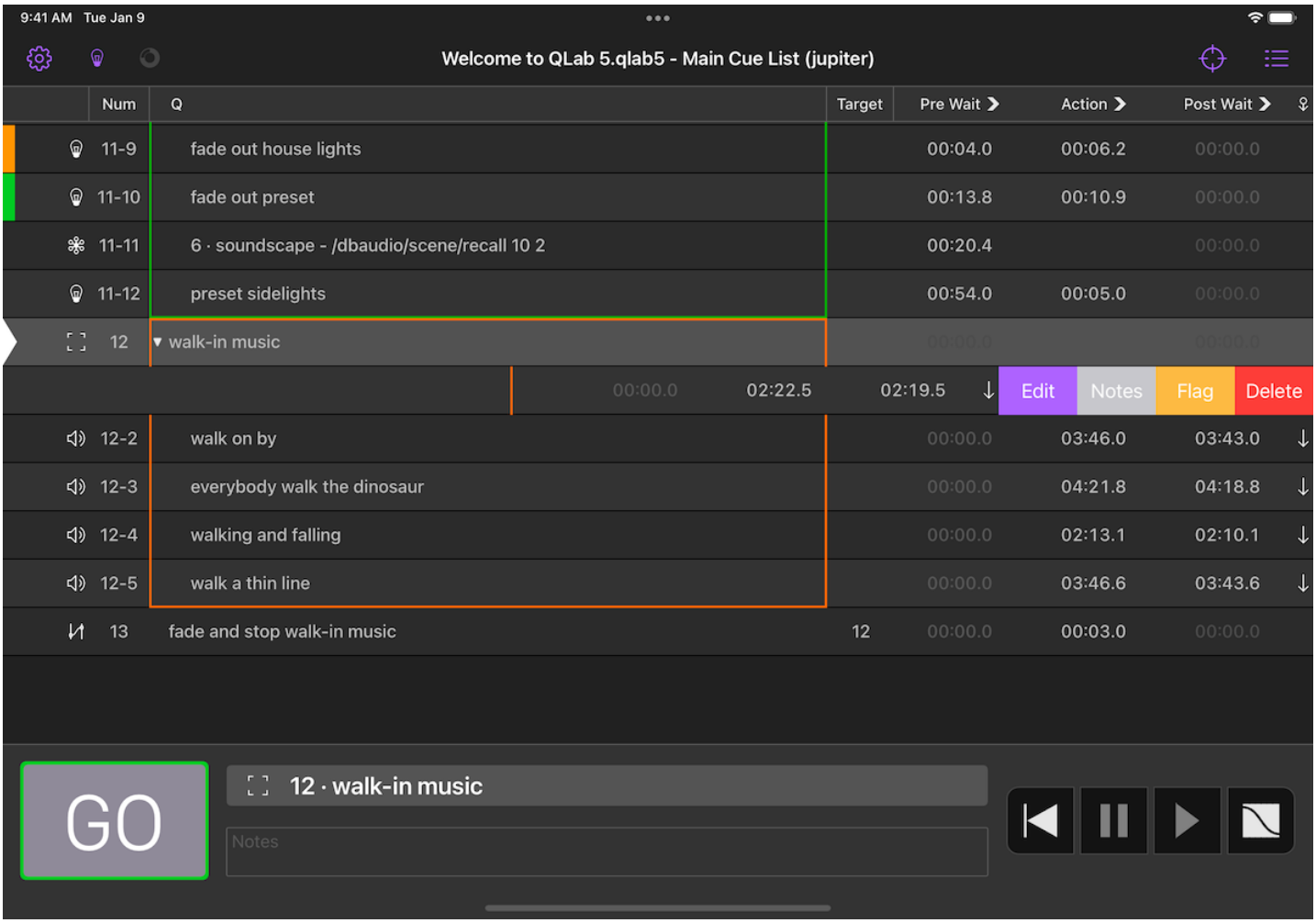


The bottom of the main display shows the familiar GO button, standby indicator, notes field, and transport buttons (reset all, pause all, resume all, and panic all) from QLab.

Above this is the cue list, which works exactly like the cue list display in QLab. To move the playhead, swipe right on a cue:



To edit a cue, swipe left to reveal four edit buttons:

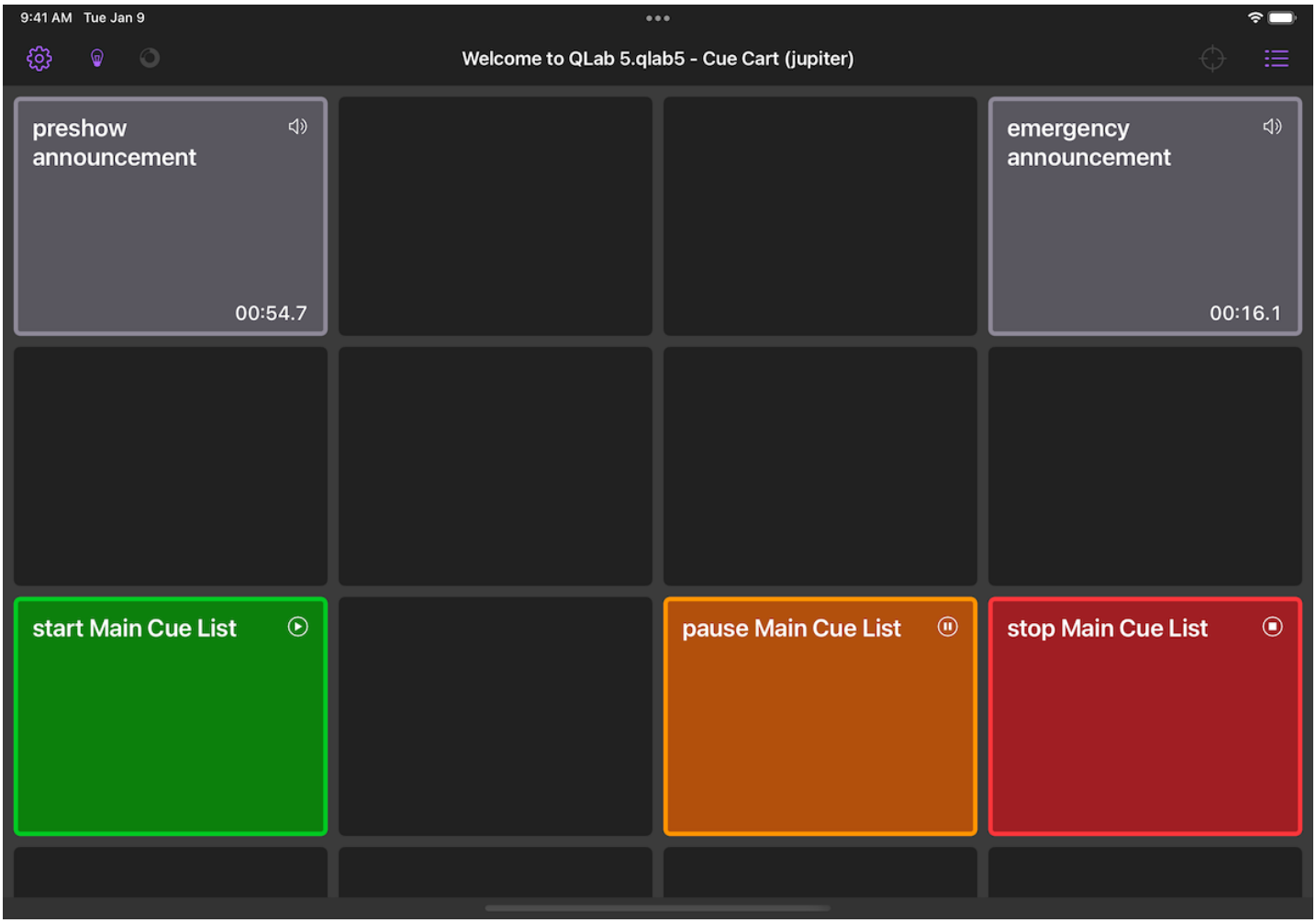


You can also double-tap on a cue to edit that cue.

To move a cue within the list, long-press on that cue and then drag it up or down the list.

Cue Carts

QLab Remote is a great way to view and operate Cue Carts.



While in the cart view:

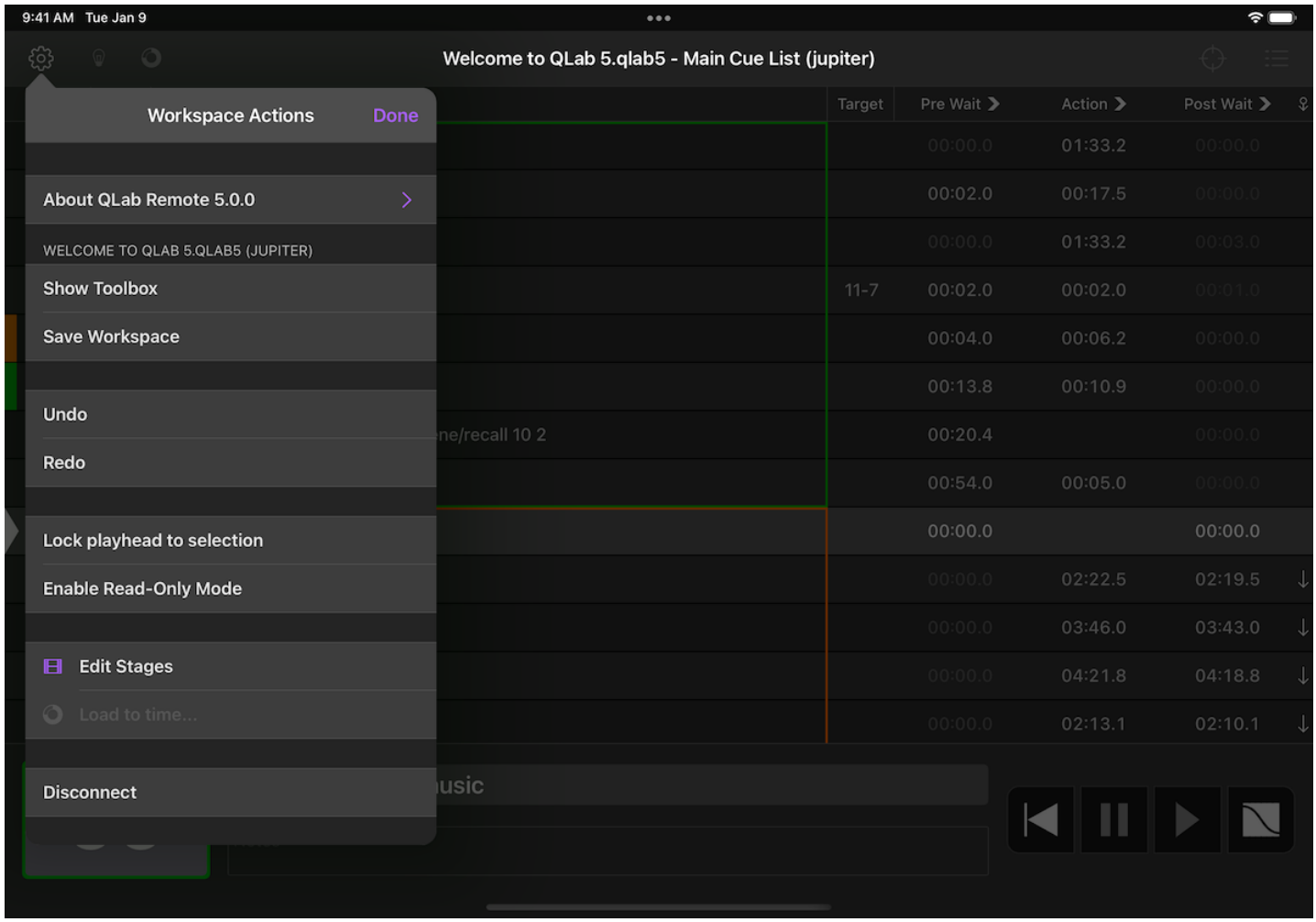
- **Tap** on a cue to start it.
- **Two-finger double-tap** on a cue to edit it.
- **Two-finger long-press-and-hold** on a cue to drag it to a new cell in the grid.
- **Pinch** to zoom in or out on the cart view.

The Toolbar

Above the cue list is a toolbar showing the name of the workspace, the name of the currently displayed cue list, and the name of the Mac that QLab Remote is connected to. Surrounding this label are five tools.

Workspace Actions

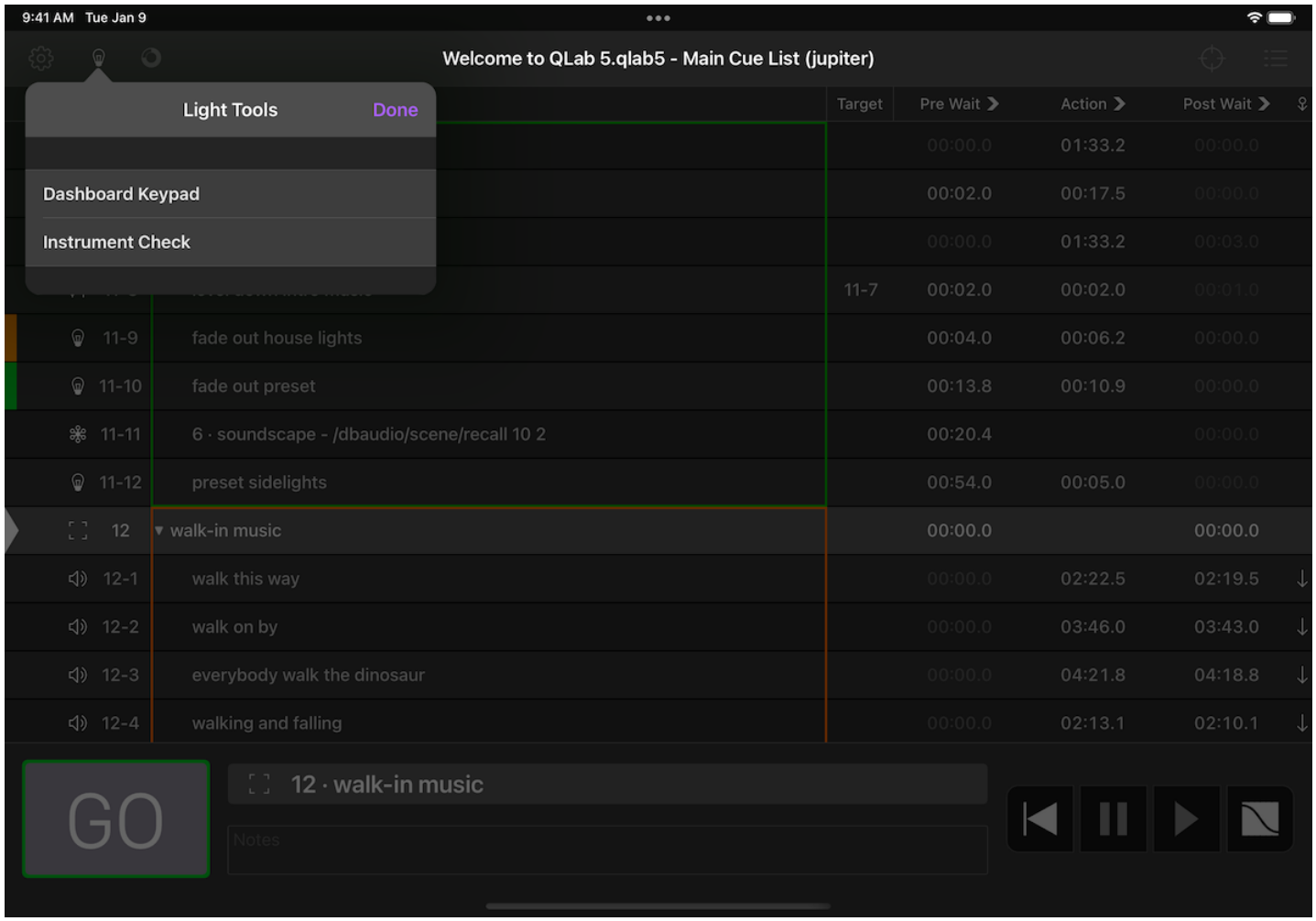
Tapping on the Workspace Actions menu (⚙) reveals ten actions:



- **About QLab Remote.** Tap this item to reveal a submenu with three items: Preferences, View Help, and Contact Support.
  - **Preferences** allows you set the log level and text size of QLab Remote.
  - **View Help** links to this manual in the default web browser.
  - **Contact Support** lets you send a message, optionally including console logs from your device, to support@figure53.com.
- **Show/Hide Toolbox.** Tap to show or hide the toolbox in the cue list view. The toolbox works just like the toolbox in QLab; you use it to add new cues to your cue list.
- **Save Workspace** has the exact same effect as choosing *Save* on the Mac.
- **Undo** has the exact same effect as choosing *Undo* on the Mac.
- **Redo** has the exact same effect as choosing *Redo* on the Mac.
- **Lock playhead to selection** lets you toggle this workspace setting from QLab Remote. You can learn more about the setting from the Workspace Settings → General section of this manual.
- **Enable/disable Read-Only Mode.** Switching off Read-only mode requires the *Show Control & Editing* in-app purchase, as discussed above.
- **Edit Stages.** If your workspaces has any video stages, and you have a Video or Bundle license installed, you can make basic edits to stage geometry using QLab Remote. See below for more details.
- **Load to time...** works just like the Load To Time tool in QLab. You can also access it using its own button in the toolbar.
- **Disconnect.** Tap here to disconnect QLab Remote from the current workspace.

Light Tools

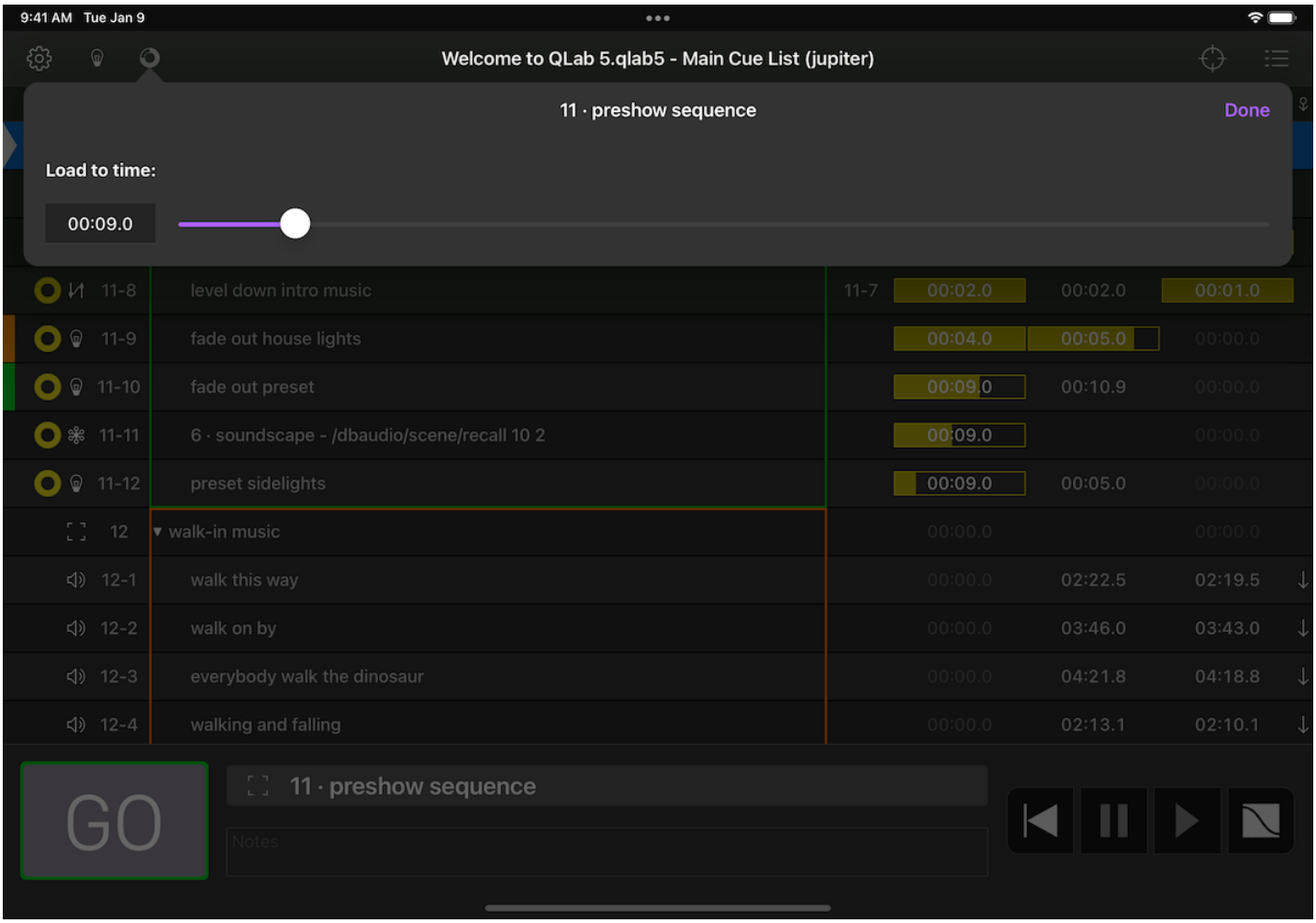
Tapping on the Light Tools menu ( ) reveals two actions:



- **Light Keypad.** The Light Keypad is an optional tool for interacting with the Light Dashboard, adjusting lights, and creating and updating Light cues. You can read more about it below.
- **Instrument Check.** This is a tool to let you quickly check every instrument in your workspace, bringing one instrument or group up at a time so that you can confirm that everything is working correctly. More details can be found below.

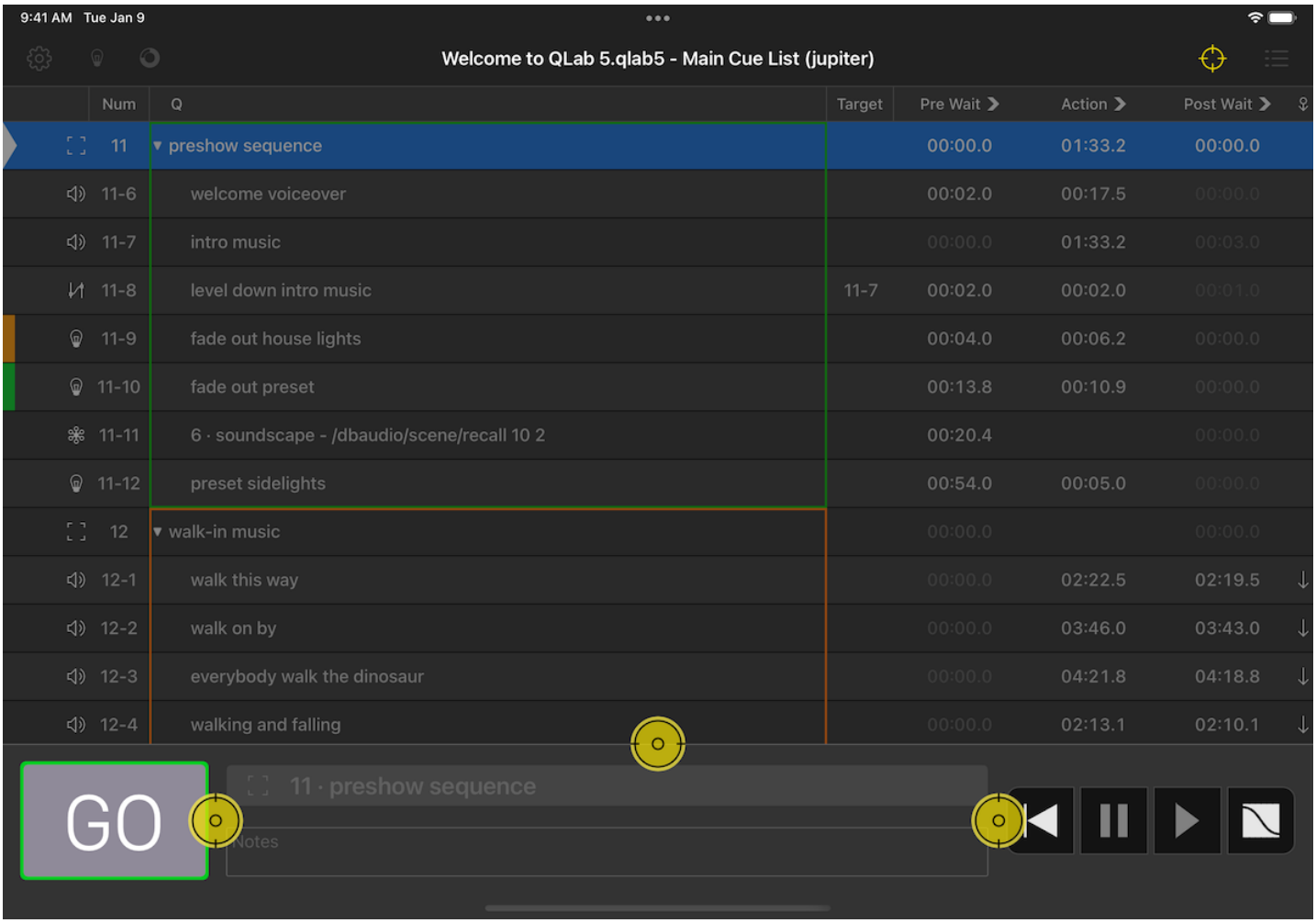
Load To Time

The Load To Time ( ) tool works just like the Load To Time tool in QLab.



Resize Controls

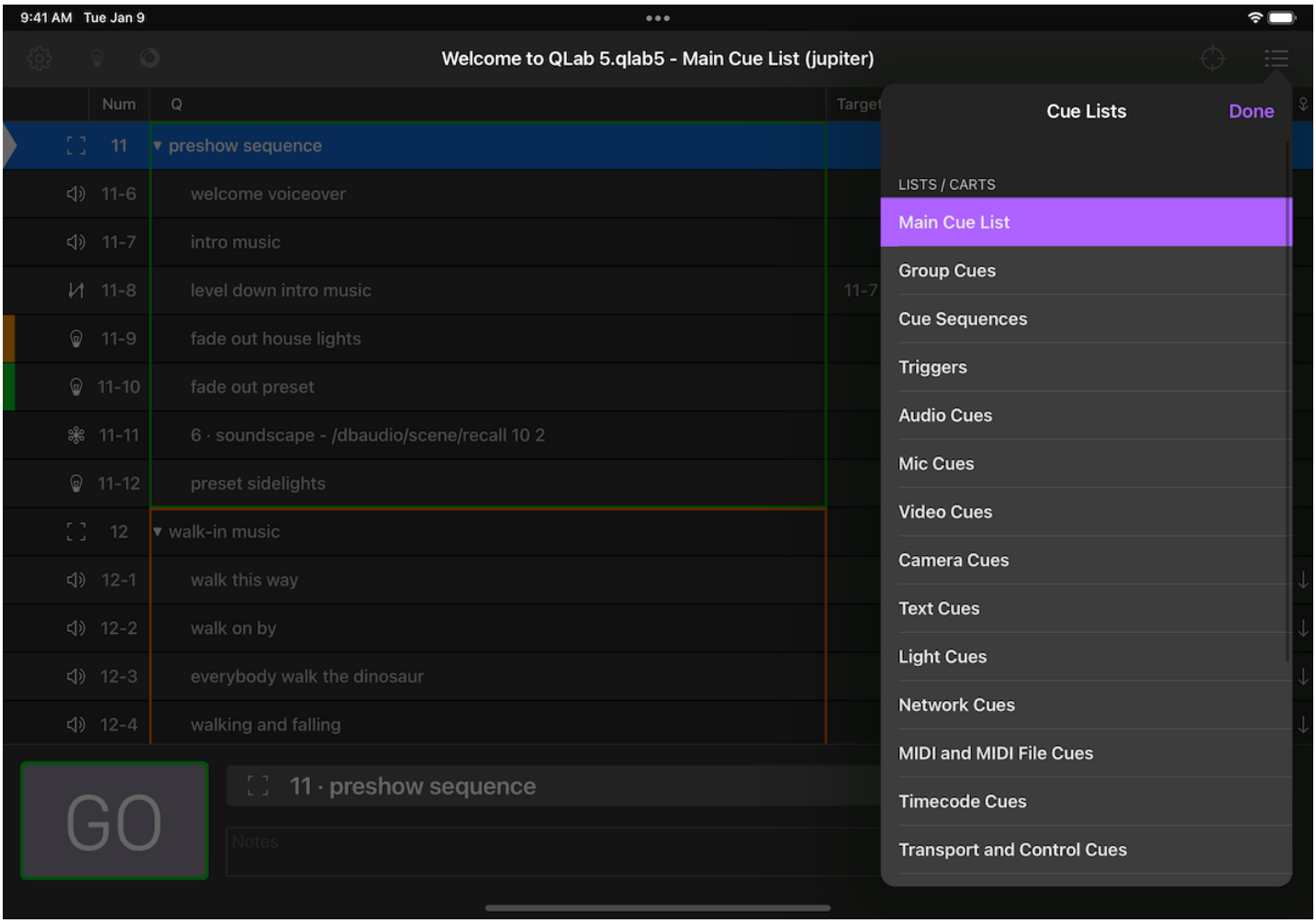
The Resize Controls (⌖) button allows you to resize the footer, the GO button, and the transport controls.



Tap the ⊙ button to reveal the resize controls, then tap and drag them to arrange them as you prefer. You can enlarge, shrink, or entirely hide the GO button and transport controls, hide the entire footer, and so forth. Tap the button again to finish.

Cue Lists

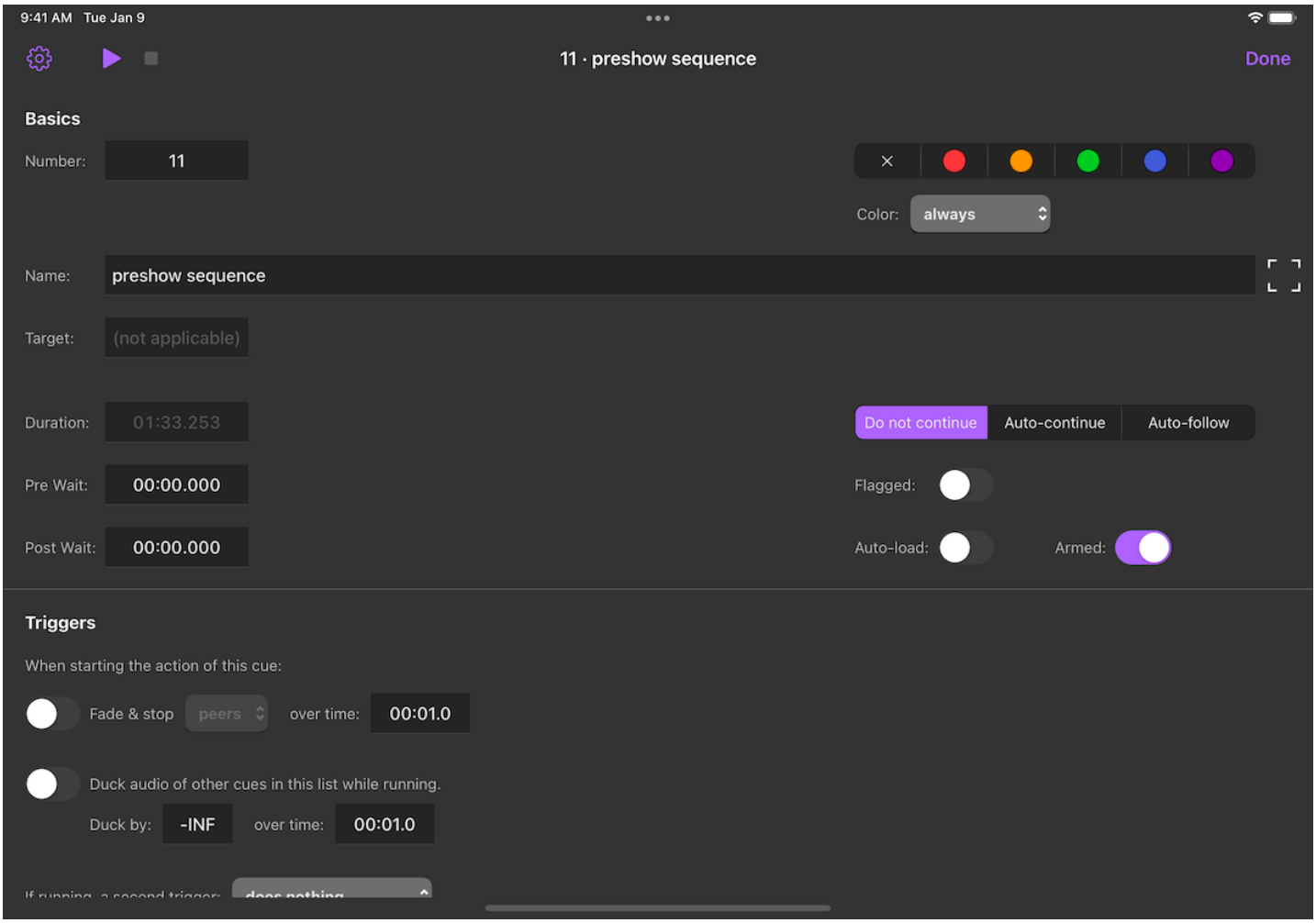
The Cue Lists (☰) menu allows you to switch between the lists and carts in your workspace.



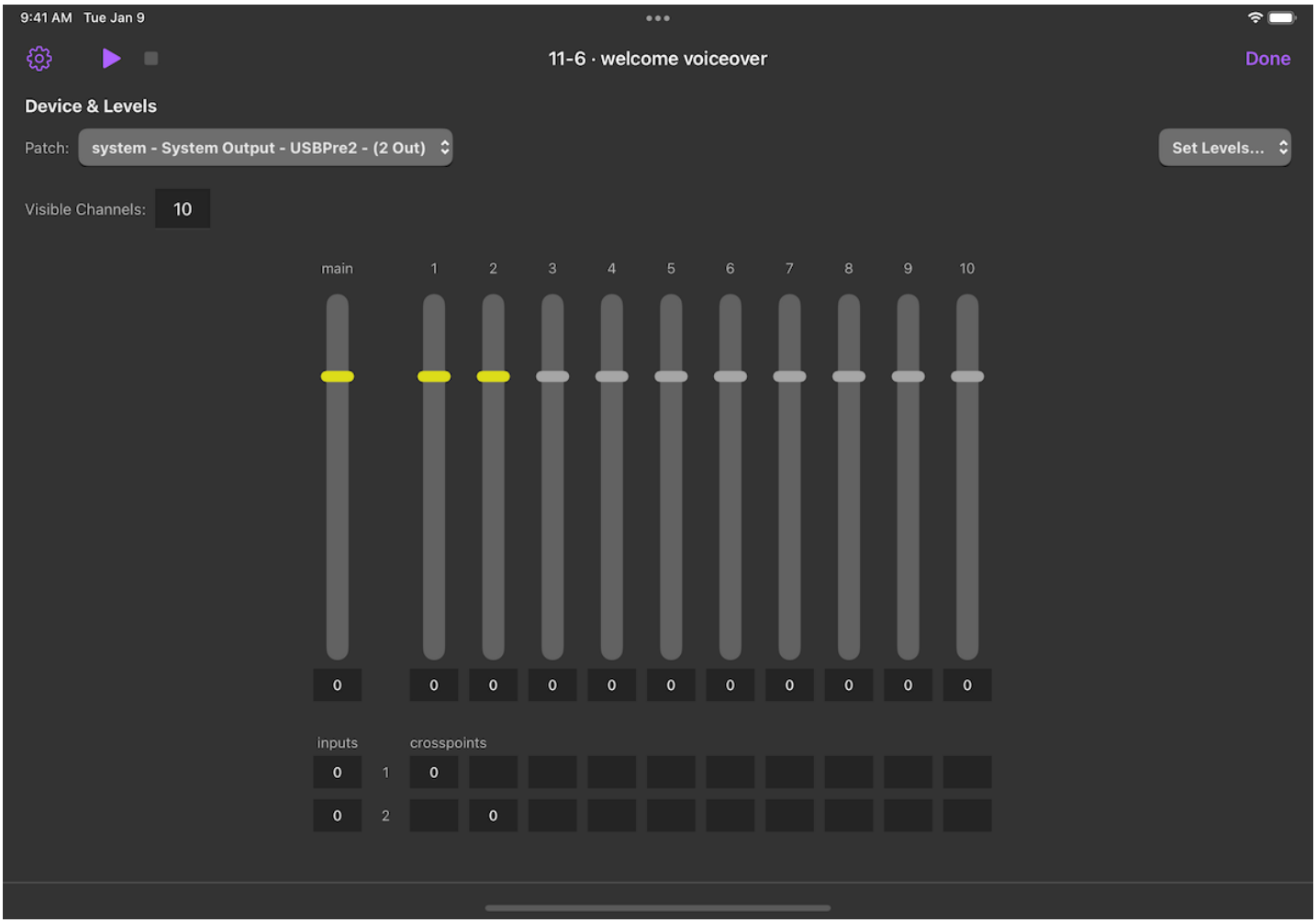
The last item in the Cue Lists menu is *Active Cues*, which is similar to the Active Cues display in QLab's sidebar.

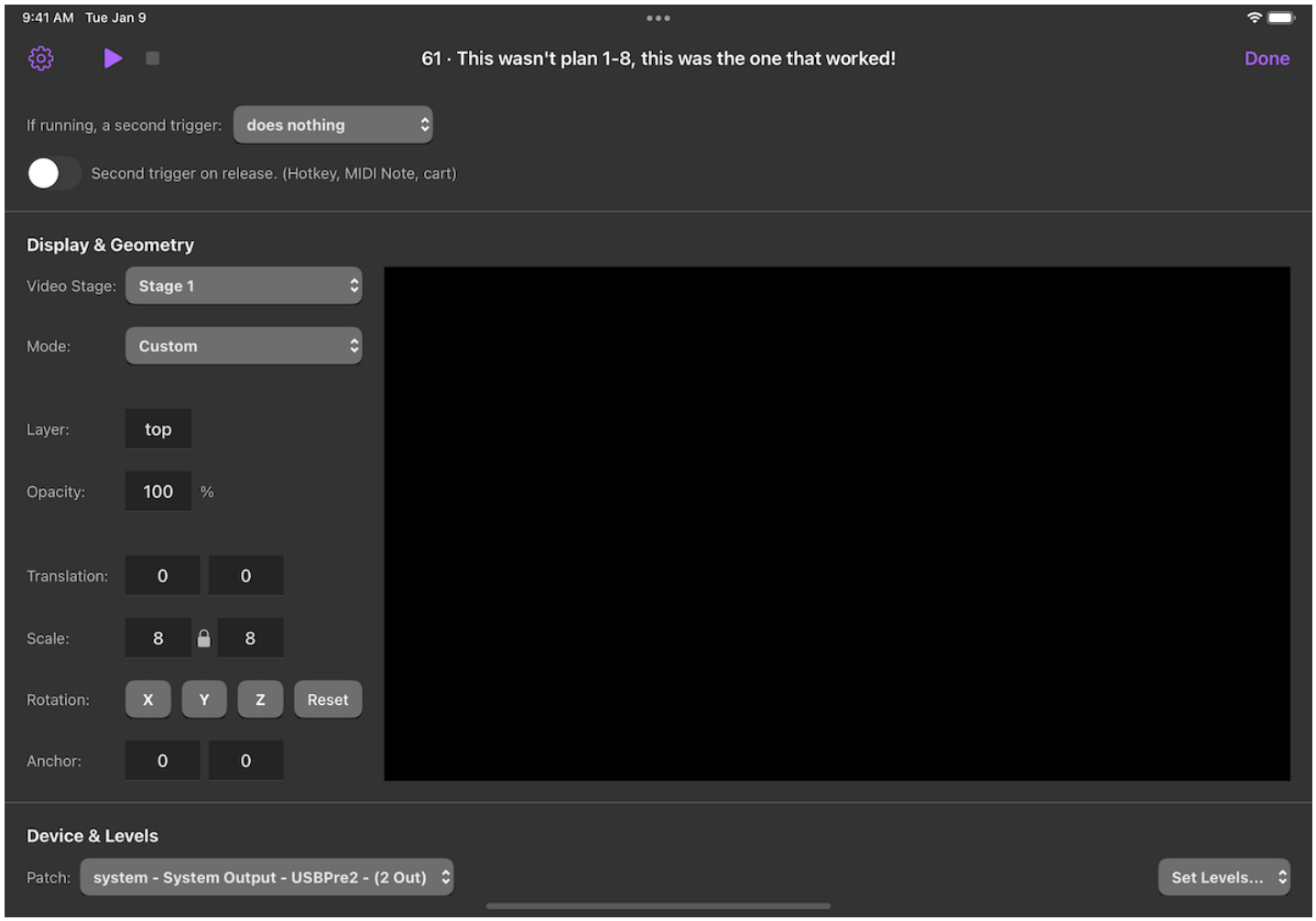
Editing Cues

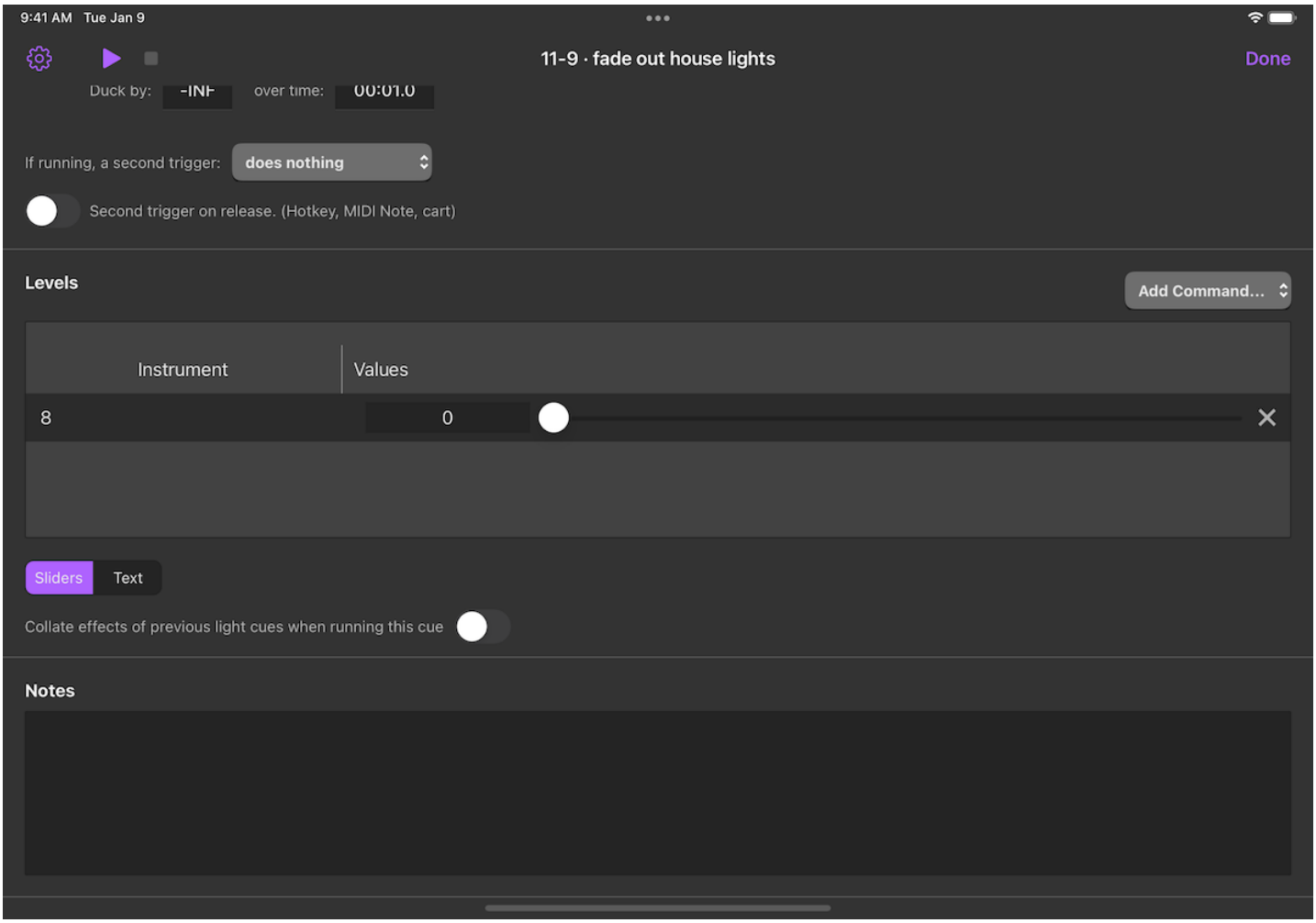
QLab Remote can edit some attributes of all cues:



You can also make some adjustments to Audio cues, Video cues, and Light cues:

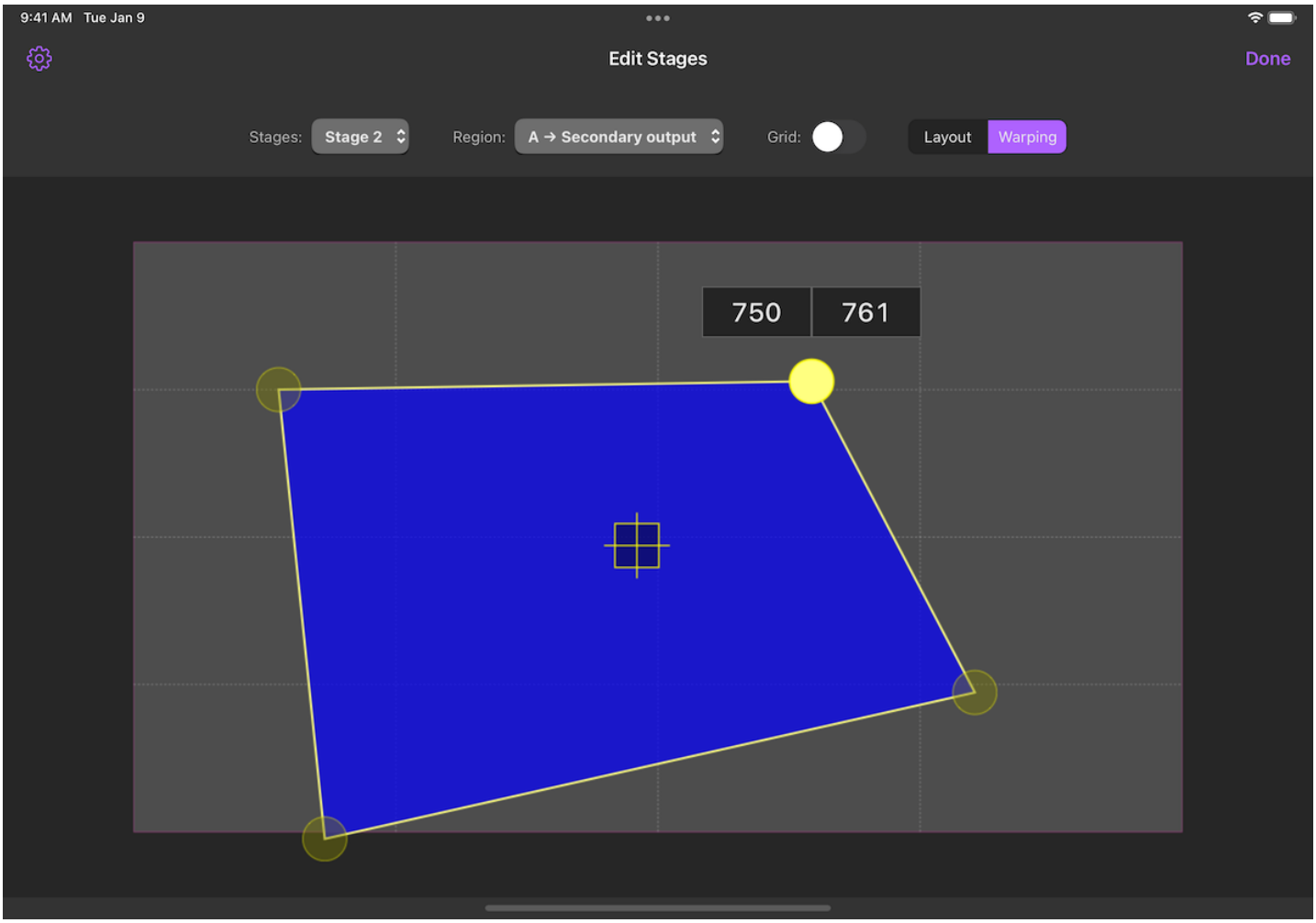

9:41 AM Tue Jan 9
11-6 · welcome voiceover
Done
Device & Levels
Patch:
system - System Output - USBPre2 - (2 Out)
Set Levels...
Visible Channels:
10
main
1
2
3
4
5
6
7
8
9
10
0
0
0
0
0
0
0
0
0
0
0
inputs
crosspoints
0
1
0
0
2
0


9:41 AM Tue Jan 9
61 · This wasn't plan 1-8, this was the one that worked!
Done
If running, a second trigger:
does nothing
Second trigger on release. (Hotkey, MIDI Note, cart)
Display & Geometry
Video Stage:
Stage 1
Mode:
Custom
Layer:
top
Opacity:
100
%
Translation:
0
0
Scale:
8
8
Rotation:
X
Y
Z
Reset
Anchor:
0
0
Device & Levels
Patch:
system - System Output - USBPre2 - (2 Out)
Set Levels...



Editing Video Stages

If you have a video license installed in QLab, you can use QLab Remote to edit the geometry of stages in your workspace.



You will need to set up the stage on the Mac, but once it's basically set up you can use QLab Remote to make precise adjustments.

Light Keypad

The Light Keypad is a tool for interacting with the Light Dashboard in QLab. It's designed to let you adjust lights quickly and easily, and then save those adjustments by creating new cues, or recording or updating existing cues.

The Light Command Line

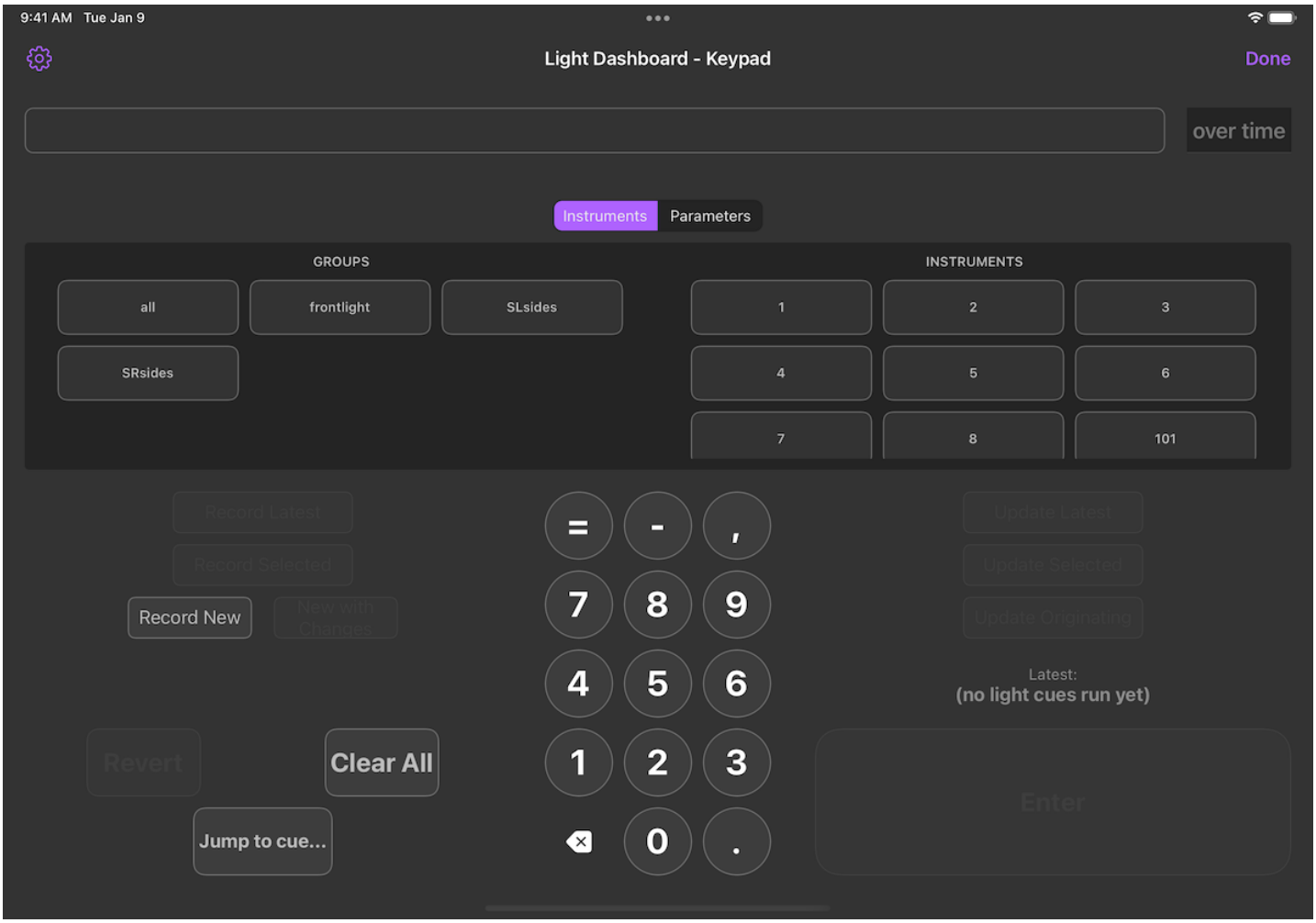
You can tap on the command line and enter text using the regular iOS keyboard (or a physical keyboard connected to your iOS device.) This command line behaves exactly like the [the command line in the Light Dashboard]. Any text you enter here will be sent to the Dashboard when you press enter.

Over time

Typically, any commands you type into the command line will execute immediately when you hit enter. If you type a time into this field, however, the command will fade over that amount of time. This concept, called sneak on some other lighting consoles, allows you to make changes to the live state of your lights in a gentle, subtle way.

Instruments & Parameters

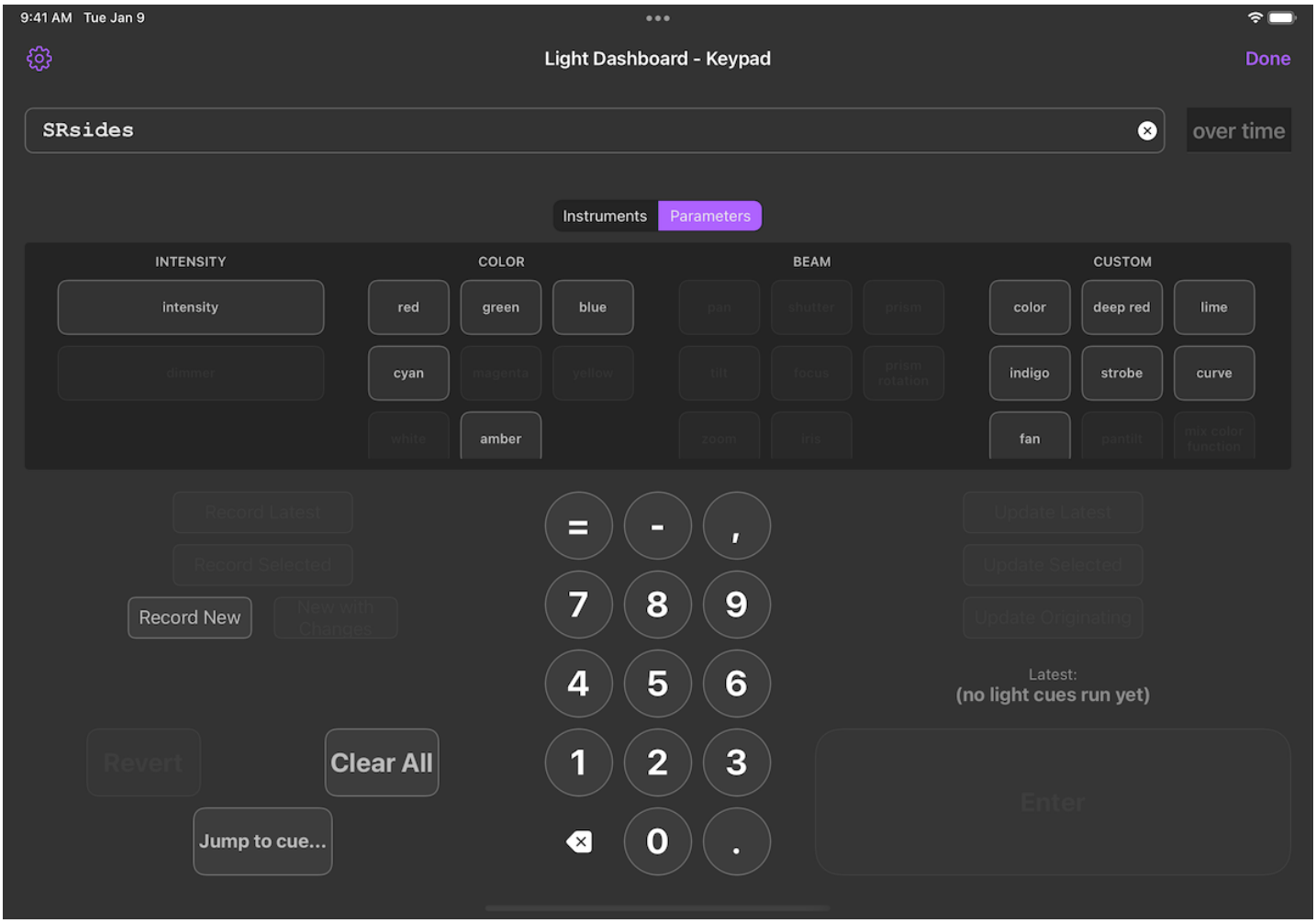
This tabbed area allows you to swiftly select an instrument or light group, then select the parameter you wish to edit. When the Light Keypad first opens, you'll be presented with the Instruments tab:



You can tap on an instrument or light group to select it, and place its name in the command line.

If your instruments or light groups are numbered, you can also use the number keys from the beginning to enter the number of your instrument or light group. Once you enter a valid instrument or light group number, QLab Remote will display the Parameters tab. If you then enter a dash or comma, QLab Remote realizes that you're trying to enter a range and returns to the Instruments tab.

Once you enter a name, QLab Remote will display the Parameters tab:



You can then tap on the parameter that you want to adjust, and then use the number keys to make the adjustment.

Once an equals sign is entered, the "=", "-", and "," buttons on the keypad become "cue", "pass", and "home" buttons which you can tap to add those keywords to the command line.

When your command is complete, the **Enter** button will become enabled, and you can tap on it to transmit the command to QLab. You will see the adjustment reflected in the Dashboard right away. If your QLab system is connected to your lights, you will of course also see your adjustment reflected in real life.

You can also always tap in the command line to use the onscreen keyboard or an attached physical keyboard to type commands from scratch or edit commands before committing them.

A note on the Light Keypad's layout

The Instruments tab always shows light groups on the left, with the group "all" listed first. Individual instruments are shown on the right, alphabetically sorted.

The Parameters tab dynamically adjusts to show the most relevant set of information. If the command line is empty, QLab Remote shows all parameters of all instruments, listed in four columns: intensity, color, beam, and custom. Each column is scrollable as needed, and on narrow layouts you can swipe left and right to move between them.

If the command line contains the name of an instrument or light group, only the parameters belonging to that instrument or light group are enabled.

The layout of all parameters, however, remains constant no matter what instruments are selected, active, or patched. The purpose of this is to help you develop a sense of "muscle memory" to aid you in locating the buttons for each parameter. Some buttons may be visible or not, some may be enabled or not, but each button will always be in the same spot (taking into account scrolling, device rotation, and device screen size, of course.)

Recording and Updating Light Cues

QLab Remote provides the same buttons for recording and updating cues as the Light Dashboard does in QLab.

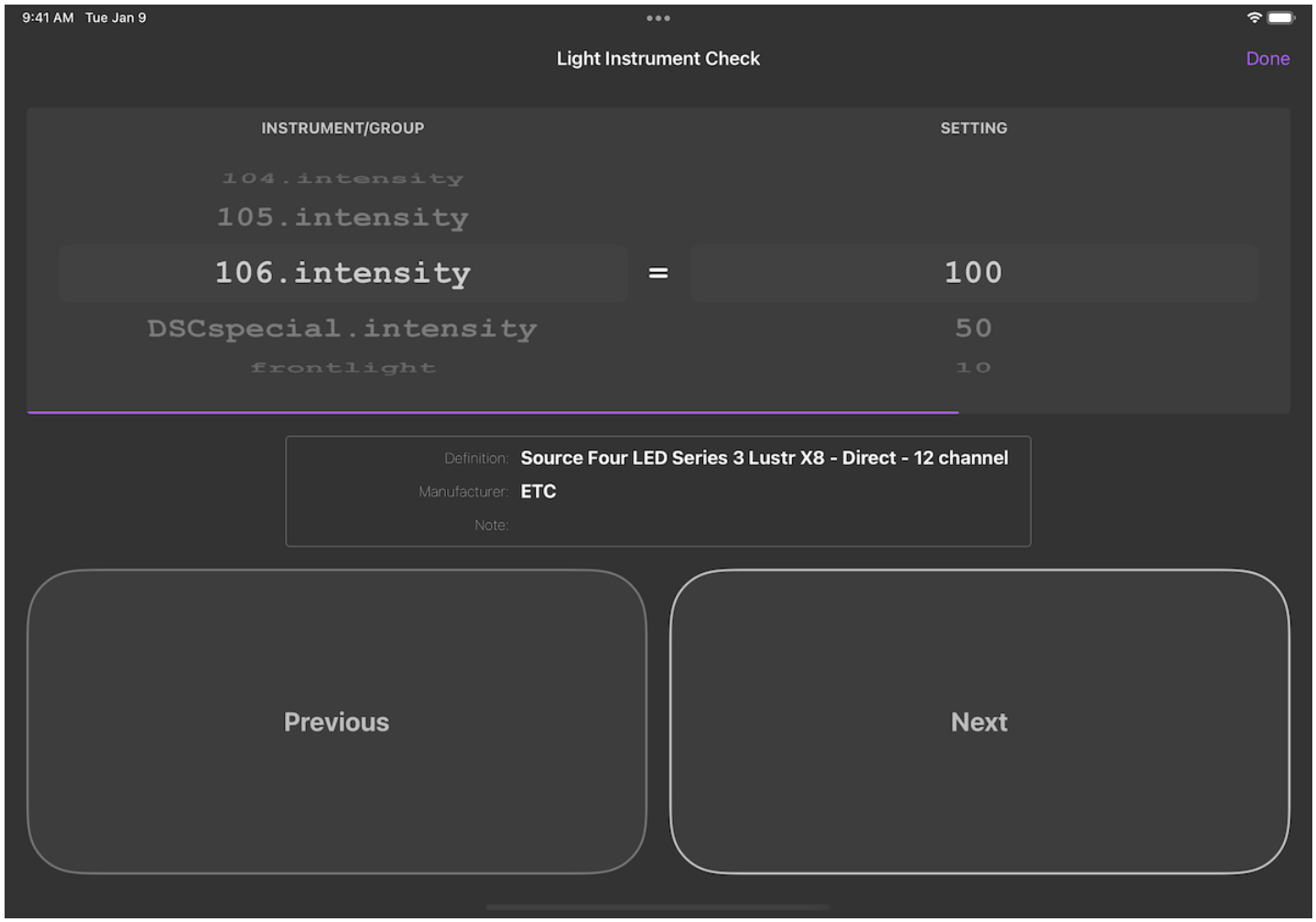
Each button is only enabled when the corresponding action is available.

Revert, Clear all, Jump to cue

These buttons also perform the same function as their QLab counterparts.

Light Instrument Check

The Light Instrument Check tool lets you flip through each lighting instrument and light group in your show in order to ensure that everything is working properly.



When you begin an instrument check, all lights are reset to their home positions. When each instrument is checked, its default parameter is brought to either 100%, 50%, or 10%. You can choose this level using the *Setting* adjustment on the right side of the screen.

Keyboard Shortcuts

QLab Remote supports keyboard shortcuts when a hardware keyboard is connected. Pressing and holding the command key displays a list of all supported shortcuts.

Key Symbols

Symbol	Meaning
⌘	command
⇧	shift
⌥	option

General

Command	Key	Note
GO	space	
Preview Selected	V	
Audition GO	⌥space	
Audition Preview Selected	⌥V	
Panic	escape or `	The backtick character is available as an alternate to escape because the escape key is absent on Apple-branded iPad keyboards.
Toggle Toolbox	⌘K	
Load to time...	⌘T	
Toggle Active Cues	⌘L	
Save	⌘S	
Undo	⌘Z	
Redo	⇧⌘Z	
Edit Stages	⌘,	
Disconnect	⌘W	

Light Keypad

Command	Key	Note
Save	⌘S	
Undo	⌘Z	
Redo	⇧⌘Z	
New Cue with Changes	⌘N	
Record New with All	⇧⌘N	
Revert	⌘R	
Jump to cue...	⌘J	
Previous Light Command	up	
Next Light Command	down	
Dismiss keypad	escape	

Audio Map Editor

Command	Key	Note
Save	⌘S	
Undo	⌘Z	
Redo	⇧⌘Z	
Hide heatmap	⌘0	
Show heatmap for output 1 – 9	⌘1 – 9	

Video Stage Editor

Command	Key	Note
Save	⌘S	
Undo	⌘Z	
Redo	⇧⌘Z	
Dismiss keypad	escape or `	

Requirements

The following features in QLab Remote are available when connected to a specific minimum version of QLab.

QLab 5.5 or newer

- Edit audio output maps.

QLab 5.0 or newer

- Connect to QLab 5 workspaces.
- Communicate using encrypted network connection.
- Start cues with audition GO and audition preview.

QLab 4.6 or newer:

- Options for Second Triggers in the Triggers inspector panel.
- Display audio output names in the Device & Levels inspector panel.

QLab 4.5 or newer:

- **Set Levels...** button in the Device & Levels inspector panel.
- **Set Geometry from Target** button in the Geometry panel of Fade cue inspector.

QLab 4.3 or newer:

- Browsing folder aliases in the File Target document picker of the cue inspector.

QLab 4.2 or newer:

- Light Keypad.
- Instrument Check.
- Cues can be deselected in the cue list.
- Override status indicators.
- Adjusting video geometry in Fade cues.
- **Add Command...** button in Light cue inspector.

QLab 4.1 or newer:

- GO in QLab Remote is covered by workspace double-go protection.
- QLab Remote respects Show/Edit mode.

QLab 4.0 or newer:

- Connect to QLab 4 workspaces.
- Load to Time.
- The Cues Toolbox.
- Long-pressing on cues to reorder them in the cue list.
- Validating audio levels set in QLab Remote against QLab's *Minimum volume limit*.
- Editing video surfaces.
- On-screen animations to indicate panicking cues.
- Progress bars in the Active Cues list.
- Assigning targets to Audio and Video cues.
- Adjusting Video cues' surface assignments.
- Adjusting rotation of Video cues.
- Adjusting absolute/relative mode of Fade cues.
- Activating/deactivating individual audio levels in Fade cues.
- Adjusting Triggers.

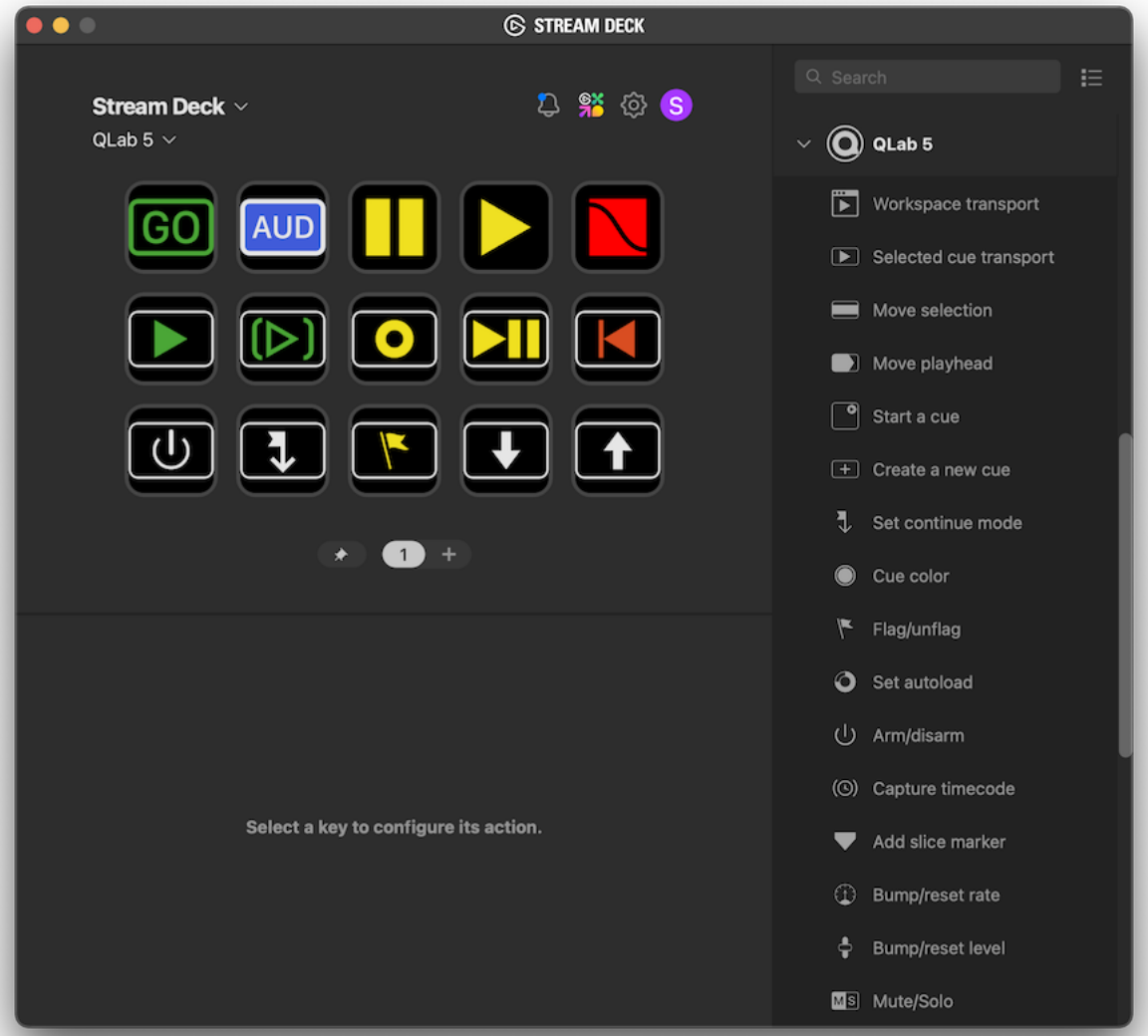
---

1. While QLab 3 is still compatible with QLab Remote, it is unsupported as of the release of QLab 5 and may not work perfectly, or not in all configurations.↩

Stream Deck

The Elgato Stream Deck family is a collection of control surfaces which can be configured to provide immediate, tactile access to software functions that might otherwise be hidden in a buried menu or behind a hard-to-remember keyboard shortcut. There are various models of Stream Deck with different numbers of keys and dials to suit different needs and price points.

The official QLab plugin for Stream Deck gives you direct access to QLab-specific actions using Elgato's free Stream Deck software.



The current version of the QLab plugin is **1.0.0.1**. It requires Stream Deck software version **6.4** or newer, and macOS **10.15** or newer. It is fully compatible with all models of Stream Deck hardware, although support for the encoders on the Stream Deck Studio is slightly limited.

Terminology

Stream Deck hardware has *keys*. Some models also have *dials* and some models also have a *touchscreen*. Dials can be rotated clockwise and counterclockwise, and can also be pressed like a key. Touchscreens can accept taps, tap-and-holds, and swipes. Stream Deck software is centered around *plugins* which offer a set of possible *actions* that can be performed by pressing keys, turning or pressing dials, and tapping or long-tapping on a touchscreen.

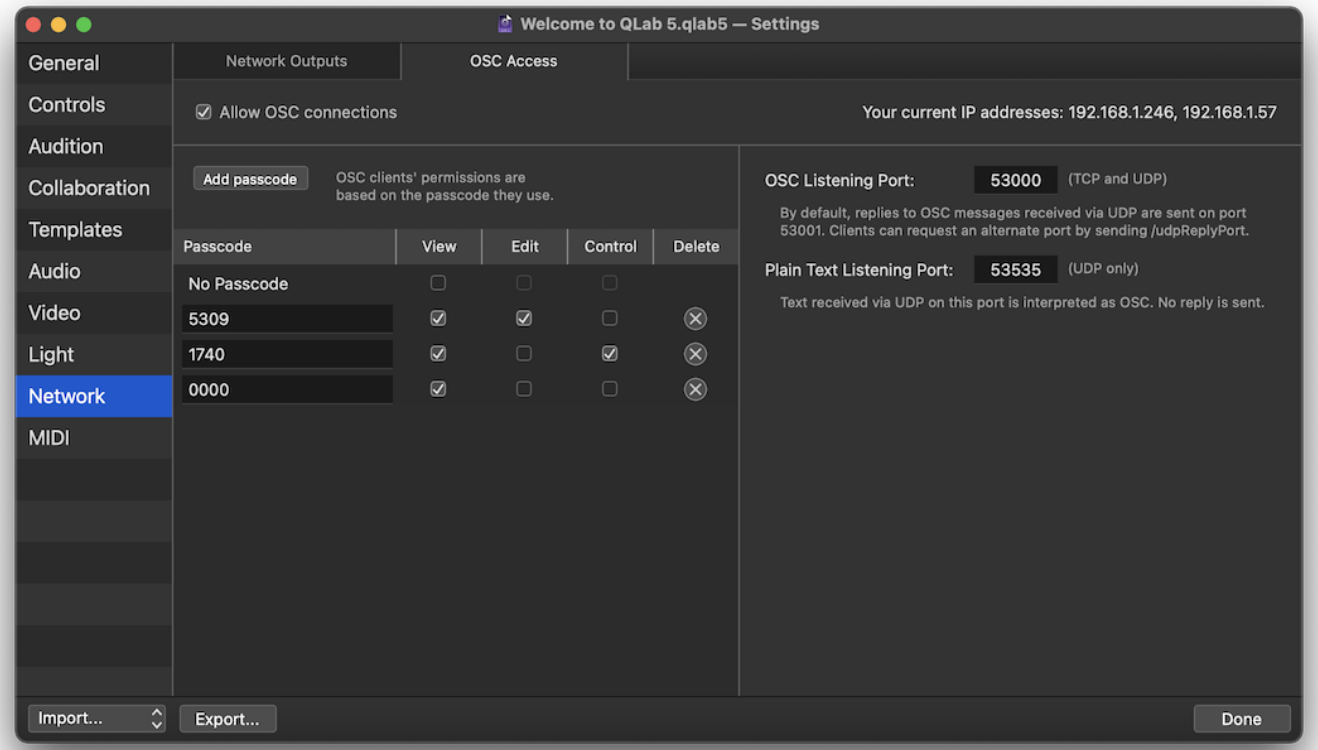
Setting Up

To use a Stream Deck with QLab, first download and install Stream Deck. Once that's done, visit the Stream Deck Marketplace to download the QLab plugin.

More detailed instructions for installation and basic use of Stream Decks can be found on Elgato's website.

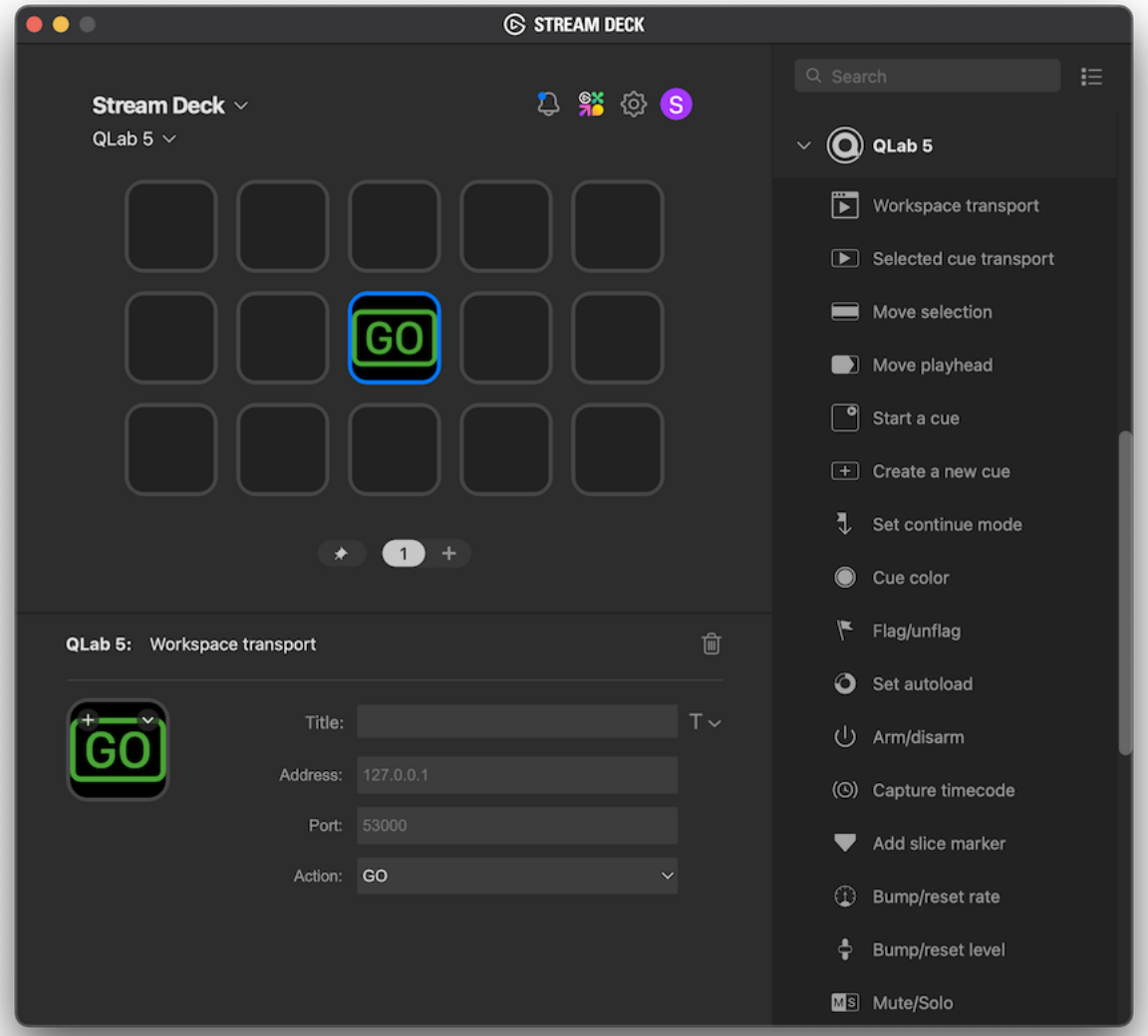
In Your Workspace

The Stream Deck plugin communicates with QLab using OSC and, for now, cannot send a passcode. Therefore, you'll need to visit Workspace Settings → Network → OSC Access, check the box labeled *Allow OSC connections*, and check the boxes next to *No Passcode* that correspond to the types of actions you plan to use.



Configuring Your First Action

Once everything above is set up, you can assign QLab actions to the Stream Deck by choosing *Configure Stream Deck* from the **Stream Deck** menu item on the right side of your Mac's toolbar, selecting an action from under the **QLab 5** heading in the sidebar, and dragging the action to a key or dial of your choosing.



With that key or dial selected, you'll see the properties of the key or dial in the bottom part of the window, called the property inspector.

Properties which must be manually configured before the action will work are marked on the right with a small exclamation mark in a circle.

Networking Considerations

While the Stream Deck software needs to run on the same computer that the Stream Deck hardware is plugged into, each QLab action can be routed to any Mac on your local network.

Every QLab action has an *Address* property and a *Port* property. If you leave these blank, they will use their default values of `127.0.0.1` and `53000`. Assuming you've changed nothing in QLab, those values will direct the action to QLab running on the same Mac as Stream Deck. If you want to direct the action to another Mac, enter that Mac's IP address here. If you've changed the port that your workspace is listening to, enter the port number here.

Key Actions

Workspace transport

This action has one required property, *Action*, which can be set to:

- GO
- Audition GO
- Panic
- Pause All
- Resume All
- Reset All

Selected cue transport



This action sends a command to the currently selected cue or cues. It has one required property, *Action*, which can be set to:

- Preview
- Audition Preview
- Panic
- Hard Stop
- Pause
- Resume
- Pause/Resume (toggle)
- Reset
- Load

Move selection



This action can be set to move the selection to either the previous cue or the next cue.

If the playhead is locked to the selection, this action will move the playhead as well.

Move playhead



This action can be set to move the playhead to either the previous cue or the next cue, or it can be set to move the playhead to either the previous cue sequence or next cue sequence.

If the playhead is locked to the selection, this action will move the playhead as well.

Start a cue



This action requires a cue number (or an acceptable OSC wildcard pattern), which is the cue that will be started by the action, and a color which is displayed on the Stream Deck.

Create a new cue



This action has one required property, *Cue type*, which defines the type of cue that it will create.

Set continue mode



This action has one required property, *Mode*, which defines the continue mode it will set the selected cue to use.

Cue color



This action has two required properties, *Selection* and *Color*.

- **Selection** defines whether the action will set the selected cue's color or second cue color.
- **Color** defines the color that will be set.

Flag/unflag



This action has one required property, *Set state*, which defines whether the action will flag the selected cue, unflag it, or toggle the flag.

Set auto-load



This action has one required property, *Set state*, which defines whether the action will set the selected cue to auto-load, set it to not auto-load, or toggle the auto-load state.

Arm/disarm



This action has one required property, *Set state*, which defines whether the action will arm the selected cue to auto-load, disarm it, or toggle the armed state.

Capture timecode



This action has no required properties. This action is the equivalent of clicking the timecode trigger **Capture** button in the Triggers tab of the inspector.

Add slice marker



This action has no required properties. This action is the equivalent of clicking the **Add Slice** button in the Time & Loops tab of the inspector.

Bump/reset rate



This action has two required properties. *Adjust* defines whether the action will add the *Amount* to the selected cue's current rate, subtract the *Amount* from the current rate, or reset the cue's rate to the default of `1`.

Bump/reset level



This action has four required properties.

Row and *Column* let you choose which crosspoint in the selected cue's levels matrix will be adjusted. Row 0 is the main output row; column 0 is the main input column. So, setting both to 0 will adjust the main level control of the cue. Setting both to 1 will adjust the crosspoint which routes input 1 to output 1.

Adjust defines whether the action will add the *Amount* to the selected cue's current rate, subtract the *Amount* from the current rate, or reset the cue's rate to the default of `1`.

Mute/Solo



This action has three required properties.

- *Output* defines which cue output will be targeted.
- *Parameter* defines whether the action will target the mute button or solo button.
- *Set* defines whether the action sets the button (turns it on) or unsets it (turns it off.)

Bump layer



This action has two required properties. *+/-* defines whether the action will add the *Amount* to the selected cue's layer or subtract the *Amount* from the current layer.

Bump crop



This action has three required properties.

- *Side* defines which side of the selected cue will be targeted.
- *+/-* defines whether the action will add to or subtract from the crop.
- *Amount* defines how much to adjust the crop with each bump.

Bump opacity



This action has two required properties. *+/-* defines whether the action will add the *Amount* to the selected cue's opacity or subtract the *Amount* from the opacity. In this context, an opacity of `1` means 100% opaque, so an adjustment of 1% is equal to `0.01`.

Smooth



This action has one required property, *Set state*, which defines whether the action will set the selected cue's *Smooth* checkbox on, set it off, or toggle it.

Bump translation/scale/anchor



This action has four required properties.

- *Parameter* defines whether the action will target the translation, scale, or anchor parameter of the selected cue.
- *Axis* defines whether the action will target the X or Y axis of the selected cue.
- *+/-* defines whether the action will add to or subtract from the target.
- *Amount* defines how much to adjust the targeted parameter.

Lock/unlock scale axes



This action has one required property, *Set state*, which defines whether the action will lock the selected cue's X and Y scale axes, unlock them, or toggle the lock.

Bump rotation



This action has three required properties.

- *Axis* defines whether the action will target the X, Y, or Z axis of the selected cue's rotation.
- *+/-* defines whether the action will add to or subtract from the rotation.
- *Amount* defines how much to adjust the rotation.

Reset geometry



This action has one required property, *Parameter*, which defines which geometry parameter of the selected cue will be reset.

Workspace toggles



This action has two required properties: *Action* defines which workspace setting or window to target, and *Set state* defines whether the action will turn the setting or window visibility on, turn it off, or toggle it.

Available actions are:

- Lock playhead to selection
- Live fade preview
- Show mode
- Fullscreen (main workspace window)
- Audition monitor windows (show/hide all)
- Light dashboard
- Override controls window

Set overrides



This action has two required properties: *Override* defines which workspace override to target, and *Set state* defines whether the action will turn the override on, turn it off, or toggle it.

Custom OSC message



This action lets you send a hand-written OSC message to QLab. Be sure to format the OSC message as you would in a QLab Network cue.

Dial Actions

Selection



- **Rotate** the dial to move the selection up or down.
- **Press** the dial to move the playhead to the selection.

Playhead



- **Rotate** the dial to move the playhead up or down.
- **Press** the dial to lock and unlock the playhead to the selection.
- **Tap** the touchscreen to change whether the dial moves the playhead by cue or by sequence. You can also change this setting in the Stream Deck inspector.

Load to time



- **Rotate** the dial to load to time.
- **Press** the dial to toggle the resolution of the dial between 1 second, 0.1 second, and 0.01 second per “tick”.
- **Tap** the touchscreen to change whether the dial targets the selected cue(s), the cue at the playhead, or the active cue(s). You can also change this setting in the Stream Deck inspector.
- **Tap and hold** the touchscreen to reset the target to time 0.

Duration



- **Rotate** the dial to change the duration of the selected cue(s). If the selected cue does not have an editable duration, this action has no effect.
- **Press** the dial to toggle the resolution of the dial between 1 second, 0.1 second, and 0.01 second per “tick”.

Pre-wait



- **Rotate** the dial to change the pre-wait of the selected cue(s).
- **Press** the dial to toggle the resolution of the dial between 1 second, 0.1 second, and 0.01 second per “tick”.
- **Tap and hold** the touchscreen to reset the pre-wait to time 0.

Post-wait



- **Rotate** the dial to change the post-wait of the selected cue(s).
- **Press** the dial to toggle the resolution of the dial between 1 second, 0.1 second, and 0.01 second per “tick”.
- **Tap and hold** the touchscreen to reset the post-wait to time 0.

Start time



- **Rotate** the dial to change the start time of the selected cue(s).
- **Press** the dial to toggle the resolution of the dial between 1 second, 0.1 second, and 0.01 second per “tick”.
- **Tap and hold** the touchscreen to reset the start time to time 0.

End time



- **Rotate** the dial to change the end time of the selected cue(s).
- **Press** the dial to toggle the resolution of the dial between 1 second, 0.1 second, and 0.01 second per “tick”.
- **Tap and hold** the touchscreen to reset the end time to time 0.

Rate



- **Rotate** the dial to change the rate of the selected cue(s).
- **Press** the dial to toggle the resolution of the dial between 0.1 and 0.01 per “tick”.
- **Tap and hold** the touchscreen to reset the rate to 1.

Level



This action has two required properties, *Row* and *Column*, which let you choose which crosspoint in the selected cue’s levels matrix will be adjusted. Row 0 is the main output row; column 0 is the main input column. So, setting both to 0 will adjust the main level control of the cue. Setting both to 1 will adjust the crosspoint which routes input 1 to output 1.

- **Rotate** the dial to change the level of the target crosspoint of the selected cue(s).
- **Press** the dial to toggle the resolution of the dial between 1 dB and 0.1 dB per “tick”.
- **Tap and hold** the touchscreen to reset the crosspoint to 0 dB.

Layer



- **Rotate** the dial to change the layer of the selected cue(s).
- **Press** the dial to toggle the resolution of the dial between 1 and 10 per “tick”.
- **Tap and hold** the touchscreen to reset the layer to *top*.

Opacity



- **Rotate** the dial to change the opacity of the selected cue(s).
- **Tap and hold** the touchscreen to reset the opacity to 100%.

Translation



This action has one required property, *Axis*, which defines whether the action will target the X or Y axis of the selected cue.

- **Rotate** the dial to translate the selected cue(s) along the target axis.
- **Press** the dial to toggle the resolution of the dial between 1 and 10 pixels per “tick”.
- **Tap and hold** the touchscreen to reset the translation to 0.

Scale



This action has one required property, *Axis*, which defines whether the action will target the X or Y axis of the selected cue.

- **Rotate** the dial to scale the selected cue(s) along the target axis, or both axes if the scale is axis-locked.
- **Press** the dial to toggle the resolution of the dial between 0.1 and 0.01 per “tick”.
- **Tap and hold** the touchscreen to reset the scale to 0.

Rotation



This action has one required property, *Axis*, which defines whether the action will target the X, Y, or Z axis of the selected cue.

- **Rotate** the dial to rotate the selected cue(s) about the target axis.
- **Press** the dial to toggle the resolution of the dial between 1 and 0.1 degrees per “tick”.
- **Tap and hold** the touchscreen to reset the selected cue(s)’ rotation to 0.

Anchor



This action has one required property, *Axis*, which defines whether the action will target the X or Y axis of the selected cue.

- **Rotate** the dial to move the selected cue(s)’ anchor along the target axis.
- **Press** the dial to toggle the resolution of the dial between 1 and 0.1 pixels per “tick”.
- **Tap and hold** the touchscreen to reset the target axis’ anchor to 0.

Crop



This action has one required property, *Side*, which defines which side of the selected cue(s) will be adjusted.

- **Rotate** the dial to adjust the crop of selected cue(s) on the target side.
- **Press** the dial to toggle the resolution of the dial between 1 and 0.1 pixels per “tick”.
- **Tap and hold** the touchscreen to reset the target side’s crop to 0.

Using OSC

QLab has extensive support for the Open Sound Control protocol, a network communication standard for computers and multimedia devices. OSC is a great way to control QLab from other software and hardware because it's relatively simple to set up, requires no specialized hardware, and uses networking infrastructure that is often either already in place or easy to implement.

Understanding OSC

All software or devices which support OSC have their own dictionary of commands. You can use programs like Max, Medialon Manager, and TouchOSC, or hardware like ETC's EOS family to send messages that exist in QLab 5's OSC dictionary.

Alternately, if your software or hardware does not allow you to program your own messages, you can use the OSC controls in Workspace Settings → Controls to capture your device's OSC messages, and map them to several of QLab's workspace-level commands like GO, Panic, Load, and so on.

Physical Requirements

QLab accepts incoming OSC messages via TCP and UDP over a local network. The sending device must be on the same network, and both the Mac running QLab and the other device must be configured correctly to share network traffic. A good, but rather dry, introduction to setting up devices on a network can be found here.

The long and short of it is that devices must be on the same network, on the same subnet, and use IP addresses that permit them to communicate with each other.

Controlling The Workspace

QLab accepts commands like `/go`, `/panic`, and `/save`, which are referred to as *workspace level messages* because they are directed at a workspace. When QLab receives these messages, it behaves exactly as though the corresponding button, menu item, or keyboard shortcut occurred within QLab. So sending `/go` to QLab from an external device will cause the exact same thing to happen as sitting down in front of QLab and clicking on the GO button with the mouse. You can find a list of workspace level OSC messages in the Workspace Messages section of QLab's OSC dictionary.

If you are using a device or program which sends specific OSC messages that you want to associate with workspace level events such as GO and panic, you can do that in Workspace Settings → Controls → OSC.

There are also several *application level messages* which interact with QLab itself, "above" the workspace level. You can find a list of application level OSC messages in the Application Messages section of QLab's OSC dictionary.

Controlling Individual Cues

There are also a variety of commands that can be directed at cues themselves. These commands have the structure `/cue/{identifier}/{command}`, where `{identifier}` is the piece of information that QLab uses to determine where to send the command. There are five identifiers available for OSC commands:

- **/cue/{x}** - replace **{x}** with the cue number of the cue you want to target.
- **/cue/selected** - the command will target all selected cues. If one or more of the selected cues can't accept the command, they will be ignored; the command will be applied only to the cues which can accept it.
- **/cue/playhead** - the command will target the cue that's standing by in the current cue list.
- **/cue/*** - the ***** character is a wildcard. A command directed to `/cue/*` will target *all* cues in a workspace. You can combine the wildcard with other text, though, so a command directed to `/cue/*1` will target all cues whose cue numbers end with `1`, like cue `1`, cue `11`, cue `41`, and cue `alternate1`. You can also use the wildcard in the middle or at the end of a cue number, so you can use `ham*` to target `hamlet`, `hamster`, and, of course, `hamilton`.
- **/cue_id/{id}** - replace **{id}** with the cue ID of the cue you want to target. Cue ID is a unique identifier for each cue which never changes, even if you change the cue's name and number. A cue's ID can be discovered using the AppleScript property `uniqueID`

Important: Spaces are not permitted in OSC addresses, so if you are using OSC to control your workspace, it's a good idea to avoid spaces in cue numbers. For example, `/cue/oh hello/start` is an invalid command, since there's a space between `oh` and `hello`. Also, sadly, OSC does not support unicode or emoji characters so cues with numbers like 繁 or □ are not addressable via OSC.

The Benefits of OSC

OSC's two main strengths are it's infrastructure requirements and its flexibility. It runs over standard networking infrastructure, including WiFi, which is fairly cheap and very easy to obtain. You can buy network equipment pretty much anywhere, so if you're on tour and a piece of your OSC infrastructure breaks, it may be relatively easy to find a replacement. MIDI cables and devices are a bit harder to come by if you're not in a large city.

Complex networks of devices with bi-directional OSC control are fairly simple to set up; just connect everything to a network switch and assign IP addresses in the same subnet range. MIDI, on the other hand, requires a more carefully designed and laid out infrastructure, with "in" and "out" cables for each device, careful management of power, and a mere 16 channels for separation of messages.

OSC is more flexible than MIDI, too, because each program or device defines its own set of OSC commands, rather than re-purposing MIDI messages such as Note On and Program Change. The set of commands that a given device can use is based on the exact needs of that device.

The Perils of OSC

Despite these advantages, OSC is not without its drawbacks.

The more we use networks, the more important it is to secure them. If your show network includes WiFi, it's important to follow best practices for securing your WiFi network. Audience members are more likely to have a cell phone in their pocket than a MIDI device, after all, and it's not hard to imagine a curious individual doing a little recreational hacking during intermission.

OSC is also, frankly, a little complicated when you're trying to do complicated things. You have to know a good amount about networking in order to troubleshoot subtle problems, and it can be time consuming to gather and retain all the necessary information about which commands do what for each device or program.

A small but frustrating technical detail is that OSC does not support use of the comma (,) as a decimal separator, only the period (or "full stop".) This is important to keep in mind if your Mac, or other devices on your network, are configured to use comma decimal separators, as is common in many European nations.

Finally, there are some political implications of using OSC. Folks are typically pretty skeptical when approached by someone from a different department with the end of a network cable. Coordinating IP addresses and address schemes can be complicated, and someone has to do it, and everyone has to agree who that someone is. While this is becoming more and more common on large shows, it's still unfamiliar territory to many.

Ultimately, though, we believe that the benefits of OSC far outweigh the drawbacks, and with a little careful planning, all the dangers can be overcome.

Using MIDI & MSC

QLab can be controlled from another program or device using MIDI. MIDI, which stands for *Musical Instrument Digital Interface,* is a digital protocol created for allowing electronic musical instruments, computers, and other related equipment to connect and communicate with each other. While not originally designed for show control, MIDI has been adopted (and adapted) for use in theaters, theme parks, concerts, and all manner of live entertainment venues.

Workspace MIDI control

QLab allows you to map incoming MIDI voice messages to controls at the workspace level, such as GO, Pause All, Panic, and so forth, so that you can use a MIDI device as a remote control for running QLab. This could be as simple as a single physical GO button, or as complex as a full MIDI surface with a button for each MIDI-mappable control in QLab. Any device capable of sending MIDI voice messages can be used, including keyboards, drum pads, purpose-built remote devices, or the user-defined keys on many popular digital audio consoles.

QLab also accepts MIDI Show Control (MSC) messages.

To configure QLab to respond to MIDI Voice Messages and/or MSC messages, visit Workspace Settings → Controls → Workspace MIDI window.

At the top of the window, you can select which of the 16 MIDI channels QLab will listen to. You can also select *Any* to allow QLab to respond to incoming messages on any channel. QLab will listen on the selected channel for incoming MIDI messages from all MIDI devices connected to the computer.

The next section of MIDI Controls pertains to MIDI Show Control, which is discussed below.

Checking the box labeled *Use "Musical" MIDI Controls* will enable incoming MIDI controls for the workspace. You can then enter the MIDI message that you want to assign to each workspace control. Workspace controls can respond to four types of MIDI messages: `Note On`, `Note Off`, `Program Change`, and `Control Change`. Choose the message type from the dropdown menu, and then enter the two bytes of data for the message. For `Note On` and `Note Off` messages, byte 1 is the note number, and byte 2 is the velocity. For `Control Change` messages, byte 1 is the control number, and byte 2 is the control value. For `Program Change` messages, byte 1 is the program number, and byte 2 is ignored.

You can use greater-than (>) or less-than (<) signs in the byte fields. So, for example, if you want to use a `Note On` message to GO, and you're using a velocity-sensitive keyboard, you may wish to enter `>0` for byte 2, so that the GO occurs no matter the velocity of the keypress.

You can also use the word *any* in the byte fields. It is recommended that you proceed with caution, though, especially when using `any` in conjunction with `Note On` messages. Many MIDI devices don't actually use `Note Off`, and instead send a `Note On` message with a velocity of `0` to indicate a `Note Off`. If your MIDI device does that, and you're using a Note On message with `any` as byte 2, you will get double triggers: one when the key is pressed, and another when it's released.

Instead of manually entering each MIDI message, you can also click the *Capture* button next to the workspace control that you wish to use. Once you click it, you'll see "Waiting..." in yellow text appear in the two byte fields. QLab will listen for the next compatible incoming MIDI message, and assign it to that control. To cancel listening for a new message without capturing one, just click the "Capture..." button a second time.

Workspace MSC control

MSC stands for MIDI Show Control, and it was created to help simplify the process of using MIDI for show control purposes. Rather than arbitrarily assigning Note On or Program Change messages to show control commands like "go" and "stop", MSC has a set of commands designed for a show control environment.

Checking the box labeled *Use MIDI Show Control*, will allow QLab to respond to a selection of incoming MSC messages associated with common tasks, without requiring you to manually set up connections between MIDI voice messages and controls or triggers in QLab. The MSC specification includes a wide variety of categories for devices that can be addressed by MSC; QLab will respond to messages sent as **Audio (General)**, **Lighting (General)**, and **Video (General)**.

With that box checked, you need to set QLab's MSC Device ID, which is a number between `0` and `126`. Every device on an MSC network must have a Device ID number, and must respond to incoming messages within their own categories if the messages are addressed to that Device ID. Also, all devices

must respond to messages sent to Device ID `127`.

Once you've set the Device ID, there's nothing else to configure in QLab. QLab will respond to the following MSC commands:

- **ALL_OFF.** Stop all currently playing cues.
- **STANDBY+/-.** Move the playhead to the next or previous cue.
- **SEQUENCE+/-.** Move the playhead to the next or previous cue sequence.
- **RESET.** If RESET is sent without a cue number, reset the workspace to the state it would be in when it is first opened. If RESET is sent with a cue number, stop that cue if it's playing, and revert any temporary changes made to it, such as with a Target cue.
- **GO.** If GO is sent without a cue number, start the standing by cue and advance the playhead to the next cue or cue sequence, just as though the on-screen GO button was pressed. If GO is sent with a cue number, start that cue.
- **STOP.** If STOP is sent without a cue number, pause the currently selected cue(s). If STOP is sent with a cue number, pause that cue. We do not know why the MSC spec uses the word "STOP" to mean "pause", but it does, so this is what QLab does.
- **RESUME.** If RESUME is sent without a cue number, resume all currently paused cues. If RESUME is sent with a cue number, resume that cue.
- **LOAD.** If LOAD is sent without a cue number, load the currently selected cue(s). If LOAD is sent with a cue number, load that cue.

One *extremely* important thing to remember when using MSC is that cue numbers in MSC, and in QLab, are *strings* not actual numbers. What that means is that you need to be sure to match the cue numbers exactly on both ends. `1`, `01`, `1.0`, `1.00` are all the same *number*, but they're all different cue numbers!

MIDI triggers

You can also set MIDI voice messages to start individual cues in your workspace. You can learn more about that from the Triggers Tab section of the Inspector section of this manual.

The Benefits of MIDI

MIDI can be a great choice for controlling QLab because it's a simple and reliable protocol with a long-proven track record, and it's spoken by thousands of devices and programs all over the world. Lighting consoles, video mixers, musical instruments... you can find MIDI in use in almost all of them.

MIDI is quick to set up, often only requiring a single cable if you only need control in one direction, and doesn't usually require a ton of programming knowledge to configure.

It's also extremely easy to send from place to place in a theater, since it can be sent over long distances on regular XLR cable using a simple, inexpensive adapter. A common misconception is that MIDI can't be sent farther than 50 feet (about 15 meters) without using an active amplifier, but this usually isn't the case. In fact, MIDI amplifiers often cause more trouble than they solve. For more details, see this guide to MIDI cable length by Richmond Sound Design.

The Perils of MIDI

Its advantages notwithstanding, using MIDI to control QLab, or any other aspect of a show for that matter, isn't without its downsides.

Getting MIDI in and out of a computer requires a MIDI interface. There are hundreds of different brands of MIDI interfaces out there at all price points, and unfortunately the vast majority of them are unsuitable for show-critical use. This is because the amount of MIDI data generated by, say, a keyboard controller is comparatively low, and even a world class pianist can't play faster than the MIDI interface can follow. But a computer generating MIDI, MSC, or MIDI timecode sends a lot more data, and sends it a lot faster than a typical musical performance. This higher load can outstrip the capacity of lower end MIDI interfaces, causing them to either skip or corrupt messages. Needless to say, that's problematic.

For this reason, the only MIDI interfaces we currently recommend are those manufactured by iConnectivity, Kenton, or ESI. We've experienced problems with nearly every other brand of MIDI interface we've encountered in the field. Naturally, your mileage may vary, and you may have perfect success with other interfaces. By and large, though, if your show depends on MIDI control, we strongly encourage sticking with one of these proven brands.

Another reason to be cautious of MIDI has to do with what MSC calls "controlled devices" and "controllers." Most lighting consoles are designed as "controlled devices," which means the level of control you have over the MSC messages they send out is limited. In ETC EOS family consoles, for example,

MSC can only be enabled on a per-list basis. Once enabled on a cue list, the console will send an MSC `GO` with the cue number of every cue that it starts in that cue list. While this is very convenient if it's what you need, it can also create quite a challenge if that's not what you need. Since the lighting console sends an MSC `GO` for every single cue, any cue in QLab with a matching cue number will be started. The only way to ensure that certain cues are started via MSC and others are not is to use matching cue numbers for the MSC-triggered cues, and make sure that other cues in QLab do NOT have the same cue number as any cue on the lighting console. The more advanced ETC consoles also let you work around this with a complicated arrangement of multiple cue lists and macros.

You can also solve the problem by using QLab to send MSC messages to the lighting console, rather than the other way around, since QLab only sends MIDI and MSC messages when you tell it to. But there are any number of reasons why that might not be practical, which leaves you with the hassle described here.

Our intention is not to discourage the use of MIDI and MSC, only to try to point out potential issues, hopefully before you encounter them. With a little care, MIDI can be a very powerful and useful tool in a show control environment.

Using Timecode

Timecode is a standard developed for use in the film industry to keep cameras and sound recording equipment synchronized both on set and in post production. It has since been adopted by theme parks, cruise ships, and theaters as a mechanism for linking lighting, audio, video, and automation equipment.

QLab can be controlled by either LTC (linear or longitudinal timecode) or MTC (MIDI timecode.) LTC is also often referred to casually as "audio timecode" or "SMPTE" (pronounced "SIMP-tee".) Neither term is wrong; LTC uses an audio signal to transmit timecode, so "audio timecode" is a perfectly fair description. SMPTE stands for the Society of Motion Picture and Television Engineers which is the organization that created and maintains the technical standards for timecode. Most accurately, though, SMPTE is the name of the organization, not the name of the timecode standard.

QLab 5 also *syncs* to timecode, which means that when incoming timecode stops, skips ahead, or skips back, QLab is able to respond appropriately.

> **A note on terminology:** many different terms are used in the world of timecode, some of which have linguistic roots which are racist or otherwise violent. Since there is absolutely no reason to adhere to these terms, we do not. Abandoning archaic terms which cause hurt is a small and ridiculously easy thing that we can do to make the places that QLab is used more accessible and inclusive to all.

Controlling Individual Cues

To use timecode to start cues in QLab, you first need to enable incoming timecode for the list or cart that contains those cues. This can be done in the Timecode tab of the inspector when a list or cart is selected.

Check the box labeled *Sync cues in this list from incoming timecode*, and then choose the *sync source*, *sync input channel* (if applicable), and *SMPTE format* appropriate for your system. QLab must be set to use the same format as the device that's sending timecode in order to operate correctly.

The *On Start* pop-up menu allows you to configure QLab's behavior when incoming timecode starts; should QLab start any cues whose timecode trigger precedes the first incoming frame? This can be helpful, for example, when you want to make sure that "setup" cues are run when jumping into the middle of a sequence in rehearsal. More details can be found in the Timecode tab section of this manual.

The *On Stop* pop-up menu allows you to configure QLab's behavior when incoming timecode stops; should cues started via timecode stop, pause, or keep running?

The *Freewheel time* box lets you enter any duration between `0` and `2` seconds which QLab will use as a window of time within which timecode drop-outs will be ignored. This can help protect against stopping or pausing cues because of a momentary drop or corruption of timecode which is common (especially with MIDI timecode.)

Once that's set, you can visit the Triggers tab for any cue within the cue list, check the *Timecode* checkbox, and enter a trigger time for the cue.

You can set the pop-up menu to *Timecode* and enter the time in the format `hour:minute:second:frame`, or set the pop-up menu to *Real Time* and enter the time in the format `hour:minute:second:millisecond`. You should use whichever format makes the most sense for your situation.

The Benefits of Timecode

Timecode can be advantageous when you need to link several devices to a single timeline that is non-negotiable and deliberately rigid. Each device can receive timecode from a central source, and then each device can be individually responsible for doing the right thing at the right time. If you use LTC for your timecode distribution, it can be very simple to route it since it's just a line-level audio signal and you can use conventional audio infrastructure.

MTC, on the other hand, uses MIDI infrastructure which is frankly pretty irritating to deal with unless you're just connecting one thing to another.

Shows that are run "on rails," such as themed attractions, dance-only or dance-centric shows performed to prerecorded tracks, or shows in which 100% repeatability is required for safety purposes are often organized around timecode.

The main reason why you might want to use timecode instead of another show control protocol is that lots and lots of legacy equipment supports it.

The Perils of Timecode

Timecode is a bit at odds with the fundamental design premise of QLab, which is that you don’t necessarily know the amount of time that will elapse between cues. After all, in traditional live theater that’s the whole point of having cues in the first place! Using timecode locks a series of events in place without any flexibility. Obviously, that’s sometimes quite useful, but not always.

We recommend using timecode when you’re adding QLab into a situation that’s already using timecode, or when you’re connecting QLab to equipment that does not support OSC, MIDI, or MSC.

Show Control Broadcast

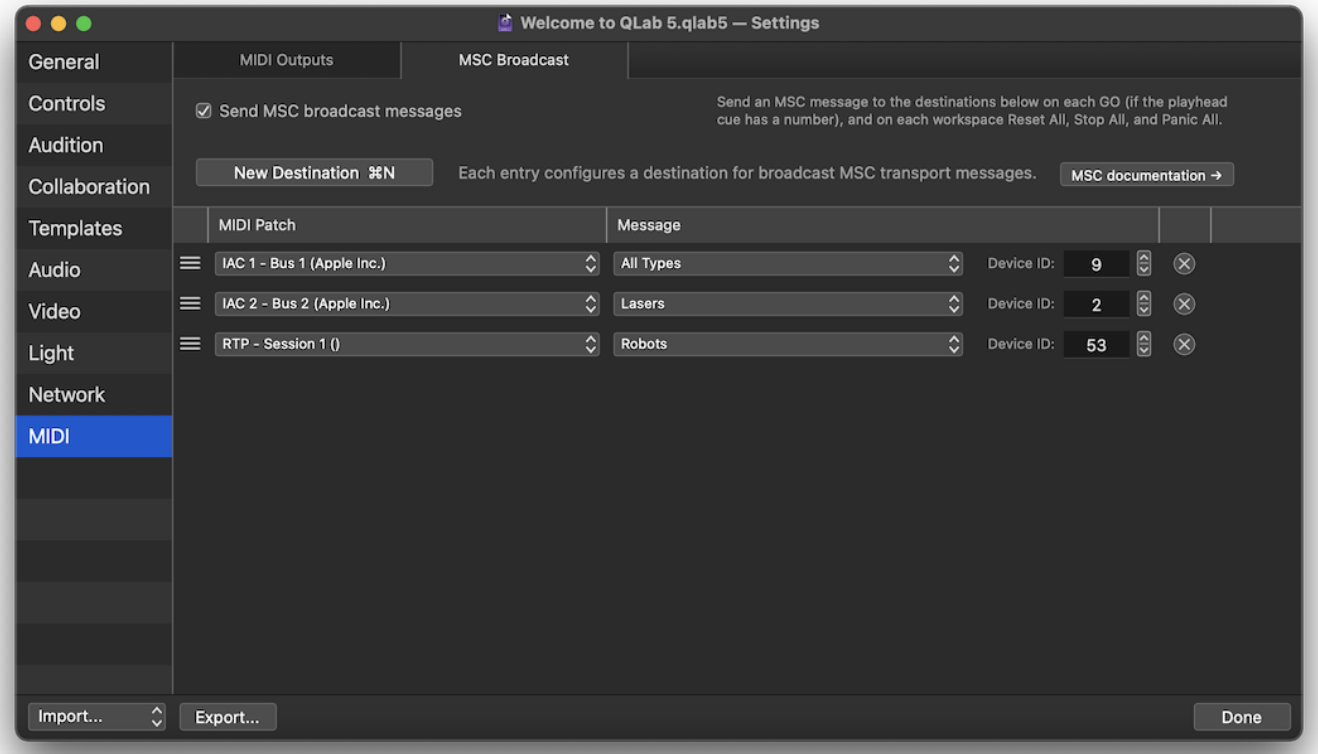
Show control broadcast is QLab's name for a set of MIDI Show Control (MSC) and Open Sound Control (OSC) messages which report on the status of a workspace while in performance. These messages can be used to monitor QLab's status, for example by a stage manager or director of production who wants to be kept informed about "where we are" in the show, or used to remote-control another device by having that device start cues of its own whenever a matching cue in QLab starts.

Show control broadcast can be switched on and off per workspace, and is entirely optional.

Show Control Broadcast With MSC

MSC show control broadcast uses the same MIDI output patches that MIDI cues and MIDI File cues use. To configure MIDI output patches, visit Workspace Settings → MIDI → MIDI Outputs.

Once you have at least one MIDI output patch configured, go to the MSC Broadcast tab.



The upper left corner has a checkbox which allows you to enable or disable MSC broadcast messages for the entire workspace. To add an MSC broadcast output, click on the **New Destination** button or use the keyboard shortcut **⌘N** while this tab of this window has focus.

Each destination is an individually configurable output for MSC broadcast messages, which requires three settings:

- A **MIDI Patch**, which the MIDI output patch that will receive the messages. The MIDI output patch, in turn, connects the workspace to a MIDI device (or virtual device, such as an IAC bus or RTP-MIDI session.)
- Under the **Message** heading is a pop-up menu which lets you set the *command format* for the destination. Command format is an MSC term for the category of device or software that will be receiving the messages. The MSC spec defines a number of specific categories and each device or software must self-identify as belonging to one or more category. QLab itself, for example, responds to messages in the Sound (General), Video (General), and Lighting (General) categories. Refer to the documentation of the device that you're sending messages to in order to learn which category to use.
- Also under the **Message** heading is a field to enter the **Device ID** of the device that will be receiving the messages. Device ID 127 is the "all-call" ID, so if you set a destination to use, say, the *Lighting (General)* format and device ID 127, then all devices which are physically connected to that MIDI output patch and which self-identify as Lighting (General) will receive the messages.

To remove a destination, click the ⊗ button at the right edge of that destination in the list.

Example

In the screen shot above, there are three MSC broadcast destinations.

The first one sends MSC messages to "All Types" via an IAC bus. Since IAC busses are internal to the Mac, any other programs running on the same Mac which are using device ID 9 will receive these messages.

The second one sends MSC messages to "Lasers" via a different IAC bus. Any laser-specific software running on the same Mac which uses device ID 2 will receive these messages.

The third one sends MSC messages to "Robots" via an RTP-MIDI session. Any robots out there listening to the same RTP-MIDI session and using device ID 53 will receive these messages.

MSC Show Control Messages

Action in QLab	Corresponding MSC message
GO	MSC GO {cue number of the cue which ran}
Panic All	MSC ALL_OFF
Stop All	MSC ALL_OFF
HardStop All	MSC ALL_OFF
Reset All	MSC RESET

Show Control Broadcast With OSC

OSC show control broadcast consists of a set of OSC messages which another program or device can send to QLab which will cause QLab to keep that other program or device informed about show control status. If the QLab workspace is closed and reopened, the remote program or device will need to re-send the message to continue receiving updates.

Messages To QLab

The highest-level commands are `/listen` and `/ignore` which start and stop the flow of all show control messages.

Message sent to QLab	Result
`/listen`	Client will receive all show control messages.
`/ignore`	Client will stop receiving all show control message.

Once a client requests updates, QLab can send show control messages in three basic forms.

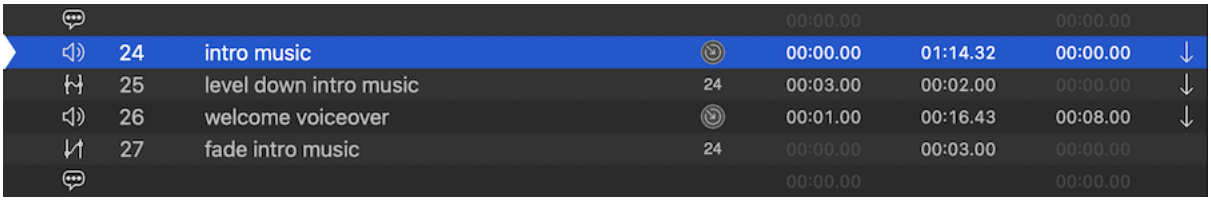
Workspace Event messages with arguments

The first form of OSC show control broadcast message is a workspace event message with arguments, which looks like this:

```
/qlab/event/workspace/{event} {cue number} {cue name} {cue uniqueID} {cue type}
```

Action in QLab	Corresponding `{event}`
GO	`go`
Audition Go	`auditionGo`
Stop	`cue/stop`
Start	`cue/start`
Playhead move	`playhead`

When any of those events occur in the workspace, the corresponding message will be sent with arguments filled in based on the cue that was related to the event. For example, consider this screen shot:



If someone pressed GO at this moment, QLab would send this show control broadcast message:

```
/qlab/event/workspace/go "24" "intro music" "F8CD48F4-B06B-4992-ACD6-5753D0224" "Audio"
```

QLab also sends four individual messages containing each of those four arguments individually:

- `/qlab/event/workspace/go/number 24`
- `/qlab/event/workspace/go/name "intro music"`
- `/qlab/event/workspace/go/uniqueID F8CD48F4-B06B-4992-ACD6-5753D0224`
- `/qlab/event/workspace/go/type Audio`

GO and Start

Whenever a cue starts for a reason *other* than a GO, that is considered a "start." So, if cue `1` is a ▶ Start cue which targets cue `2`, and you select cue `1` and press GO, QLab would report a `/qlab/event/workspace/go` message for cue `1`, and a `/qlab/event/workspace/cue/start` message for cue `2`.

As far as show control broadcast is concerned, "previewing" a cue is the same as "starting" it.

Stop

Whenever a cue stops, whether it's stopped manually or as a result of a ■ Stop cue or because it simply runs out, QLab reports a `/qlab/event/workspace/cue/stop` message for that cue.

Playhead Move

Whenever the playhead in the current cue list moves for any reason, QLab reports a `/qlab/event/workspace/playhead` message for the cue that the playhead moves to.

Workspace Event messages without arguments

The second form of OSC show control broadcast message is a workspace event message without arguments, which looks like this:

```
/qlab/event/workspace/{event}
```

Action in QLab	Corresponding `{event}`
Pause All	`pauseAll`
Resume All	`resumeAll`
Panic All	`panicAll`
Stop All	`stopAll`
HardStop All	`hardStopAll`
Reset All	`resetAll`

Whenever any of those events occurs in the workspace, the corresponding message is sent.

Setting scope

While a comprehensive set of show control messages is hardly overwhelming, it still may be desirable to receive only a subset of these messages instead of every single one.

To that end, the `/listen` and `/ignore` messages have a set of extended forms which allow you to receive or stop receiving exactly the show control messages you want. The extended forms look like this:

```
/{command}/{event}/{optional detail}
```

- `{command}` can be either `listen` or `ignore`
- `{event}` can be any event listed above, such as `go`, `playhead`, `cue/start`, or `stopAll`.

You can stop there with messages like `/listen/go` or `/listen/resumeAll`, or add any of the following optional details:

- `number`
- `name`
- `uniqueID`
- `type`

Adding those details further reduces the variety of messages which get sent.

Example

If a client sends QLab only this message:

```
/listen/go
```

It will only receive `/qlab/event/workspace/go` messages, including all forms of that message with all arguments.

If a client sends QLab only these messages:

```
/listen/go/number, /listen/panicAll
```

It will receive one `/qlab/event/workspace/go {number}` message with every GO with only that single argument. It will also receive a `/qlab/event/workspace/panicAll` message whenever the workspace panics.

By using the extended forms of `/listen` and `/ignore` messages in combination, you can subscribe to the exact set of messages you want, with nothing extraneous.

Customizing outgoing OSC show control broadcast messages

Sometimes, you may need to send show control broadcast messages from QLab which are formatted differently than the default messages listed above. For example, you may be using a device which can only respond to OSC messages formatted in a specific way.

You can set your own format by sending the OSC message `/eventFormat` to QLab from the client that wants the custom-formatted messages. This message accepts one argument which is a text string containing the desired message format. The string can contain any or all of the following tokens which will be replaced with live data when the message is sent:

- `#workspace_id#` - the workspace ID.
- `#address#` - the address of show control broadcast message.
- `#status#` - this token returns “error” if either the incoming or outgoing message is malformed or in any other way problematic, or “ok” if everything is ok.
- `#data#` - the argument of the show control broadcast message.

When you use `/eventFormat`, QLab will only send single-argument show control broadcast messages. The multi-argument message will not be sent.

At present, you are only able to use a single `/eventFormat` at a time (per client) which will format all OSC show control broadcast messages for the given client. This means it’s possible to create a format which will generate invalid messages. For this reason, it’s probably best to `/listen` to specific events which are compatible with your chosen format.

To **find out the current format,** send `/eventFormat` without any arguments. QLab will reply with the current format.

To **reset to the default format,** send `/eventFormat ""` (an empty string argument.)

Examples

- The default format can be re-created using `/eventFormat "#address# #data#"`
- To format a message which starts cues in another copy of QLab on another Mac, you could use this format: `/eventFormat "/cue/#data#/start`. Then send `/listen/go/number` and optionally also `/listen/cue/start/number`
- To send the message expected by an ETC Eos family console, use `/eventFormat "/eos/cue/#data#/fire"`. Then send `/listen/go/number` and optionally also `/listen/cue/start/number`

Network Cues

Network cues you to send messages from QLab to other software or devices using the network connections of your Mac. They can also be used to send control messages to QLab itself. Messages can be sent via TCP or UDP.

The Network cue can send three types of network messages:

- **OSC messages**, which use the Open Sound Control protocol; a flexible, extensible, network-based messaging system designed as a sort of spiritual successor to MIDI.
- **Plain Text messages**, which send plain ASCII text.
- **Hex Code messages**, which send hexadecimal codes.

In addition, QLab includes the following collection of *network device descriptions* which allow a network patch to be configured to communicate with a specific device or program:

- QLab 5
- Go Button 3
- Adamson Fletcher Machine
- Amadeus Acoustics
- Audio Definition Model (ADM)
- Audioström LiveProfessor
- atemOSC
- Behringer WING
- Behringer X-Air family
- Behringer X32 family / Midas M32 family
- BitFocus Companion
- Blackmagic Designs Videohub
- Borealis Vor
- Chamsys MagicQ
- Christie products which support Christie's RS232C protocol via a network connection.
- Christie HS Series
- Cockos Reaper
- Creative Connors Spikemark 5 and 6
- d&b DS100 (Soundscape)
- D::Light
- Dan Murfin ShowPlay
- DiGiCo SD series and Quantum series consoles
- disguise
- Elation Obsidian Onyx
- ETC ColorSource AV family
- ETC Eos Family
- Flamingo Software Atlas
- Flux Spat Revolution
- GarageCube MadMapper 5
- Glypheo
- High End Hog 4
- Innovate Audio panLab 2
- Innovate Audio panLab Console
- L-Acoustics L-ISA
- Lawo mc2 consoles
- Lightkey 4
- MA Lighting grandMA3
- Mark Of The Unicorn (MOTU) AVB Audio Interfaces
- Meyer GALAXY (in Normal mode and in Spacemap mode)
- Millumin 4 and 5
- Mixeffect
- OBSBot Center, Tail Air, and Talent
- PJLink
- Shure Axient Digital Wireless System

- Shure P10T Wireless PSM Transmitter
- Shure ULX-D Digital Wireless System
- Synthe FX Luminair 4
- TheatreMix 3
- VISCA-over-IP
- Yamaha DM3 series
- Yamaha DM7 series
- Yamaha Rivage family
- zactrack
- ZoomOSC and ZoomISO

These network device descriptions include the full library of commands required to communicate with these devices or programs.[1] When a network patch is configured to use one of these network device descriptions, the inspector shows a series of menus and controls specific to that description. This makes it easy to work with a device without having to constantly be referring to the device's manual to look up specific commands or syntax. Examples are provided below.

❋ Network cues require a license of any type.

The Inspector for Network Cues

When a ❋ Network cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as a Settings tab.

Settings

The controls in the Settings tab very depending upon two factors: the selected network patch, and whether or not the cue has a duration. The examples and screen shots in this section of the manual use a variety of combinations of settings in order to give a sense of the possible layouts.

Patch. This pop-up menu allows you to select a network patch for the cue to use. Clicking on the menu allows you to select one of the network patches already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open Network Settings to edit patch list...* to quickly get to Workspace Settings → Network → Network Outputs, or choose *New patch with network device* to quickly generate a new network patch and select it for use. If a cue's patch has multiple destinations, the message sent by the cue will be sent to each destination simultaneously. This is convenient, for example, for easily controlling a primary and a backup device at the same time without needing to create separate cues to control each.

The contents of the lower portion of the Settings tab will change based upon the type of patch in use.

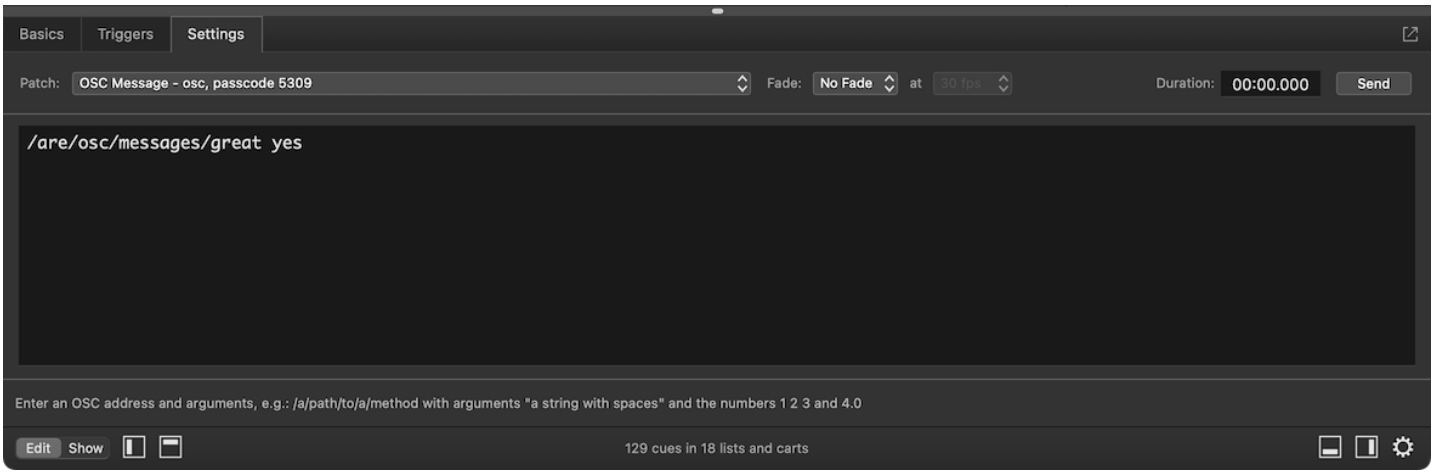
Fade. This pop-up menu will default to *No Fade* if the ❋ Network cue has no duration, or *Resend* if it does have a duration. This menu lets you set the ❋ Network cue to send its message just once (no duration, no fade), to send it repeatedly over time (duration, resend), or to perform a one-dimensional (1D) or two-dimensional (2D) fade to send more complex messages over time. These options are discussed more below.

Duration. You can edit the duration of the cue here, as well as in the Basics tab or in the Duration column of the cue list.

Send. Clicking this button will test-send your message and is the same as previewing the cue.

OSC Messages Without Durations

If the ❋ Network cue uses a patch configured to send an OSC message, the lower part of the Settings tab shows a single, large text field.



The text field allows to enter any single OSC command. OSC messages are made up of an *address* which is the part that looks like a web address, and zero or more *arguments* which follow the address. QLab uses the following formatting rules for OSC messages:

- A single space separates the address from the first argument and arguments from each other
- String arguments that contain spaces must be "enclosed in quotation marks."
- Arguments made of only digits are assumed to be integers.
- Arguments made of digits with a decimal separator are assumed to be floating point numbers (floats).
- The following OSC 1.1 arguments are allowed:
  - `\T` is a boolean true.
  - `\F` is a boolean false.
  - `\I` is an impulse.
  - `\N` is a Null.

While OSC itself only supports using a period (a.k.a. a "full stop") as a decimal separator, QLab is aware of the location settings on your Mac and will invisibly substitute a period in outgoing OSC messages if your Mac is set to use a comma as the decimal separator.

Examples

Here are a few examples showing proper OSC message formatting in QLab.

```
/my/groovy/message 2 10 12
```

This sends a message to address `/my/groovy/message` with three integers, `2`, `10`, and `12`, as separate arguments.

```
/day/start "mornin', ralph" "morning', sam"
```

This sends a message to address `/day/start` with two strings, `morning', ralph` and `mornin', sam`, as separate arguments.

```
/osc/rocks \T
```

This sends a `TRUE` value to the address `/osc/rocks`.

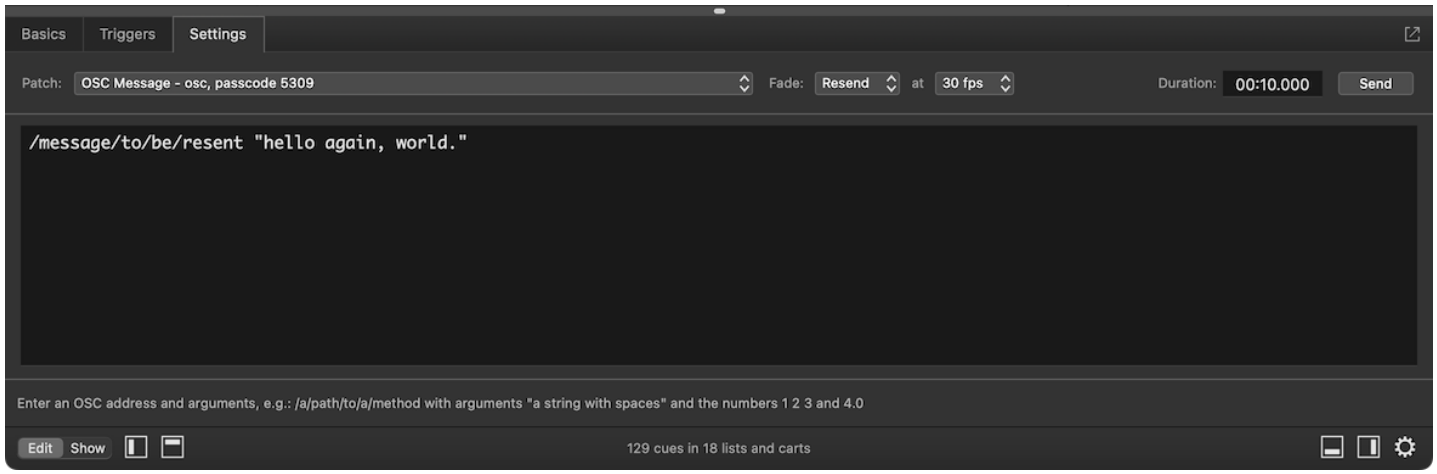
OSC Messages With Durations

If the ❀ Network cue uses a patch configured to send an OSC message and the cue has a duration, the **Fade** control comes into play.

Resend

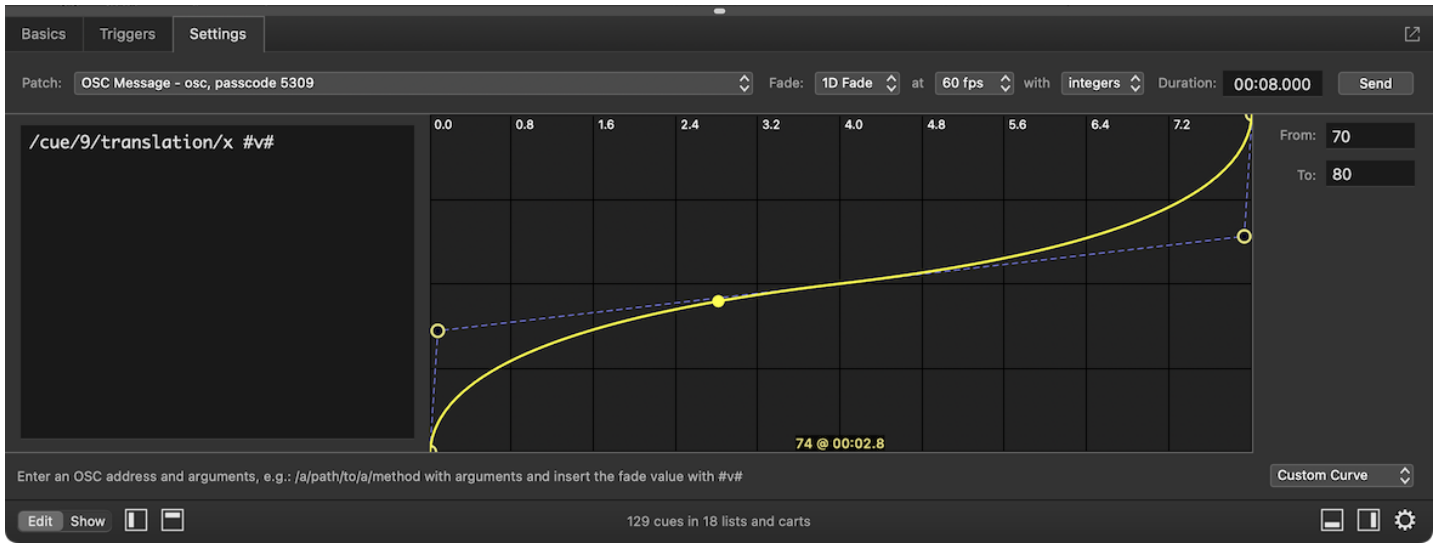
If the **Fade** pop-up menu is set to *Resend*, the cue sends its message repeatedly over its duration. The frequency that the message is sent is adjusted by the **fps** pop-up menu which has several options from 1 message per second up to 120 messages per second.

The cue in this screen shot will send its message 30 times per second for ten seconds:



1D Fade

If the **Fade** pop-up menu is set to *1D Fade*, the cue also sends its message repeatedly over its duration, but allows you to configure the message to change over time.



This cue is set to use a *1D Fade*, resending its message at *60 fps* using *floats*. This means:

- A single value will be faded over the duration of the cue,
- That value will be updated and sent 60 times per second,
- The value will include decimal values (to six decimal places.)

The right side of the tab displays a fade curve (which you can edit to suit your needs) and two text fields. The **From** field sets a starting value, and the **To** field sets a final value.

The text field fills the left side of the tab. Note that the OSC message includes the text `#v#`. This is a **token**, which is like a stand-in for an actual value.

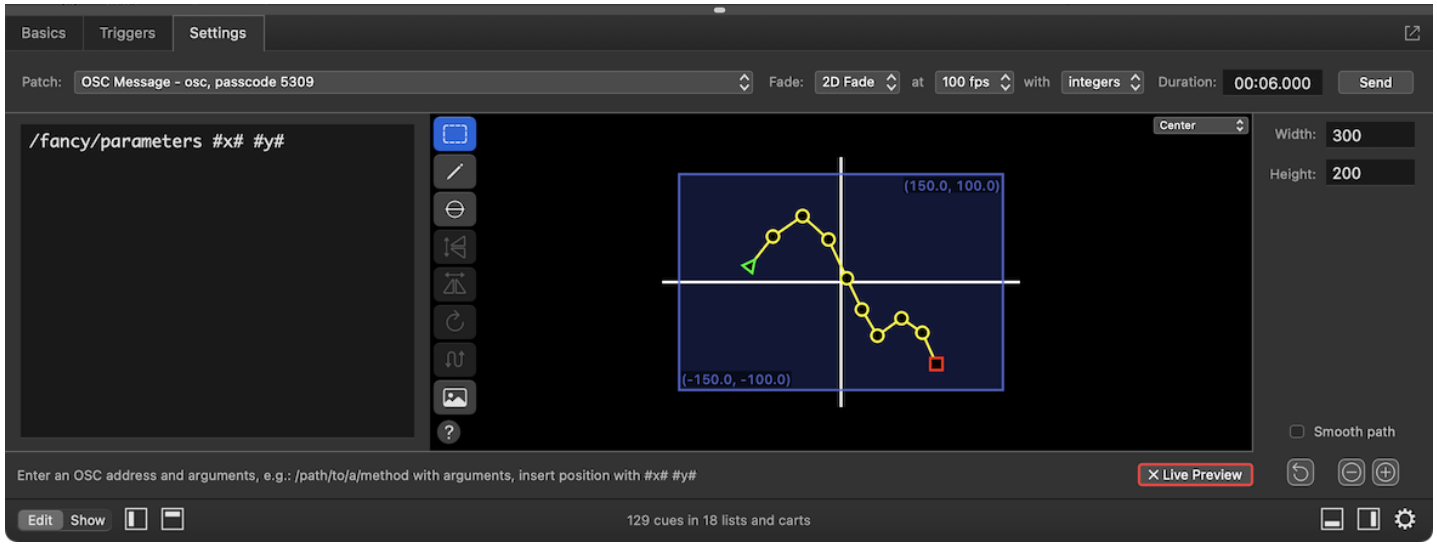
When the cue runs, QLab calculates a fade starting at the **From** value and ending at the **To** value. Each time the message is sent, the calculated value is sent in place of the `#v#` token.

The token is most commonly used in the place of an argument, but if the OSC message that you're sending includes a numerical value in the address, the token can be used there as well.

The pop-up menu in the lower right corner lets you choose between using a custom curve shape or a linear curve shape, and lets you reset the curve to its default state.

2D fade

If the **Fade** pop-up menu is set to *2D Fade*, the cue allows you to fade two values at the same time over the duration of the cue.



This cue is set to use a *2D Fade*, resending its message at *100 fps* using *integers*. This means:

- Two values will be faded over the duration of the cue,
- The values will be updated and sent 100 times per second,
- The values will be integer values only; no decimals.

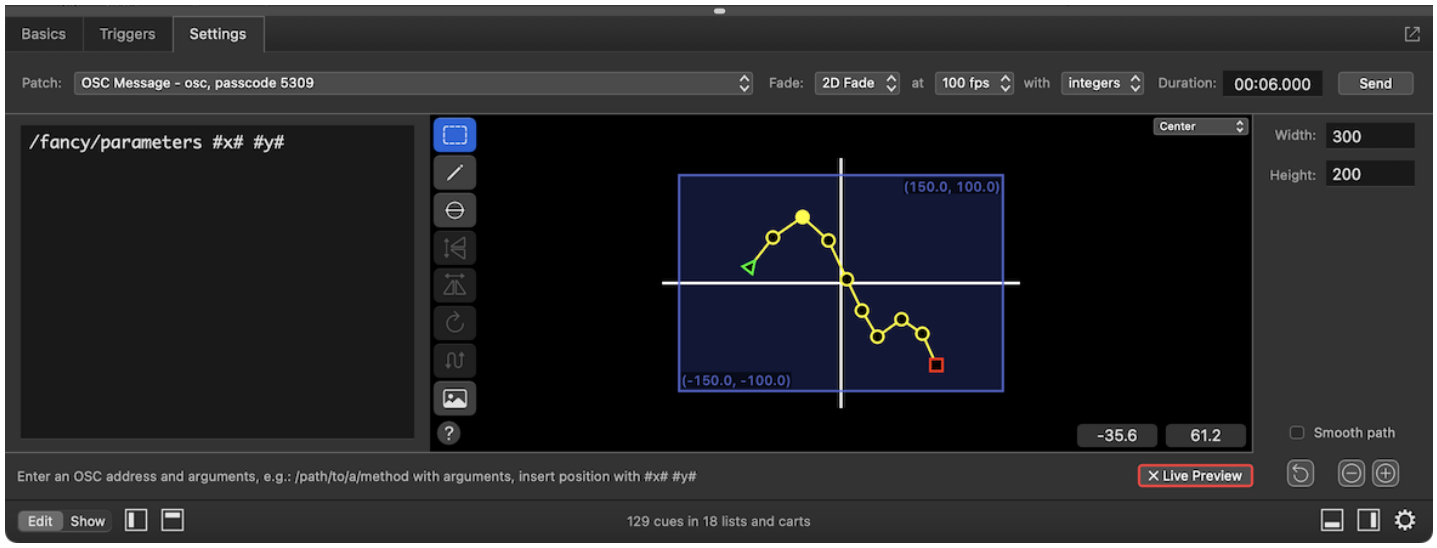
The left side of the tab is filled by a large text field. Here, you can enter an OSC message including two tokens, `#x#` and `#y#`. When the cue runs and the fade animates, the X-axis value will replace the `#x#` token and the Y-axis value will replace the `#y#` token.

The right side of the tab shows a blue rectangular area inside which you can draw a path using your mouse or trackpad. The fade path is shown as a yellow line over a blue background, with the following details:

- The starting point of the path is drawn as a green triangle.
- The ending point is drawn as a red square.
- Points along the path are drawn as yellow circles.
- If there is enough space, small directional carets pointing away from the starting point and towards the ending point will appear along the path.

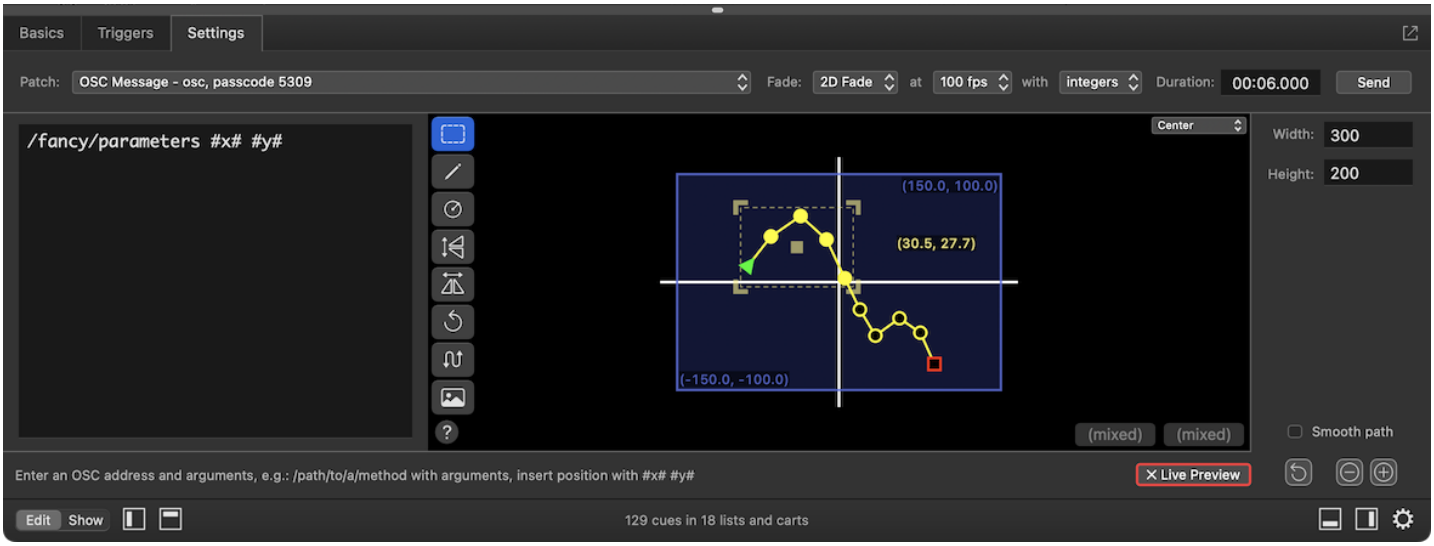
Along the left edge of the canvas is a set of tools for creating and editing paths.

The **select** tool lets you click on individual control points to select them for precise editing.



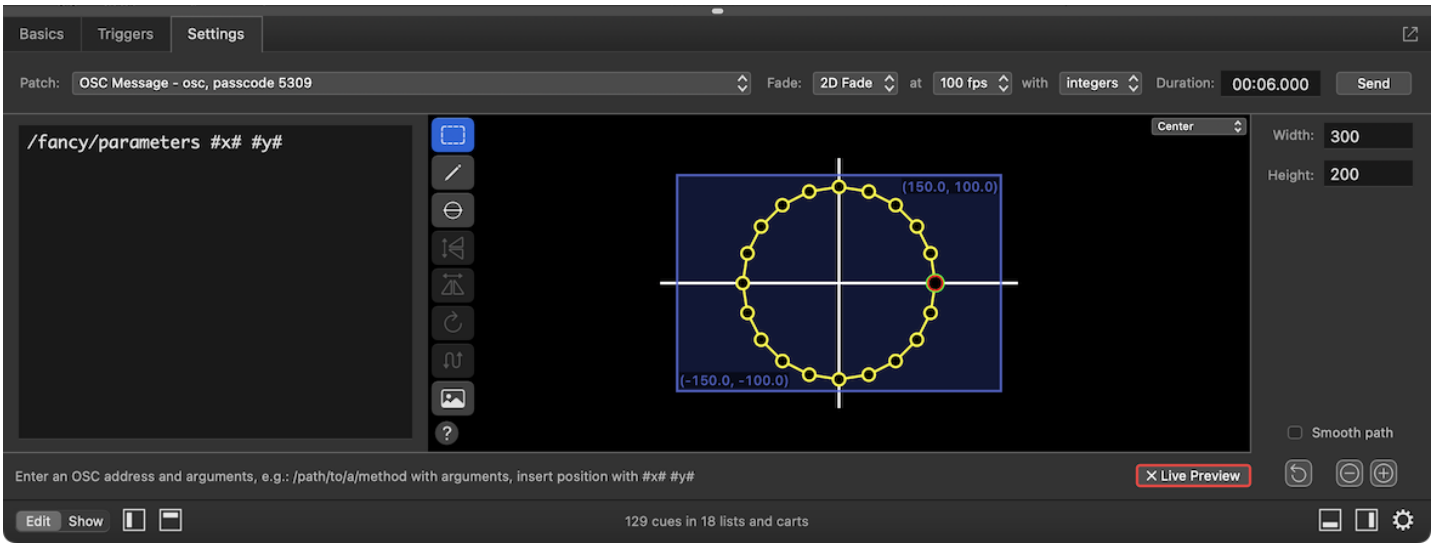
With one point selected, you can use the text fields in the lower right corner to type in a value or click and drag up and down to adjust the value.

You can also click and drag a selection box around several control points:



The **draw** tool lets you click to add points to the path, or click, hold, and drag to draw a freeform line. Holding down the option key (⌥) key allows you to add a point in the middle of a path, otherwise each drawn point will be added to the end of the path. Double-clicking while using the draw tool finishes drawing and switches to the select tool.

The **circle** tool lets you draw a perfectly circular path. The first click sets the starting (and ending) point of the circle, and the second click defines the diameter of the circle. Holding down the option key (⌥) allows you to instead click once to define the center point, and then click again to define the radius of the circle.

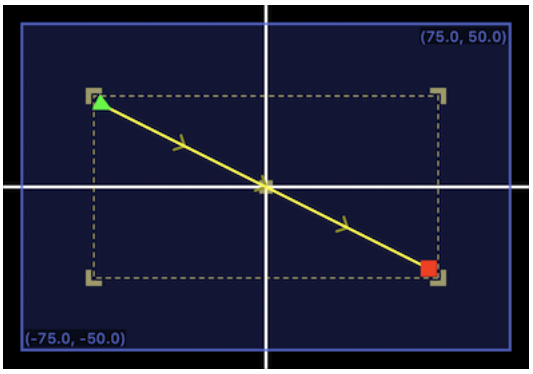


Flip vertical flips the selected group of points around the X axis.

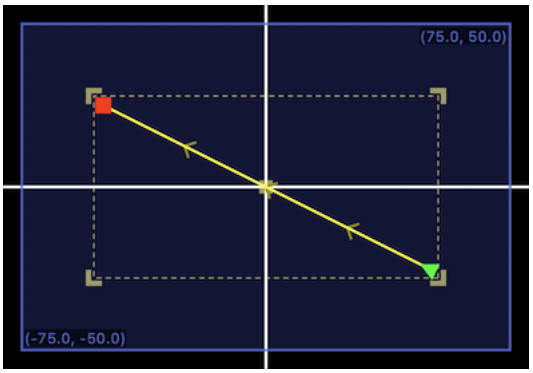
Flip horizontal flips the selected group of points around the Y axis.

Rotate spins the selected group of points in 90 degree increments around the center point of the selection.

Reverse leaves the points in the same location, but reverses their order in the path. For example, consider this simple path with two points:



After using the reverse tool, what was the start point is now the end point, and vice versa:



The ⓘ **help button** displays a list of the keyboard shortcuts and modifiers for working with 2D fades.

The **alignment** menu to the upper right of the path lets you choose where the axis is placed on the canvas. By default, it is placed at the center, both horizontally and vertically, corresponding with QLab's center origin point for video geometry. However, this menu allows you to place the axis in one of the corners to help achieve your mathematical goals. It has no functional difference beyond how the canvas is displayed, and therefore what sorts of values are easiest to enter.

To the right side of the canvas are a few more controls for configuring the canvas and the overall behavior of the path:

Width and **Height** allow you set the width and height of the blue rectangle, which defines the range of values sent by the cue.

Smooth path. When this box is un-checked, paths will be drawn precisely between each control point. When the box is checked, the path will be smoothed between points to create a naturally curving line.

You can use the ⊖ and ⊕ buttons to zoom in and out on the canvas. If you're using a trackpad, you can zoom in and out by placing the cursor over the canvas and pinching in and out. If you're using a mouse or trackball with a scroll wheel, you can zoom in and out by holding down the option key (⌥) and scrolling vertically.

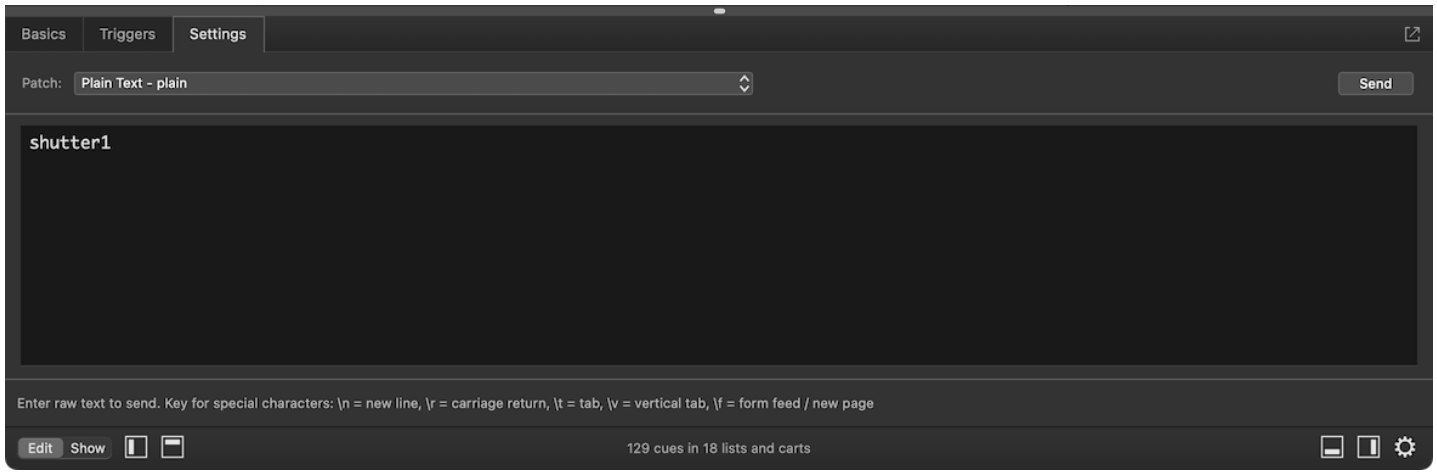
While zoomed in, you can pan around using two-finger swipes on a trackpad. If you have a mouse or trackball with a middle button (oftentimes the scroll wheel can be clicked; this is usually the middle button,) you can pan around my holding that button down.

The **Reset Points** button resets the path to the default shape defined by the cue template.

If **Live Preview** is on, QLab will transmit OSC messages as you click and drag so that you can see or hear the results in real time.

Plain Text Messages

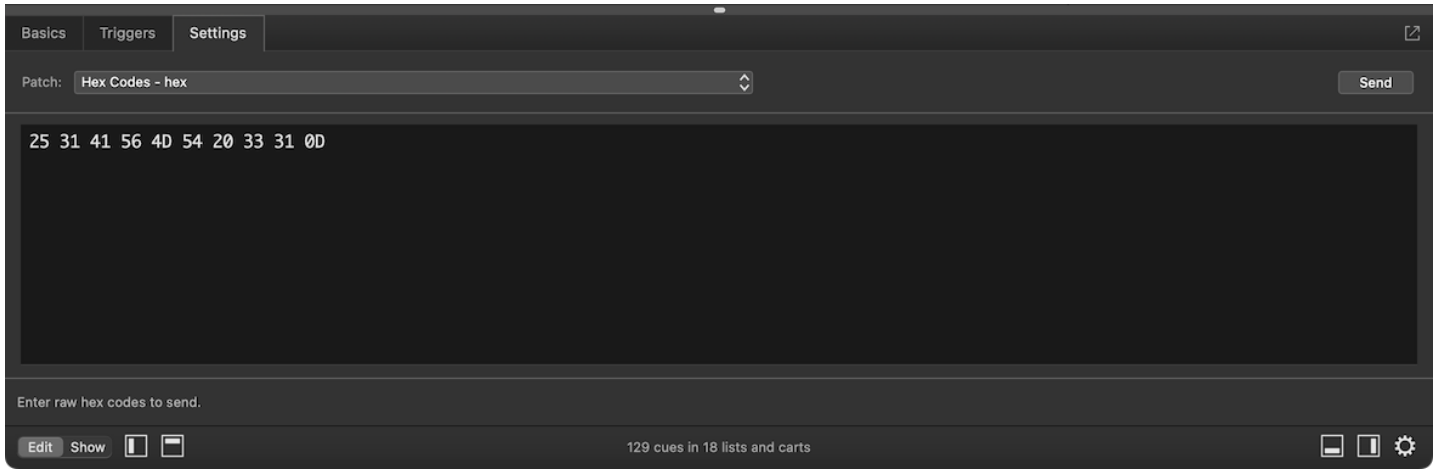
If the ⌘ Network cue uses a patch configured to send a Plain Text message, the lower part of the Settings tab shows a single, large text field.



Plain text messages cannot be given a duration, and so cannot be faded or resent. Only ASCII characters can be used in plain text messages.

Hex Code Messages

If the Network cue uses a patch configured to send a Hex Code message, the lower part of the Settings tab shows a single, large text field.



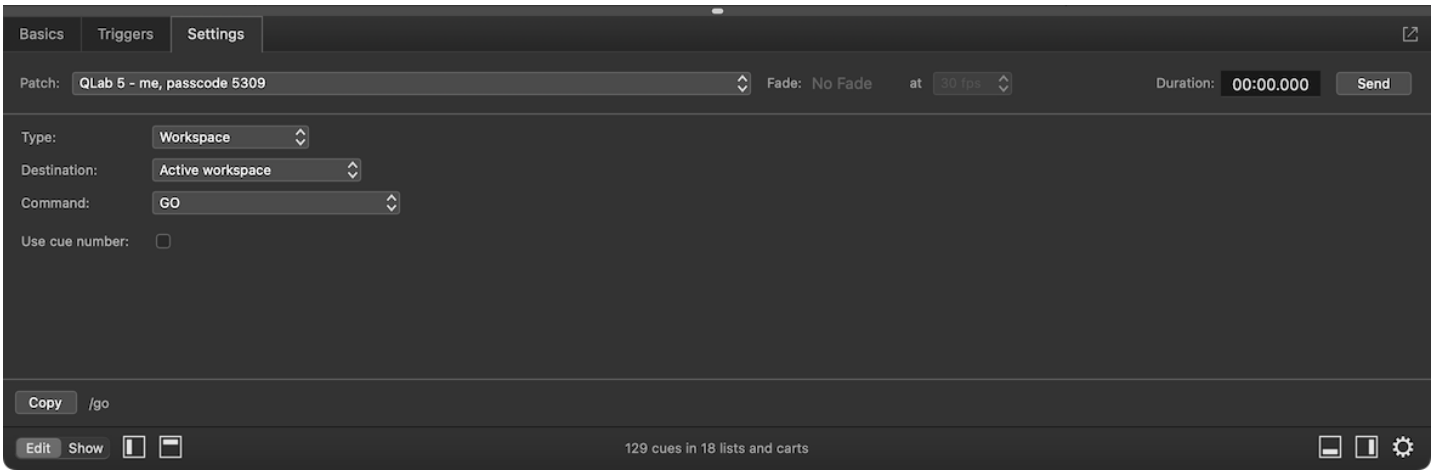
Plain text messages cannot be given a duration, and so cannot be faded or resent. Only hexadecimal symbols can be used in plain text messages; the numerals 0 through 9 and the letters A through F.

Network Device Description messages

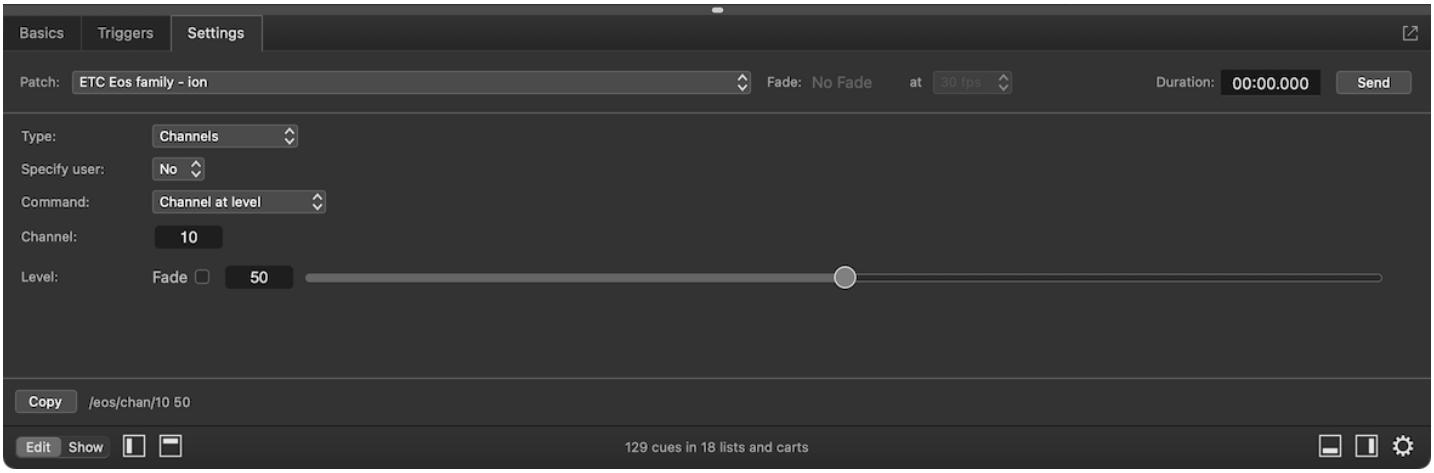
If the Network cue uses a patch configured with a network device description, the lower part of the Settings tab can show a variety of pop-up menus, text fields, checkboxes, sliders, 1D fade curves, and 2D fade curves. The specific controls that are available depend upon the network device description in use.

When using Network cues to communicate with another computer running QLab, or to send OSC messages from QLab back to itself, this option provides a very simple interface for accessing the commonly used OSC commands that QLab recognizes. Enter a cue number, and choose a command to send. Some commands have additional arguments, and additional fields will appear when those commands are chosen.

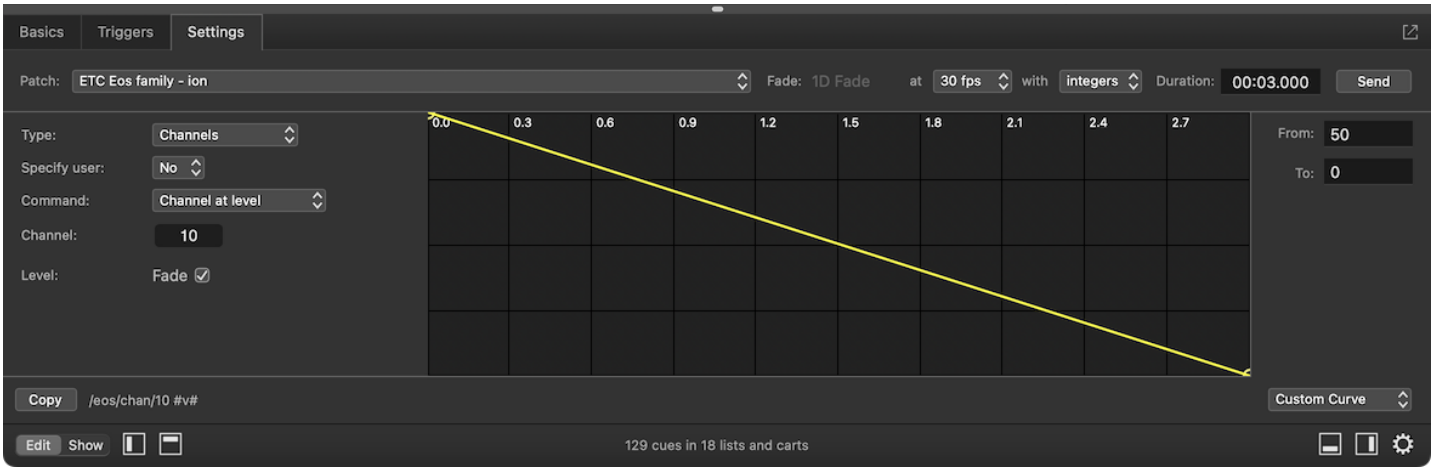
QLab 5 - Workspace GO



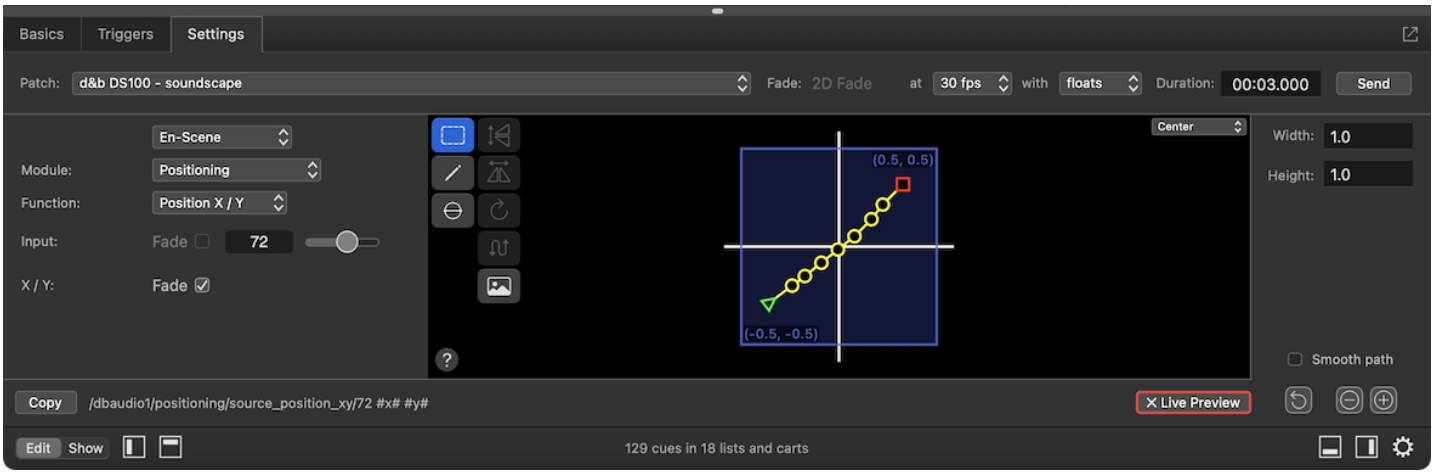
ETC EOS Family – Channel 10 at 50%



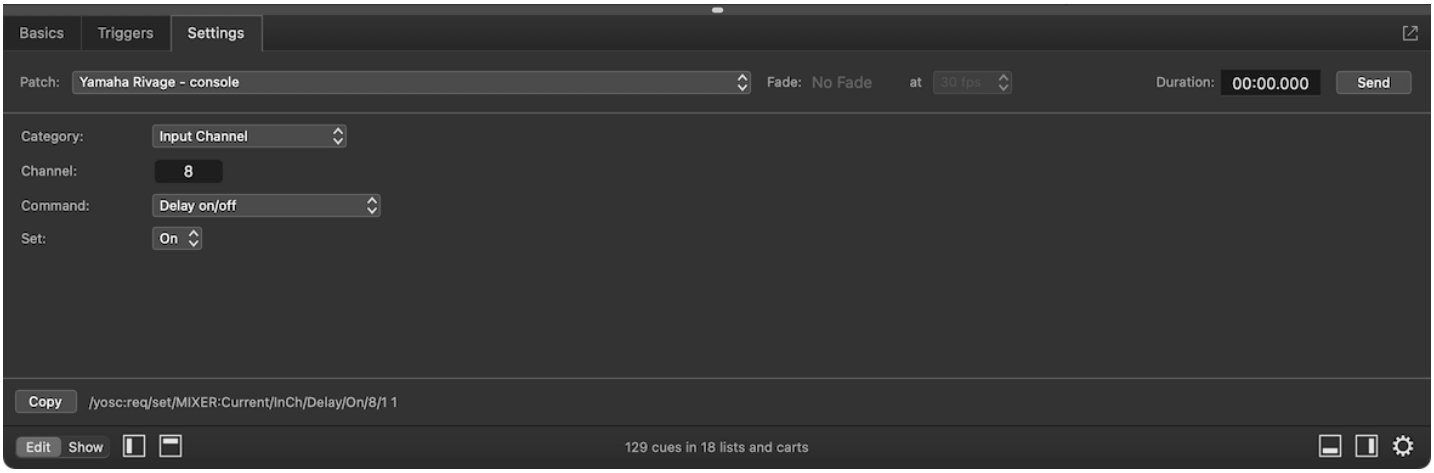
ETC EOS Family – Fade channel 10 from 50% to 0%



d&b DS100 (Soundscape) – Fade input 1 from downstage left to upstage right



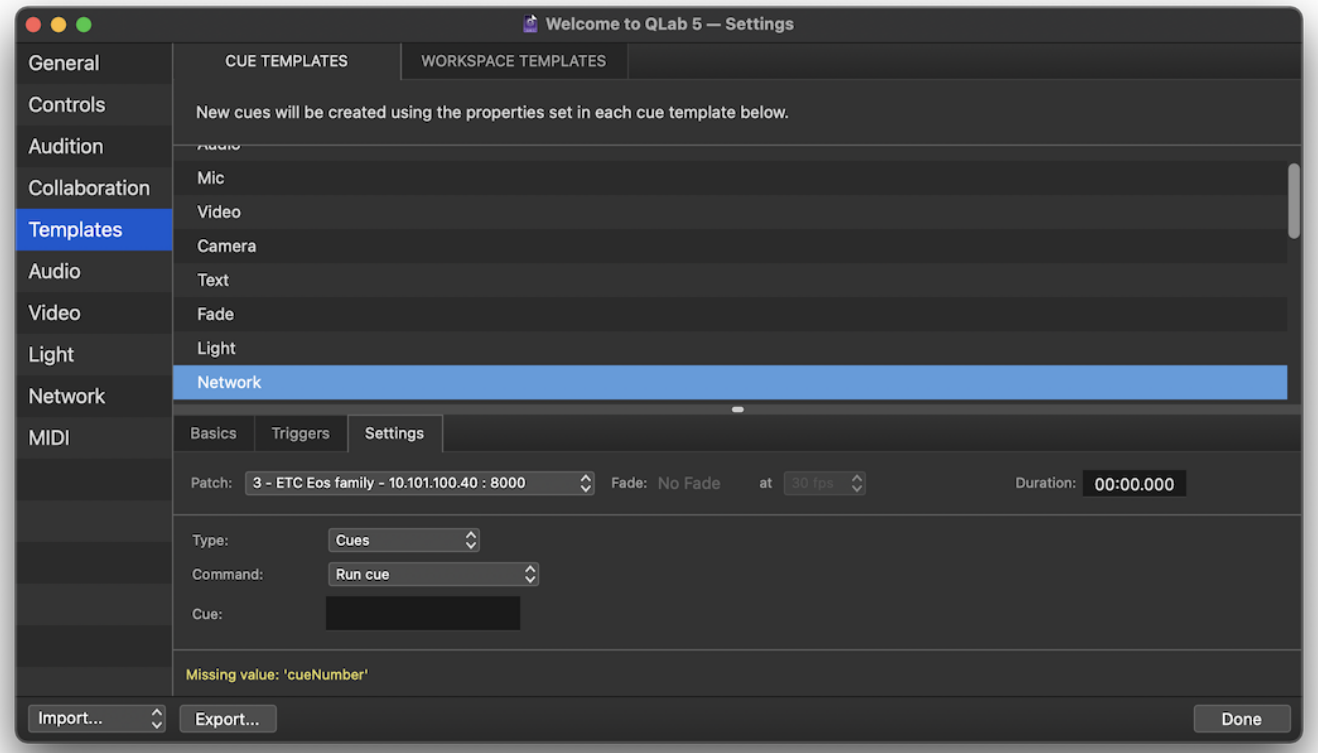
Yamaha Rivage - Enable delay on input channel 8



If you would like us to create a network device description for a specific network-controllable device or program, please contact us at support@figure53.com and tell us about it. We may be able to create a description, and if so we may be able to include it in a subsequent release of QLab.

Network Cues and Cue Templates

As discussed in the section on Cue Templates in this manual, cues can be given a customized default state. Newly created cues in a workspace come into existence using this default state. If, for example, most of the Network cues in your workspace will be used to start a cue on an ETC console, you might configure the Network cue template like this:



Then, every new Network cue would start off looking like that, and would need only a cue numbered to be entered in the Settings tab in order to be used.

However, if your QLab workspace communicates with several different OSC-controllable devices, say an ETC console, a d&b DS100, and another remote copy of QLab, you might find yourself wishing there were a way to make multiple cue templates. To address this need, follow these steps:

1. In Workspace Settings → Network → Network Outputs, set up a network patch for each device that your workspace will communicate with.
2. In Workspace Settings → Templates → Cue Templates, select the Network cue, click on the Settings tab, and choose one of the network patches.
3. Configure the rest of the Settings tab as you would like cues which use that patch to appear.
4. Now, select a different network patch and configure the Settings tab for the defaults that you want associated with that patch.
5. Repeat for each network patch in your workspace.
6. Finally, select whichever network patch you plan to use most often, and leave the cue template set to use that patch.

Now, newly created Network cues will appear using that patch and all its associated settings, but if you switch the cue to another patch, the default settings you created in the cue template will suddenly appear. In this way, you can capture the default state of each type of Network cue that you plan to use.

Broken Cues and Warnings

Network cues can become broken or warned for the following reasons:

No Patch

The cue has no network patch assigned. Assign a network patch to clear this warning.

Incomplete network patch

The cue has a network patch assigned, but something is wrong with it. The network patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the network patch or select a new network patch to clear

this warning.

Missing message.

❌ Fill in a valid message in the Settings tab of the inspector to clear this warning.

Missing values.

❌ The cue uses a network device description and one or more parameters have no value. Fill in values for every control in the Settings tab of the inspector to clear this warning.

Invalid OSC message.

❌ The cue uses a patch configured to send OSC, but the OSC messages is malformed. Edit the OSC message in the Settings tab of the inspector to clear this warning.

Invalid hex codes.

❌ The cue uses a network device description which sends hexadecimal values, but the cue contains at least one character that is invalid. Edit the message in the Settings tab of the inspector to clear this warning.

Missing image file.

⚠ This non-breaking warning will appear on a Network cue which had a 2D fade background image assigned which QLab is not able to locate. Replace the image or click **Remove Background** in the Settings tab of the inspector to clear this warning.

External network output disabled.

⊘ This non breaking warning will appear on a Network cue whose patch directs messages to some other device on the network when external network output is disabled via Override Controls. Re-enable external network output to clear this warning.

Local network output disabled.

⊘ This non breaking warning will appear on a Network cue whose patch directs messages to QLab or to other software on the local Mac when local network output is disabled via Override Controls. Re-enable local network output to clear this warning.

License required

❌ A license of any kind must be installed to use Network cues.

---

1. All trademarks are the property of their respective owners, and their inclusion in this manual and in QLab does not represent any official relationship between Figure 53, LLC and any other entity. The inclusion of a particular device or program should not be construed as an endorsement, and omission of a device or program should not necessarily be construed as an admonishment. Every effort has been made to ensure that these network device descriptions are accurate and functional, but it is impossible to make a complete guarantee since manufacturers may change their devices or command libraries without notice. ↩

MIDI Cues

MIDI cues allow you to send MIDI voice messages, MIDI Show Control (MSC) messages, or MIDI System Exclusive (SysEx) messages. MIDI cues have no target and, by default, no duration since they send a single message more or less instantaneously. However, certain types of MIDI messages can be set to fade from one value to another over time, and when sending those messages, MIDI cues can have a duration.

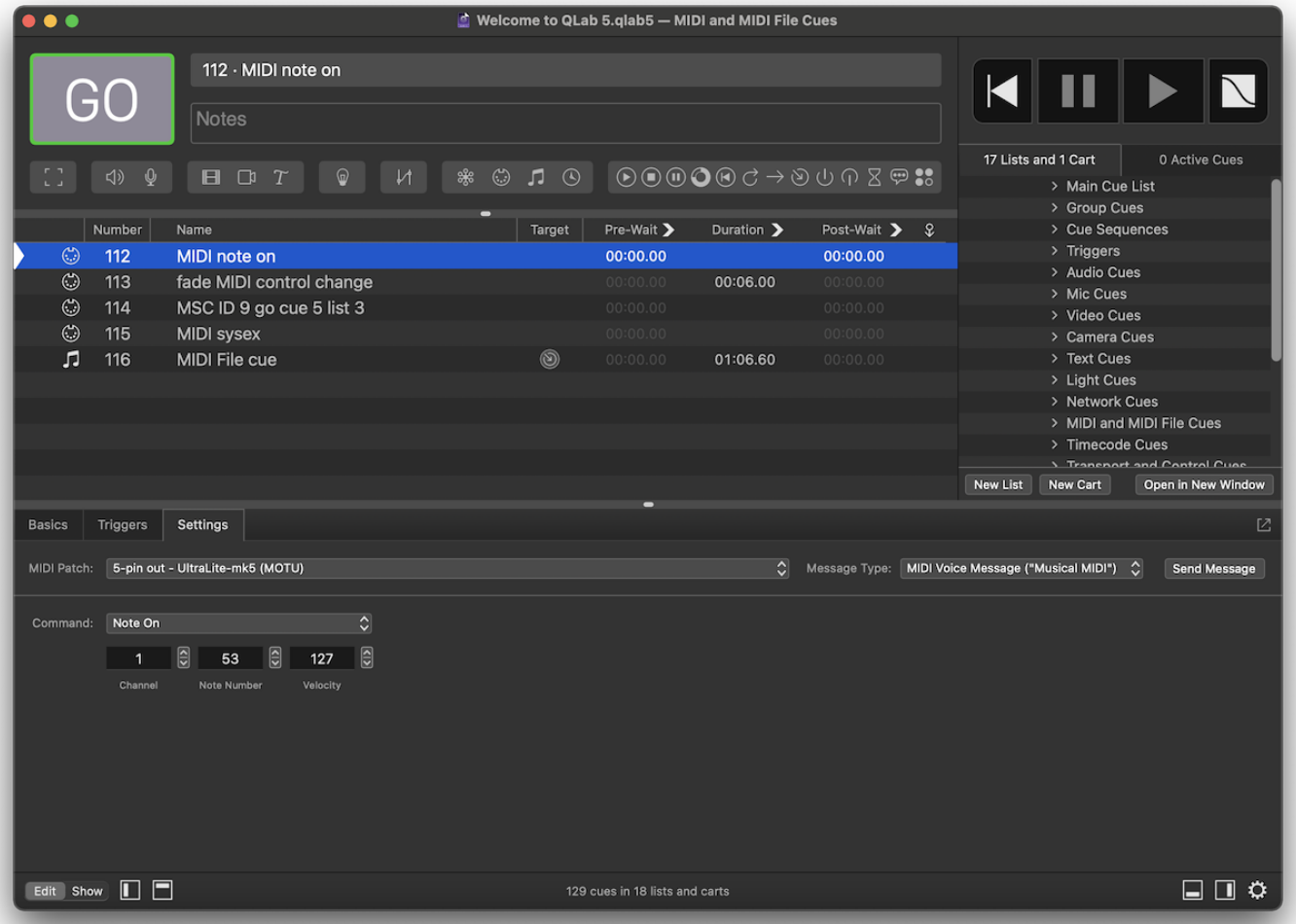
MIDI cues require a license of any kind.

The Inspector for MIDI cues

When a MIDI cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, and a Settings tab.

The Settings Tab

Along the top of the Settings tab are three controls which are present for all MIDI cues. Everything below those three controls changes depending upon the type of message the cue sends.



MIDI Patch. This pop-up menu allows you to select a MIDI patch for the cue to use. Clicking on the menu allows you to select one of the MIDI patches already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open MIDI*

Settings to edit patch list... to quickly get to Workspace Settings → MIDI, or choose *New patch with MIDI device* to quickly generate a new MIDI patch and select it for use.

Message Type. This pop-up menu lets you choose among MIDI Voice Message, MIDI Show Control Message, and MIDI SysEx Message.

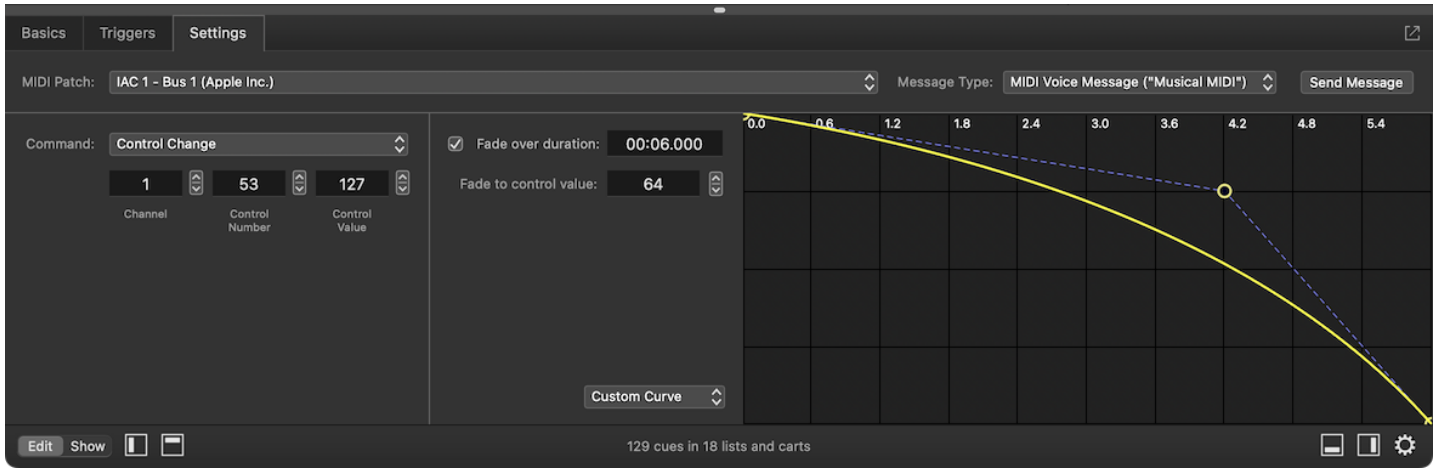
Send Message. Click this button to test-send your message.

MIDI Voice Message ("Musical MIDI")

When the message type is set to MIDI Voice Message, the following controls appear:

Command. QLab can send the following types of MIDI Voice commands: Note On, Note Off, Program Change, Control Change, Key Pressure (Aftertouch), Channel Pressure, and Pitch Bend Change. All commands require a channel, but the other controls available will vary depending on the type of command selected.

Control Change, Key Pressure, Channel Pressure, and Pitch Bend commands can be faded from one value to another over time. If you select any of these options, a set of controls will appear to enable fading.



To fade a message, check the box marked *Fade over duration* and enter the desired duration in the field provided.

You can also adjust the curve of the fade using the graph on the right. The pop-up menu allows you to choose between a custom curve shape and a linear curve shape, and lets you reset the curve to its default state.

Because MIDI cues have no way of knowing the current value of the parameters that they are controlling, you need to enter both a starting value, which is the field labeled *Value* underneath the Command drop-down menu, and an ending value, which is the field labeled *Fade to value* towards the middle of the inspector.

MIDI Show Control Message (MSC)

When the message type is set to MIDI Show Control Message, the following controls appear:

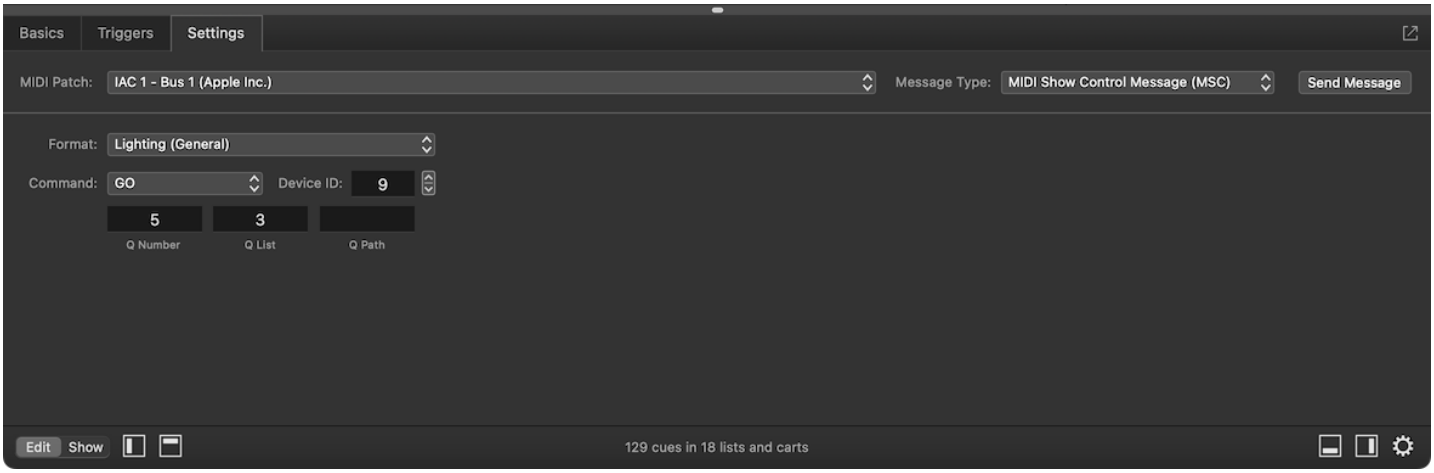
Command Format. The MSC spec defines a number (a surprisingly large number, really) of specific categories within which devices or software can self-identify. Choose the category for the receiving device here. Some devices belong to more than one category; QLab itself responds to messages in the Sound (General), Video (General), and Lighting (General) categories. Refer to the documentation of the device that you're sending messages to in order to learn which category to use.

Command. Select the MSC command you wish to send here.

Device ID. Enter the MSC device ID number of the receiving device here. Device ID `127` is the "all-call" ID, and all MSC devices on your MSC network will respond to the message.

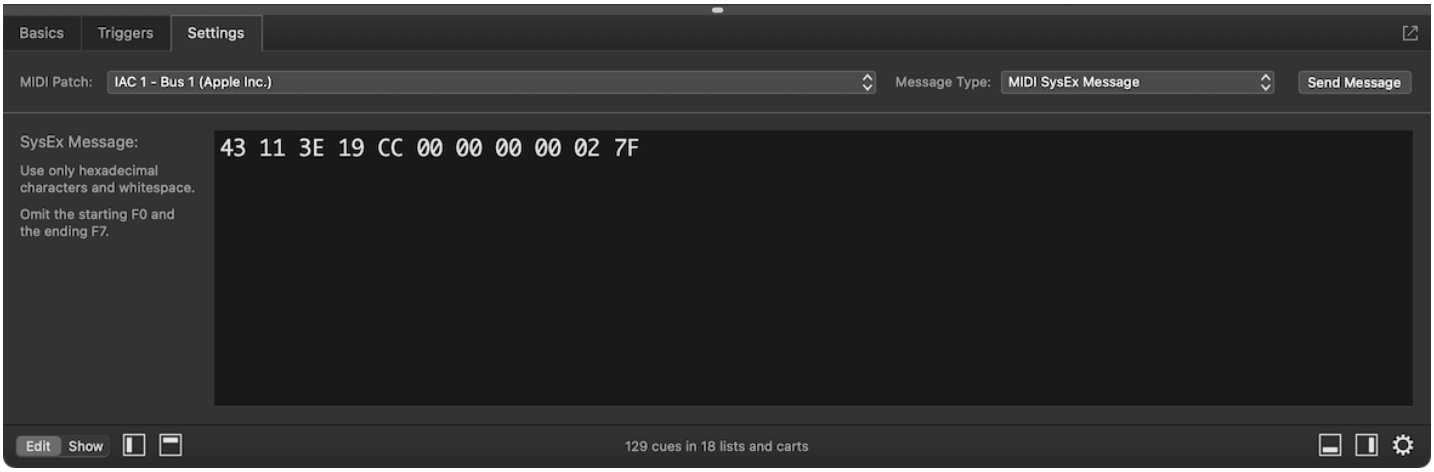
Q Number, Q List, and **Q Path** should be filled in according to the needs of the receiving device. If you're having trouble in this area, there are two quick guidelines that can help:

1. If you're sending MSC to an ETC Eos family console, leave *Q List* blank if the cue you're starting is in the active cue list. Only fill it in if the cue you're starting is in a different cue list.
2. As far as we've seen, very few devices or programs use *Q Path*. It's usually best to leave it blank.



MIDI SysEx Message

When the message type is set to MIDI SysEx Message, a single large text entry field appears. Enter your SysEx message here in hexadecimal format. Omit the leading `F0` and `F7`; QLab adds those for you.



This SysEx message, as an example, sets the fader level of input channel 1 on a Yamaha CL series console to `-INF`.

Broken MIDI Cues

MIDI cues can become broken for the following reasons:

No MIDI patch

The cue has no MIDI patch assigned. Assign a MIDI patch to clear this warning.

Incomplete MIDI patch

The cue has a MIDI patch assigned, but something is wrong with it. The MIDI patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the MIDI patch or select a new MIDI patch to clear this warning.

Invalid SysEx message length

Edit the SysEx message in the Settings tab of the inspector.

Invalid characters in SysEx message

Edit the SysEx message in the Settings tab of the inspector.

License required

MIDI cues require a license of any kind.

MIDI File Cues

♫ MIDI File cues allow you to play a MIDI file containing a sequence of MIDI events to a single MIDI output. While QLab does not have a native concept of tempo and meter maps, ♫ MIDI File cues provide a way to create individual timelines on which events occur with tempo- and meter-based timings. This can be useful, for example, when synchronizing MIDI-triggered events with a piece of music.

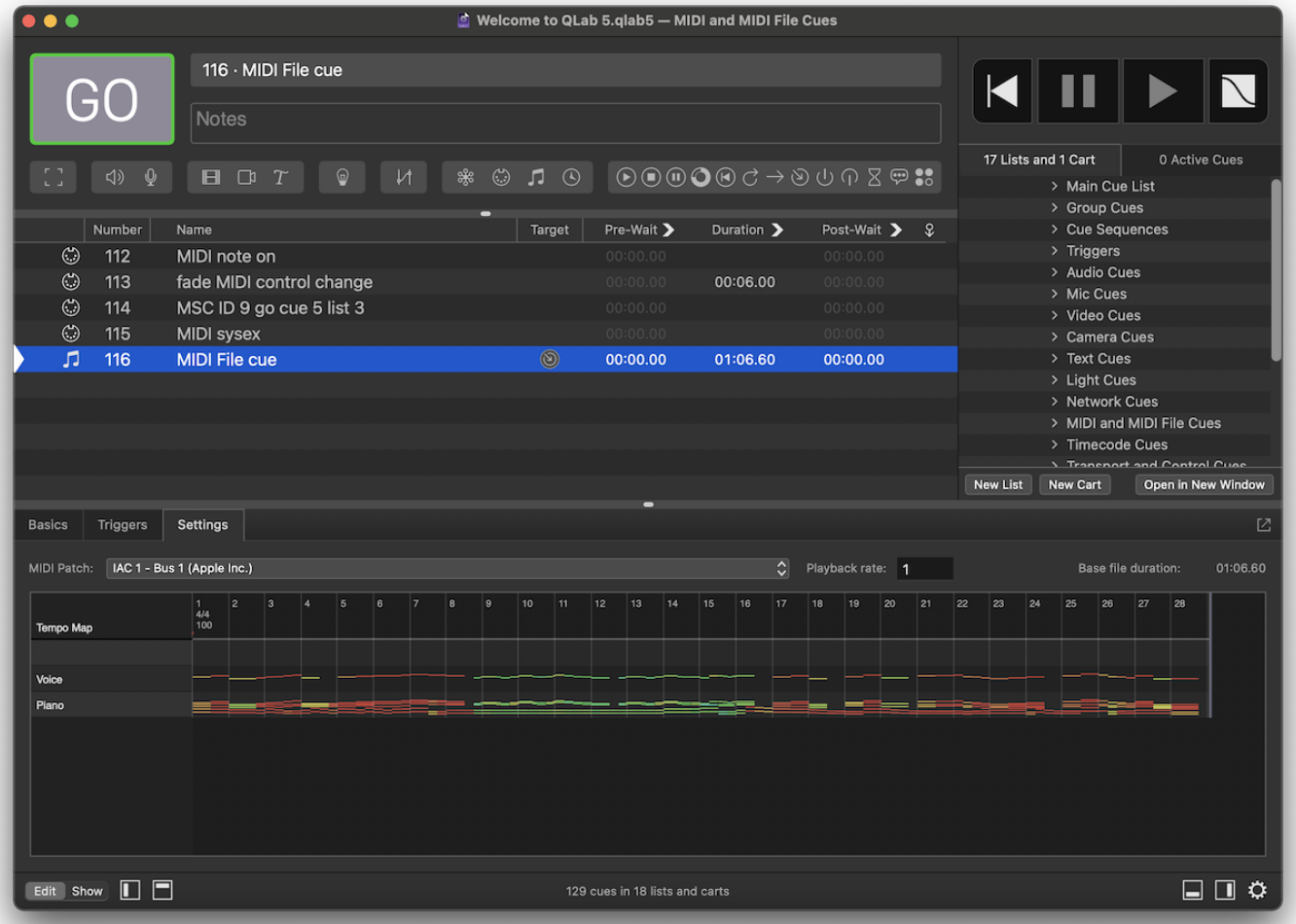
♫ MIDI File cues require a license of any kind.

MIDI Files

♫ MIDI File cues support both Type 0 (single-track) and Type 1 (multitrack) standard MIDI files. Type 2 files (an uncommon multiple-sequence format) are not supported.

The Inspector for MIDI File cues

When a ♫ MIDI File cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, and a Settings tab.



The Settings Tab

MIDI Patch. This pop-up menu allows you to select a MIDI patch for the cue to use. Clicking on the menu allows you to select one of the MIDI patches already configured in the workspace, or *(unpatched)* if you want to ensure that the cue does not play when started. You can also choose *Open MIDI Settings to edit patch list...* to quickly get to Workspace Settings → MIDI, or choose *New patch with MIDI device* to quickly generate a new MIDI patch and select it for use.

Playback rate. This field is a multiplier for all tempi in the MIDI file. A rate of `0.5`, for example, will result in half-speed playback.

The 🎵 MIDI File cue supports any number of tracks, and events can occur on any MIDI channel (1-16), but all messages in the file are sent to a single MIDI port. This can be a physical port connected to a tone generator, lighting board, or other gear, or a software MIDI destination such as ipMIDI. It can also be an IAC bus, which allows MIDI events to loop back into QLab and start other cues, or to route to another application running on the same computer.

🎵 MIDI File cue timings are based on the computer's internal clock, and are not clocked to any audio or video devices.

Broken MIDI File Cues

🎵 MIDI File cues can become ❌ broken for the following reasons:

No MIDI file selected

The cue has no target, and 🎵 MIDI File cues require a MIDI file as a target. Select a MIDI file as the target of this cue to clear this warning.

Missing MIDI file

The cue had a target file assigned, but that file is missing. Perhaps the file was on a removable drive or network drive which is not currently connected. Re-locate the target file, or assign a new target file to clear this warning.

Invalid MIDI File.

Either the file is missing or damaged, or it's not a valid MIDI file. assign a new target file to clear this warning.

File target in Trash

The cue's target file is in the Trash, and files in the Trash cannot be used. Either move the target file out of the Trash, or assign a new target file to clear this warning.

No MIDI patch

The cue has no MIDI patch assigned. Assign a MIDI patch to clear this warning.

Incomplete MIDI patch

The cue has a MIDI patch assigned, but something is wrong with it. The MIDI patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the MIDI patch or select a new MIDI patch to clear this warning.

License required

🎵 MIDI File cues require a license of any kind.

Timecode Cues

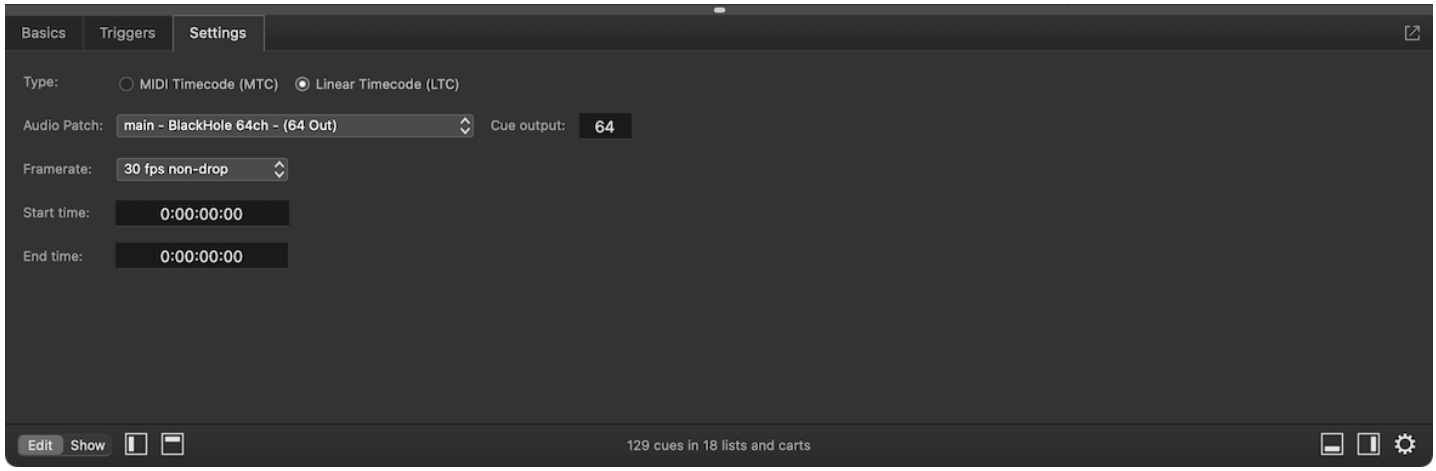
Timecode cues allow you to generate timecode and send it out of QLab to be received by other software or devices. QLab supports sending both LTC (Linear or Longitudinal Timecode) and MTC (MIDI Timecode), and unlike most timecode-enabled applications which have a single, fixed timeline from which timecode can be generated directly, QLab allows you to generate as many simultaneous, independent timecode streams as you need, each with its own configuration. Timecode cues require a license of any kind.

The Inspector for Timecode Cues

When a Timecode cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as a single Settings tab.

The Settings Tab

The Settings tab allows you to configure the type of timecode that the cue will send, its routing, its frame rate, its starting frame, and (optionally) its ending frame.



Type. Timecode cues can generate either MIDI Timecode (MTC) or Linear Timecode (LTC). Depending on which type you select here, the inspector will show different output options.

MTC (MIDI Timecode) is a stream of constantly changing MIDI data. When MTC is selected, the **MIDI Patch** pop-up menu allows you to select a MIDI output patch to use. No channel number is required because MTC messages are not associated with a MIDI channel.

> Sending MTC over a network is discouraged, as the inconsistent latency of a network connection can often render the timecode signal inaccurate, or even unreadable, on the receiving end. Inexpensive MIDI interfaces often create the same problem, even those that work well for normal musical MIDI. Always use a reliable, high-quality MIDI interface when working with MTC.

LTC is an audio signal, meant to be carried on a line-level audio connection. When LTC is selected, the **Audio Patch** pop-up menu allows you to select an audio output patch to use, and the **cue output** field allows you to enter the number of the output channel to use. The channel number defaults to `0` as a safety measure; `0` is not a valid channel, so you need to proactively specify the output channel you want to use.

> **Warning:** Use caution when setting up the routing of an LTC signal, from QLab or any source. LTC signals are meant to be run at fairly high levels and have strong high-frequency content which, if accidentally routed out to speakers, can cause serious damage to hearing and to friendships.

The **Framerate** pop-up menu behaves the same way with both MTC and LTC. QLab supports both **video speed** and **film speed** options at all common framerates. Be sure to match the framerate on the receiving end precisely.

Broadly speaking, "film speed" refers to the timecode formats with integer framerates (24, 25, 30 drop, 30 non-drop), while "video speed" refers to their non-integer counterparts (23.976, 24.975, 29.97 drop, and 29.97 non-drop).

Each video speed framerate is an identical timecode format to its film speed counterpart, but pulled down by 0.1%. For example, one frame of 29.97 non-drop timecode consists of the same data as the same frame of 30 non-drop, but played at a 0.1% slower rate. Neither LTC nor MTC differentiates between video speed and film speed in how the bits are encoded, so timecode at the wrong speed will initially appear correct on the receiving end. However, the timecode will drift noticeably over time from what is expected unless the speeds match.

Timecode cues are clocked differently depending on the type selected. LTC follows the clock of the audio device to which it outputs, and is guaranteed not to drift from that clock. MTC, on the other hand, follows the computer's internal clock. Under normal use, drift between high-quality devices is usually minimal, but if drift-free synchronization with another machine is required over long stretches of time, the best option is to output LTC to an audio device that can resolve to the same audio clock (for example, via a word clock connection or something similar) as the other machine.

Start time. This field allows you to specify the first frame of timecode that is transmitted when the cue is started. Bear in mind that devices listening to timecode sometimes require a few frames to be received before they sync up. If an event needs to be triggered on a specific frame, it is best to start timecode output at least a few frames earlier as a preroll into that event. Because there is no room for preroll before `0:00:00:00`, the best practice is to treat `1:00:00:00` as the beginning of the timeline and use hour `0` for preroll.

End time. You can optionally specify the final frame of timecode that is to be transmitted by this cue. If you do not enter an ending frame, this field will show the same frame as the start time field. You can also edit the duration of the cue in the Basics tab of the inspector, or in the *Duration* column in the cue list, and this field will update itself correspondingly.

If you set an end time or duration, the Timecode cue will stop itself once the duration elapses. If you do not set a duration, the Timecode cue will run until stopped.

Monitoring Timecode

Outgoing and incoming timecode can both be monitored using the *Timecode* window which can be found in the **Window** menu. Note that if you have multiple simultaneously playing Timecode cues, only the most recently started cue will display in the Timecode window. Fear not; the other timecode is still running.

Broken Timecode Cues

Timecode cues can become broken for the following reasons:

No MIDI patch

The cue has no MIDI patch assigned. Assign a MIDI patch to clear this warning.

Incomplete MIDI patch

The cue has a MIDI patch assigned, but something is wrong with it. The MIDI patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the MIDI patch or select a new MIDI patch to clear this warning.

No audio output patch

The cue has no audio output patch assigned. Assign an audio output patch to clear this warning.

Issue with audio output patch

The cue has an audio output patch assigned, but something is wrong with it. The audio output patch will have warnings of its own, listed in the Warnings tab of the Workspace Status window, which you can use to figure out what's wrong with it. Either fix the audio output patch or select a new audio output patch to clear this warning.

Invalid output channel.

The cue has a valid audio output patch, but an invalid cue output. Select a valid cue output number to clear this warning.

License required

Timecode cues require a license of any kind.

Chapter 9: Scripting

9.1 OSC Dictionary
9.2 OSC Queries
9.3 Script Cues
9.4 AppleScript Dictionary
9.5 Parameter Reference
9.6 OSC & Scripting Examples

OSC Dictionary

QLab has an extensive API (application programming interface) for OSC which allows you to control QLab from any device or software which can broadcast OSC messages. What follows here is a complete dictionary of QLab 5's OSC implementation. OSC messages are sorted more or less alphabetically in each section, which is to say they're alphabetically sorted, but messages which are closely related to each other are sometimes grouped together even if they are not alphabetically adjacent if that makes it easier to understand their relationship.

Getting Started

The QLab OSC API can be used over both UDP and TCP transport layers. QLab listens for incoming OSC on port 53000. Individual workspaces can be configured to listen for incoming OSC on any port in the Network → OSC Access section of Workspace Settings, although it's important to remember that QLab itself still listens on port 53000.

When a client talks to QLab via UDP, each OSC message corresponds to one UDP datagram. Replies to OSC messages sent via UDP are sent on port 53001 by default, but can be customized using `/udpReplyPort`; see below.

When a client talks to QLab via TCP, messages are framed using the double end SLIP protocol (RFC 1055) as required by the OSC 1.1 specification.

QLab also listens for plain text on UDP port 53535, and attempts to interpret it as OSC. For example, sending the text `/cue/selected/start` to QLab on UDP port 53535 will have the same result as sending the actual OSC command `/cue/selected/start` to port 53000. The plain text UDP port can also be customized on a per-workspace basis, just like the OSC port.

The OSC API behaves almost identically when using both UDP and TCP. Exceptions are noted below, such as cases where a reply may be larger than the maximum size of a UDP datagram.

Important: OSC messages which arrive on a given port will be received by *every* open workspace listening to that port. This is a change from QLab 4 in which only the front-most workspace received OSC messages. If you are using multiple workspaces simultaneously and wish to route OSC messages selectively, you can use different port numbers for each workspace, prefix messages with `/workspace/{id}`, or prefix messages with `/workspace/{name}` (discussed below.)

Three Ways To Use OSC With QLab

OSC can be used in essentially three ways with QLab: as a relatively simple "remote control" protocol in the spirit of MIDI, as a robust two-way protocol for tight integration with other systems, and (as of QLab 5.3) as a show control broadcast method.

Those readers interested in using OSC for simple remote control can skip ahead to the section titled "How To Read This Dictionary." Those only interested in show control messaging should read the Show Control Broadcast section of this manual, although the OSC messages relating to show control broadcast are also listed on this page for completeness.

For those who are interested in using OSC to talk to QLab and getting answers back, and then doing things with those answers, read on here.

Getting status updates from QLab

When a client has requested status updates, that client will receive messages from QLab whenever the client needs to update its knowledge of a cue, cue list, or workspace.

Clients who have requested status updates might receive the following messages at any time:

- `/update/workspace/{workspace_id}` - the client needs to reload the cue lists for the workspace. This message is also sent whenever various other aspects of a workspace are updated.
- `/update/workspace/{workspace_id}/cue_id/{cue_id}` - the client needs to reload the state for the specified cue. If the cue is a Group cue or cue list, the client should also reload the children of that cue.
- `/update/workspace/{workspace_id}/cueList/{cue_list_id}/playbackPosition {cue_id}` - the playback position of `cue_list_id` has changed to `cue_id`. If there is no current playback position, there will be no `cue_id` argument.
- `/update/workspace/{workspace_id}/disconnect` - the client must disconnect from the given workspace (e.g. because it is closing.)

To receive status updates from QLab, send the following OSC command:

```
/updates 1
```

To stop receiving updates, send:

```
/updates 0
```

The `/updates` message can be used by clients with any level of access permission.

Replies from QLab

Most, but not all, OSC messages sent to QLab will result in a reply being sent back to the client who sent the message. This is separate from the idea of updates, as discussed above, which are sent proactively by QLab. Replies are only sent in response to incoming OSC messages.

Messages which perform discrete actions in QLab, like `/go` and `/panic`, do not generate replies by default, although you can request a reply for every message using the `/alwaysReply` command, discussed below.

Replies from QLab take the form:

```
/reply/{/invoked/osc/method} json_string
```

`json_string` takes the form:

```
{
    "workspace_id" : string,
    "address": "/osc/message/that/was/sent",
    "status": string,
    "data": value
}
```

`workspace_id` is optional, and only present if the reply is specifically from the given workspace, rather than from QLab as a whole.

`status` can be:

- `ok` - the OSC message was received and everything is good.
- `error` - the OSC message was malformed, invalid, or something else has gone wrong.
- `denied` - the client has not yet successfully connected to QLab and needs to do that before sending this OSC message, or the client is connected using a passcode that does not have suitable access privileges to send this OSC message.

`data` is the JSON-encoded result of the OSC message that was sent.

Show control broadcast messages **are not sent using this reply format.** They are sent as standard OSC messages, without any JSON encoding.

Example

Sending a workspace `cueLists` method:

```
/workspace/34200B51-835A-4918-A137-B6511784B6CA/cueLists
```

would cause QLab to respond with:

```
/reply/workspace/34200B51-835A-4918-A137-B6511784B6CA/cueLists {json_string}
```

where `{json_string}` would be a list of the cue lists in the workspace.

How To Read This Dictionary

Every OSC message that QLab will respond to is listed in this dictionary. Each definition starts with a horizontal separator followed by the OSC message itself written out like so:

/cue/{cue_number}/preWait {number}

The parts that are enclosed in {braces} are the parts which you have to fill in to make the message work, and the dictionary tries to give you clues about what sort of thing you'll need to fill it with. For example, in the message above, you need to replace `{cue_number}` with the cue number of the cue you want to talk to, and you need to replace `{number}` with a number of some kind. This OSC message uses the number you fill in to set the pre-wait of the cue that you specify.

Next comes a table which looks like this:

view	edit	control	query	+/-?	Live
read	read/write	read only	✓	✗	edit, control

View, Edit, Control

The first three columns in this table describe the degree of access to this message for clients with view, edit, and control permission.

- *Read* means that a client can send the message without arguments and get data back from QLab. For example, `/cue/10/preWait` will return the pre-wait of cue `10`.
- *Read/write* means that a client can read data from QLab, and can also send data to QLab to make changes. For example, `/cue/10/preWait 5` will set the pre-wait of cue `10` to 5 seconds.
- *Read only* means that this message can only be used to read; there is no "write" form.
- ✓ means that a client can send this message. This is used only for messages which are "actions" such as `/go` and `/panic`, which neither read nor write data.
- ✗ means that a client cannot send this message (well, it can send it, but QLab will ignore it.)

Query

The fourth column shows whether or not this message can be used by QLab's internal OSC query mechanism.

+/-?

The fifth column is only present for OSC messages which are directed at cues, and it shows whether or not the message can be used with QLab's **increment/decrement** syntax:

- `/cue/x/property/+ {delta}` adds `{delta}` to the current value.
- `/cue/x/property/- {delta}` subtracts `{delta}` from the current value.

Starting with QLab 5.5, you can optionally send `{delta}` as part of the OSC address, instead of as an argument.

For example:

- `/cue/10/preWait/+ 1`
- `/cue/10/preWait/+/1`

These two messages are equivalent. Sending either one increases the preWait of cue `10` by one second.

Starting with QLab 5.5, OSC messages which accept +/- syntax to move through a finite list of options will "roll over" if you add or subtract beyond the scope of the list.

For example, `/cue/{number}/continueMode/+ 1` will cycle from "Auto-follow" back to "Do not continue", etc.

Live

The last column is only present for OSC messages which have a **live** form. "Live" messages are the same as their non-live counterparts but operate on the active, "live" value of a parameter rather than changing the saved initial state of the parameter. Sending these messages does not cause the document to have unsaved changes.

For example, `/cue/1/rate 2` sets the playback rate of cue 1 to `2`, but `/cue/1/rate/live 2` sets the *live* playback rate of cue 1 to `2`. When cue 1 (or the whole workspace) is reset, the live adjustment will vanish and cue 1 will revert to its saved rate.

To use the live form of an OSC message, add `/live` as the very last part of its address.

+/- and Live together

Live goes last, always.

- `/cue/x/opacity/live/+ 10` - this will **not** work.
- `/cue/x/opacity/+/live 10` - this **will** work.

After the table comes a description of the behavior of the OSC message, including separate explanations for *read* and *write* usage if applicable.

Finally, some OSC messages come with examples showing how to use them. Readers are heartily encouraged to request that specific additional examples be added by emailing the QLab support team.

OSC Booleans in QLab

Many OSC messages in QLab require an argument which sets a value to either true or false. QLab 5 allows several different data types for these arguments. All of the following are valid:

- **Booleans.** OSC 1.1 has a boolean data type, allowing you to send `True` or `False` as an argument.
- **Integers** or **Floats.** If QLab receives any number as an argument where it's expecting a true or false value, `0` will be interpreted as false, and any other number (including, for example, `0.5`) will be interpreted as true.
- **Strings.** If QLab receives a text string as an argument where it's expecting a true or false value, any string which begins with `N`, `n`, `F`, `f`, or the digit `0` will be interpreted as false. Any string which begins with `Y`, `y`, `T`, `t`, or any digit `1` through `9` will be be interpreted as true.

Examples

The following messages will all flag cue `12`:

- `/cue/12/flagged Yes`
- `/cue/12/flagged yippee`
- `/cue/12/flagged "you betcha"`
- `/cue/12/flagged 1`
- `/cue/12/flagged 1.0`
- `/cue/12/flagged true`

The following messages will all un-flag cue `12`:

- `/cue/12/flagged No`
- `/cue/12/flagged never`
- `/cue/12/flagged "forget it"`
- `/cue/12/flagged 0`
- `/cue/12/flagged 00`
- `/cue/12/flagged false`

Starting with QLab 5.2, any OSC message in QLab's dictionary that accepts a boolean argument can also accept the argument `toggle` or append `/toggle` to the end of their address. Doing this will toggle the current value.

Example

- `/cue/12/flagged/toggle` or `/cue/12/flagged toggle` will flag cue `12` if it is currently un-flagged, or un-flag it if it is currently flagged.

Application Messages

The following OSC messages pertain to QLab as a whole, not to a specific workspace. For security, however, if every open workspace requires a passcode, then a client must connect to at least one of those workspaces with the passcode before any application messages will be accepted.

There are two exceptions to this rule: `/version` and `/workspaces` will always be accepted, even without a passcode.

/alwaysReply {number}

view	edit	control	query
read/write	read/write	read/write	✓

By default, QLab will only reply to an incoming OSC message if that message generates a reply to send. For example, `/go` does not generate a reply.

Read: If `number` is not given, return the `alwaysReply` status for the sending client.

Write: If `number` is given and is not zero, send a reply for every OSC message received from the client. Messages that would not normally generate a reply will generate one with a JSON string argument that contains:

```
{
  "workspace_id" : {string},
  "address": "/osc/message/that/was/sent",
  "status": {"ok" or "error"}
}
```

If `number` is given and is `0`, stop sending replies to messages that do not generate replies.

/disconnect

view	edit	control	query
✗	✓	✗	✗

Disconnect from QLab. Clients should send this message when they will no longer be sending messages to QLab.

If you are communicating with QLab via UDP, QLab will automatically disconnect your client if it has not heard any messages from it in the last 61 seconds. Any message (e.g. `/thump`) will serve to keep the client connected, or you can send `/forgetMeNot` or `/udpKeepAlive` (see below) to override this 61-

second timeout. If you are disconnected, you will need to reconnect before further commands will be accepted. If you are using a connection with a passcode, the passcode needs to be sent again, just as though you were connecting for the first time.

If you are communicating with QLab via TCP, QLab will not automatically disconnect your client, because TCP is nice like that. Clients will remain connected until they send `/disconnect` or until the TCP connection itself is disconnected.

/fontNames

view	edit	control	query
read only	read only	read only	✕

Return an array of the names/PostScript names of all available fonts. For example:

```
[
  "AppleColorEmoji",
  "AppleSDGothicNeo-Bold",
  "AppleSDGothicNeo-ExtraBold",
  "AppleSDGothicNeo-Heavy",
  "AppleSDGothicNeo-Light",
  ...
]
```

/fontFamiliesAndStyles

view	edit	control	query
read only	read only	read only	✕

Return a dictionary with each available font family name (e.g. “Helvetica”, “Courier New”) paired with an array of its available styles (e.g. “Regular”, “Light Oblique”). For example:

```
{
  "Apple Color Emoji" :
    [
      "Regular"
    ],
  "Apple SD Gothic Neo" :
    [
      "Regular",
      "Medium",
      "Light",
      "UltraLight",
      "Thin",
      "SemiBold",
      "Bold",
      "ExtraBold",
      "Heavy"
    ],
  ...
}
```

/forgetMeNot {boolean}

/udpKeepAlive {boolean}

view	edit	control	query
read/write	read/write	read/write	✗

Sending `/forgetMeNot` or `/udpKeepAlive` with a `true` argument will cause QLab to remember the client and all its settings (such as `/alwaysReply`) until QLab quits or until the client sends `/forgetMeNot` or `/udpKeepAlive` with a `false` argument. This allows a client to send a passcode, ask for specific replies, etc. only once at the beginning of a session, and not worry about being disconnected after 61 seconds of inactivity.

It is best practice to always send `/forgetMeNot` or `/udpKeepAlive` with a `false` argument when you're done, to allow QLab to clear its record of the now-inactive client.

/overrides/dmxOutputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the DMX output override.

Write: Set the DMX output override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleDmxOutput

view	edit	control	query
✗	✓	✓	✗

Enable or disable DMX output.

/overrides/midiInputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the MIDI input override.

Write: Set the MIDI input override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleMidiInput

view	edit	control	query
✗	✓	✓	✗

Enable or disable MIDI input.

/overrides/midiOutputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the MIDI output override.

Write: Set the MIDI output override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleMidiOutput

view	edit	control	query
✗	✓	✓	✗

Enable or disable MIDI output.

/overrides/mscInputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the MSC input override.

Write: Set the MSC input override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleMscInput

view	edit	control	query
✗	✓	✓	✗

Enable or disable MSC input.

/overrides/mscOutputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the MSC output override.

Write: Set the MSC output override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleMscOutput

view	edit	control	query
✗	✓	✓	✗

Enable or disable MSC output.

/overrides/sysexInputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the MIDI SysEx input override.

Write: Set the MIDI SysEx input override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleSysexInput

view	edit	control	query
✗	✓	✓	✗

Enable or disable SysEx input.

/overrides/sysexOutputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the MIDI SysEx output override.

Write: Set the MIDI SysEx output override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleSysexOutput

view	edit	control	query
✗	✓	✓	✗

Enable or disable SysEx output.

/overrides/networkExternalInputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the external network input override.

Write: Set the external network input override to `true` or `false`. See details on booleans at the beginning of this section.

The external network input override pertains to network messages that come from other devices on the network. It does not pertain to network messages from QLab itself, or from other software running on the same Mac as QLab.

/overrides/toggleNetworkExternalInput

view	edit	control	query
✗	✓	✓	✗

Enable or disable external network input.

The external network input override pertains to network messages that come from other devices on the network. It does not pertain to network messages from QLab itself, or from other software running on the same Mac as QLab.

/overrides/networkExternalOutputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the external network output override.

Write: Set the external network output override to `true` or `false`. See details on booleans at the beginning of this section.

The external network output override pertains to network messages that QLab sends to other devices on the network. It does not pertain to network messages that QLab sends to itself or to other software running on the same Mac as QLab.

/overrides/toggleNetworkExternalOutput

view	edit	control	query
✗	✓	✓	✗

Enable or disable external network output.

The external network output override pertains to network messages that QLab sends to other devices on the network. It does not pertain to network messages that QLab sends to itself or to other software running on the same Mac as QLab.

/overrides/networkLocalInputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the local network input override.

Write: Set the local network input override to `true` or `false`. See details on booleans at the beginning of this section.

The local network input override pertains to network messages that come from QLab itself or other software running on the same Mac as QLab. It does not pertain to network messages from other devices on the network.

/overrides/toggleNetworkLocalInput

view	edit	control	query
✗	✓	✓	✗

Enable or disable local network input.

The local network input override pertains to network messages that come from QLab itself or other software running on the same Mac as QLab. It does not pertain to network messages from other devices on the network.

/overrides/networkLocalOutputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the local network output override.

Write: Set the local network output override to `true` or `false`. See details on booleans at the beginning of this section.

The local network output override pertains to network messages that QLab sends to itself or to other software running on the same Mac as QLab. It does not pertain to network messages sent to other devices on the network.

/overrides/toggleNetworkLocalOutput

view	edit	control	query
✗	✓	✓	✗

Enable or disable local network output.

The local network output override pertains to network messages that QLab sends to itself or to other software running on the same Mac as QLab. It does not pertain to network messages sent to other devices on the network.

/overrides/timecodeInputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the timecode input override.

Write: Set the timecode input override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleTimecodeInput

view	edit	control	query
✗	✓	✓	✗

Enable or disable timecode input.

/overrides/timecodeOutputEnabled {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current state of the timecode output override.

Write: Set the timecode output override to `true` or `false`. See details on booleans at the beginning of this section.

/overrides/toggleTimecodeOutput

view	edit	control	query
✗	✓	✓	✗

Enable or disable timecode output.

/overrideWindow {boolean}

view	edit	control	query

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current visibility of the Override Window.

Write: Show or hide the Override Window. See details on booleans at the beginning of this section.

/toggleOverrideWindow

view	edit	control	query
✗	✓	✓	✗

Show or hide the Override Window.

/replyFormat {format_string}

view	edit	control	query
read/write	read/write	read/write	✗

Set the format of QLab's reply messages to suit your needs. `format_string` is a string containing your desired reply format. The string can optionally contain the following tokens that will be replaced when sending the reply:

- `#workspace_id#` - the workspace ID
- `#address#` - the OSC address of the reply
- `#status#` - ok / error
- `#data#` - the data of the reply

QLab will do its best to create a reply message with the format you specify.

Sending an empty string will reset QLab's reply format to its default form.

Example

Let's say you set QLab's reply format with the following message:

```
/replyFormat "#data# #address#"
```

Then, if you sent `/cue/1/colorName`, you would get the reply:

```
green /cue/1/colorName
```

The `#data#` token resolves to `green`, assuming the color of cue 1 is in fact green, and the `#address#` token resolves to `/cue/1/colorName`, since that was the address portion of the OSC command that you sent.

/timecodeWindow {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return the current visibility of the Timecode Window.

Write: Show or hide the Timecode Window. See details on booleans at the beginning of this section.

/toggleTimecodeWindow

view	edit	control	query
✗	✓	✓	✗

Show or hide the Timecode Window.

/udpReplyPort {number}

view	edit	control	query
read/write	read/write	read/write	✗

Read: If no argument is given, return the port number that QLab is using to send UDP messages to the sender.

Write: Set the port number on which the sender wants to receive UDP messages to `number`. `number` must be an integer from `0` to `65535`. Sending `0` resets the port number to the default of `53001`.

/updates {boolean}

view	edit	control	query
read/write	read/write	read/write	✗

Read: If no argument is given, return the current subscription status for this client.

Write: Subscribe or unsubscribe to updates from QLab. See details on booleans at the beginning of this section.

/version

view	edit	control	query
✓	✓	✓	✓

Return QLab's version number.

/workingDirectory {path}

view	edit	control	query
read	read/write	read	✗

Read: If no argument is given, return the current working directory, which is the directory that appears in *Open* or *Save* dialogue boxes.

Write: If `path` is provided, set the current working directory to `path`. You can provide two kinds of paths:

- Full paths, e.g. `/a/full/path/to/some/directory/`
- Paths beginning with a tilde, e.g. `~/a/path/to some/directory`

Paths beginning with a tilde (~) will be expanded; the tilde signifies “relative to the user’s home directory.”

This message provides direct access to the macOS working directory command and therefore might seem to behave strangely to those who have not dealt with the inner workings of how macOS apps open and save things. If in doubt, you can consider this message to only really matter if you are opening or saving via OSC messages. If you are, you should try to set the working directory explicitly via OSC before trying to open or save via OSC.

/workspaces

view	edit	control	query
read only	read only	read only	✗

Return an array of dictionaries for each open workspace. Each dictionary looks like this:

```
[
    {
        "uniqueID": string,
        "displayName": string,
        "port": number,
        "udpReplyPort": number,
        "version": string
    }
]
```

Workspace messages

Workspace OSC messages use the form `/workspace/{id}/command`, where `{id}` may be either the display name of the workspace, such as `hamlet.qlab5`, or the unique ID of the workspace, which can be found in the Info tab of the Workspace Status Window.

Note, however, that addressing by display name will work only if the display name is composed of characters allowed in OSC method names. This does NOT include spaces, unicode characters, diacriticals, or other “special” characters.

Addressing a workspace by its unique ID looks like this:

```
/workspace/1B11984A-3EBC-4A9C-A004-B9E3AA32DA6B/go
```

Addressing a workspace by its display name looks like this:

```
/workspace/hamlet/go
```

Note that the extension of the filename, the `.qlab5` part, is omitted.

If you send a workspace message without the `/workspace/{id}` portion of the address, then the message will be sent to all open workspaces listening on the port to which the message is sent. So, if your `hamlet.qlab5` workspace is the only open workspace, or the only workspace listening to a particular port, and you send QLab the OSC command `/go` to that port, then `hamlet.qlab5` will GO. If both `hamlet.qlab5` and `twelfthnight.qlab5` are open and listening to the same port, sending the OSC command `/go` to that port will cause both workspaces to GO.

/workspace/{id}/alwaysAudition {boolean}

view	edit	control	query
read	read/write	read	✓

Read: If no argument is given, return `true` if the specified workspace is set to always audition or `false` if not.

Write: Turn always audition on or off for the specified workspace. See details on booleans at the beginning of this section.

/workspace/{id}/auditionMonitors {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is given, return `true` if all audition monitor windows for the specified workspace are open or `false` if at least one audition monitor window is closed.

Write: Show or hide all audition monitor windows for the specified workspace. See details on booleans at the beginning of this section.

/workspace/{id}/toggleAuditionMonitors

view	edit	control	query
✗	✓	✗	✗

Show or hide all audition monitor windows for the specified workspace.

/workspace/{id}/basePath

view	edit	control	query
read only	read only	read only	✓

Return a string which is the path to the directory containing the QLab workspace. If the workspace is not yet saved, this will be an empty string.

/workspace/{id}/connect {passcode_string}

view	edit	control	query
✓	✓	✓	✗

Connect to the specified workspace. `{passcode_string}` is optional; if it is not sent, the client will connect to the workspace with whatever permissions have been set for OSC connections without passcodes. If the workspace has no permissions enabled for passcode-less connections, then connecting without a passcode is not possible.

If connecting to a workspace using a passcode, you must supply it before any other OSC messages will be accepted by the workspace or the cues it contains.

Returns `ok` if the supplied passcode matches a passcode entry in the workspace.

Returns `badpass` if the passcode does not match any passcode entries in the workspace.

Returns `error` if the specified workspace does not exist or is not open.

Starting with QLab 5, repeatedly sending this message with incorrect passcodes will introduce a progressively longer delay until the next `/connect` message will be accepted. This helps to protect against unauthorized users attempting to guess the passcode.

/workspace/{id}/cueLists

/workspace/{id}/selectedCues

/workspace/{id}/runningCues

/workspace/{id}/runningOrPausedCues

view	edit	control	query
read only	read only	read only	✕

Return an array of cue dictionaries listing the following information about all cues that fall within the scope of the message:

```
[
    {
        "uniqueID": string,
        "number": string
        "name": string
        "listName": string
        "type": string
        "colorName": string
        "colorName/live": string
        "flagged": number
        "armed": number
    }
]
```

The scope of each message is as follows:

- `cueLists` are all cue lists and carts which appear in the sidebar.
- `selectedCues` are all cues which are currently selected.
- `runningCues` are all cues which are currently running (visible in Active Cues, and with an elapsing duration.)
- `runningOrPausedCues` are all cues which are currently visible in Active Cues, whether or not their duration is elapsing.

If any of the included cues are Group cues, the dictionary will include an array of cue dictionaries for all children in the group:

```
[
    {
        "number": "{string}",
        "uniqueID": {string},
        "cues": [ {a cue dictionary}, {another dictionary}, {and another} ],
        "flagged": true|false,
        "listName": "{string}",
        "type": "{string}",
        "colorName": "{string}",
        "colorName/live": "{string}",
        "name": "{string}",
        "armed": true|false,
    }
]
```

Note: These messages may generate large replies, which can easily be larger than the maximum size supported by UDP datagrams. You should communicate with QLab via a TCP connection if you wish to use these messages.

Starting with QLab 4.4.3, versions of these commands are available which return smaller amounts of data.

The following messages are identical to the similar messages above, except they do not include any data for the children of Group cues:

/cueLists/shallow
/selectedCues/shallow

/runningCues/shallow
/runningOrPausedCues/shallow

The following messages return only the cue IDs of the cues in question, and not all the other information about them. Cue IDs of children of Group cues are included.

/cueLists/uniqueIDs
/selectedCues/uniqueIDs
/runningCues/uniqueIDs
/runningOrPausedCues/uniqueIDs

The following messages return only the cue IDs of the cues in question, and do not include children of Group cues.

/cueLists/uniqueIDs/shallow
/selectedCues/uniqueIDs/shallow
/runningCues/uniqueIDs/shallow
/runningOrPausedCues/uniqueIDs/shallow

/workspace/{id}/currentCueList {string}

view	edit	control	query
read	read	read/write	✓

Read: If no argument is provided, return the cue number of the current cue list or cart of the specified workspace.

Write: Set the current cue list or cart of the specified workspace to `string`. `string` must be the cue number of a cue list or cart in the workspace.

/workspace/{id}/currentCueListID {string}

/workspace/{id}/currentCueListID/{string}

view	edit	control	query
read	read	read/write	✓

Read: If no argument is provided, return the cue ID of the current cue list or cart of the specified workspace.

Write: Set the current cue list or cart of the specified workspace to `string`. `string` must be the cue ID of a cue list or cart in the workspace.

/workspace/{id}/dashboard/clear

view	edit	control	query
✗	✓	✗	✗

Clear all modifications made in the Dashboard and set all parameters of all lights to their home value. Parked parameters are not affected. Please be aware that this generally causes a blackout. This message is the QLab equivalent of what many other consoles refer to as “Go To Cue out.”

/workspace/{id}/dashboard/mode {string}

view	edit	control	query
read	read/write	read/write	✗

Read: If no argument is provided, this message has no effect.

Write: Set the Dashboard's view mode to `string`. Supported modes are `sliders` and `tiles`.

/workspace/{id}/dashboard/newCueWithAll

view	edit	control	query
✗	✓	✗	✗

Record all current light levels into a new Light cue. Parameters which have no explicit level set will be recorded at their home value.

/workspace/{id}/dashboard/newCueWithChanges

view	edit	control	query
✗	✓	✗	✗

Record all manually adjusted light levels in the Dashboard into a new Light cue. Parameters which have not been manually adjusted will not be recorded.

/workspace/{id}/dashboard/nextMode

view	edit	control	query
✗	✓	✗	✗

Toggle between "sliders" and "tiles" view modes in the Dashboard.

/workspace/{id}/dashboard/recordAllToLatest

view	edit	control	query
✗	✓	✗	✗

Record all manually adjusted light levels in the Dashboard into the latest-run Light cue, overwriting any levels already in that cue. If no levels have been manually adjusted, or Light cues have been run and no cue is displayed in the Dashboard as the latest Light cue, this message has no effect.

/workspace/{id}/dashboard/recordAllToSelected

view	edit	control	query
✗	✓	✗	✗

Record all manually adjusted light levels in the Dashboard into the currently selected Light cue or cues, overwriting any levels already in those cues. If no levels have been adjusted, or there are no currently selected Light cues, this message has no effect.

/workspace/{id}/dashboard/redo

view	edit	control	query
✗	✓	✗	✗

Redo the last un-done action taken in the Dashboard. If nothing has been un-done in the Dashboard, this message has no effect.

/workspace/{id}/dashboard/revert

view	edit	control	query
✗	✓	✗	✗

Revert all manually adjusted light levels in the Dashboard to the levels that they held before they were adjusted. If no levels have been adjusted, this message has no effect.

/workspace/{id}/dashboard/setLight {string} {setting} {time}

/workspace/{id}/dashboard/setLight/{string} {setting} {time}

/workspace/{id}/dashboard/setLight/live {string} {setting} {time}

/workspace/{id}/dashboard/setLight/{string}/live {setting} {time}

view	edit	control	query	live
✗	✓	✗	✗	control

Set instrument or light group `string` to level `setting` in the Dashboard. `string` may include a parameter name; if it does not, the default parameter for the specified instrument or light group will be addressed.

`setting` must be an acceptable value for the specified parameter of the specified instrument or group. If `setting` is a decimal number, the Light Dashboard may round it to the nearest equivalent DMX value.

`time` is an optional whole or decimal number. If provided, the parameter will be faded from its current value to `level` over that many seconds. If `time` is omitted, it will be assumed to be `0.0` seconds.

Examples

`/dashboard/setLight frontlight 50 5` sets the default parameter of the light or group called “frontlight” to 50, fading from its current level over 5 seconds.

`/dashboard/setLight myMover.cyan 75` sets the `cyan` parameter of the light or group called “myMover” to 75 immediately.

`/dashboard/setLight/6 25` sets the default parameter of the light or group called “6” to 25 using the single-argument form of this OSC message.

`/dashboard/setLight myMover.pantilt pantilt(20,30)` sets the `pantilt` virtual parameter of the light or group called “myMover” to pan 20, tilt 30.

/workspace/{id}/dashboard/undo

view	edit	control	query
✗	✓	✗	✗

Un-does the last action taken in the Dashboard. If nothing has been done in the Dashboard, this message has no effect.

/workspace/{id}/dashboard/updateLatestCue

view	edit	control	query
✗	✓	✗	✗

Copy all manually adjusted light levels into the latest Light cue. If the adjusted levels belong to lights or groups that are already in the latest cue, QLab will overwrite those levels. If not, QLab will add them and leave everything else alone. If no Light cues have been run, and no cue is displayed to the left as

the latest Light cue, this message has no effect.

/workspace/{id}/dashboard/updateOriginatingCues

view	edit	control	query
✗	✓	✗	✗

Copy all manually adjusted light levels into the cue or cues which originated their current levels. Originating cues are discussed in detail in the page on Light Dashboard in the Lighting section of this documentation.

/workspace/{id}/dashboard/updateSelectedCues

view	edit	control	query
✗	✓	✗	✗

Copy all manually adjusted light levels into the currently selected Light cue or cues. If the adjusted levels belong to lights or groups that are already in the selected cue or cues, QLab will overwrite those levels. If not, QLab will add them and leave everything else alone. If there are no currently selected Light cues, this message has no effect.

/workspace/{id}/delete/{cue_number}

/workspace/{id}/delete_id/{cue_id}

/workspace/{id}/delete/selected

view	edit	control	query
✗	✓	✗	✗

Delete the specified cue(s).

/workspace/{id}/delete/active

view	edit	control	query
✗	✓	✗	✗

Delete all cues that are currently running or paused.

/workspace/{id}/doubleGoWindowRemaining

view	edit	control	query
read only	read only	read only	✓

When workspace “double go protection” is engaged, return the number of seconds that must elapse until the next GO is permitted. Returns `0` when a GO is currently allowed or if double go protection is not enabled.

/workspace/{id}/fullScreen {boolean}

view	edit	control	query

view	edit	control	query
read	read/write	read	✓

Read: If no argument is provided, return the current full screen status of the workspace.

Write: Set the full screen mode status of the main workspace window. `true` will switch the main workspace window into macOS' full screen mode; `false` will switch the main workspace window into regular window mode.

/workspace/{id}/toggleFullScreen

view	edit	control	query
✗	✓	✗	✗

Turn full screen mode on or off for the main workspace window.

/workspace/{id}/go {cue_number}

view	edit	control	query
✗	✗	✓	✗

If no `cue_number` is given, tell the current list of the given workspace to GO.

If `cue_number` is given and matches the cue number of a list in the given workspace, tell that list to GO.

If `cue_number` is given and matches the cue number of a cue in any list in the given workspace, jump the playhead to that cue and then GO.

If `cue_number` is given and either does not match the cue number of a cue in any list, or matches the cue number of a cue in a cart, this message has no effect. (Cue carts do not have a playback position, so GO means nothing to a cart. Use `/start` instead.)

`cue_number` is optional; if given, it must be a string and must match a cue number in the given workspace. QLab will jump to the specified cue and then GO. If no argument is provided, the current cue list in the given workspace will GO on whatever cue is currently standing by.

When handling this OSC message, QLab cannot use the same technique it uses in other places to turn numbers into strings when necessary. This is why `cue_number`, if given, must be a string. If you're sending the message from QLab, the way to ensure that a number is sent as a string is to enclose the argument in quotation marks.

- **Correct:** `/go`
- **Correct:** `/go "53"`
- *Incorrect:* `/go 53`

Other OSC-sending devices or programs will have their own ways to specify an argument as a string.

/workspace/{id}/go/{cue_number}

view	edit	control	query
✗	✗	✓	✗

This message is equivalent to the above `/go` command, except here `cue_number` is part of the address, not an argument, and is not optional. Since it's part of the address, it should not include quotation marks as discussed above.

/workspace/{id}/auditionGo {cue_number}

/workspace/{id}/auditionGo/{cue_number}

view	edit	control	query
✗	✗	✓	✗

These messages are equivalent to the above `/go` commands, except they trigger an audition GO instead of a regular GO. Audition GO causes the cue or cues which are started to play through their audition outputs, defined in Workspace Settings.

/workspace/{id}/hardStop

view	edit	control	query
✗	✗	✓	✗

Stop all playback and cut all audio effects immediately.

/workspace/{id}/liveFadePreview {boolean}

view	edit	control	query
read	read/write	read/write	✓

Enable or disable live fade preview. See details on booleans at the beginning of this section. If no argument is given, return the current status of live fade preview.

/workspace/{id}/toggleLiveFadePreview

view	edit	control	query
✗	read/write	✗	✗

Enable or disable live fade preview.

/workspace/{id}/lightDashboard {boolean}

view	edit	control	query
read	read/write	read	✓

Read: If no argument is given, return the current visibility of the Dashboard.

Write: Show or hide the Light Dashboard. See details on booleans at the beginning of this section.

/workspace/{id}/toggleLightDashboard

view	edit	control	query
✗	✓	✗	✗

If the Light Dashboard is closed, open it and place focus in the command line. If the Light Dashboard is open, but focus is not in the command line, place focus in the command line. If the Light Dashboard is open and focus is in the command line, move focus to the main workspace window.

/workspace/{id}/move/{cue_id} {new_index} {new_parent_cue_id}



`new_parent_cue_id` is optional, and must be a string.

If `new_parent_cue_id` is not provided, move the specified cue (`cue_id`) from its current position to the given `new_index` position within the cue's current parent Group, Cart, or List. `new_index` is required and must be an integer.

If `new_parent_cue_id` is provided, move the specified cue from its current position to the given `new_index` position within the Group, Cart, or List whose unique ID is `new_parent_cue_id` .

If the move fails for any reason (i.e. a Group cue cannot be moved inside of another Group cue that it already contains), QLab will send an error reply.

If the move succeeds, QLab will reply with `"status": "ok"` and `"data"` containing a dictionary with 2 key/value pairs:

```
[
    {
        "parent_cue_id": string,
        "index": integer
    }
]
```

`parent_cue_id` is a string with the unique ID of the Group, Cart, or List that contains the cue that was moved. `index` is an integer with the index of the position of the moved cue in its new parent.

/workspace/{id}/new {cue_type}

view	edit	control	query
✗	✓	✗	✗

Create a new cue. `cue_type` is a string stating the type of cue to create. Supported strings include: `audio` , `mic` , `video` , `camera` , `text` , `light` , `fade` , `network` , `midi` , `midi file` , `timecode` , `group` , `start` , `stop` , `pause` , `load` , `reset` , `devamp` , `goto` , `target` , `arm` , `disarm` , `wait` , `memo` , `script` , `list` , `cuelist` , `cue list` , `cart` , `cuecart` , or `cue cart` .

This method returns the unique ID of the new cue. The newly created cue will also be selected, so subsequent commands can address the new cue either using the unique ID or simply by addressing the currently selected cue.

This method has three optional additional arguments:

`/workspace/{id}/new {cue_type} {cue_ID} {cart_row} {cart_column}`

If `{cue_ID}` is supplied, the new cue will be created after that cue.

If `{cue_ID}` specifies a cart, the new cue will be created within the cart. You must then specify the position in the cart using `{cart_row}` and `{cart_column}` , which must be integers.

/workspace/{id}/panic

view	edit	control	query
✗	✗	✓	✗

Tell the workspace to panic. A panic is a brief gradual fade out leading into a hard stop. Sending a second instruction to panic during that gradual fade out will cause an immediate hard stop.

/workspace/{id}/panicInTime {number}

view	edit	control	query
✗	✗	✓	✗

Panic over the specified time, rather than over the panic time defined in the workspace. `{number}` can be any number 0 or greater, decimals allowed.

/workspace/{id}/pause

view	edit	control	query
✗	✗	✓	✗

Pause all currently running cues in the workspace.

/workspace/{id}/playhead/{cue_number}

/workspace/{id}/playheadID/{cue_id}

/workspace/{id}/playbackPosition/{cue_number}

/workspace/{id}/playbackPositionID/{cue_id}

view	edit	control	query
✗	✗	✓	✗

Set the playhead (also called the playback position) of the active cue list to the given cue. When using `/playheadID` or `/playbackPositionID`, sending the value `none` will unset the playhead.

/workspace/{id}/playhead/active

view	edit	control	query
✗	✗	✓	✗

Move the playhead (also called the playback position) of the active cue list to the first active cue.

/workspace/{id}/playhead/next

/workspace/{id}/playbackPosition/next

view	edit	control	query
✗	✗	✓	✗

Move the playhead (also called the playback position) of the active cue list to the next cue.

/workspace/{id}/playhead/none

/workspace/{id}/playbackPosition/none

view	edit	control	query
✗	✗	✓	✗

Unset the playhead (also called the playback position) of the active cue list.

/workspace/{id}/playhead/previous

/workspace/{id}/playbackPosition/previous

view	edit	control	query
✗	✗	✓	✗

Move the playhead (also called the playback position) of the active cue list to the previous cue.

/workspace/{id}/playhead/nextSequence

/workspace/{id}/playbackPosition/nextSequence

view	edit	control	query
✗	✗	✓	✗

Move the playhead (also called the playback position) of the active cue list to the next cue sequence.

/workspace/{id}/playhead/previousSequence

/workspace/{id}/playbackPosition/previousSequence

view	edit	control	query
✗	✗	✓	✗

Move the playhead (also called the playback position) of the active cue list to the previous cue sequence.

/workspace/{id}/playhead/selected

view	edit	control	query
✗	✗	✓	✗

Move the playhead (also called the playback position) of the active cue list to the first selected cue.

/workspace/{id}/redo

view	edit	control	query
✗	✓	✗	✗

Redo the last un-done action in the workspace. If nothing has been un-done in the workspace, this message has no effect.

/workspace/{id}/renumber {startNumber} {incrementNumber} {prefix} {suffix}

view	edit	control	query
✗	✓	✗	✗

Renumber the selected cues, starting at `startNumber` and incrementing by `incrementNumber`. Both numbers must be a positive number, decimals allowed. `prefix` and `suffix` are both optional and can be any text.

/workspace/{id}/reset

view	edit	control	query
✗	✗	✓	✗

Reset the workspace. Resetting stops all cues, returns the playhead to the top of the current cue list, and restores any temporary changes made to cues (such as retargeting via a Target cue or adjustments using a “live” OSC message.)

/workspace/{id}/resume

view	edit	control	query
✗	✗	✓	✗

Un-pause all paused cues in the workspace.

/workspace/{id}/save

view	edit	control	query
✗	✓	✗	✗

Tell the given workspace to save itself to disk.

/workspace/{id}/select/{cue_number}

/workspace/{id}/select_id/{id}

view	edit	control	query
✗	✓	✓	✗

Select the specified cue(s).

/workspace/{id}/select/next

view	edit	control	query
✗	✓	✓	✗

Move the selection down one cue.

/workspace/{id}/select/previous

view	edit	control	query
✗	✓	✓	✗

Move the selection up one cue.

/workspace/{id}/showMode {boolean}

view	edit	control	query
read	read/write	read	✓

Read: If no argument is provided, return `true` if the workspace is currently in show mode, and `false` if the workspace is currently in edit mode.

Write: If `boolean` is true, set the workspace to show mode. If false, set the workspace to edit mode. See details on booleans at the beginning of this section.

/workspace/{id}/toggleEditShowMode

view	edit	control	query
✗	✓	✗	✗

Switch between show mode and edit mode.

/workspace/{id}/stop

view	edit	control	query
✗	✓	✗	✗

Stop playback but allow audio effects which decay over time (for example, reverbs) to continue rendering.

/workspace/{id}/thump

view	edit	control	query
read only	read only	read only	✓

Returns a string `thump`. This is a simple “heartbeat” message (thump-thump, thump-thump) which you can use to verify a connection, keep a session active, etc.

/workspace/{id}/undo

view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the workspace. If there is no valid action to undo, this message has no effect.

Workspace Settings messages

Workspace Settings OSC messages use the form `/workspace/{id}/settings/command`, where `{id}` may be either the display name of the workspace, such as `hamlet.qlab5`, or the unique ID of the workspace, which can be found in the Info tab of the Workspace Status Window.

Note, however, that addressing by display name will work only if the display name is composed of characters allowed in OSC method names. This does NOT include spaces, unicode characters, diacriticals, or other "special" characters.

Addressing a workspace by its unique ID looks like this:

```
/workspace/1B11984A-3EBC-4A9C-A004-B9E3AA32DA6B/settings/light/patch
```

Addressing a workspace by its display name looks like this:

```
/workspace/hamlet/settings/light/patch
```

Note that the extension of the filename, the `.qlab5` part, is omitted.

If you send a workspace message without the `/workspace/{id}` portion of the address, then the message will be sent to all open workspaces listening on the port to which the message is sent. So, if your `hamlet.qlab5` workspace is the only open workspace, or the only workspace listening to a particular port, and you send QLab the OSC command `/light/patch` to that port, then `hamlet.qlab5` will receive that message and respond appropriately. If both `hamlet.qlab5` and `twelfthnight.qlab5` are open and listening to the same port, sending the OSC command `/light/patch` to that port will cause both workspaces to respond appropriately.

/settings/audio/cueOutputChannelCounts

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary of cue output counts for audio output patches:

```
[
  {
    "{patch uniqueID}":{number of cue outputs},
    "{another patch uniqueID}":{number of cue outputs},
    et cetera...
  }
]
```

The number of patches in a workspace varies, so the number of items in this dictionary will vary.

/settings/audio/map/{map_name}

/settings/audio/mapID/{map_ID}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return a array containing all the details of the specified audio map. These details include the width, height, and unique ID of the map, and nested arrays for the filter, marks, and map objects in the map. Those nested arrays contain all the attributes of each item like spread, name, unique ID, etc. The attributes in each array will differ, of course, since filters, marks, and map objects all have different attributes.

/settings/audio/map/{map_name}/filters

/settings/audio/mapID/{map_ID}/filters

/settings/audio/map/{map_name}/marks

/settings/audio/mapID/{map_ID}/marks

/settings/audio/map/{map_name}/objects

/settings/audio/mapID/{map_ID}/objects

/settings/audio/map/{map_name}/size

/settings/audio/mapID/{map_ID}/size

/settings/audio/map/{map_name}/size/height

/settings/audio/mapID/{map_ID}/size/height

/settings/audio/map/{map_name}/size/width

/settings/audio/mapID/{map_ID}/size/width

/settings/audio/map/{map_name}/uniqueID

/settings/audio/mapID/{map_ID}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return a subset of the data returned by `/settings/audio/map` or `/settings/audio/mapID`.

/settings/audio/map/{map_name}/filter/{name}

/settings/audio/mapID/{map_ID}/filter/{name}

/settings/audio/map/{map_name}/filterID/{id}

/settings/audio/mapID/{map_ID}/filterID/{id}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array containing all the attributes of the specified filter of the specified audio map.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/settings/audio/map/{map_name}/filter/{name}/angle {number}

/settings/audio/mapID/{map_ID}/filter/{name}/angle {number}

/settings/audio/map/{map_name}/filterID/{id}/angle {number}

/settings/audio/mapID/{map_ID}/filterID/{id}/angle {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the angle of the specified filter in degrees away from horizontal.

Write: If `number` is given, set the angle of the specified filter to `number` . `number` can be any whole or decimal number from `-180` to `180` .

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/settings/audio/map/{map_name}/filter/{name}/passthrough/{output} {boolean}

/settings/audio/mapID/{map_ID}/filter/{name}/passthrough/{output} {boolean}

/settings/audio/map/{map_name}/filterID/{id}/passthrough/{output} {boolean}

/settings/audio/mapID/{map_ID}/filterID/{id}/passthrough/{output} {boolean}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

`output` is required and must be an integer from 1 to 128.

Read: If no argument is given, return `TRUE` if the specified output is set to pass through the filter or `FALSE` if it is not.

Write: Check or uncheck the passthrough checkbox of the specified output in the specified filter. See details on booleans at the beginning of this section.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/settings/audio/map/{map_name}/filter/{name}/passthroughs

/settings/audio/mapID/{map_ID}/filter/{name}/passthroughs

/settings/audio/map/{map_name}/filterID/{id}/passthroughs

/settings/audio/mapID/{map_ID}/filterID/{id}/passthroughs

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the passthrough checkboxes of the specified filter.

/settings/audio/map/{map_name}/filter/{name}/name {string}

/settings/audio/mapID/{map_ID}/filter/{name}/name {string}

/settings/audio/map/{map_name}/filterID/{id}/name {string}

/settings/audio/mapID/{map_ID}/filterID/{id}/name {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the name of the specified filter.

Write: If `string` is given, set the name of the specified filter to `string`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/settings/audio/map/{map_name}/filter/{name}/position {x} {y}

/settings/audio/mapID/{map_ID}/filter/{name}/position {x} {y}

/settings/audio/map/{map_name}/filterID/{id}/position {x} {y}

/settings/audio/mapID/{map_ID}/filterID/{id}/position {x} {y}

/settings/audio/map/{map_name}/filter/{name}/position/live {x} {y}

/settings/audio/mapID/{map_ID}/filter/{name}/position/live {x} {y}

/settings/audio/map/{map_name}/filterID/{id}/position/live {x} {y}

/settings/audio/mapID/{map_ID}/filterID/{id}/position/live {x} {y}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no arguments are given, return the position of the specified filter.

Write: If `x` and `y` are given, set the position of the specified filter to `(x,y)`. `x` and `y` can be any whole or decimal numbers.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/settings/audio/map/{map_name}/filter/{name}/position/x {number}

/settings/audio/mapID/{map_ID}/filter/{name}/position/x {number}

/settings/audio/map/{map_name}/filterID/{id}/position/x {number}

/settings/audio/mapID/{map_ID}/filterID/{id}/position/x {number}

/settings/audio/map/{map_name}/filter/{name}/position/x/live {number}

/settings/audio/mapID/{map_ID}/filter/{name}/position/x/live {number}

/settings/audio/map/{map_name}/filterID/{id}/position/x/live {number}

/settings/audio/mapID/{map_ID}/filterID/{id}/position/x/live {number}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no argument is given, return the X-axis position of the specified filter.

Write: If `number` is given, set the X-axis position of the specified filter to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/settings/audio/map/{map_name}/filter/{name}/position/y {number}

/settings/audio/mapID/{map_ID}/filter/{name}/position/y {number}

/settings/audio/map/{map_name}/filterID/{id}/position/y {number}

/settings/audio/mapID/{map_ID}/filterID/{id}/position/y {number}

/settings/audio/map/{map_name}/filter/{name}/position/y/live {number}

/settings/audio/mapID/{map_ID}/filter/{name}/position/y/live {number}

/settings/audio/map/{map_name}/filterID/{id}/position/y/live {number}

/settings/audio/mapID/{map_ID}/filterID/{id}/position/y/live {number}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no argument is given, return the Y-axis position of the specified filter.

Write: If `number` is given, set the Y-axis position of the specified filter to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/settings/audio/map/{map_name}/filter/{name}/uniqueID

/settings/audio/mapID/{map_ID}/filter/{name}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the unique ID of the specified filter.

/settings/audio/map/{map_name}/filter/{name}/width {number}

/settings/audio/mapID/{map_ID}/filter/{name}/width {number}

/settings/audio/map/{map_name}/filterID/{id}/width {number}

/settings/audio/mapID/{map_ID}/filterID/{id}/width {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the width of the specified filter.

Write: If `number` is given, set the width of the specified filter to `number`. `number` can be any whole or decimal number `0` or greater, limited to half of the largest dimension of the map.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}

/settings/audio/mapID/{map_ID}/mark/{name}

/settings/audio/map/{map_name}/markID/{id}

/settings/audio/mapID/{map_ID}/markID/{id}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array containing all the attributes of the specified mark.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}/gravity {number}

/settings/audio/mapID/{map_ID}/mark/{name}/gravity {number}

/settings/audio/map/{map_name}/markID/{id}/gravity {number}

/settings/audio/mapID/{map_ID}/markID/{id}/gravity {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the gravity of the specified mark.

Write: If `number` is given, set the gravity of the specified mark to `number`. `number` can be any whole or decimal number from `0.1` to `5.0`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}/level/{outChannel} {decibel}

/settings/audio/mapID/{map_ID}/mark/{name}/level/{outChannel} {decibel}

/settings/audio/map/{map_name}/markID/{id}/level/{outChannel} {decibel}

/settings/audio/mapID/{map_ID}/markID/{id}/level/{outChannel} {decibel}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

`outChannel` is required and must be an integer from 0 to 128. 0 is the main row.

Read: If no `decibel` is given, return the volume level of the specified crosspoint.

Write: If `decibel` is given, set the specified crosspoint to `decibel`. `decibel` can be any number, decimals allowed, or a string such as `-INF`. If `decibel` is a string (any string, in fact) QLab will set the crosspoint to `-INF`,

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}/levels

/settings/audio/mapID/{map_ID}/mark/{name}/levels

/settings/audio/map/{map_name}/markID/{id}/levels

/settings/audio/mapID/{map_ID}/markID/{id}/levels

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the levels of the specified mark.

/settings/audio/map/{map_name}/mark/{name}/name {string}

/settings/audio/mapID/{map_ID}/mark/{name}/name {string}

/settings/audio/map/{map_name}/markID/{id}/name {string}

/settings/audio/mapID/{map_ID}/markID/{id}/name {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the name of the specified mark.

Write: If `string` is given, set the name of the specified mark to `string`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}/position {x} {y}

/settings/audio/mapID/{map_ID}/mark/{name}/position {x} {y}

/settings/audio/map/{map_name}/markID/{id}/position {x} {y}

/settings/audio/mapID/{map_ID}/markID/{id}/position {x} {y}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no arguments are given, return the position of the specified mark.

Write: If `x` and `y` are given, set the position of the specified mark to `(x,y)`. `x` and `y` can be any whole or decimal numbers.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}/position/x {number}

/settings/audio/mapID/{map_ID}/mark/{name}/position/x {number}

/settings/audio/map/{map_name}/markID/{id}/position/x {number}

/settings/audio/mapID/{map_ID}/markID/{id}/position/x {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the X-axis position of the specified mark.

Write: If `number` is given, set the X-axis position of the specified mark to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}/position/y {number}

/settings/audio/mapID/{map_ID}/mark/{name}/position/y {number}

/settings/audio/map/{map_name}/markID/{id}/position/y {number}

/settings/audio/mapID/{map_ID}/markID/{id}/position/y {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the Y-axis position of the specified mark.

Write: If `number` is given, set the Y-axis position of the specified mark to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}/shadow {number}

/settings/audio/mapID/{map_ID}/markID/{id}/shadow {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the shadow angle of the specified mark.

Write: If `number` is given, set the shadow angle of the specified mark to `number`. `number` can be any whole or decimal number from `0` to `90`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/settings/audio/map/{map_name}/mark/{name}/uniqueID

/settings/audio/mapID/{map_ID}/mark/{name}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the unique ID of the specified mark.

/settings/audio/map/{map_name}/object/{name}

/settings/audio/mapID/{map_ID}/object/{name}

/settings/audio/map/{map_name}/objectID/{id}

/settings/audio/mapID/{map_ID}/objectID/{id}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array containing all the attributes of the specified map object.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/settings/audio/map/{map_name}/object/{name}/colorName {number}

/settings/audio/mapID/{map_ID}/object/{name}/colorName {number}

/settings/audio/map/{map_name}/objectID/{id}/colorName {number}

/settings/audio/mapID/{map_ID}/objectID/{id}/colorName {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the color of the specified map object.

Write: If `string` is given, set the color of the specified map object to `string` . Valid colors are:

```
berry
blue
crimson
cyan
forest
gray
green
hot pink
indigo
lavender
magenta
midnight
olive
orange
peach
```

```
plum
purple
red
sky blue
yellow
```

and

```
none
```

Certain other colors may also be valid...

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/settings/audio/map/{map_name}/object/{name}/name {string}

/settings/audio/mapID/{map_ID}/object/{name}/name {string}

/settings/audio/map/{map_name}/objectID/{id}/name {string}

/settings/audio/mapID/{map_ID}/objectID/{id}/name {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the name of the specified map object.

Write: If `string` is given, set the name of the specified map object to `string`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/settings/audio/map/{map_name}/object/{name}/position {x} {y}

/settings/audio/mapID/{map_ID}/object/{name}/position {x} {y}

/settings/audio/map/{map_name}/objectID/{id}/position {x} {y}

/settings/audio/mapID/{map_ID}/objectID/{id}/position {x} {y}

/settings/audio/map/{map_name}/object/{name}/position/live {x} {y}

/settings/audio/mapID/{map_ID}/object/{name}/position/live {x} {y}

/settings/audio/map/{map_name}/objectID/{id}/position/live {x} {y}

/settings/audio/mapID/{map_ID}/objectID/{id}/position/live {x} {y}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no arguments are given, return the position of the specified map object.

Write: If `x` and `y` are given, set the position of the specified map object to `(x,y)`. `x` and `y` can be any whole or decimal numbers.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/settings/audio/map/{map_name}/object/{name}/position/x {number}

/settings/audio/mapID/{map_ID}/object/{name}/position/x {number}

/settings/audio/map/{map_name}/objectID/{id}/position/x {number}

/settings/audio/mapID/{map_ID}/objectID/{id}/position/x {number}

/settings/audio/map/{map_name}/object/{name}/position/x/live {number}

/settings/audio/mapID/{map_ID}/object/{name}/position/x/live {number}

/settings/audio/map/{map_name}/objectID/{id}/position/x/live {number}

/settings/audio/mapID/{map_ID}/objectID/{id}/position/x/live {number}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no argument is given, return the X-axis position of the specified map object.

Write: If `number` is given, set the X-axis position of the specified map object to `number` . `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/settings/audio/map/{map_name}/object/{name}/position/y {number}

/settings/audio/mapID/{map_ID}/object/{name}/position/y {number}

/settings/audio/map/{map_name}/objectID/{id}/position/y {number}

/settings/audio/mapID/{map_ID}/objectID/{id}/position/y {number}

/settings/audio/map/{map_name}/object/{name}/position/y/live {number}

/settings/audio/mapID/{map_ID}/object/{name}/position/y/live {number}

/settings/audio/map/{map_name}/objectID/{id}/position/y/live {number}

/settings/audio/mapID/{map_ID}/objectID/{id}/position/y/live {number}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no argument is given, return the Y-axis position of the specified map object.

Write: If `number` is given, set the Y-axis position of the specified map object to `number` . `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/settings/audio/map/{map_name}/object/{name}/spread {number}

/settings/audio/mapID/{map_ID}/object/{name}/spread {number}

/settings/audio/map/{map_name}/objectID/{id}/spread {number}

/settings/audio/mapID/{map_ID}/objectID/{id}/spread {number}

/settings/audio/map/{map_name}/object/{name}/spread/live {number}

/settings/audio/mapID/{map_ID}/object/{name}/spread/live {number}

/settings/audio/map/{map_name}/objectID/{id}/spread/live {number}

/settings/audio/mapID/{map_ID}/objectID/{id}/spread/live {number}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no argument is given, return the spread of the specified map object.

Write: If `number` is given, set the spread of the specified map object to `number`. `number` can be any whole or decimal number from `0` to twice the largest dimension of the map.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/settings/audio/map/{map_name}/object/{name}/uniqueID

/settings/audio/mapID/{map_ID}/object/{name}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the unique ID of the specified map object.

/settings/audio/map/{map_name}/testObject

/settings/audio/mapID/{map_ID}/testObject

view	edit	control	query
read only	read only	read only	✗

Return a array containing the level, position, and running status of the test object in the specified map.

/settings/audio/maps/testObjectFileTarget {string}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the target of the audio map test object. Empty string (`""`) means that the cue has no file target.

Write: If `string` is given, set the target of the audio map test object to `string`. If `string` is `"none"` or empty (`""`), unset the file target. If `string` is anything else, `string` should be a file path and name in any of the following three forms:

- Full paths, e.g. `/Volumes/MyDisk/path/to/some/file.wav`
- Paths beginning with a tilde, e.g. `~/path/to some/file.mov`
- Relative paths, e.g. `this/is/a/relative/path.mid`

Paths beginning with a tilde (~) will be expanded; the tilde signifies “relative to the user’s home directory”.

Relative paths will be interpreted according to the current working directory. Use the `/workingDirectory` application message to set or get the current working directory.

/settings/audio/map/{map_name}/testObject/level {number}

/settings/audio/mapID/{map_ID}/testObject/level {number}

view	edit	control	query	+/-?
✗	✓	✓	✓	✓

Read: If no argument is given, return the level of the test object in the specified map.

Write: If `number` is given, set the level of the test object in the specified map to `number`. `number` can be any whole or decimal number between the minimum and maximum workspace audio level.

/settings/audio/map/{map_name}/testObject/position {x} {y}

/settings/audio/mapID/{map_ID}/testObject/position {x} {y}

view	edit	control	query	+/-?
✗	✓	✓	✗	✗

Read: If no arguments are given, return the position of the test object in the specified map.

Write: If `x` and `y` are given, set the position of the test object in the specified map to `(x,y)`. `x` and `y` can be any whole or decimal numbers.

/settings/audio/map/{map_name}/testObject/position/x {number}

/settings/audio/mapID/{map_ID}/testObject/position/x {number}

view	edit	control	query	+/-?
✗	✓	✓	✓	✓

Read: If no argument is given, return the X-axis position of the test object in the specified map.

Write: If `number` is given, set the X-axis position of the test object in the specified map to `number`. `number` can be any whole or decimal number.

/settings/audio/map/{map_name}/testObject/position/y {number}

/settings/audio/mapID/{map_ID}/testObject/position/y {number}

view	edit	control	query	+/-?
✗	✓	✓	✓	✓

Read: If no argument is given, return the Y-axis position of the test object in the specified map.

Write: If `number` is given, set the Y-axis position of the test object in the specified map to `number`. `number` can be any whole or decimal number.

/settings/audio/map/{map_name}/testObject/running {boolean}

/settings/audio/mapID/{map_ID}/testObject/running {boolean}

view	edit	control	query	+/-?
✗	✓	✓	✓	✗

Read: If no argument is given, return `TRUE` if the test object in the specified map is running, or `FALSE` if it is not.

Write: Start or stop the test object in the specified map. See details on booleans at the beginning of this section.

/settings/audio/map/{map_name}/testObject/start

/settings/audio/mapID/{map_ID}/testObject/start

view	edit	control	query
✗	✓	✓	✗

Start the test object in the specified map.

/settings/audio/map/{map_name}/testObject/stop

/settings/audio/mapID/{map_ID}/testObject/stop

view	edit	control	query
✗	✓	✓	✗

Stop the test object in the specified map.

/settings/audio/maps

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return a array containing all the details of all the audio maps in the workspace. These details include the name, width, height, and unique ID of the map, and nested arrays for the filter, marks, and map objects in the map. Those nested arrays contain all the attributes of each item like spread, name, unique ID, etc. The attributes in each array will differ, of course, since filters, marks, and map objects all have different attributes.

/settings/audio/maps/undo



view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the Audio Map Editor. If there is no valid action to undo, this message has no effect.

/settings/audio/maps/redo

view	edit	control	query
✗	✓	✗	✗

Redo the most recent un-done change in the Audio Map Editor. If there is no valid action to redo, this message has no effect.

/settings/audio/maxVolume

/settings/audio/minVolume

view	edit	control	query
read only	read only	read only	✗

Return the decibel value of the “Min:” and “Max:” levels from the Volume Limits section of Workspace Settings → Audio.

/settings/audio/outputChannelNames

/settings/mic/outputChannelNames

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary of output names for audio output patches:

```
[
  {
    "{patch uniqueID}" :
      {
        "{cue output number}": "{output name}",
        "{another cue output number}": "{another output name}",
        et cetera...
      }
  },
  { ... }
]
```

`{ ... }` represents a second audio patch; the number of patches in a workspace varies, so the number of items in this dictionary will vary.

If an audio output patch does not have customized output names, that patch will not be included in the dictionary. If no audio output patches have customized output names, the data returned will be an empty JSON object.

The `/mic` form of this message is deprecated in QLab 5.0. This message works in QLab 5, but will be removed in a future version of QLab. Use the `/audio` form instead.

/settings/audio/patch/{name}

/settings/audio/patchID/{id}

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing the specified audio output patch.

```
[
  {
    "name" : "{string}",
    "uniqueID" : "{string}",
    "routing" : [number,number,number,...]
  }
]
```

The numbers listed for `routing` are the device output numbers which have cue outputs routed to them. For example, if the patch is using a device with four outputs, and all four device outputs have cue outputs routed to them, the list will be `[1,2,3,4]`. If device output 3 has no cue outputs routed to it, the list will be `[1,2,4]`.

/settings/audio/patch/{name}/cueOutputChannels {number}

/settings/audio/patchID/{id}/cueOutputChannels {number}

view	edit	control	query	+/-?
read	read/write	✗	✓	✓

Read: If no argument is given, return the number of cue outputs of the specified patch.

Write: If `number` is given, set the number of cue outputs of the specified patch to `number`, which must be a whole number from 1 to 128.

/settings/audio/patch/{name}/level/{inChannel}/{outChannel} {decibel}

/settings/audio/patch/{name}/level/{inChannel}/{outChannel}/live {decibel}

/settings/audio/patchID/{id}/level/{inChannel}/{outChannel} {decibel}

/settings/audio/patchID/{id}/level/{inChannel}/{outChannel}/live {decibel}

view	edit	control	query	+/-?	live
read	read/write	✗	✓	✓	control

Read: If no argument is given, return the volume level of the specified crosspoint.

Write: If `decibel` is given, set the specified crosspoint to `decibel`. `decibel` can be any number, decimals allowed, or a string such as `-INF`. If `decibel` is a string (any string, in fact) QLab will set the crosspoint to `-INF`.

/settings/audio/patch/{name}/mute {output} {boolean}

/settings/audio/patch/{name}/mute/{output} {boolean}

/settings/audio/patchID/{id}/mute {output} {boolean}

/settings/audio/patchID/{id}/mute/{output} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `TRUE` if the device output of the specified patch is muted, or `FALSE` if it is not.

Write: Mute or unmute the specified device output of the specified patch. See details on booleans at the beginning of this section.

/settings/audio/patch/{name}/mute/clear

/settings/audio/patchID/{id}/mute/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-mute all device outputs of the specified patch.

/settings/audio/patch/{name}/muteChannels

/settings/audio/patchID/{id}/muteChannels

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the numbers of device outputs that are muted.

/settings/audio/patch/{name}/name {string}

/settings/audio/patchID/{id}/name {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Set the name of the specified patch to `string`.

/settings/audio/patch/{name}/reset

/settings/audio/patchID/{id}/reset

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Reset all faded parameters of the specified audio patch (cue output effects, routing, and device output effects) to their saved state.

/settings/audio/patch/{name}/routing

/settings/audio/patchID/{id}/routing

view	edit	control	query	+/-?
read only	read only	✗	✗	✗

Return the device output numbers which have cue outputs routed to them. For example, if the patch is using a device with four outputs, and all four device outputs have cue outputs routed to them, the list will be `[1,2,3,4]`. If device output 3 has no cue outputs routed to it, the list will be `[1,2,4]`.

/settings/audio/patch/{name}/routing/reset

/settings/audio/patchID/{id}/routing/reset

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Reset faded audio levels of the specified audio patch to their saved state.

/settings/audio/patch/{name}/solo {output} {boolean}

/settings/audio/patch/{name}/solo/{output} {boolean}

/settings/audio/patchID/{id}/solo {output} {boolean}

/settings/audio/patchID/{id}/solo/{output} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `TRUE` if the device output of the specified patch is soloed, or `FALSE` if it is not.

Write: Solo or un-solo the specified device output of the specified patch. See details on booleans at the beginning of this section.

/settings/audio/patch/{name}/solo/clear

/settings/audio/patchID/{id}/solo/clear

view	edit	control	query	+/-?

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-solo all device outputs of the specified patch.

/settings/audio/patch/{name}/soloChannels

/settings/audio/patchID/{id}/soloChannels

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the numbers of device outputs that are soloed.

/settings/audio/patch/{name}/uniqueID

/settings/audio/patchID/{id}/uniqueID

view	edit	control	query
read only	read only	read only	✗

Return the unique ID of the specified audio output patch.

/settings/audio/patchList

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing all audio output patches defined in the workspace:

```
[
  {
    "name" : "{string}",
    "uniqueID" : "{string}",
    "routing" : [number,number,number,...]
  },
  { ... }
]
```

The numbers listed for `routing` are the device output numbers which have cue outputs routed to them. For example, if the patch is using a device with four outputs, and all four device outputs have cue outputs routed to them, the list will be `[1,2,3,4]`. If device output 3 has no cue outputs routed to it, the list will be `[1,2,4]`.

If an audio output patch does not have a name, QLab will use the name of the audio device assigned to that patch as its `name`.

/settings/audio/undo

view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the Audio section of Workspace Settings. If there is no valid action to undo, this message has no effect.

/settings/audio/redo

view	edit	control	query
✗	✓	✗	✗

Redo the most recent un-done change in the Audio section of Workspace Settings. If there is no valid action to redo, this message has no effect.

/settings/general/minGoTime {number}

view	edit	control	query
read	read/write	read	✗

Read: If `number` is not given, return the minimum time required between each GO (a.k.a. “double GO protection time” or “minimum GO time.“)

Write: Set the minimum time required between each GO to `number` seconds. `number` can be any number greater than or equal to `0`, decimals allowed.

/settings/general/selectionIsPlayhead {boolean}

view	edit	control	query
read	read/write	read/write	✓

Read: If no argument is provided, return `true` if the selection is currently locked to the playhead, and `false` if it is not.

Write: If `boolean` is true, lock the selection and the playhead together. If `boolean` is false, unlock the selection from the playhead.

/settings/general/toggleSelectionIsPlayhead

view	edit	control	query
✗	✓	✓	✗

Lock or unlock the selection to the playhead.

/settings/general/undo

view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the General section of Workspace Settings. If there is no valid action to undo, this message has no effect.

/settings/general/redo

view	edit	control	query
✗	✓	✗	✗

Redo the most recent un-done change in the General section of Workspace Settings. If there is no valid action to redo, this message has no effect.

/settings/light/patch

view	edit	control	query
read only	read only	read only	✕

Return a (rather verbose) JSON dictionary describing the light patch for the workspace:

```
{
  "settingKeywords":[
    "home",
    "pass",
    "cue"
  ],
  "groups":[
    {
          "instruments":[
            {
              "conflicted":true|false,
              "patched":true|false,
              "name":"{instrument name}",
              "definition":{
                "isBroken":true|false,
                "manufacturer":"{manufacturer name}",
                "defaultParameter":{number of default parameter},
                "name":"{definition name}",
                "definitionVersion":{version of definition},
                "parameters":{
                  "{parameter number}":{
                    "isBroken":true|false,
                    "homeValue":{value},
                    "valueIsPercentage":true|false,
                    "name":"{parameter name}",
                    "type":"scalar"|"pantilt"|"rgbcolor"|"cmycolor"|"muxer",
                    "twoBytes":true|false
                  },
                  (more parameters as needed...)
                }
              },
              "parameters":[
                {
                  "valueIsPercentage":true|false,
                  "homeValueInDMX":{value},
                  "twoBytes":true|false,
                  "uniqueName":"{instrument name}.{parameter name}",
                  "definitionParameter":{
                    "isBroken":true|false,
                    "homeValue":{value},
                    "valueIsPercentage":true|false,
                    "name":"{parameter name}",
                    "type":"scalar"|"pantilt"|"rgbcolor"|"cmycolor"|"muxer",
                    "twoBytes":true|false
                  },
                  "instrumentName":"{instrument name}",
                  "homeValue":{value},
                  "name":"{parameter name}"
                },
                (more parameters as needed...)
```

```
            ]
          },
          (more instruments as needed...)
        ],
        "name":"{light group name}",
        "members":[
          {"name":"{instrument name"},
          (more instruments as needed...)
        ],
        "parameters":[
          {
            "isBroken":true|false,
            "homeValue":{value},
            "valueIsPercentage":true|false,
            "name":"{parameter name}",
            "type":"scalar"|"pantilt"|"rgbcolor"|"cmycolor"|"muxer",
            "twoBytes":true|false
          },
          (more parameters as needed...)
        ]
      },
  (more groups as needed...)
],
"instruments":[
  {
        "conflicted":true|false,
        "patched":true|false,
        "name":"{instrument name}",
        "definition":{
          "isBroken":true|false,
          "manufacturer":"{manufacturer name}",
          "defaultParameter":{number of default parameter},
          "name":"{definition name}",
          "definitionVersion":{version of definition},
          "parameters":{
            "{parameter number}":{
              "isBroken":true|false,
              "homeValue":{value},
              "valueIsPercentage":true|false,
              "name":"{parameter name}",
              "type":"scalar"|"pantilt"|"rgbcolor"|"cmycolor"|"muxer",
              "twoBytes":true|false
            },
            (more parameters as needed...)
          }
        },
        "parameters":[
          {
            "valueIsPercentage":true|false,
            "homeValueInDMX":{value},
            "twoBytes":true|false,
            "uniqueName":"{instrument name}.{parameter name}",
            "definitionParameter":{
              "isBroken":true|false,
              "homeValue":{value},
              "valueIsPercentage":true|false,
              "name":"{parameter name}",
              "type":"scalar"|"pantilt"|"rgbcolor"|"cmycolor"|"muxer",
```

```
            "twoBytes":true|false
          },
          "instrumentName":"{instrument name}",
          "homeValue":{value},
          "name":"{parameter name}"
        },
        (more parameters as needed...)
      ]
    },
  (more instruments as needed...)
 ]
}
```

/settings/light/undo

view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the Light section of Workspace Settings. If there is no valid action to undo, this message has no effect.

/settings/light/redo

view	edit	control	query
✗	✓	✗	✗

Redo the most recent un-done change in the Light section of Workspace Settings. If there is no valid action to redo, this message has no effect.

/settings/mic/patchList

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing all audio input patches defined in the workspace:

```
[
  {
    "uniqueID" : "{string}",
    "name" : "{string}",
  },
  { ... }
]
```

If an audio input patch does not have a name, QLab will use the name of the audio device assigned to that patch as its `name`.

NOTE: The `/mic` form of this message is changed in QLab 5.0 to list audio input patches. For output patches, use the `/audio` form instead.

/settings/mic/undo

view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the Mic section of Workspace Settings. If there is no valid action to undo, this message has no effect.

/settings/mic/redo

view	edit	control	query
✗	✓	✗	✗

Redo the most recent un-done change in the Mic section of Workspace Settings. If there is no valid action to redo, this message has no effect.

/settings/midi/patchList

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing all MIDI patches defined in the workspace:

```
[
  {
    "uniqueID" : string,
    "name" : string
  },
  { ... }
]
```

`{...}` represents a second MIDI patch; the number of patches in a workspace varies, so the number of items in this dictionary will vary.

/settings/midi/undo

view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the MIDI section of Workspace Settings. If there is no valid action to undo, this message has no effect.

/settings/midi/redo

view	edit	control	query
✗	✓	✗	✗

Redo the most recent un-done change in the MIDI section of Workspace Settings. If there is no valid action to redo, this message has no effect.

/settings/network/patchList

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing all network patches defined in the workspace:

```
[
  {
    "uniqueID" : string,
    "name" : string
  },
  { ... }
]
```

`{...}` represents a second network patch; the number of patches in a workspace varies, so the number of items in this dictionary will vary.

/settings/network/undo

view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the Network section of Workspace Settings. If there is no valid action to undo, this message has no effect.

/settings/network/redo

view	edit	control	query
✗	✓	✗	✗

Redo the most recent un-done change in the Network section of Workspace Settings. If there is no valid action to redo, this message has no effect.

/settings/video/inputPatchList

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing all video input patches defined in the workspace:

```
[
  {
    "uniqueID" : string,
    "name" : string
  },
  { ... }
]
```

`{...}` represents a second video input patch; the number of patches in a workspace varies, so the number of items in this dictionary will vary.

/settings/video/routes

view	edit	control	query
read only	read only	read only	✕

Return a JSON dictionary describing all video output routes defined in the workspace:

```
{
  "enableGuides":true|false,
  "destinationInfo":
  {
    "name":"Name of video device",
    "destinationType":
    "videoOutputKeyPath":"screen.screenID.1", -- macOS ID number of screen
    "videoOutputRasterSize":{"width":xxxx,"height":yyyy},
    "videoOutputRefreshRate":zzz -- frame rate in Hz
  },
  "rotationDegrees":0|90|180|270,
  "uniqueID":"some unique ID code",
  "scalingMode":0|1|2|3,
  "naturalSize":{"width":xxxx,"height":yyyy},
  "name":"Name of output route",
  "rear":true|false
},
{
  (more routes as needed...)
}
```

The contents of each `destinationInfo` dictionary will contain additional information which depends upon the type of output being described.

- Standard macOS-visible displays also have `screenID` and `screenSerialNumber`
- Blackmagic Designs devices also have `deckLinkHandle`, `displayName`, `displayMode`, `displayModeInfo`, and `keyingMode`
- NDI devices also have `pixelSize`, `audioChannels`, `audioSampleRate`, and `audioBufferSize`
- Syphon devices also have `pixelSize`

The `displayModeInfo` for Blackmagic Designs devices is itself a dictionary containing `identifier`, `displayName`, `pixelDimensions`, `fpsDuration`, `fpsScale`, and `fieldDominance`

/settings/video/route/{name}

/settings/video/routeIndex/{number}

/settings/video/routeID/{route_id}

view	edit	control	query
read only	read only	read only	✕

Return a JSON dictionary describing the specified video output route.

Routes are zero-indexed, and the index order is the order in which the routes appear in Workspace Settings → Video → Output Routing.

/settings/video/route/{name}/enableGuides {boolean}

/settings/video/routeIndex/{number}/enableGuides {boolean}

/settings/video/routeID/{route_id}/enableGuides {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the current visibility of the guides for the specified video output route.

Write: Set the visibility of the guides for the specified video output route to `true` or `false`. See details on booleans at the beginning of this section.

/settings/video/route/{name}/uniqueID

/settings/video/routeIndex/{number}/uniqueID

/settings/video/routeID/{route_id}/uniqueID

view	edit	control	query
read only	read only	read only	✗

Return the unique ID of the specified video output route.

/settings/video/stages

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing all stages defined in the workspace:

```
{
  "uniqueID":"unique ID of stage",
  "name":"name of stage",
  "width":xxxx,
  "height":yyyy,
  "regions":[
    {
      "boundsOnStage":{"y":yyyy,"x":xxxx,"width":wwww,"height":hhhh},
      "meshWidth":number,
      "warpType":number,
      "uniqueID":"unique ID of region",
      "controlPoints":[
        {"x":xxx,"y":yyy},
        { mode control points as needed...}
      ],
      "name":"A",
      "meshHeight":number
    },
    { more regions as needed... }
```

```
    ]
  },
  {
    ( more stages as needed )
  }
```

/settings/video/stage/{name}

/settings/video/stageID/{stage_id}

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing the specified stage.

/settings/video/stage/{current_name}/name {string}

/settings/video/stageID/{stage_id}/name {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the specified stage.

Write: If `string` is given, set the name of the specified stage to `string`.

/settings/video/stage/{name}/regions

/settings/video/stageID/{stage_id}/regions

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing the regions in the specified stage.

/settings/video/stage/{name}/size

/settings/video/stageID/{stage_id}/size

view	edit	control	query
read only	read only	read only	✓

Return a JSON dictionary describing the size of the specified stage.

/settings/video/stage/{name}/size/height

/settings/video/stageID/{stage_id}/size/height

view	edit	control	query
read only	read only	read only	✓

Return the height of the specified stage.

/settings/video/stage/{name}/size/width

/settings/video/stageID/{stage_id}/size/width

view	edit	control	query
read only	read only	read only	✓

Return the width of the specified stage.

/settings/video/stage/{name}/uniqueID

/settings/video/stageID/{stage_id}/uniqueID

view	edit	control	query
read only	read only	read only	✓

Return the unique ID of the specified stage.

/settings/video/stage/{name}/region/{name}

/settings/video/stage/{name}/regionID/{region_id}

/settings/video/stage/{name}/regionIndex/{index}

/settings/video/stageID/{stage_id}/region/{name}

/settings/video/stageID/{stage_id}/regionID/{region_id}

/settings/video/stageID/{stage_id}/regionIndex/{index}

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing the specified region.

/settings/video/stage/{name}/region/{name}/bounds {x} {y} {width} {height}

/settings/video/stage/{name}/region/{name}/bounds {string}

/settings/video/stage/{name}/regionID/{region_id}/bounds {x} {y} {width} {height}

/settings/video/stage/{name}/regionID/{region_id}/bounds {string}

/settings/video/stage/{name}/regionIndex/{index}/bounds {x} {y} {width} {height}

/settings/video/stage/{name}/regionIndex/{index}/bounds {string}

/settings/video/stageID/{stage_id}/region/{name}/bounds {x} {y} {width} {height}

/settings/video/stageID/{stage_id}/region/{name}/bounds {string}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds {x} {y} {width} {height}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds {string}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds {x} {y} {width} {height}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no arguments are given, return a JSON dictionary describing the position, width, and height of the specified region.

Write: Set the position and size of the specified region. Arguments can be provided either as four separate numbers or as single string in the form `"{{x,y}{width,height}}"`. In either case, all four numbers must be integers and must refer to a value within the bounds of the specified stage. `x` sets the horizontal position of the bottom left corner of the region, `y` sets the vertical position of the bottom left corner of the region, `width` sets the width of the region, and `height` sets the height of the region.

/settings/video/stage/{name}/region/{name}/bounds/origin {x} {y}

/settings/video/stage/{name}/region/{name}/bounds/origin {string}

/settings/video/stage/{name}/regionID/{region_id}/bounds/origin {x} {y}

/settings/video/stage/{name}/regionID/{region_id}/bounds/origin {string}

/settings/video/stage/{name}/regionIndex/{index}/bounds/origin {x} {y}

/settings/video/stage/{name}/regionIndex/{index}/bounds/origin {string}

/settings/video/stageID/{stage_id}/region/{name}/bounds/origin {x} {y}

/settings/video/stageID/{stage_id}/region/{name}/bounds/origin {string}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds/origin {x} {y}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds/origin {string}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds/origin {x} {y}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds/origin {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no arguments are given, return a JSON dictionary describing the position of the specified region.

Write: Set the position of the bottom left corner of the specified region. Arguments can be provided either as two separate numbers or as a single string in the form `"{x,y}"`. In either case, both numbers must be integers and must refer to a value within the bounds of the specified stage. `x` sets the horizontal position of the bottom left corner of the region, and `y` sets the vertical position of the bottom left corner of the region.

/settings/video/stage/{name}/region/{name}/bounds/origin/x {number}

/settings/video/stage/{name}/regionID/{region_id}/bounds/origin/x {number}

/settings/video/stage/{name}/regionIndex/{regionIndex}/bounds/origin/x {number}

/settings/video/stageID/{stage_id}/region/{name}/bounds/origin/x {number}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds/origin/x {number}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds/origin/x {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the horizontal position of the specified region.

Write: If `number` is given, set the horizontal position of the bottom left corner of the specified region to `number`. `Number` must be an integer and must refer to a value within the bounds of the specified stage.

/settings/video/stage/{name}/region/{name}/bounds/origin/y {number}

/settings/video/stage/{name}/regionID/{region_id}/bounds/origin/y {number}

/settings/video/stage/{name}/regionIndex/{regionIndex}/bounds/origin/y {number}

/settings/video/stageID/{stage_id}/region/{name}/bounds/origin/y {number}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds/origin/y {number}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds/origin/y {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the vertical position of the specified region.

Write: If `number` is given, set the vertical position of the bottom left corner of the specified region to `number`. `Number` must be an integer and must refer to a value within the bounds of the specified stage.

/settings/video/stage/{name}/region/{name}/bounds/size {width} {height}

/settings/video/stage/{name}/region/{name}/bounds/size {string}

/settings/video/stage/{name}/regionID/{region_id}/bounds/size {width} {height}

/settings/video/stage/{name}/regionID/{region_id}/bounds/size {string}

/settings/video/stage/{name}/regionIndex/{index}/bounds/size {width} {height}

/settings/video/stage/{name}/regionIndex/{index}/bounds/size {string}

/settings/video/stageID/{stage_id}/region/{name}/bounds/size {width} {height}

/settings/video/stageID/{stage_id}/region/{name}/bounds/size {string}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds/size {width} {height}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds/size {string}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds/size {width} {height}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds/size {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no arguments are given, return a JSON dictionary describing the size of the specified region.

Write: Set the size of the specified region. Arguments can be provided either as two separate numbers or as a single string in the form `"{width,height}"`. In either case, both numbers must be integers and must refer to a value within the bounds of the specified stage. `width` sets the width of the region, and `y` sets the height of the region.

/settings/video/stage/{name}/region/{name}/bounds/size/height {number}

/settings/video/stage/{name}/regionID/{region_id}/bounds/size/height {number}

/settings/video/stage/{name}/regionIndex/{index}/bounds/size/height {number}

/settings/video/stageID/{stage_id}/region/{name}/bounds/size/height {number}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds/size/height {number}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds/size/height {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the height of the specified region.

Write: If `number` is given, set the height of the specified region to `number`. `Number` must be an integer and must refer to a value within the bounds of the specified stage.

/settings/video/stage/{name}/region/{name}/bounds/size/width {number}

/settings/video/stage/{name}/regionID/{region_id}/bounds/size/width {number}

/settings/video/stage/{name}/regionIndex/{index}/bounds/size/width {number}

/settings/video/stageID/{stage_id}/region/{name}/bounds/size/width {number}

/settings/video/stageID/{stage_id}/regionID/{region_id}/bounds/size/width {number}

/settings/video/stageID/{stage_id}/regionIndex/{index}/bounds/size/width {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the width of the specified region.

Write: If `number` is given, set the width of the specified region to `number`. `Number` must be an integer and must refer to a value within the bounds of the specified stage.

/settings/video/stage/{name}/region/{name}/subregionIndex/{index}/controlPointIndex/{index} {x} {y}

/settings/video/stage/{name}/regionID/{region_id}/subregionIndex/{index}/controlPointIndex/{index} {x} {y}

/settings/video/stage/{name}/regionIndex/{index}/subregionIndex/{index}/controlPointIndex/{index} {x} {y}

/settings/video/stageID/{id}/region/{name}/subregionIndex/{index}/controlPointIndex/{index} {x} {y}

/settings/video/stageID/{id}/regionID/{region_id}/subregionIndex/{index}/controlPointIndex/{index} {x} {y}

/settings/video/stageID/{id}/regionIndex/{index}/subregionIndex/{index}/controlPointIndex/{index} {x} {y}

view	edit	control	query	+/-?
read	read/write	read	✗	✗

Read: If no arguments are given, return the coordinates of the specified control point.

Write: If `x` and `y` are given, set the coordinates of the specified control point to { `x` , `y` }. `x` and `y` must both be given together, must be integers, and must refer to a location within the bounds of the specified area.

/settings/video/stage/{name}/region/{name}/enableGrid {boolean}

/settings/video/stage/{name}/regionID/{region_id}/enableGrid {boolean}

/settings/video/stage/{name}/regionIndex/{index}/enableGrid {boolean}

/settings/video/stageID/{stage_id}/region/{name}/enableGrid {boolean}

/settings/video/stageID/{stage_id}/regionID/{region_id}/enableGrid {boolean}

/settings/video/stageID/{stage_id}/regionIndex/{index}/enableGrid {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the current visibility of the grid for the specified region.

Write: Set the visibility of the grid for the specified region to `true` or `false`. See details on booleans at the beginning of this section.

/settings/video/stage/{name}/region/{name}/enableGuide {boolean}

/settings/video/stage/{name}/regionID/{region_id}/enableGuide {boolean}

/settings/video/stage/{name}/regionIndex/{index}/enableGuide {boolean}

/settings/video/stageID/{stage_id}/region/{name}/enableGuide {boolean}

/settings/video/stageID/{stage_id}/regionID/{region_id}/enableGuide {boolean}

/settings/video/stageID/{stage_id}/regionIndex/{index}/enableGuide {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the current visibility of the guides for the specified region.

Write: Set the visibility of the guides for the specified region to `true` or `false`. See details on booleans at the beginning of this section.

/settings/video/stage/{name}/region/{name}/resetControlPoints

/settings/video/stage/{name}/regionID/{region_id}/resetControlPoints

/settings/video/stage/{name}/regionIndex/{index}/resetControlPoints

/settings/video/stageID/{stage_id}/region/{name}/resetControlPoints

/settings/video/stageID/{stage_id}/regionID/{region_id}/resetControlPoints

/settings/video/stageID/{stage_id}/regionIndex/{index}/resetControlPoints

view	edit	control	query
✗	✓	✗	✗

Set all control points for the specified region to their default initial positions.

/settings/video/undo

view	edit	control	query
✗	✓	✗	✗

Undo the most recent change in the Video section of Workspace Settings. If there is no valid action to undo, this message has no effect.

/settings/video/redo

view	edit	control	query
✗	✓	✗	✗

Redo the most recent un-done change in the Video section of Workspace Settings. If there is no valid action to redo, this message has no effect.

Show Control Broadcast messages

Show control broadcast messages are sent by QLab in response to requests from clients. Clients request updates by sending a `/listen` message in one or more of the forms below. Every `/listen` message has a corresponding `/ignore` message which stops updates from being sent by QLab.

The format of these messages can be modified to suit individual needs using the `/eventFormat` message. You can read about this message and how to use it in the section on customizing OSC show control broadcast messages.

Show control broadcast messages require view access.

/listen

Requests all show control broadcast messages to be sent to the client.

/listen/auditionGo

Requests show control broadcast messages to be sent to the client for audition GO events only. Each audition GO event will result in four messages being sent:

- `/qlab/event/workspace/auditionGo "{cue number}" "{cue name}" "{cue uniqueID}" "{cue type}"`
- `/qlab/event/workspace/auditionGo/number "{cue number}"`
- `/qlab/event/workspace/auditionGo/name "{cue name}"`
- `/qlab/event/workspace/auditionGo/uniqueID "{cue uniqueID}"`

/listen/auditionGo/number

Requests show control broadcast messages to be sent to the client for audition GO events only, and only in the `/qlab/event/workspace/auditionGo/number "{cue number}"` form.

/listen/auditionGo/name

Requests show control broadcast messages to be sent to the client for audition GO events only, and only in the `/qlab/event/workspace/auditionGo/number "{cue name}"` form.

/listen/auditionGo/uniqueID

Requests show control broadcast messages to be sent to the client for audition GO events only, and only in the `/qlab/event/workspace/auditionGo/number "{cue uniqueID}"` form.

/listen/cue/start

Requests show control broadcast messages to be sent to the client for cue start events only. Each audition GO event will result in four messages being sent:

- `/qlab/event/workspace/cue/start "{cue number}" "{cue name}" "{cue uniqueID}" "{cue type}"`
- `/qlab/event/workspace/cue/start/number "{cue number}"`
- `/qlab/event/workspace/cue/start/name "{cue name}"`
- `/qlab/event/workspace/cue/start/uniqueID "{cue uniqueID}"`

/listen/cue/start/number

Requests show control broadcast messages to be sent to the client for cue start events only, and only in the `/qlab/event/workspace/cue/start/number "{cue number}"` form.

/listen/cue/start/name

Requests show control broadcast messages to be sent to the client for cue start events only, and only in the `/qlab/event/workspace/cue/start/name "{cue name}"` form.

/listen/cue/start/uniqueID

Requests show control broadcast messages to be sent to the client for cue start events only, and only in the `/qlab/event/workspace/cue/start/uniqueID "{cue uniqueID}"` form.

/listen/cue/stop

Requests show control broadcast messages to be sent to the client for cue stop events only. Each audition GO event will result in four messages being sent:

- `/qlab/event/workspace/cue/stop "{cue number}" "{cue name}" "{cue uniqueID}" "{cue type}"`
- `/qlab/event/workspace/cue/stop/number "{cue number}"`
- `/qlab/event/workspace/cue/stop/name "{cue name}"`
- `/qlab/event/workspace/cue/stop/uniqueID "{cue uniqueID}"`

/listen/cue/stop/number

Requests show control broadcast messages to be sent to the client for cue stop events only, and only in the `/qlab/event/workspace/cue/stop/number "{cue number}"` form.

/listen/cue/stop/name

Requests show control broadcast messages to be sent to the client for cue stop events only, and only in the `/qlab/event/workspace/cue/stop/name "{cue name}"` form.

/listen/cue/stop/uniqueID

Requests show control broadcast messages to be sent to the client for cue stop events only, and only in the `/qlab/event/workspace/cue/stop/uniqueID "{cue uniqueID}"` form.

/listen/go

Requests show control broadcast messages to be sent to the client for GO events only. Each GO event will result in four messages being sent:

- `/qlab/event/workspace/go "{cue number}" "{cue name}" "{cue uniqueID}" "{cue type}"`
- `/qlab/event/workspace/go/number "{cue number}"`
- `/qlab/event/workspace/go/name "{cue name}"`
- `/qlab/event/workspace/go/uniqueID "{cue uniqueID}"`

/listen/go/number

Requests show control broadcast messages to be sent to the client for GO events only, and only in the `/qlab/event/workspace/go/number "{cue number}"` form.

/listen/go/name

Requests show control broadcast messages to be sent to the client for GO events only, and only in the `/qlab/event/workspace/go/number "{cue name}"` form.

/listen/go/uniqueID

Requests show control broadcast messages to be sent to the client for GO events only, and only in the `/qlab/event/workspace/go/number "{cue uniqueID}"` form.

/listen/hardStopAll

Requests show control broadcast messages to be sent to the client for HardStop All events only.

/listen/panicAll

Requests show control broadcast messages to be sent to the client for Panic All events only.

/listen/pauseAll

Requests show control broadcast messages to be sent to the client for Pause All events only.

/listen/playhead

Requests show control broadcast messages to be sent to the client for playhead move events only. Each playhead move event will result in four messages being sent:

- `/qlab/event/workspace/playhead "{cue number}" "{cue name}" "{cue uniqueID}" "{cue type}"`
- `/qlab/event/workspace/playhead/number "{cue number}"`
- `/qlab/event/workspace/playhead/name "{cue name}"`
- `/qlab/event/workspace/playhead/uniqueID "{cue uniqueID}"`

/listen/playhead/number

Requests show control broadcast messages to be sent to the client for playhead move events only, and only in the `/qlab/event/workspace/playhead/number "{cue number}"` form.

/listen/playhead/name

Requests show control broadcast messages to be sent to the client for playhead move events only, and only in the `/qlab/event/workspace/playhead/number "{cue name}"` form.

/listen/playhead/uniqueID

Requests show control broadcast messages to be sent to the client for playhead move events only, and only in the `/qlab/event/workspace/playhead/number "{cue uniqueID}"` form.

/listen/resetAll

Requests show control broadcast messages to be sent to the client for Reset All events only.

/listen/resumeAll

Requests show control broadcast messages to be sent to the client for Resume All events only.

/listen/stopAll

Requests show control broadcast messages to be sent to the client for Stop All events only.

/ignore

Stops all show control broadcast messages from being sent to the client.

/ignore/auditionGo

Stops show control broadcast messages from being sent to the client for audition GO events only.

/ignore/auditionGo/number

Stops show control broadcast messages from being sent to the client for audition GO events only, and only in the `/qlab/event/workspace/auditionGo/number "{cue number}"` form.

/ignore/auditionGo/name

Stops show control broadcast messages from being sent to the client for audition GO events only, and only in the `/qlab/event/workspace/auditionGo/number "{cue name}"` form.

/ignore/auditionGo/uniqueID

Stops show control broadcast messages from being sent to the client for audition GO events only, and only in the `/qlab/event/workspace/auditionGo/number "{cue uniqueID}"` form.

/ignore/cue/start

Stops show control broadcast messages from being sent to the client for cue start events only.

/ignore/cue/start/number

Stops show control broadcast messages from being sent to the client for cue start events only, and only in the `/qlab/event/workspace/cue/start/number "{cue number}"` form.

/ignore/cue/start/name

Stops show control broadcast messages from being sent to the client for cue start events only, and only in the `/qlab/event/workspace/cue/start/name "{cue name}"` form.

/ignore/cue/start/uniqueID

Stops show control broadcast messages from being sent to the client for cue start events only, and only in the `/qlab/event/workspace/cue/start/uniqueID "{cue uniqueID}"` form.

/ignore/cue/stop

Stops show control broadcast messages from being sent to the client for cue stop events only.

/ignore/cue/stop/number

Stops show control broadcast messages from being sent to the client for cue stop events only, and only in the `/qlab/event/workspace/cue/stop/number "{cue number}"` form.

/ignore/cue/stop/name

Stops show control broadcast messages from being sent to the client for cue stop events only, and only in the `/qlab/event/workspace/cue/stop/name "{cue name}"` form.

/ignore/cue/stop/uniqueID

Stops show control broadcast messages from being sent to the client for cue stop events only, and only in the `/qlab/event/workspace/cue/stop/uniqueID "{cue uniqueID}"` form.

/ignore/go

Stops show control broadcast messages from being sent to the client for GO events only.

/ignore/go/number

Stops show control broadcast messages from being sent to the client for GO events only, and only in the `/qlab/event/workspace/go/number "{cue number}"` form.

/ignore/go/name

Stops show control broadcast messages from being sent to the client for GO events only, and only in the `/qlab/event/workspace/go/number "{cue name}"` form.

/ignore/go/uniqueID

Stops show control broadcast messages from being sent to the client for GO events only, and only in the `/qlab/event/workspace/go/number "{cue uniqueID}"` form.

/ignore/hardStopAll

Stops show control broadcast messages from being sent to the client for HardStop All events only.

/ignore/panicAll

Stops show control broadcast messages from being sent to the client for Panic All events only.

/ignore/pauseAll

Stops show control broadcast messages from being sent to the client for Pause All events only.

/ignore/playhead

Stops show control broadcast messages from being sent to the client for playhead move events only.

/ignore/playhead/number

Stops show control broadcast messages from being sent to the client for playhead move events only, and only in the `/qlab/event/workspace/playhead/number "{cue number}"` form.

/ignore/playhead/name

Stops show control broadcast messages from being sent to the client for playhead move events only, and only in the `/qlab/event/workspace/playhead/number "{cue name}"` form.

/ignore/playhead/uniqueID

Stops show control broadcast messages from being sent to the client for playhead move events only, and only in the `/qlab/event/workspace/playhead/number "{cue uniqueID}"` form.

/ignore/resetAll

Stops show control broadcast messages from being sent to the client for Reset All events only.

/ignore/resumeAll

Stops show control broadcast messages from being sent to the client for Resume All events only.

/ignore/stopAll

Stops show control broadcast messages from being sent to the client for Stop All events only.

Cue messages

Cue OSC messages use the form `/workspace/{id}/cue/{cue_number}/command`, where `{id}` may be either the display name of the workspace, such as `hamlet.qlab5`, or the unique ID of the workspace, which can be found in the Info tab of the Workspace Status Window.

Note, however, that addressing by display name will work only if the display name is composed of characters allowed in OSC method names. This does NOT include spaces, unicode characters, diacriticals, or other "special" characters.

Addressing a workspace by its unique ID looks like this:

```
/workspace/1B11984A-3EBC-4A9C-A004-B9E3AA32DA6B/cue/x/start
```

Addressing a workspace by its display name looks like this:

```
/workspace/hamlet/cue/x/start
```

Note that the extension of the filename, the `.qlab5` part, is omitted.

If you send a cue message without the `/workspace/{id}` portion of the address, then the message will be sent to all open workspaces listening on the port to which the message is sent. So, if your `hamlet.qlab5` workspace is the only open workspace, or the only workspace listening to a particular port, and you send QLab the OSC command `/cue/x/start` to that port, then cue `x` in the workspace called `hamlet.qlab5` will start. If both `hamlet.qlab5` and `twelfthnight.qlab5` are open and listening to the same port, and both have a cue numbered `x`, then sending the OSC command `/cue/x/start` to that port will cause cue `x` in both workspaces to start.

Addressing Cues

Cues can be addressed either by their cue number or their unique ID. Cues always have a unique ID. They do not always have a number, but if it exists it will be unique within the workspace.

For all cue OSC messages, any instance of the address pattern `/cue/{cue_number}` can be replaced by the equivalent unique ID address pattern `/cue_id/{id}`.

Additionally, QLab supports a few special addresses for cues:

- **/cue/selected** addresses the currently selected cue or cues.
- **/cue/playhead** addresses the cue that's currently standing by at the playhead.
- **/cue/playbackPosition** is the same as `/cue/playhead`.
- **/cue/active** addresses all active cues (currently playing or paused).

Finally, because QLab supports OSC address patterns, you may use an asterisk `*` and a question mark as wildcards within `{cue_number}` or `{id}`.

Examples

- `/cue/*/armed 0` would disarm all cues in the workspace; `*` matches any character or characters.
- `/cue/?/armed 0` would disarm all cues in the workspace with a single-character cue number; `?` matches any single character.
- `/cue/[*]/armed 0` would disarm only cues which have a cue number, and have no effect on cues which have no cue number.
- `/cue/understudy-*/colorName green` would set green as the display color for all cues that have a number that starts with the string "understudy-".

Important: Spaces are not permitted inside OSC addresses, so cue numbers with spaces in them will not work properly with OSC. If you are using OSC to control your workspace, avoid using spaces in cue numbers.

/cue/{cue_number}/actionElapsed

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the elapsed action (in seconds) of the specified cue.

/cue/{cue_number}/percentActionElapsed

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the elapsed action (as a percentage of the total action) of the specified cue.

/cue/{cue_number}/allowsEditingDuration

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue has an editable duration, such as an Audio, Video, or Fade cue. Otherwise, return `false`.

/cue/{cue_number}/armed {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the armed state of the specified cue.

Write: If `boolean` is true, arm the specified cue. If false, disarm the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/auditionGo

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Tell the specified cue to audition GO. Auditioning a cue starts it according to the audition behavior defined for its output type.

/cue/{cue_number}/auditionPreview

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Audition preview the specified cue. Audition previewing a cue starts it according to the audition behavior defined for its output type, skipping over its pre-wait time, and does not advance the playhead. Also, if the cue has an auto-follow or auto-continue, the followed or continued cue is not started.

/cue/{cue_number}/autoLoad {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the auto-load state of the specified cue.

Write: If `boolean` is true, set the specified cue to auto-load. If false, set the specified cue to not auto-load. See details on booleans at the beginning of this section.

/cue/{cue_number}/canHaveAudioMapTargets

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue can target an audio map, such as a Fade or Reset cue. Otherwise, return `false`.

/cue/{cue_number}/canHavePatchTargets

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue can target a patch, such as a Fade or Reset cue. Otherwise, return `false`.

/cue/{cue_number}/captureTimecode

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Set the timecode trigger time of the the specified cue to the currently incoming timecode. If there is no incoming timecode, this message has no effect.

/cue/{cue_number}/cartPosition/

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return an array with the row and column numbers for the specified cue's position within a cart. A cue that is not contained within a cart will return `[0,0]`.

/cue/{cue_number}/cartPosition/column

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the column number for the specified cue's position within a cart. A cue that is not contained within a cart will return `0`.

/cue/{cue_number}/cartPosition/row

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the row number for the specified cue's position within a cart. A cue that is not contained within a cart will return `0`.

/cue/{cue_number}/children/

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

If the specified cue is a list, cart, or Group cue, return an array of cue dictionaries listing the following information about all cues within the specified list, cart, or Group cue:

```
[
  {
    "number":"{string}",
    "uniqueID":"{string}",
    "cues":[ {a cue dictionary}, {another dictionary}, {and another} ],
    "flagged":true|false,
    "listName":"{string}",
    "type":"{string}",
    "colorName":"{string}",
```

```
    "colorName/live": "{string}",
    "name":"{string}",
    "armed":true|false
  },
  { ... }
]
```

The “cues” item is relevant only if the cue is a Group cue which contains other cues, in which case the “cues” item will contain cue dictionaries for each of the children of the enclosing Group cue. If any of those children are themselves Group cues, with their own children, then it’s Group turtles all the way down.

{...} represents a second cue dictionary; the number of cues within the specified list, cart, or Group cue will vary, so the number of items in this dictionary will vary.

If the specified cue is not a list, cart, or Group cue, this message will return an OSC reply containing an empty “data” field.

/cue/{cue_number}/children/shallow

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

This message behaves identically to the above message (`/cue/{cue_number}/children`) except that it omits the “cues” item, and therefore returns a smaller set of information. Nested Groups will not be queried, and so only the first “layer” of the cue hierarchy will be returned.

/cue/{cue_number}/children/uniqueIDs

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

If the specified cue is a list, cart, or Group cue, return an array of cue dictionaries listing the following information about all cues within the specified list, cart, or Group cue:

```
[
  {
    "uniqueID":"{string}",
    "cues":[ {a cue dictionary}, {another dictionary}, {and another} ]
  },
  { ... }
]
```

The “cues” item is relevant only if the cue is a Group cue which contains other cues, in which case the “cues” item will contain cue dictionaries which contain the `uniqueID` and `cues` for each of the children of the enclosing Group cue.

{...} represents a second cue dictionary; the number of cues within the specified list, cart, or Group cue will vary, so the number of items in this dictionary will vary.

/cue/{cue_number}/children/uniqueIDs/shallow

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

This message behaves identically to the above message (`/cue/{cue_number}/children/uniqueIDs`) except that it omits the “cues” item, and therefore returns a smaller set of information. Nested Groups will not be queried, and so only the first “layer” of the cue hierarchy will be returned.

/cue/{cue_number}/colorCondition {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the color condition mode of the specified cue.

Write: If `number` is given, set the color condition mode of the specified cue to `number` . Valid color condition modes are:

```
0 - Always
1 - Before action
2 - After action
```

Removed in QLab 5.2. This message works in QLab 5.0.x and QLab 5.1.x only. In QLab 5.2 and later, use the `/useSecondColor` message instead.

/cue/{cue_number}/colorName {string}

/cue/{cue_number}/colorName/live {string}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the color of the specified cue.

Write: If `string` is given, set the color of the specified cue to `string` . As of QLab 5.2, valid colors are:

```
berry
blue
crimson
cyan
forest
gray
green
hot pink
indigo
lavender
magenta
midnight
olive
orange
peach
plum
purple
```

```
red
sky blue
yellow
```

and

```
none
```

Certain other colors may also be valid...

/cue/{cue_number}/secondColorName {string}

/cue/{cue_number}/secondColorName/live {string}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the second color of the specified cue.

Write: If `string` is given, set the second color of the specified cue to `string`.

/cue/{cue_number}/useSecondColor {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Use second color* lock button of the specified cue.

Write: Set the state of the *Use second color* lock button of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/continueMode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the continue mode of the specified cue.

Write: If `number` is given, set the continue mode of the specified cue to `number`. Valid continue modes are:

```
0 - No continue
1 - Auto-continue
2 - Auto-follow
```

/cue/{cue_number}/cueTargetID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the cue ID of the target of the specified cue. Empty string (`""`) means that the cue has no cue target.

Write: If `string` is given, and if the specified cue can have cue targets, set the target of the specified cue to `string` . If `string` is `"none"` or empty (`""`), unset the cue target. If `string` is anything else, it must be a valid uniqueID of a cue in the workspace.

/cue/{cue_number}/cueTargetNumber {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the cue number of the target of the specified cue. Empty string (`""`) means that the cue has no cue target.

Write: If `string` is given, and if the specified cue can have cue targets, set the target of the specified cue to `string` . `string` must be a valid cue number of a cue in the workspace.

/cue/{cue_number}/currentCueTarget

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the cue ID of the current target of the specified cue. Empty string (`""`) means that the cue has no current target.

Deprecated in QLab 5.5. This message works but will be removed in a future version of QLab. Use `/currentCueTargetID` instead.

/cue/{cue_number}/currentCueTargetID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the cue ID of the current target of the specified cue. Empty string (`""`) means that the cue has no current target.

/cue/{cue_number}/currentCueTargetNumber

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the cue number of the current target of the specified cue. Empty string (`""`) means that the cue has no current target.

/cue/{cue_number}/tempCueTargetNumber {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, and the specified cue has a temporary target, return the cue number of that temporary target. Empty string (`""`) means that the cue has no temporary target.

Write: If `string` is given, and if the specified cue can have cue targets, temporarily set the target of the specified cue to `string` . The specified cue will revert to its previous target if it is reset, if the workspace is reset, or if the workspace is closed and reopened. `string` must be a valid cue number of a cue in the workspace.

/cue/{cue_number}/tempCueTargetID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

This works exactly the same as `/cueTargetID` , but refers to the temporary target. See `/tempCueTargetNumber` for more detail about temporary targets.

/cue/{cue_number}/defaultName

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the default name of the specified cue.

/cue/{cue_number}/displayName

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the display name of the specified cue.

/cue/{cue_number}/duckLevel {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the duck or boost level of the specified cue.

Write: If `number` is given, set the duck or boost level of the specified cue to `number` . If `number` is a negative number, it will set a duck level. If `number` is a positive number, it will set a boost level. `number` can be any number, decimals allowed, from `-120` to `12` .

NOTE: This message worked differently in QLab 4; it used a scaled range of `0` to `4` , in which `0 = -120` dB, `1 = 0` dB, and `4 = +12` dB. In QLab 5, the message simply uses decibel values directly.

/cue/{cue_number}/duckOthers {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Duck audio of other cues* checkbox of the specified cue.

Write: Set the state of the *Duck audio of other cues* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/duckTime {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the *Duck audio of other cues* time of the specified cue.

Write: If `number` is given, set the *Duck audio of other cues* time of the specified cue to `number`.

/cue/{cue_number}/duration {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the duration of the specified cue.

Write: If `number` is given, and the selected cue has an editable duration, set the duration of the specified cue to `number`.

/cue/{cue_number}/currentDuration {number}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the current duration of the specified cue.

/cue/{cue_number}/tempDuration {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, and if the specified cue has a temporary duration, return that temporary duration.

Write: If `number` is given, and if the specified cue has a duration, temporarily set the duration of the specified cue to `number`. The specified cue will revert to its previous duration if it is reset, if the workspace is reset, or if the workspace is closed and reopened.

/cue/{cue_number}/fadeAndStopOthers {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the *Fade and stop* mode of the specified cue.

Write: If `number` is given, set the *Fade and stop* mode of the specified cue to `number`. Valid modes are:

```
0 - None
1 - Peers
2 - List or cart
3 - All
```

Mode `0` is equivalent to the *Fade and stop* checkbox being unchecked.

/cue/{cue_number}/fadeAndStopOthersTime {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the *Fade and stop others* time of the specified cue.

Write: If `number` is given, set the *Fade and stop others* time of the specified cue to `number`.

/cue/{cue_number}/fileTarget {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the target of the specified cue. Empty string (`""`) means that the cue has no file target.

Write: If `string` is given, and if the specified cue can have file targets, set the target of the specified cue to `string`. If `string` is `"none"` or empty (`""`), unset the file target. If `string` is anything else, `string` should be a file path and name in any of the following three forms:

- Full paths, e.g. `/Volumes/MyDisk/path/to/some/file.wav`
- Paths beginning with a tilde, e.g. `~/path/to some/file.mov`
- Relative paths, e.g. `this/is/a/relative/path.mid`

Paths beginning with a tilde (~) will be expanded; the tilde signifies “relative to the user’s home directory”.

Relative paths will be interpreted according to the current working directory. Use the `/workingDirectory` application message to set or get the current working directory.

/cue/{cue_number}/flagged {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Flagged* checkbox of the specified cue.

Write: Set the flagged state of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/go

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is not a cue list, tell QLab to move the playhead to cue `cue_number` and then GO.

If the specified cue is a cue list, tell that cue list to GO. This GO respects the current playback position for that list, as well as double go protection for the workspace.

/cue/{cue_number}/hardPause

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Pause the specified cue without allowing audio effects to decay naturally. If the specified cue is not playing, this message has no effect.

/cue/{cue_number}/hardStop

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Stop the specified cue without allowing audio effects to decay naturally. If the specified cue is not playing, this message has no effect.

/cue/{cue_number}/hasCueTargets

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is able to target another cue. Examples of such a cue include Fade cues and Stop cues.

/cue/{cue_number}/hasFileTargets

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is able to target a file. Examples of such a cue are Audio, Video, and MIDI File cues.

/cue/{cue_number}/isActionRunning

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the action of the specified cue (not the pre-wait or post-wait) is running.

/cue/{cue_number}/isAuditioning

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is auditioning.

/cue/{cue_number}/isBroken

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is broken.

/cue/{cue_number}/isCrossfadingOut

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is crossfading out. This only applies to cues inside Playlist Group cues.

/cue/{cue_number}/isLoaded

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is loaded.

/cue/{cue_number}/isNextInPlaylist

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is the next cue in a Playlist Group. “Next” in this case means the next to begin playing, either automatically when the currently playing cue in the Playlist Group ends, or manually if the currently playing cue in the Playlist Group is of infinite duration.

/cue/{cue_number}/isOverridden

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue’s output is currently suppressed by an override control.

/cue/{cue_number}/isPanicking

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is panicking.

/cue/{cue_number}/isPaused

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is paused.

/cue/{cue_number}/isRunning

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue is running.

/cue/{cue_number}/isTailingOut

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue has an audio effect which is decaying.

/cue/{cue_number}/isWarning

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return `true` if the specified cue has a non-breaking warning.

/cue/{cue_number}/listName

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the list name of the specified cue. The list name is the name that is displayed in the cue list, which can be either the default name, a manually set display name, or nothing.

/cue/{cue_number}/load

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Load the specified cue.

/cue/{cue_number}/loadAt {number}

/cue/{cue_number}/loadAt {hours} {minutes} {seconds}

view	edit	control	query	+/-?
✗	✗	✓	✗	✓

If a single argument, `number`, is given and is a positive number, load the specified cue to `number` seconds. If `number` is a negative number, load the specified cue to `number` seconds before the end of the cue. If the cue has a pre-wait, that time is counted as part of the load time.

If three arguments are given, and each are positive numbers, load the specified cue to `hours`:`minutes`:`seconds`. Negative numbers are not accepted. If the cue has a pre-wait, that time is counted as part of the load time.

If no argument is given, this command is equivalent to `load`.

Examples:

- `/cue/10/loadAt 15` loads cue `10` to 15 seconds.
- `/cue/20/loadAt -30` loads cue `20` to 30 seconds before the end (so that, for example, you can rehearse the last 30 seconds of a dance number.)
- `/cue/30/loadAt 0 4 6.5` load cue `30` to `4:06.5` (four minutes, six point five seconds.)

/cue/{cue_number}/loadActionAt {number}

/cue/{cue_number}/loadActionAt {hours} {minutes} {seconds}

view	edit	control	query	+/-?
✗	✗	✓	✗	✓

If a single argument, `number`, is given and is a positive number, load the specified cue to `number` seconds. If `number` is a negative number, load the specified cue to `number` seconds before the end of the cue. QLab will automatically add the pre-wait of the specified cue so that the cue is loaded to the given time as though it has no pre-wait.

If three arguments are given, and each are positive numbers, load the specified cue to `hours`:`minutes`:`seconds`. Negative numbers are not accepted. QLab will automatically add the pre-wait of the specified cue so that the cue is loaded to the given time as though it has no pre-wait.

If no argument is given, this command is equivalent to `load`.

/cue/{cue_number}/loadFileAt {number}

/cue/{cue_number}/loadFileAt {hours} {minutes} {seconds}

view	edit	control	query	+/-?
✗	✗	✓	✗	✓

If a single argument, `number`, is given and is a positive number, load the specified cue to `number` seconds ignoring any slice loop counts. If `number` is a negative number, load the specified cue to `number` seconds before the end of the cue. QLab will automatically add the pre-wait of the specified cue so that the cue is loaded to the given time as though it has no pre-wait.

If three arguments are given, and each are positive numbers, load the specified cue to `hours`:`minutes`:`seconds` ignoring any slice loop counts. Negative numbers are not accepted. QLab will automatically add the pre-wait of the specified cue so that the cue is loaded to the given time as though it has no pre-wait.

If no argument is given, this command is equivalent to `load`, but ignores slice loop counts.

/cue/{cue_number}/loadAndSetPlayhead

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Move the playhead to the specified cue and load that cue.

/cue/{cue_number}/maxTimeInCueSequence

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the maximum time required to complete the cue sequence starting at the specified cue, as used e.g. for the “load to time” slider. Any infinite loops within the sequence are only counted once.

/cue/{cue_number}/name {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the specified cue.

Write: If `string` is given, set the name of the specified cue to `string`.

/cue/{cue_number}/notes {string}

view	edit	control	query	+/-?
read/write	read/write	read/write	✓	✗

Read: If no argument is given, return the notes of the specified cue.

Write: If `string` is given, set the notes of the specified cue to `string`.

/cue/{cue_number}/number {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the cue number of the specified cue.

Write: If `string` is given, set the cue number of the specified cue to `string`.

/cue/{cue_number}/panic

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Panic the specified cue. Panicked cues fade out and stop over the panic duration specified in the General section of Workspace Settings.

/cue/{cue_number}/panicInTime {number}

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Panic the specified cue, using `number` for the panic duration instead of the panic duration specified in Workspace Settings.

/cue/{cue_number}/parent

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the cue ID of the parent of the specified cue. If the specified cue is inside a Group, then the Group is the parent. Otherwise, the cue list or cue cart that contains the cue is the parent.

/cue/{cue_number}/patchTargetID {id}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the ID of the patch that is targeted by the specified cue.

Write: If `id` is given, set the target of the specified cue to the patch with ID `id`.

/cue/{cue_number}/pause

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Pause the specified cue, allowing any audio effects on the cue to decay naturally. If the specified cue is not playing, this message has no effect.

/cue/{cue_number}/togglePause

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is playing, pause it. If the specified cue is paused, resume it. If the specified cue is not playing and not paused, this message has no effect.

/cue/{cue_number}/postWait {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the post-wait of the specified cue.

Write: If `number` is given, set the post-wait of the specified cue to `number`.

/cue/{cue_number}/postWaitElapsed

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the elapsed post-wait time (in seconds) of the specified cue.

/cue/{cue_number}/percentPostWaitElapsed

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the elapsed post-wait time (as a percentage of the total post-wait time) of the specified cue.

/cue/{cue_number}/preview

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Preview the specified cue. Previewing a cue starts it, skipping over its pre-wait time, and does not advance the playhead. Also, if the cue has an auto-follow or auto-continue, the followed or continued cue is not started.

/cue/{cue_number}/preWait {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the pre-wait of the specified cue.

Write: If `number` is given, set the pre-wait of the specified cue to `number`.

/cue/{cue_number}/preWaitElapsed

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the elapsed pre-wait time (in seconds) of the specified cue.

/cue/{cue_number}/percentPreWaitElapsed

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the elapsed pre-wait time (as a percentage of the total pre-wait time) of the specified cue.

/cue/{cue_number}/reset

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Reset the specified cue. Resetting a cue returns any temporary changes (such as those caused by a “live” OSC method) to be reverted.

/cue/{cue_number}/resume

view	edit	control	query	+/-?

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Resume the specified cue. If the specified cue is not paused, this message has no effect.

/cue/{cue_number}/secondTriggerAction {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the second trigger action of the specified cue.

Write: If `number` is given, set the second trigger action of the specified cue to `number`. Valid actions are:

```
0 - Do nothing
1 - Panic
2 - Stop
3 - Hard stop
4 - Hard stop & restart
5 - Devamp
6 - Playlist Next
7 - Playlist Previous
```

Options `6` and `7` are only valid when sent to a Playlist Group cue.

/cue/{cue_number}/secondTriggerOnRelease {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Second trigger on release* checkbox of the specified cue.

Write: Set the state of the *Second trigger on release* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/skipIfDisarmed {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Skip if disarmed* checkbox of the specified cue.

Write: Set the state of the *Skip if disarmed* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/soloCueInTime {number}

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Fade and stop all other cues in the same cue list as the specified cue over `number` seconds. `number` is required, and can be any number greater than or equal to zero, decimals allowed.

/cue/{cue_number}/start

/cue/start {cue_number}

/cue_id/start {cue_id}

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Start the specified cue. This does not move the playhead.

/cue/{cue_number}/startAndAutoloadNext

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Start the specified cue and load the following cue or cue sequence if that cue or cue sequence is set to auto-load. This does not move the playhead.

/cue/{cue_number}/stop

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Stop the specified cue. If the specified cue is not playing, this message has no effect.

/cue/{cue_number}/targetMode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return `0` if the specified cue targets a cue or `1` if the specified cue targets a patch.

Write: If `number` is given, set the target mode of the specified cue. Valid target modes are:

```
0 - Cue target
1 - Patch target
```

/cue/{cue_number}/timecodeTrigger

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return the timecode time of the specified cue's timecode trigger.

/cue/{cue_number}/timecodeTrigger/hours {number}

view	edit	control	query	+/-?
read only	read/write	read only	✓	✗

Read: If no argument is given, return the “hours” portion of the specified cue’s timecode trigger.

Write: If `number` is given, set the “hours” portion of the specified cue’s timecode trigger.

/cue/{cue_number}/timecodeTrigger/minutes {number}

view	edit	control	query	+/-?
read only	read/write	read only	✓	✗

Read: If no argument is given, return the “minutes” portion of the specified cue’s timecode trigger.

Write: If `number` is given, set the “minutes” portion of the specified cue’s timecode trigger.

/cue/{cue_number}/timecodeTrigger/seconds {number}

view	edit	control	query	+/-?
read only	read/write	read only	✓	✗

Read: If no argument is given, return the “seconds” portion of the specified cue’s timecode trigger.

Write: If `number` is given, set the “seconds” portion of the specified cue’s timecode trigger.

/cue/{cue_number}/timecodeTrigger/frames {number}

view	edit	control	query	+/-?
read only	read/write	read only	✓	✗

Read: If no argument is given, return the “frames” portion of the specified cue’s timecode trigger.

Write: If `number` is given, set the “frames” portion of the specified cue’s timecode trigger.

/cue/{cue_number}/timecodeTrigger/bits {number}

view	edit	control	query	+/-?
read only	read/write	read only	✓	✗

Read: If no argument is given, return the “bits” portion of the specified cue’s timecode trigger.

Write: If `number` is given, set the “bits” portion of the specified cue’s timecode trigger.

/cue/{cue_number}/timecodeTrigger/text {string}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
read only	read/write	read only	✓	✗

Read: If no argument is given, return the specified cue's timecode trigger as a text string.

Write: If `string` is given, set the specified cue's timecode trigger to `string`. `string` must be in the form of a valid timecode entry.

/cue/{cue_number}/type

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the type (Audio, Video, Fade, etc.) of the specified cue.

/cue/{cue_number}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the uniqueID of the specified cue.

/cue/{cue_number}/valuesForKeys {json_string}

view	edit	control	query	+/-?
✓	✓	✓	✗	✗

This special method can be used to request a custom collection of state information about the specified cue. `json_string` must be a JSON-formatted string representing an array of keys you wish to query.

Example:

```
/cue/2/valuesForKeys "[\"opacity\",\"surfaceSize\"]"
```

would return the values for the "opacity" and "surfaceSize" properties of cue 2, assuming cue 2 is a Video, Camera, or Text cue.

Invalid use of this message has no effect.

/cue/{cue_number}/valuesForKeysWithArguments {json_string}

view	edit	control	query	+/-?
✓	✓	✓	✗	✗

This special method can be used to request a custom collection of state information about the given cue. `json_string` must be a JSON-formatted string representing a dictionary of keys and arguments you wish to query.

Example:

```
/cue/2/valuesForKeysWithArguments "{\"level\":[0,0]}"
```

would return a dictionary that contains the value of the main volume level of cue 2, assuming cue 2 has audio levels. Note that this method is limited to returning one value per key, even if you send multiple keys with different arguments.

Group cue/List/Cart messages

Messages specific to Group cues, cue lists, and cue carts (which are really also Group cues.)

/cue/{cue_number}/cartColumns

/cue/{cue_number}/cartRows

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

If the specified cue is a cart, these messages return the number of rows or columns in the cart. If the specified cue is not a cart, these messages have no effect.

/cue/{cue_number}/collapse

view	edit	control	query
✗	✓	✗	✗

If the specified Group cue is expanded, displaying its children, collapse it.

/cue/{cue_number}/expand

view	edit	control	query
✗	✓	✗	✗

If the specified Group cue is collapsed, not displaying its children, expand it.

/cue/{cue_number}/currentTimecode

/cue/{cue_number}/currentTimecode/hours

/cue/{cue_number}/currentTimecode/minutes

/cue/{cue_number}/currentTimecode/seconds

/cue/{cue_number}/currentTimecode/frames

/cue/{cue_number}/currentTimecode/bits

/cue/{cue_number}/currentTimecode/text

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

If the specified cue is a cue list or cart which is set to receive timecode, these messages return the all or part of the current incoming timecode, either as a number or as a string.

/cue/{cue_number}/isChildAuditioning

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

If the specified cue is a list, cart, or Group cue, return `true` if there is at least one currently auditioning cue amongst the children of the specified cue. If there are none, return `false`.

If the specified cue is not a list, cart, or Group cue, this message has no effect.

/cue/{cue_number}/isChildFlagged

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

If the specified cue is a list, cart, or Group cue, return `true` if there is at least one flagged cue amongst the children of the specified cue. If there are none, return `false`.

If the specified cue is not a list, cart, or Group cue, this message has no effect.

/cue/{cue_number}/mode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the mode of the specified Group cue, list, or cart.

Write: If `number` is given, set the mode of the specified Group cue. Valid modes are:

```
0 - List
1 - Start first and enter
2 - Start first
3 - Timeline
4 - Start random
5 - Cart
6 - Playlist
```

Modes `0` and `5` are read-only; a cue list will return mode `0`, but mode cannot be set to `0`, and a cue cart will return mode `5`, but mode cannot be set to `5`.

/cue/{cue_number}/moveCartCue/{child} {row} {column}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

If the specified cue is a cart, then move child cue `child` to position `row`, `column` within the cart.

`child` can be the cue number or cue ID of the child cue. `row` and `column` must be valid for the specified cart cue.

If the specified cue is not a cart, this message has no effect.

/cue/{cue_number}/playhead {string}

/cue/{cue_number}/playhead/{string}

/cue/{cue_number}/playbackPosition {string}

view	edit	control	query	+/-?
read	read	read/write	✓	✗

Read: If no argument is given, return the cue number of the standing-by cue, or “none” if there is no cue standing by.

Write: If the specified cue is a cue list, set the playhead (playback position). `string` can be:

- The cue number of a cue in the list
- `next` , which will move the playhead down by one cue.
- `previous` , which will move the playhead up by one cue.
- `none` , which will unset the playhead.

/cue/{cue_number}/playheadID {string}

/cue/{cue_number}/playheadID/{string}

/cue/{cue_number}/playbackPositionID {string}

view	edit	control	query	+/-?
read	read	read/write	✓	✗

Read: If no argument is given, return the cue ID of the standing-by cue, or “none” if there is no cue standing by.

Write: If the specified cue is a cue list, set the playhead (playback position). `string` can be:

- The cue number of a cue in the list
- `next` , which will move the playhead down by one cue.
- `previous` , which will move the playhead up by one cue.
- `none` , which will unset the playhead.

/cue/{cue_number}/playhead/next

/cue/{cue_number}/playbackPosition/next

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is a cue list, set the playhead in that cue list to the next cue.

/cue/{cue_number}/playhead/none

/cue/{cue_number}/playbackPosition/none

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is a cue list, unset the playhead in that cue list.

/cue/{cue_number}/playhead/previous

/cue/{cue_number}/playbackPosition/previous

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is a cue list, set the playhead in that cue list to the previous cue.

/cue/{cue_number}/playhead/nextSequence

/cue/{cue_number}/playbackPosition/nextSequence

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is a cue list, set the playhead in that cue list to the next cue sequence.

/cue/{cue_number}/playhead/previousSequence

/cue/{cue_number}/playbackPosition/previousSequence

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is a cue list, set the playhead in that cue list to the previous cue sequence.

/cue/{cue_number}/playlist/currentCue {string}

/cue/{cue_number}/playlist/currentCueID {string}

view	edit	control	query	+/-?
✗	✗	✓	✓	✗

Read: If no argument is given, return the cue number or ID of the current cue of the specified Playlist Group cue.

Write: Set the current cue of the specified Playlist Group cue to `string`.

If the specified cue is not a Playlist Group cue, this message has no effect.

/cue/{cue_number}/playlist/doCrossfade {boolean}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the state of the *Crossfade* checkbox of the specified cue.

Write: Enable or disable crossfading on the specified Playlist Group cue. See details on booleans at the beginning of this section.

If the specified cue is not a Playlist Group cue, this message has no effect.

/cue/{cue_number}/playlist/doLoop {boolean}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the state of the *Loop* checkbox of the specified Playlist Group cue.

Write: Enable or disable looping on the specified Playlist Group cue. See details on booleans at the beginning of this section.

If the specified cue is not a Playlist Group cue, this message has no effect.

/cue/{cue_number}/playlist/doShuffle {boolean}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the state of the *Shuffle* checkbox of the specified Playlist Group cue.

Write: Enable or disable shuffle on the specified Playlist Group cue. See details on booleans at the beginning of this section.

If the specified cue is not a Playlist Group cue, this message has no effect.

/cue/{cue_number}/playlist/next

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is a Playlist Group cue, start the next cue in that playlist.

/cue/{cue_number}/playlist/previous

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is a Playlist Group cue, start the previous cue in that playlist.

/cue/{cue_number}/playlist/crossfade/duration {number}

/cue/{cue_number}/playlistCrossfadeDuration {number}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the crossfade duration of the specified Playlist Group cue.

Write: If `number` is given, set the crossfade time of the specified Playlist Group cue to `number` seconds. `number` can be any whole or decimal number greater than or equal to zero.

If the specified cue is not a Playlist Group cue, this message has no effect.

`playlistCrossfadeDuration` is deprecated in QLab 5.0.4. This message works but will be removed in a future version of QLab. Use `/playlist/crossfade/duration` instead.

/cue/{cue_number}/playlistCrossfade {boolean}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the state of the *Crossfade* checkbox of the specified cue.

Write: Enable or disable crossfading on the specified cue. See details on booleans at the beginning of this section.

If the specified cue is not a Playlist Group cue, this message has no effect.

Deprecated in QLab 5.0.4. This message works but will be removed in a future version of QLab. Use `/playlist/doCrossfade` instead.

/cue/{cue_number}/playlistLoop {boolean}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the state of the *Loop* checkbox of the specified cue.

Write: Enable or disable looping on the specified cue. See details on booleans at the beginning of this section.

If the specified cue is not a Playlist Group cue, this message has no effect.

Deprecated in QLab 5.0.4. This message works but will be removed in a future version of QLab. Use `/playlist/doLoop` instead.

/cue/{cue_number}/playlistShuffle {boolean}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the state of the *Shuffle* checkbox of the specified cue.

Write: Enable or disable shuffle on the specified cue. See details on booleans at the beginning of this section.

If the specified cue is not a Playlist Group cue, this message has no effect.

Deprecated in QLab 5.0.4. This message works but will be removed in a future version of QLab. Use `/playlist/doShuffle` instead.

/cue/{cue_number}/shuffle

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

If the specified cue is a [ ] Group cue, shuffle the contents of the specified cue.

/cue/{cue_number}/timecodeFreewheelTime

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If `number` is not given, return the timecode freewheel time of the specified list or cart.

Write: Set the timecode freewheel time of the specified list or cart to `number` seconds. `number` can be any number from 0 to 2, decimals allowed.

If the specified cue is not a list or cart, this message has no effect. Also, note that this setting is separate from the question of whether the list or cart is set to sync to timecode at all.

/cue/{cue_number}/timecodeLookbackTime {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If `number` is not given, return the timecode lookback time of the specified list or cart.

Write: Set the timecode lookback time of the specified list or cart to `number` seconds. `number` can be any number greater than or equal to 0, decimals allowed.

If the specified cue is not a list or cart, this message has no effect. Also, note that this setting is separate from the question of whether the list or cart is set to sync to timecode at all.

/cue/{cue_number}/timecodeSMPTEFormat {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the SMPTE format of the specified list or cart.

Write: If `number` is given, set the SMPTE format of the list or cart. Valid formats are:

```
0  - 24 fps
1  - 25 fps
2  - 30 fps drop-frame
3  - 30 fps
```

If the specified cue is not a list or cart, this message has no effect. Also, note that this setting is separate from the question of whether the list or cart is set to sync to timecode at all.

/cue/{cue_number}/timecodeStartBehavior {number}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the timecode start behavior of the specified list or cart.

Write: If `number` is given, set the timecode start behavior of the list or cart. Valid behaviors are:

```
0 - do not start cues whose triggers precede the incoming timecode
1 - start cues whose triggers fall within the most recent minute of timecode
2 - start cues whose triggers fall within the most recent hour of timecode
3 - start cues whose triggers fall within the lookback time
4 - start all cues which precede the timecode
```

If the specified cue is not a list or cart, this message has no effect. Also, note that this setting is separate from the question of whether the list or cart is set to sync to timecode at all.

/cue/{cue_number}/timecodeStopBehavior {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the timecode stop behavior of the specified list or cart.

Write: If `number` is given, set the timecode stop behavior of the list or cart. Valid behaviors are:

```
0 - do nothing
1 - pause timecode triggered cues
2 - stop timecode triggered cues
```

If the specified cue is not a list or cart, this message has no effect. Also, note that this setting is separate from the question of whether the list or cart is set to sync to timecode at all.

/cue/{cue_number}/timecodeSyncMode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the timecode sync mode of the specified list or cart.

Write: If `number` is given, set the timecode sync mode of the list or cart. Valid timecode sync modes are:

```
0 - MTC (MIDI timecode)
1 - LTC (Linear timecode)
```

If the specified cue is not a list or cart, this message has no effect. Also, note that this setting is separate from the question of whether the list or cart is set to sync to timecode at all.

Audio cue messages

Messages specific to Audio cues. Most of these messages work with Mic cues, Video cues, and Camera cues, as well.

/cue/{cue_number}/addSliceMarker {time} {play_count}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Add a slice marker to the specified cue. If `time` is given, add the marker at that time. If not, add the marker at the current time of the cue. `time` can be any positive whole or decimal number.

If `play_count` is given, set the play count of the new slice (the slice preceding the new marker) to `play_count`. If `play_count` is not given, the play count will be `1`. `play_count` can be any positive whole number, or `-1` to set the slice to infinite loop.

Slice markers are zero-indexed, meaning the first marker of a cue is marker 0.

/cue/{cue_number}/audioMap

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return a array containing all the details of the map in use by the specified cue. These details include the width, height, and unique ID of the map, and nested arrays for the filters, marks, and map objects in the map, containing all the attributes of each item like spread, name, unique ID, etc. The attributes in each array will differ, of course, since filters, marks, and map objects all have different attributes.

/cue/{cue_number}/audioMap/filters

/cue/{cue_number}/audioMap/marks

/cue/{cue_number}/audioMap/objects

/cue/{cue_number}/audioMap/size

/cue/{cue_number}/audioMap/size/height

/cue/{cue_number}/audioMap/size/width

/cue/{cue_number}/audioMap/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return a subset of the data returned by `/cue/{cue_number}/audioMap`.

/cue/{cue_number}/audioMap/filter/{name}

/cue/{cue_number}/audioMap/filterID/{id}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array containing all the attributes of the specified filter in the map of the specified cue.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/cue/{cue_number}/audioMap/filter/{name}/angle {number}

/cue/{cue_number}/audioMap/filterID/{id}/angle {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the angle of the specified filter in degrees away from horizontal.

Write: If `number` is given, set the angle of the specified filter to `number`. `number` can be any whole or decimal number from `-180` to `180`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/cue/{cue_number}/audioMap/filter/{name}/passthrough/{output} {boolean}

/cue/{cue_number}/audioMap/filterID/{id}/passthrough/{output} {boolean}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

`output` is required and must be an integer from 1 to 128.

Read: If no argument is given, return `TRUE` if the specified output is set to pass through the filter or `FALSE` if it is not.

Write: Check or uncheck the passthrough checkbox of the specified output in the specified filter. See details on booleans at the beginning of this section.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/cue/{cue_number}/audioMap/filter/{name}/passthroughs

/cue/{cue_number}/audioMap/filterID/{id}/passthroughs

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the passthrough checkboxes of the specified filter.

/cue/{cue_number}/audioMap/filter/{name}/name {string}

/cue/{cue_number}/audioMap/filterID/{id}/name {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the name of the specified filter.

Write: If `string` is given, set the name of the specified filter to `string`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/cue/{cue_number}/audioMap/filter/{name}/position {x} {y}

/cue/{cue_number}/audioMap/filterID/{id}/position {x} {y}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no arguments are given, return the position of the specified filter.

Write: If `x` and `y` are given, set the position of the specified filter to `(x,y)`. `x` and `y` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/cue/{cue_number}/audioMap/filter/{name}/position/x {number}

/cue/{cue_number}/audioMap/filterID/{id}/position/x {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the X-axis position of the specified filter.

Write: If `number` is given, set the X-axis position of the specified filter to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/cue/{cue_number}/audioMap/filter/{name}/position/y {number}

/cue/{cue_number}/audioMap/filterID/{id}/position/y {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the Y-axis position of the specified filter.

Write: If `number` is given, set the Y-axis position of the specified filter to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/cue/{cue_number}/audioMap/filter/{name}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the unique ID of the specified filter.

/cue/{cue_number}/audioMap/filter/{name}/width {number}

/cue/{cue_number}/audioMap/filterID/{id}/width {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the width of the specified filter.

Write: If `number` is given, set the width of the specified filter to `number`. `number` can be any whole or decimal number `0` or greater, limited to half of the largest dimension of the map.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/filter/*B`) and bracketed ranges /(e.g. `/filter/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}

/cue/{cue_number}/audioMap/markID/{id}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array containing all the attributes of the specified mark in the map of the audio output patch of the specified cue.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}/gravity {number}

/cue/{cue_number}/audioMap/markID/{id}/gravity {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the gravity of the specified mark.

Write: If `number` is given, set the gravity of the specified mark to `number`. `number` can be any whole or decimal number from `0.1` to `5.0`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}/level/{outChannel} {decibel}

/cue/{cue_number}/audioMap/markID/{id}/level/{outChannel} {decibel}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

`outChannel` is required and must be an integer from 0 to 128.

Read: If no `decibel` is given, return the volume level of the specified output.

Write: If `decibel` is given, set the level of the specified output to `decibel`. `decibel` can be any number, decimals allowed, or a string such as `-INF`. If `decibel` is a string (any string, in fact) QLab will set the level to `-INF`,

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}/levels

/cue/{cue_number}/audioMap/markID/{id}/levels

view	edit	control	query	+/-?

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the levels of the specified mark.

/cue/{cue_number}/audioMap/mark/{name}/name {string}

/cue/{cue_number}/audioMap/markID/{id}/name {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the name of the specified mark.

Write: If `string` is given, set the name of the specified mark to `string`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}/position {x} {y}

/cue/{cue_number}/audioMap/markID/{id}/position {x} {y}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no arguments are given, return the position of the specified mark.

Write: If `x` and `y` are given, set the position of the specified mark to `(x,y)`. `x` and `y` can be any whole or decimal numbers.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}/position/x {number}

/cue/{cue_number}/audioMap/markID/{id}/position/x {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the X-axis position of the specified mark.

Write: If `number` is given, set the X-axis position of the specified mark to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}/position/y {number}

/cue/{cue_number}/audioMap/markID/{id}/position/y {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the Y-axis position of the specified mark.

Write: If `number` is given, set the Y-axis position of the specified mark to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}/shadow {number}

/cue/{cue_number}/audioMap/markID/{id}/shadow {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the shadow angle of the specified mark.

Write: If `number` is given, set the shadow angle of the specified mark to `number`. `number` can be any whole or decimal number from `0` to `90`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/mark/*B`) and bracketed ranges /(e.g. `/mark/[2-6]`).

/cue/{cue_number}/audioMap/mark/{name}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the unique ID of the specified mark.

/cue/{cue_number}/audioMap/object/{name}

/cue/{cue_number}/audioMap/objectID/{id}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array containing all the attributes of the specified map object in the map of the specified cue.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/cue/{cue_number}/audioMap/object/{name}/colorName {number}

/cue/{cue_number}/audioMap/objectID/{id}/colorName {number}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the color of the specified map object.

Write: If `string` is given, set the color of the specified map object to `string`. Valid colors are:

```
berry
blue
crimson
cyan
```

```
forest
gray
green
hot pink
indigo
lavender
magenta
midnight
olive
orange
peach
plum
purple
red
sky blue
yellow
```

and

```
none
```

Certain other colors may also be valid...

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/cue/{cue_number}/audioMap/object/{name}/name {string}

/cue/{cue_number}/audioMap/objectID/{id}/name {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Read: If no argument is given, return the name of the specified map object.

Write: If `string` is given, set the name of the specified map object to `string`.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/cue/{cue_number}/audioMap/object/{name}/position {x} {y}

/cue/{cue_number}/audioMap/object/{name}/position/live {x} {y}

/cue/{cue_number}/audioMap/objectID/{id}/position {x} {y}

/cue/{cue_number}/audioMap/objectID/{id}/position/live {x} {y}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no arguments are given, return the position of the specified map object.

Write: If `x` and `y` are given, set the position of the specified map object to `(x,y)`. `x` and `y` can be any whole or decimal numbers.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/audioMap/object/{name}/position/x {number}

/cue/{cue_number}/audioMap/object/{name}/position/x/live {number}

/cue/{cue_number}/audioMap/objectID/{id}/position/x {number}

/cue/{cue_number}/audioMap/objectID/{id}/position/x/live {number}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no argument is given, return the X-axis position of the specified map object.

Write: If `number` is given, set the X-axis position of the specified map object to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/audioMap/object/{name}/position/y {number}

/cue/{cue_number}/audioMap/object/{name}/position/y/live {number}

/cue/{cue_number}/audioMap/objectID/{id}/position/y {number}

/cue/{cue_number}/audioMap/objectID/{id}/position/y/live {number}

view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no argument is given, return the Y-axis position of the specified map object.

Write: If `number` is given, set the Y-axis position of the specified map object to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/audioMap/object/{name}/spread {number}

/cue/{cue_number}/audioMap/object/{name}/spread/live {number}

/cue/{cue_number}/audioMap/objectID/{id}/spread {number}

/cue/{cue_number}/audioMap/objectID/{id}/spread/live {number}



view	edit	control	query	+/-?	live
✗	✓	✗	✓	✓	control

Read: If no argument is given, return the spread of the specified map object.

Write: If `number` is given, set the spread of the specified map object to `number`. `number` can be any whole or decimal number from `0` to twice the largest dimension of the map.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/audioMap/object/{name}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the unique ID of the specified map object.

/cue/{cue_number}/audioMapID {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the map ID of the audio map currently in use by the specified cue. Empty string (`""`) means that the cue is not using an audio map.

Write: If `string` is given and matches the map ID of an audio map in the workspace, set the audio map of the specified cue to that map. If `string` is `"none"` or empty (`""`), set the specified cue to use no map.

/cue/{cue_number}/audioMapName {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the name of the audio map currently in use by the specified cue. Empty string (`""`) means that the cue is not using an audio map.

Write: If `string` is given and matches the name of an audio map in the workspace, set the audio map of the specified cue to that map. If `string` is `"none"` or empty (`""`), set the specified cue to use no map.

/cue/{cue_number}/audioMapNumber {number}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
✗	✓	✗	✓	✓

Read: If no argument is given, return the index of the audio map currently in use by the specified cue. `0` means that the cue is not using an audio map.

Write: If `number` is given, set the audio map of the specified cue to that map, according to the order of maps as listed in *Workspace Settings* → *Audio* → *Audio Maps*. If `number` is `0`, set the specified cue to use no map.

/cue/{cue_number}/audioOutputPatch

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing the specified audio output patch currently in use by the specified cue.

```
[
  {
    "name" : "{string}",
    "uniqueID" : "{string}",
    "routing" : [number,number,number,...]
  }
]
```

The numbers listed for `routing` are the device output numbers which have cue outputs routed to them. For example, if the patch is using a device with four outputs, and all four device outputs have cue outputs routed to them, the list will be `[1,2,3,4]`. If device output 3 has no cue outputs routed to it, the list will be `[1,2,4]`.

/cue/{cue_number}/audioOutputPatch/cueOutputChannels {number}

/cue/{cue_number}/audioOutputPatch/cueOutputChannels {number}

view	edit	control	query	+/-?
read	read/write	✗	✓	✗

Read: If no argument is given, return the number of cue outputs of the audio output patch currently in use by the specified cue.

Write: If `number` is given, set the number of cue outputs of the audio output patch currently in use by the specified cue to `number`, which must be a whole number from 1 to 128.

/cue/{cue_number}/audioOutputPatch/level/{inChannel}/{outChannel} {decibel}

/cue/{cue_number}/audioOutputPatch/level/{inChannel}/{outChannel}/live {decibel}

view	edit	control	query	+/-?	live
read	read/write	✗	✓	✓	control

Read: If no argument is given, return the volume level of the specified crosspoint of the audio output patch currently in use by the specified cue.

Write: If `decibel` is given, set the specified crosspoint of the audio output patch currently in use by the specified cue to `decibel`. `decibel` can be any number, decimals allowed, or a string such as `-INF`. If `decibel` is a string (any string, in fact) QLab will set the crosspoint to `-INF`.

/cue/{cue_number}/audioOutputPatch/mute {output} {boolean}

/cue/{cue_number}/audioOutputPatch/mute/{output} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `TRUE` if the device output of the audio output patch currently in use by the specified cue is muted, or `FALSE` if it is not.

Write: Mute or unmute the specified device output of the audio output patch currently in use by the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/audioOutputPatch/mute/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-mute all device outputs of the specified cue.

/cue/{cue_number}/audioOutputPatch/muteChannels

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the numbers of device outputs that are muted.

/cue/{cue_number}/audioOutputPatch/name {string}

view	edit	control	query	+/-?
✗	✓	✗	✓	✗

Set the name of the specified patch to `string`.

/cue/{cue_number}/audioOutputPatch/reset

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Reset all faded parameters of the audio output patch currently in use by the specified cue (cue output effects, routing, and device output effects) to their saved state.

/cue/{cue_number}/audioOutputPatch/routing

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return the device output numbers which have cue outputs routed to them. For example, if the patch is using a device with four outputs, and all four device outputs have cue outputs routed to them, the list will be `[1,2,3,4]`. If device output 3 has no cue outputs routed to it, the list will be `[1,2,4]`.

/cue/{cue_number}/audioOutputPatch/routing/reset

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Reset faded audio levels of the audio output patch currently in use by the specified cue to their saved state.

/cue/{cue_number}/audioOutputPatch/solo {output} {boolean}

/cue/{cue_number}/audioOutputPatch/solo/{output} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `TRUE` if the device output of the audio output patch currently in use by the specified cue is soloed, or `FALSE` if it is not.

Write: Solo or un-solo the specified device output of the audio output patch currently in use by the specified cue. See details on booleans at the beginning of this section.

Starting with QLab 5.5, `{output}` can include OSC wildcard patterns such as asterisks (e.g. `/solo "*L"`) and bracketed ranges (e.g. `/solo [2-6]`).

/cue/{cue_number}/audioOutputPatch/solo/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-solo all device outputs of the specified cue.

/cue/{cue_number}/audioOutputPatch/soloChannels

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the numbers of device outputs that are soloed.

/cue/{cue_number}/audioOutputPatch/uniqueID

view	edit	control	query

view	edit	control	query
read only	read only	read only	✗

Return the unique ID of the audio output patch currently in use by the specified cue.

/cue/{cue_number}/audioOutputPatchName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the audio output patch currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the name of an audio output patch in the workspace, set the audio output patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/audioOutputPatchNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the index of the audio output patch currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first patch in the audio output patch list in Workspace Settings, `2` means the second patch, and so on.

Write: If `number` is given, set the audio output patch of the specified cue to that patch. If `number` is `0` , un-patch the specified cue. If `number` is greater than the number of audio output patches in the workspace, this message has no effect. `number` must be a whole number.

/cue/{cue_number}/audioOutputPatchID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the patch ID of the audio output patch currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of an audio output patch in the workspace, set the audio output patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. For Video and Camera cues which use an NDI device for output, `string` may also be `"ndi-output"` . If `string` is anything else, this message has no effect.

/cue/{cue_number}/audioTrackFormats

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return a JSON dictionary containing any available metadata about the file target of the specified cue. The exact form and contents of the dictionary varies depending upon the type of file, but generally includes things like channel count, sample rate, and file format.

This message may be most helpful when used with 🎞 Video cues.

/cue/{cue_number}/audioTrackID

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return the ID of the track currently in use by the specified cue. This is only relevant if the file target of the specified cue is a video file which contains multiple audio tracks.

This message may be most helpful when used with 🎞 Video cues.

/cue/{cue_number}/currentFileTime

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

If the specified cue has a file target, return the current playback time of the target file in seconds.

Examples

- If the cue is not running, the playback time is `0`.
- If the cue has been playing for ten seconds, and the playback rate of the cue is `1.0`, then the playback time is `10`.
- If the cue has been playing for fifteen seconds, and the playback rate of the cue is `0.5`, then the playback time is `7.5`.

/cue/{cue_number}/deleteSliceMarker {index}

/cue/{cue_number}/deleteSliceMarker/{index}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Delete slice marker `index` of the specified cue. `index` can be zero or any positive whole number.

/cue/{cue_number}/deleteSliceMarkers

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Delete all slice markers of the specified cue.

/cue/{cue_number}/doFade {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Use integrated fade* checkbox of the specified cue.

Write: Enable or disable the integrated fade curve of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/endTime {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the end time of the specified cue.

Write: If `number` is given, set the end time of the specified cue to `number` seconds. `number` can be any whole or decimal number greater than or equal to zero.

/cue/{cue_number}/gang {inChannel} {outChannel} {gang}

/cue/{cue_number}/gang/{inChannel}/{outChannel} {gang}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

`inChannel` is required and must be an integer from `0` to `24`. `0` is the main column.

`outChannel` is required and must be either an integer from `0` to `128`, or a string that is the name of a cue output. `0` is the main row.

Read: If no `gang` is given, return the gang of the specified crosspoint.

Write: If `gang` is given, set the gang of the specified crosspoint. `gang` must be a string. While it's not technically necessary, in practice at least two crosspoints must have exactly matching `gang` strings for it to really do anything.

/cue/{cue_number}/infiniteLoop {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the specified cue is set to loop infinitely, and `false` if it is not.

Write: Set the infinite loop of the specified cue to `true` or `false`. See details on booleans at the beginning of this section.

/cue/{cue_number}/inputChannelName/{number} {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of audio input `number` of the specified cue. Inputs are numbered starting at `1`.

Write: Set the name of input `number` to `string`.

/cue/{cue_number}/lastSliceInfiniteLoop {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the last slice of the specified cue is set to loop infinitely, and `false` if it is not.

Write: Set the infinite loop of the last slice of the specified cue to `true` or `false`. See details on booleans at the beginning of this section.

/cue/{cue_number}/lastSlicePlayCount {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the play count of the last slice of the specified cue.

Write: If `number` is given, set the play count of the last slice of the specified cue to `number`. `number` can be any positive whole number, or `-1` to set the slice to infinite loop.

/cue/{cue_number}/level {inChannel} {outChannel} {decibel}

/cue/{cue_number}/level/{inChannel}/{outChannel} {decibel}

/cue/{cue_number}/level/{inChannel}/{outChannel}/live {decibel}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

`inChannel` is required and must be an integer from 0 to 24. 0 is the main column.

`outChannel` is required and must be either an integer from 0 to 128, or a string that is the name of a cue output. 0 is the main row.

Read: If no `decibel` is given, return the volume level of the specified crosspoint.

Write: If `decibel` is given, set the specified crosspoint to `decibel`. `decibel` can be any number, decimals allowed, or a string such as `-INF`. If `decibel` is a string (any string, in fact) QLab will set the crosspoint to `-INF`.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/levels

/cue/{cue_number}/levels/live

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	view

Return all the audio levels currently available in the Levels tab of the specified cue's inspector. The levels are returned as an array of arrays, like so:

```
[
  [ main level, output 1 level, output 2 level, ...],
  [ input 1 main level, input 1/output 1 level, input 1/output 2 level, ...],
  [ ... ]
]
```

`[ ... ]` represents a second row in the crosspoint matrix, which is present if and only if the specified cue has at least two audio input channels. There could be as many as 24 rows.

Row 0 of `/levels` is equivalent to the results of the `/sliderLevels` method.

/cue/{cue_number}/liveAverageLevel/{outputChannel} {low} {high}

/cue/{cue_number}/liveAverageLevel/{outputChannel}/{low}/{high}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the RMS level of `outputChannel`. `outputChannel` can be either the number or name of a cue output.

`low` and `high` are optional numerical values. If they are present, they will cause QLab to re-scale the output of this message to a range of values from `low` to `high`.

Examples:

- `/cue/1/liveAverageLevel/1 0 100` will return the level of cue output 1 of cue 1 on a scale of `0` (silent) to `100` (as loud as possible).
- / `cue/10/liveAverageLevel/center/-20/20` will return the level of the cue output named "center" of cue 10 on a scale of `-20` (silent) to `20` (as loud as possible).

/cue/{cue_number}/liveAverageObjectLevel/{name} {low} {high}

/cue/{cue_number}/liveAverageObjectIDLevel/{id} {low} {high}

/cue/{cue_number}/liveAverageObjectLevel/{name}/{low}/{high}

/cue/{cue_number}/liveAverageObjectIDLevel/{id}/{low}/{high}

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the RMS level of the specified object.

`low` and `high` are optional numerical values. If they are present, they will cause QLab to re-scale the output of this message to a range of values from `low` to `high`.

Examples:

- `/cue/1/liveAverageObjectLevel/A 0 100` will return the level of the object named "A" of cue 1 on a scale of `0` (silent) to `100` (as loud as possible).
- / `cue/10/liveAverageObjectLevel/B/-20/20` will return the level of the object named "B" of cue 10 on a scale of `-20` (silent) to `20` (as loud as possible).

/cue/{cue_number}/lockFadeToCue {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Lock fade to start/end* checkbox of the specified cue.

Write: Set the state of the *Lock fade to start/end* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/mute {output} {boolean}

/cue/{cue_number}/mute/channel/{output} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `TRUE` if the cue output of the specified cue is muted, or `FALSE` if it is not.

Write: Mute or unmute the specified cue output of the specified cue. See details on booleans at the beginning of this section. You can refer to the cue output by its number or its name, if it has one.

/cue/{cue_number}/mute/object/{name} {boolean}

/cue/{cue_number}/mute/objectID/{id} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `TRUE` if the cue output of the specified cue is muted, or `FALSE` if it is not.

Write: Mute or unmute the specified object of the specified cue. See details on booleans at the beginning of this section..

/cue/{cue_number}/mute/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-mute all cue outputs and all objects of the specified cue.

/cue/{cue_number}/mute/channel/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-mute all cue outputs of the specified cue.

/cue/{cue_number}/mute/object/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-mute all objects of the specified cue.

/cue/{cue_number}/muteChannels

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the cue outputs that are muted.

/cue/{cue_number}/muteObjects

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the objects that are muted.

/cue/{cue_number}/numChannelsIn

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return the number of input channels in the specified cue.

/cue/{cue_number}/object/{name}

/cue/{cue_number}/objectID/{id}

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return an array containing all the details of the specified object in the specified cue. These details include color, name, position, spread, and unique ID of the object.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/cue/{number}/object/{name}/colorName {string}

/cue/{number}/objectID/{id}/colorName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the color of the specified object in the specified cue.

Write: If `string` is given, set the color of the specified object in the specified cue. `string` can be any valid cue color.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/cue/{number}/object/{name}/name {string}

/cue/{number}/objectID/{id}/name {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the specified object in the specified cue.

Write: If `string` is given, set the name of the specified object in the specified cue. `string` can be any text, but if you plan to refer to the object using OSC it must not include spaces or any reserved character such as `\`, `*`, `?`, etc.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/cue/{number}/object/{name}/position {x} {y}

/cue/{number}/object/{name}/position/live {x} {y}

/cue/{number}/objectID/{id}/position {x} {y}

/cue/{number}/objectID/{id}/position/live {x} {y}

view	edit	control	query	+/-?	live
read	read/write	read	✗	✗	control

Read: If no arguments are given, return the position of the specified object in the specified cue.

Write: If `x` and `y` are given, set the position of the specified object in the specified cue to `(x,y)`. `x` and `y` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{number}/object/{name}/position/x {number}

/cue/{number}/object/{name}/position/x/live {number}

/cue/{number}/objectID/{id}/position/x {number}

/cue/{number}/objectID/{id}/position/x/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✗	✓	control

Read: If no argument is given, return the X-axis position of the specified object in the specified cue.

Write: If `number` is given, set the X-axis position of the specified object in the specified cue to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{number}/object/{name}/position/y {number}

/cue/{number}/object/{name}/position/y/live {number}

/cue/{number}/objectID/{id}/position/y {number}

/cue/{number}/objectID/{id}/position/y/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✗	✓	control

Read: If no argument is given, return the Y-axis position of the specified object in the specified cue.

Write: If `number` is given, set the Y-axis position of the specified object in the specified cue to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

/cue/{number}/object/{name}/spread {number}

/cue/{number}/object/{name}/spread/live {number}

/cue/{number}/objectID/{id}/spread {number}

/cue/{number}/objectID/{id}/spread/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✕	✕	control

Read: If no argument is given, return the spread of the specified object in the specified cue.

Write: If `number` is given, set the spread of the specified object in the specified cue to `number`. `number` can be any whole or decimal number.

`{name}` can include OSC wildcard patterns such as asterisks (e.g. `/object/*B`) and bracketed ranges /(e.g. `/object/[2-6]`).

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{number}/object/{name}/uniqueID

/cue/{number}/objectID/{id}/uniqueID

view	edit	control	query	+/-?
read only	read only	read only	✕	✕

Return the unique ID of the specified object in the specified cue.

/cue/{number}/objects

view	edit	control	query	+/-?
read only	read only	read only	✕	✕

Return an array of dictionaries containing all the details of all the objects in the specified cue. These details include color, name, position, spread, and unique ID.

/cue/{cue_number}/objectLevel/{row}/{name} {decibel}

/cue/{cue_number}/objectLevel/{row}/{name}/live {decibel}

/cue/{cue_number}/objectIDLevel/{row}/{id} {decibel}

/cue/{cue_number}/objectIDLevel/{row}/{id}/live {decibel}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

`row` is required and must be an integer from 0 to 24. 0 is the main row.

`name` or `id` is required and must be a string that matches the name or ID of an object in the specified cue or a map object visible to the selected cue.

Read: If no `decibel` is given, return the volume level of the specified crosspoint.

Write: If `decibel` is given, set the specified crosspoint to `decibel`. `decibel` can be any number, decimals allowed, or a string such as `-INF`. If `decibel` is a string (any string, in fact) QLab will set the crosspoint to `-INF`,

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{number}/objectLevels

/cue/{number}/objectLevels/live

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	view

Return all the audio levels currently available in the Objects tab of the specified cue's inspector. The levels are returned as an array of arrays, like so:

```
[
  [ main level, object 1 level, object 2 level, ...],
  [ input 1 main level, input 1/object 1 level, input 1/object 2 level, ...],
  [ ... ]
]
```

`[ ... ]` represents a second row in the crosspoint matrix, which is present if and only if the specified cue has at least two audio input channels. There could be as many as 24 rows.

/cue/{cue_number}/patch {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the index of the patch of the specified cue.

Write: If `number` is given, set the patch of the specified cue. `number` must be a positive whole number.

Deprecated in QLab 5.0. This message works in QLab 5, but incompletely, and will be removed in a future version of QLab. Use `/audioOutputPatchNumber` or `/audioOutputPatchID` instead.

/cue/{cue_number}/patchList

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Deprecated in QLab 5.0. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/settings/audio/patchList`.

/cue/{cue_number}/playCount {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the play count of the specified cue.

Write: If `number` is given, set the play count (number of times to loop) of the specified cue to `number`. `number` can be any whole number greater than zero.

/cue/{cue_number}/preservePitch {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Preserve pitch* checkbox of the specified cue.

Write: Enable or disable the *Preserve pitch* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/rate {number}

/cue/{cue_number}/rate/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the rate of the specified cue.

Write: If `number` is given, set the rate of the specified cue to `number`. `number` can be any positive whole or decimal number from 0.03 to 33.0.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/setDefaultLevels

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Set the audio levels of the specified cue to the default levels set in the cue template for the specified cue's type.

/cue/{cue_number}/setSilentLevels

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Set the audio levels of the specified cue to silent.

/cue/{cue_number}/sliceMarker {index} {time} {play_count}

/cue/{cue_number}/sliceMarker/{index} {time} {play_count}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

`index` is required and can be zero or any positive whole number.

Read: If `time` and `play_count` are not given, return a JSON dictionary listing the time and play count of slice `index`, like so:

```
[
  {
    "time":(time in seconds),
    "playCount":(whole number or -1 for infinite loop)
  }
]
```

Write: For the specified cue, set the time of slice marker `index` to `time`, and the play count of slice `index` to `play_count`. `time` can be any positive whole or decimal number. `play_count` can be any positive whole number, or `-1` to set the slice to infinite loop.

`time` and `play_count` must both be given if either one is.

/cue/{cue_number}/sliceMarker/{index}/playCount {play_count}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the play count of slice `index` of the specified cue.

Write: If `play_count` is given, set the play count of slice `index` of the specified cue to `play_count`. `play_count` can be any positive whole number, or `-1` to set the slice to infinite loop.

/cue/{cue_number}/sliceMarker/{index}/time {time}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the time of slice marker `index` of the specified cue.

Write: If `time` is given, set the time of slice marker `index` of the specified cue to `time`. `time` can be any positive whole or decimal number.

/cue/{cue_number}/sliceMarkers

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: Return a JSON dictionary listing the marker time and play count of all slices of the specified cue:

```
[
  {
```

```
    "time": number,
    "playCount": number
  },
  { ... }
]
```

`{ ... }` represents a second slice; the number of slices in a cue varies, so the number of items in this dictionary will vary.

Slices *end* with slice markers. Therefore, `time` corresponds to the end time of the slice whose `playCount` is being reported.

Write: The "write" form of this message is only available via the increment/decrement syntax discussed above.

Examples:

- `/cue/10/sliceMarkers/+ 5` would shift all slice markers in cue 10 to be five seconds later.
- `/cue/10/sliceMarkers 5` would be invalid, and have no effect.

/cue/{cue_number}/sliderLevel {channel} {decibel}

/cue/{cue_number}/sliderLevel/{channel} {decibel}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Get or set a single cue output slider volume level.

`outChannel` is required and must be either an integer from 0 to 128, or a string that is the name of a cue output. 0 is the main row.

Read: If no `decibel` is given, return the volume level of the specified cue output.

Write: If `decibel` is given, set the volume level of `channel` to `decibel`. `decibel` can be any number, decimals allowed, or a string such as `-INF`. If `decibel` is a string (any string, in fact) QLab will set the crosspoint to `-INF`.

/cue/{cue_number}/sliderLevels

/cue/{cue_number}/sliderLevels/live

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	view

Return an array of the output levels of the specified cue, including the main level.

/cue/{cue_number}/solo {output} {boolean}

/cue/{cue_number}/solo/{output} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `TRUE` if the cue output of the specified cue is soloed, or `FALSE` if it is not.

Write: Solo or un-solo the specified cue output of the specified cue. See details on booleans at the beginning of this section. You can refer to the cue output by its number or its name, if it has one.

/cue/{cue_number}/solo/object/{name} {boolean}

/cue/{cue_number}/solo/objectID/{id} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `TRUE` if the cue output of the specified cue is soloed, or `FALSE` if it is not.

Write: Solo or un-solo the specified object of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/solo/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-solo all cue outputs and objects of the specified cue.

/cue/{cue_number}/solo/channel/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-solo all cue outputs of the specified cue.

/cue/{cue_number}/solo/object/clear

view	edit	control	query	+/-?
✗	read/write	✗	✗	✗

Un-solo all objects of the specified cue.

/cue/{cue_number}/soloChannels

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the cue outputs that are soloed.

/cue/{cue_number}/soloObjects

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Return an array of the objects that are soloed.

/cue/{cue_number}/startTime {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the start time of the specified cue.

Write: If `number` is given, set the start time of the specified cue to `number` seconds. `number` can be any whole or decimal number greater than or equal to zero.

Mic cue messages

Messages specific to Mic cues. Mic cues also respond to any Audio cue messages that pertain to the I/O, Levels, and Objects tabs of the inspector.

/cue/{cue_number}/audioInputPatchName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the audio input patch currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the name of an audio input patch in the workspace, set the audio input patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/audioInputPatchNumber {number}

view	edit	control	query	+/-?
read only	read/write	read	✓	✓

Read: If no argument is given, return the index of the audio input patch currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first patch in the audio input patch list in Workspace Settings, `2` means the second patch, and so on.

Write: If `number` is given, set the audio input patch of the specified cue to that patch. If `number` is `0` , un-patch the specified cue. If `number` is greater than the number of audio input patches in the workspace, this message has no effect. `number` must be a whole number.

/cue/{cue_number}/audioInputPatchID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the patch ID of the audio input patch currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of an audio input patch in the workspace, set the audio input patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/channelOffset {number}

view	edit	control	query	+/-?
read only	read/write	read	✓	✓

Read: If no argument is given, return the input channel offset of the specified cue, as set in the I/O tab of the inspector.

Write: If `number` is given, set the input channel offset of the specified cue to `number`. `number` can be any whole number, zero or greater.

/cue/{cue_number}/channels {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the number of input channels used by the specified cue.

Write: If `number` is given, set the number of input channels used by the specified cue to `number`. `number` can be any positive whole number.

/cue/{cue_number}/duration {number}

/cue/{cue_number}/tempDuration {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the duration of the specified cue.

Write: If `number` is given, set the duration (or temporary duration) of the specified cue to `number`. `number` can be any number greater than zero.

Video cue messages

Messages specific to Video cues. Most of these messages will also work with Camera and Text cues as well.

Video cues will also respond to Audio cue messages, although some Audio cue messages will have no effect if the target video file does not have an audio track.

/cue/{cue_number}/anchor {x} {y}

/cue/{cue_number}/anchor/live {x} {y}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no arguments are given, return the anchor point of the specified cue.

Write: Set the anchor point of the specified cue to `(x,y)`. `x` and `y` can be any decimal numbers.

/cue/{cue_number}/anchor/x {number}

/cue/{cue_number}/anchor/x/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the X-axis anchor point of the specified cue.

Write: If `number` is given, set the X-axis anchor point of the specified cue to `number`. `number` can be any decimal number.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/anchor/y {number}

/cue/{cue_number}/anchor/y/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the Y-axis anchor point of the specified cue.

Write: If `number` is given, set the Y-axis anchor point of the specified cue to `number`. `number` can be any decimal number.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/blendMode {string}

view	edit	control	query	+/-?
read only	read/write	read only	✓	✗

Read: If no argument is given, return the blend mode of the specified cue.

Write: If `string` is given, set the blend mode of the specified cue to `string`. `string` must be the name of a blend mode, a list of which can be found in the Parameter Reference page of this manual.

/cue/{cue_number}/clockType {string}

view	edit	control	query	+/-?
read only	read/write	read only	✓	✓

Read: If no argument is given, return the clock type used by the specified cue (either audio or video.)

Write: If `string` is given, set the clock type of the specified cue to `string`. `string` can be either “audio” or “video”.

NOTE: There are two alternate forms of this message: `/cue/{cue_number}/clockType/audio` and `/cue/{cue_number}/clockType/video` which allow you to set the clock type of a cue without needing an argument. This form may be easier to use with some OSC-sending clients.

/cue/{cue_number}/crop {top} {bottom} {left} {right}

/cue/{cue_number}/crop/live {top} {bottom} {left} {right}

view	edit	control	query	+/-?	live
read only	read/write	read only	✓	✓	edit, control

Read: If no arguments are given, return the current crop for the specified cue. Crop values are expressed in terms of numbers of pixels counting inwards from each edge of the cue. So `0,0,0,0` is a cue that isn't cropped at all, and `10,20,30,40` is a cue that is cropped 10 pixels in from the top, 20 pixels in from the bottom, 30 pixels in from the left side, and 40 pixels in from the right side.

Write: If `top`, `bottom`, `left`, and `right` are given, set the crop of the specified cue to those values. Each value must be a number, and all four values must be given.

/cue/{cue_number}/cropTop {number}

/cue/{cue_number}/cropTop/live {number}

view	edit	control	query	+/-?	live
read only	read/write	read only	✓	✓	edit, control

Read: If no argument is given, return the current crop for the top edge of specified cue. Crop values are expressed in terms of numbers of pixels counting inwards from each edge of the cue, so a cropTop of 10 means that the top ten pixels of the cue are cropped out.

Write: If `number` is given, set the top crop of the specified cue to `number`. `number` can be any number.

/cue/{cue_number}/cropBottom {number}

/cue/{cue_number}/cropBottom/live {number}

view	edit	control	query	+/-?	live
read only	read/write	read only	✓	✓	edit, control

Read: If no argument is given, return the current crop for the bottom edge of specified cue. Crop values are expressed in terms of numbers of pixels counting inwards from each edge of the cue, so a cropBottom of 10 means that the bottom ten pixels of the cue are cropped out.

Write: If `number` is given, set the bottom crop of the specified cue to `number`. `number` can be any number.

/cue/{cue_number}/cropLeft {number}

/cue/{cue_number}/cropLeft/live {number}

view	edit	control	query	+/-?	live
read only	read/write	read only	✓	✓	edit, control

Read: If no argument is given, return the current crop for the left edge of specified cue. Crop values are expressed in terms of numbers of pixels counting inwards from each edge of the cue, so a cropLeft of 10 means that the left ten pixels of the cue are cropped out.

Write: If `number` is given, set the left crop of the specified cue to `number`. `number` can be any number.

/cue/{cue_number}/cropRight {number}

/cue/{cue_number}/cropRight/live {number}

view	edit	control	query	+/-?	live
read only	read/write	read only	✓	✓	edit, control

Read: If no argument is given, return the current crop for the right edge of specified cue. Crop values are expressed in terms of numbers of pixels counting inwards from each edge of the cue, so a cropRight of 10 means that the right ten pixels of the cue are cropped out.

Write: If `number` is given, set the right crop of the specified cue to `number`. `number` can be any number.

/cue/{cue_number}/cueSize

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return the natural size of the cue's video frame:

```
{
    "width": number,
    "height": number
}
```

/cue/{cue_number}/cueSize/height

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	control

Return the height of the natural size of the cue's video frame.

/cue/{cue_number}/cueSize/width

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	control

Return the width of the natural size of the cue's video frame.

/cue/{cue_number}/fillStage {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the specified cue is set to play in full-stage mode, or `false` if it is not.

Write: Set the full-stage mode of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/fillStyle {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the fill style of the specified cue.

Write: If `number` is given, set the fill style mode of the specified cue. Valid styles are:

```
0 - Fit
1 - Fill
2 - Stretch
```

/cue/{cue_number}/holdLastFrame {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the specified cue is set to hold last frame, or `false` if it is not.

Write: Set the state of the *Hold last frame* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/layer {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the layer of the specified cue.

Write: If `number` is given, set the layer of the specified cue to `number`. `number` can be any whole number from 0 to 1000. Layer 0 is the “bottom” layer and layer 1000 is the “top” layer.

/cue/{cue_number}/opacity {number}

/cue/{cue_number}/opacity/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the opacity of the specified cue.

Write: If `number` is given, set the opacity of the specified cue to `number`. `number` can be any decimal number from 0 to 1, where 0 represents 0% opacity and 1 represents 100% opacity.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/origin {x} {y}

/cue/{cue_number}/origin/live {x} {y}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no arguments are given, return the anchor point of the specified cue.

Write: Set the anchor point of the specified cue to `(x,y)`. `x` and `y` can be any decimal numbers.

Deprecated in QLab 5.0.2. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/cue/{cue_number}/anchor`.

/cue/{cue_number}/origin/x {number}

/cue/{cue_number}/origin/x/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the X-axis anchor point of the specified cue.

Write: If `number` is given, set the X-axis anchor point of the specified cue to `number`. `number` can be any decimal number.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

Deprecated in QLab 5.0.2. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/cue/{cue_number}/anchor/x`.

/cue/{cue_number}/origin/y {number}

/cue/{cue_number}/origin/y/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the Y-axis anchor point of the specified cue.

Write: If `number` is given, set the Y-axis anchor point of the specified cue to `number`. `number` can be any decimal number.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

Deprecated in QLab 5.0.2. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/cue/{cue_number}/anchor/y`.

/cue/{cue_number}/preserveAspectRatio {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the scale ratio lock of the specified cue.

Write: Set the state of the scale ratio lock of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/quaternion {a} {b} {c} {d}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no arguments are given, return an array of four numbers representing the cue's rotation as a quaternion.

Write: If four numbers are given (`a`, `b`, `c`, and `d`), set the rotation of the specified cue. Each number can be any number, decimals allowed.

Caution: you need to understand quaternion math to make any meaningful use of this message. Quaternion math is really hard. Good luck!

/cue/{cue_number}/resetRotation

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Reset the rotation of the specified cue.

/cue/{cue_number}/rotate/x {number}

/cue/{cue_number}/rotate/y {number}

/cue/{cue_number}/rotate/z {number}

/cue/{cue_number}/rotate/x/live {number}

/cue/{cue_number}/rotate/y/live {number}

/cue/{cue_number}/rotate/z/live {number}

view	edit	control	query	+/-?	live
✗	✓	✗	✗	✓	control

Add `number` to the current quaternion rotation of the specified cue. `number` can be any decimal number.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/scale {x} {y}

/cue/{cue_number}/scale/live {x} {y}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no arguments are given, return the scale of the specified cue.

Write: If `x` and `y` are given, set the scale of the specified cue to `(x,y).` `x` and `y` can be any number, decimals allowed.

/cue/{cue_number}/scale/x {number}

/cue/{cue_number}/scale/x/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the X-axis scale of the specified cue.

Write: If `number` is given, set the X-axis scale of the specified cue to `number.` `number` can be any number, decimals allowed.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/scale/y {number}

/cue/{cue_number}/scale/y/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the Y-axis scale of the specified cue.

Write: If `number` is given, set the Y-axis scale of the specified cue to `number`. `number` can be any number, decimals allowed.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/smooth {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Smooth* checkbox of the specified cue.

Write: If `boolean` is given, enable or disable the *Smooth* checkbox on the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/stage

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary containing all available information of the stage that the specified cue is assigned to. The exact contents of this dictionary can vary widely, but generally includes information such as the name, size, and warping details for all regions, the size and gamma of edge blend zones, control point positions and link information, and so on.

/cue/{cue_number}/stage/name {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the video stage currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and the specified cue is patched to a stage, set the name of that stage to `string`. If the specified cue is not currently patched to a stage, this message has no effect.

/cue/{cue_number}/stage/region/{name}

/cue/{cue_number}/stage/regionID/{id}

/cue/{cue_number}/stage/regionIndex/{index}

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary containing all available information of the specified region of the stage that the specified cue is assigned to. The exact contents of this dictionary can vary widely, but generally includes information such as the size and gamma of edge blend zones, control point positions and link information, and so on.

/cue/{number}/stage/region/{name}/bounds {x} {y} {width} {height}

/cue/{number}/stage/regionID/{id}/bounds {x} {y} {width} {height}

/cue/{number}/stage/regionIndex/{index}/bounds {x} {y} {width} {height}

view	edit	control	query	+/-?
read	read/write	read	✗	✗

Read: If no arguments are given, return the lower left corner coordinates, the width, and the height of the specified region.

Write: If all four arguments are given and are valid, set the position and size of the specified region. All four arguments must be numbers, decimals allowed, and taken together they should describe a location and size that is within the boundaries of the stage.

/cue/{number}/stage/region/{name}/bounds/origin {x} {y}

/cue/{number}/stage/regionID/{id}/bounds/origin {x} {y}

/cue/{number}/stage/regionIndex/{index}/bounds/origin {x} {y}

view	edit	control	query	+/-?
read	read/write	read	✗	✗

Read: If no arguments are given, return the lower left corner coordinates of the specified region.

Write: If both arguments are given and are valid, set the position of the specified region. Both arguments must be numbers, decimals allowed, and should describe a location that is within the boundaries of the stage.

/cue/{number}/stage/region/{name}/bounds/origin/x {number}

/cue/{number}/stage/regionID/{id}/bounds/origin/x {number}

/cue/{number}/stage/regionIndex/{index}/bounds/origin/x {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the X-axis coordinate of the lower left corner of the specified region.

Write: If `number` is given, set the X-axis position of the specified region.

/cue/{number}/stage/region/{name}/bounds/origin/y {number}

/cue/{number}/stage/regionID/{id}/bounds/origin/y {number}

/cue/{number}/stage/regionIndex/{index}/bounds/origin/y {number}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the Y-axis coordinate of the lower left corner of the specified region.

Write: If `number` is given, set the Y-axis position of the specified region.

/cue/{number}/stage/region/{name}/bounds/size {width} {height}

/cue/{number}/stage/regionID/{id}/bounds/size {width} {height}

/cue/{number}/stage/regionIndex/{index}/bounds/size {width} {height}

view	edit	control	query	+/-?
read	read/write	read	✗	✗

Read: If no arguments are given, return the size of the specified region.

Write: If both arguments are given and are valid, set the position of the specified region. Both arguments must be numbers, decimals allowed, and should describe a size that is within the boundaries of the stage.

/cue/{number}/stage/region/{name}/bounds/size/height {number}

/cue/{number}/stage/regionID/{id}/bounds/size/height {number}

/cue/{number}/stage/regionIndex/{index}/bounds/size/height {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the height of the specified region.

Write: If `number` is given, set the height of the specified region.

/cue/{number}/stage/region/{name}/bounds/size/width {number}

/cue/{number}/stage/regionID/{id}/bounds/size/width {number}

/cue/{number}/stage/regionIndex/{index}/bounds/size/width {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the width of the specified region.

Write: If `number` is given, set the width of the specified region.

/cue/{number}/stage/region/{name}/enableGrid {boolean}

/cue/{number}/stage/regionID/{id}/enableGrid {boolean}

/cue/{number}/stage/regionIndex/{index}/enableGrid {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the grid of the specified region is enabled (visible), and `false` if it is not.

Write: If `boolean` is given, enable or disable the grid of the specified region. See details on booleans at the beginning of this section.

/cue/{number}/stage/region/{name}/enableGuide {boolean}

/cue/{number}/stage/regionID/{id}/enableGuide {boolean}

/cue/{number}/stage/regionIndex/{index}/enableGuide {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the guides of the route assigned to the specified region are enabled (visible), and `false` if they are not.

Write: If `boolean` is given, enable or disable the guides of the route assigned to the specified region. See details on booleans at the beginning of this section.

/cue/{number}/stage/region/{name}/moveBy {x} {y}

/cue/{number}/stage/regionID/{id}/moveBy {x} {y}

/cue/{number}/stage/regionIndex/{index}/moveBy {x} {y}

view	edit	control	query	+/-?
✗	✓	✗	✗	✓

Move the specified region. `x` and `y` can be any number, decimals allowed.

/cue/{number}/stage/region/{name}/name

/cue/{number}/stage/regionID/{id}/name

/cue/{number}/stage/regionIndex/{index}/name

view	edit	control	query
read only	read only	read only	✗

Return the name of the specified region of the stage that the specified cue is assigned to.

/cue/{number}/stage/region/{name}/resetControlPoints

/cue/{number}/stage/regionID/{id}/resetControlPoints

/cue/{number}/stage/regionIndex/{index}/resetControlPoints

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Reset all warping control points of the specified region to their default positions.

/cue/{number}/stage/region/{name}/uniqueID

/cue/{number}/stage/regionID/{id}/uniqueID

/cue/{number}/stage/regionIndex/{index}/uniqueID

view	edit	control	query
read only	read only	read only	✗

Return the ID of the specified region of the stage that the specified cue is assigned to.

/cue/{cue_number}/stage/regions

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing the regions of the stage in use by the specified cue.

/cue/{cue_number}/stage/size

view	edit	control	query
read only	read only	read only	✗

Return a JSON dictionary describing the size of stage in use by the specified cue.

/cue/{cue_number}/stage/size/height

view	edit	control	query
read only	read only	read only	✓

Return the height of stage in use by the specified cue.

/cue/{cue_number}/stage/size/width

view	edit	control	query
read only	read only	read only	✓

Return the width of stage in use by the specified cue.

/cue/{cue_number}/stage/uniqueID

view	edit	control	query

view	edit	control	query
read only	read only	read only	✗

Return the unique ID of the stage in use by the specified cue.

/cue/{cue_number}/stageID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the stage ID of the video stage currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the stage ID of a video stage in the workspace, set the video stage of the specified cue to that stage. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/stageName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the video stage currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the name of a video stage in the workspace, set the video stage of the specified cue to that stage. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/stageNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the index of the video stage currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first stage in the video stage list in Workspace Settings, `2` means the second stage, and so on.

Write: If `number` is given, set the video stage of the specified cue to that stage. If `number` is `0` , un-patch the specified cue. If `number` is greater than the number of video stages in the workspace, this message has no effect. `number` must be a whole number.

/cue/{cue_number}/surfaceID

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Deprecated in QLab 5.0. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/cue/{cue_number}/stageID` .

/cue/{cue_number}/surfaceList

view	edit	control	query	+/-?

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Deprecated in QLab 5.0. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/settings/video/stages`.

/cue/{cue_number}/surfaceName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Deprecated in QLab 5.0. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/cue/{cue_number}/stageName`.

/cue/{cue_number}/surfaceSize

view	edit	control	query	+/-?
read only	read only	read only	✓	✗

Deprecated in QLab 5.0. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/cue/{cue_number}/stage/size`.

/cue/{cue_number}/translation {x} {y}

/cue/{cue_number}/translation/live {x} {y}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no arguments are given, return the translation of the specified cue.

Write: If `x` and `y` are given, set the translation of the specified cue to `(x,y)`. `x` and `y` can be any numbers, decimals allowed.

/cue/{cue_number}/translation/x {number}

/cue/{cue_number}/translation/x/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the X-axis translation of the specified cue.

Write: If `number` is given, set the X-axis translation of the specified cue to `number`. `number` can be any number, decimals allowed.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/translation/y {number}

/cue/{cue_number}/translation/y/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the Y-axis translation of the specified cue.

Write: If `number` is given, set the Y-axis translation of the specified cue to `number`. `number` can be any number, decimals allowed.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/videoEffects

/cue/{cue_number}/videoEffects/live

view	edit	control	query	+/-?	live
read only	read only	read only	✗	✗	view

Return a list of the video effects in use by the specified cue.

Video effects have OSC-compatible names which can be discovered by hovering your mouse over a video effect currently in use in the Video FX tab of the inspector, or in the video effects section of the Parameter Reference page of this manual.

/cue/{cue_number}/videoEffects/add {name}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Add a video effect, specified by `name`, to the specified cue. If the cue already has one or more video effects in use, the new video effect will be added as the last video effect. `name` must be the OSC name of a video effect in QLab, a list of which can be found in the video effects section of the Parameter Reference page of this manual.

/cue/{cue_number}/videoEffects/insert {name} {index}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Add a video effect, specified by `name`, to the specified cue at position `index`. Video effects are zero-indexed; the first video effect on a cue is at position `0`. If there is already a video effect at position `index`, that effect and any effects after it will be pushed down by one, and the new effect will occupy position `index`. If `index` is greater than one plus the number of video effects currently in use by the cue, the new effect will be added as the last video effect. `name` must be the OSC name of a video effect in QLab and `index` must be a whole number, `0` or higher.

/cue/{cue_number}/videoEffect/{name}/delete

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Remote the video effect named `name` from the specified cue. If there is no video effect of the specified `name` in use by the cue, this message has no effect. If there is more that one video effect with the same `name` in use by the cue, this command addresses the first instance of the video effect.

/cue/{cue_number}/videoEffectIndex/{index}/delete

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Remote the video effect at position `index` from the specified cue. Video effects are zero indexed; the first video effect on a cue is at position `0`. If there is no video effect at position `index`, this message has no effect. `index` must be a whole number, `0` or higher.

/cue/{cue_number}/videoEffect/{name}/enabled {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the specified video effect on the specified cue is enabled, and `false` if it is not. The video effect is specified by `name` which is the OSC name of the video effect.

Write: If `boolean` is given, enable or disable the specified video effect on the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/videoEffectIndex/{index}/enabled {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the specified video effect on the specified cue is enabled, and `false` if it is not. The video effect is specified by `index` which is a zero-indexed count of the video effects in use by the cue.

Write: If `boolean` is given, enable or disable the specified video effect on the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/videoEffect/{name}/move {newIndex}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Move the video effect specified by `name` to position `newIndex`. If `newIndex` is greater than one plus the number of video effects currently in use by the cue, the effect will be moved to the end of the list of video effects. `name` must correspond to a video effect in use by the cue, and `newIndex` must be a whole number, `0` or higher. If there is more that one video effect with the same `name` in use by the cue, this command addresses the first instance of the video effect.

/cue/{cue_number}/videoEffectIndex/{index}/move {newIndex}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Move the video effect at position `index` to position `newIndex`. If `newIndex` is greater than one plus the number of video effects currently in use by the cue, the effect will be moved to the end of the list of video effects. `index` must correspond to a position of a video effect, and both `index` and `newIndex` must be whole numbers, `0` or higher.

/cue/{cue_number}/videoEffect/{name}/parameter/{parameterKey} {setting}

/cue/{cue_number}/videoEffect/{name}/parameter/{parameterKey}/live {setting}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the current setting or value for the given parameter of the specified video effect on the specified cue. `parameterKey` must be the OSC parameter name of the specified video effect. The video effect is specified by `name` which is the OSC name of the video effect.

Write: If `setting` is given, set the value of the given parameter of the specified video effect on the specified cue to `setting`. `setting` must be valid for the given parameter. `parameterKey` must be the OSC parameter name of the specified video effect. The video effect is specified by `name` which is the OSC name of the video effect.

/cue/{cue_number}/videoEffectIndex/{index}/parameter/{parameterKey} {setting}

/cue/{cue_number}/videoEffectIndex/{index}/parameter/{parameterKey}/live {setting}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the current setting or value for the given parameter of the specified video effect on the specified cue. `parameterKey` must be the OSC parameter name of the specified video effect. The video effect is specified by `index` which is a zero-indexed count of the video effects in use by the cue.

Write: If `setting` is given, set the value of the given parameter of the specified video effect on the specified cue to `setting`. `setting` must be valid for the given parameter. `parameterKey` must be the OSC parameter name of the specified video effect. The video effect is specified by `index`.

/cue/{cue_number}/videoEffect/{name}/parameters {json_string}

/cue/{cue_number}/videoEffect/{name}/parameters/live {json_string}

view	edit	control	query	+/-?	live
read	read/write	read	✗	✗	control

Read: If no argument is given, return a JSON string containing the names and current settings or values for the given parameter of the specified video effect on the specified cue in the form `{"parameter1_name":value,"parameter2_name",value...}`. The video effect is specified by `name` which is the OSC name of the video effect.

Write: If `json_string` is given, set the value of the parameters of the specified video effect on the specified cue according to `json_string`. `json_string` must be formatted appropriately and contain only valid parameter names and values for the specified video effect. The video effect is specified by `name` which is the OSC name of the video effect.

Example

Assuming cue `1` is a 🎞 Video cue with the Color Controls video effect, the following OSC message would appropriately set all four parameters of that video effect:

```
/cue/1/videoEffect/ColorControls/parameters "{\"inputContrast\":1,\"inputAngle\":0,\"inputBrightness\":0.25,\"inputSaturation\":1}"
```

Note the backslashes surrounding most of the quotation marks; these are necessary because those quotation marks are part of the JSON string that needs to be sent, but QLab uses quotation marks to denote the beginning and end of a text string in OSC messages. The backslashes tell QLab's OSC interpreter to send the quote as part of the string and not use it to mark the end of the string.

/cue/{cue_number}/videoEffectIndex/{index}/parameters {json_string}

/cue/{cue_number}/videoEffectIndex/{index}/parameters/live {json_string}

view	edit	control	query	+/-?	live
read	read/write	read	✗	✗	control

Read: If no argument is given, return a JSON string containing the names and current settings or values for the given parameter of the specified video effect on the specified cue in the form `{"parameter1_name":value,"parameter2_name",value...}`. The video effect is specified by `index` which is a zero-indexed count of the video effects in use by the cue.

Write: If `json_string` is given, set the value of the parameters of the specified video effect on the specified cue according to `json_string`. `json_string` must be formatted appropriately and contain only valid parameter names and values for the specified video effect. The video effect is specified by `index` which is a zero-indexed count of the video effects in use by the cue.

Example

Assuming cue `1` is a 🎞 Video cue with the Color Controls video effect, the following OSC message would appropriately set all four parameters of that video effect:

```
/cue/1/videoEffect/0/parameters "{\"inputContrast\":1,\"inputAngle\":0,\"inputBrightness\":0.25,\"inputSaturation\":1}"
```

Note the backslashes surrounding most of the quotation marks; these are necessary because those quotation marks are part of the JSON string that needs to be sent, but QLab uses quotation marks to denote the beginning and end of a text string in OSC messages. The backslashes tell QLab's OSC interpreter to send the quote as part of the string and not use it to mark the end of the string.

Camera cue messages

Messages specific to Camera cues. Camera cues also respond to Audio cue, Mic cue, and Video cue messages that pertain to the I/O, Geometry, Levels, Objects, and Video FX tabs of the inspector.

/cue/{cue_number}/cameraPatch {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Deprecated in QLab 5.0. This message works in QLab 5, but incompletely, and will be removed in a future version of QLab. Use `/videoInputPatchNumber` or `/videoInputPatchID` instead.

/cue/{cue_number}/duration {number}

/cue/{cue_number}/tempDuration {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the duration of the specified cue.

Write: If `number` is given, set the duration (or temporary duration) of the specified cue to `number`. `number` can be any number greater than zero.

/cue/{cue_number}/videoInputPatchName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the video input patch currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the name of a video input patch in the workspace, set the video input patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/videoInputPatchNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the index of the video input patch currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first patch in the patch list in Workspace Settings, `2` means the second patch, and so on.

Write: If `number` is given, set the video input patch of the specified cue to that patch. If `number` is `0`, un-patch the video input of the specified cue. If `number` is greater than the number of video input patches in the workspace, this message has no effect. `number` must be a whole number.

/cue/{cue_number}/videoInputPatchID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the patch ID of the video input patch currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of a video input patch in the workspace, set the video input patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

Text cue messages

Messages specific to Text cues. Text cues also respond to Video cue messages that pertain to the I/O, Geometry, and Video FX tabs of the inspector.

/cue/{cue_number}/fixedWidth {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the width of the specified cue.

Write: If `number` is given, and is greater than 0, set the fixed width of the specified cue to `number`. If `number` equals 0, set the width of the specified cue to automatic. `number` can be any number greater than or equal to zero, decimals allowed.

/cue/{cue_number}/text {string}

/cue/{cue_number}/text/live {string}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the text of the specified cue.

Write: If `string` is given, set the text of the specified cue to `string`. When setting text, the formatting of the new text will match the first character of the existing text.

/cue/{cue_number}/text/format {json_string}

/cue/{cue_number}/text/format/live {json_string}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return an array of JSON dictionaries describing each substring in the text of the specified cue. A substring, in this case, is defined as a subset of the text with uniform format attributes. If the entire text string is uniformly formatted, the array will have only one dictionary.

```
[
  {
    "fontFamily": string,
    "fontStyle": string,
    "fontName": string,
    "fontSize": number,
    "lineSpacing": number,
    "color": [red, green, blue, alpha],
    "range": [index, length]
  },
  { ... }
]
```

`color` is an array of four numbers, each from 0 to 1, decimals allowed, representing a percentage value for that parameter.

`range` is an array of two numbers; `index` is the position of the first character of the substring within the entire text of the cue (the first character in the cue's text has an index of `0`), and `length` is the total number of characters in the substring.

`{ ... }` represents a second substring; the number of substrings in a Text cue varies, so the number of items in this dictionary will vary. If the entire text of the cue is uniformly formatted, the array will have only one dictionary.

Write: If `json_string` is given, this command can be used to set the formatting of the whole text or a substring of the specified cue. `range` is optional; if omitted, the message will apply to the whole text of the cue.

/cue/{cue_number}/text/format/alignment {alignment}

/cue/{cue_number}/text/format/alignment/live {alignment}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the alignment of the specified cue.

Write: If `alignment` is given, set the text alignment of the specified cue to `alignment`. Valid alignments are `left`, `center`, `right`, or `justify`.

/cue/{cue_number}/text/format/color {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/backgroundColor {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/shadowColor {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/strikethroughColor {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/underlineColor {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/color/live {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/backgroundColor/live {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/shadowColor/live {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/strikethroughColor/live {red} {green} {blue} {alpha}

/cue/{cue_number}/text/format/underlineColor/live {red} {green} {blue} {alpha}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no arguments are given, return the color for the given attribute of the text of the specified cue.

Write: If `red`, `green`, `blue`, and `alpha` are specified, set the color for the given attribute of the text of the specified cue.

`red`, `green`, `blue`, and `alpha` must be numbers between 0.0 and 1.0, decimals allowed, where 1.0 is the maximum level. Thus, `1 0 0 1` is primary red, and `1 1 1 0.5` is white at 50% transparency.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`.

/cue/{cue_number}/text/format/fontFamily

/cue/{cue_number}/text/format/fontFamily/live

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	control

Return the font family name (“Helvetica”, “Courier New”, etc.) used in the text of the specified cue.

This command has several optional alternate forms:

/cue/{cue_number}/text/format/fontFamily/{index}/{length}

`index` is a whole number or a percentage (e.g. `53%`) which specifies the start of a substring. The first character is index `0`.

`length` is a whole number or percentage which specifies the length of that substring, or is `-1` which specifies “to the end of the string”

/cue/{cue_number}/text/format/fontFamily/word/{word_index}

`word_index` is a whole number which specifies a single word within the text of the cue. The first word is word `0`.

Examples

- `/cue/1/text/format/fontFamily/4/12` will return the font family name used for characters 5 through 17 of the text of cue 1.
- `/cue/10/text/format/fontFamily/25%/75%` will return the font family name used for the latter 75% of the text of cue 10.
- `/cue/12/text/format/fontFamily/word/8` will return the font family name used for the eighth word in the the text of cue 10.

/cue/{cue_number}/text/format/fontStyle

/cue/{cue_number}/text/format/fontStyle/live

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	control

Return the style (“Bold Oblique”, “Regular”, etc.) used in the text of the specified cue.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`.

/cue/{cue_number}/text/format/fontFamilyAndStyle {family} {style}

/cue/{cue_number}/text/format/fontFamilyAndStyle/live {family} {style}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no arguments are given, return the font family and style for the text of the specified cue.

Write: If `family` and `style` are given, set the font family and style of the text of the specified cue to `family` and `style`. Individual commands for font family and style are not available because the combination of both values is required to reliably describe an individual font.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`.

/cue/{cue_number}/text/format/fontName {name}

/cue/{cue_number}/text/format/fontName/live {name}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the name of the font used for the text of the specified cue.

Write: If `name` is given, set the font of the text of the specified cue.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`.

/cue/{cue_number}/text/format/fontSize {number}

/cue/{cue_number}/text/format/fontSize/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the font size of the text of the specified cue.

Write: If `number` is specified, set the font size of the text of the specified cue to `number`.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/text/format/lineSpacing {number}

/cue/{cue_number}/text/format/lineSpacing/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the line spacing of the text of the specified cue.

Write: If `number` is specified, set the line spacing of the text of the specified cue to `number`.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/text/format/shadowBlurRadius {number}

/cue/{cue_number}/text/format/shadowBlurRadius/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the blur radius of the text drop shadow of the specified cue.

Write: If `number` is specified, set the blur radius of the text drop shadow of the specified cue. `number` must be a number greater than or equal to zero.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`, although it only works if you index the first word or character in a line. It's also important to remember that the invisible "return" character at the end of a line (created by pressing the "return" key) counts as a character.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/text/format/shadowOffset {width} {height}

/cue/{cue_number}/text/format/shadowOffset/live {width} {height}

view	edit	control	query	+/-?	live
read	read/write	read	✗	✗	control

Read: If no arguments are given, return the horizontal and vertical offset of the text drop shadow of the specified cue.

Write: If `width` and `height` are specified, set the offset of the text drop shadow of the specified cue. `width` and `height` must be numbers.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`, although it only works if you index the first word or character in a line. It's also important to remember that the invisible "return" character at the end of a line (created by pressing the "return" key) counts as a character.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/text/format/shadowOffset/width {number}

/cue/{cue_number}/text/format/shadowOffset/width/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the horizontal offset of the text drop shadow of the specified cue.

Write: If `width` is specified, set the horizontal offset of the text drop shadow of the specified cue. `width` must be a number.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`, although it only works if you index the first word or character in a line. It's also important to remember that the invisible "return" character at the end of a line (created by pressing the "return" key) counts as a character.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/text/format/shadowOffset/height {number}

/cue/{cue_number}/text/format/shadowOffset/height/live {number}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✓	control

Read: If no argument is given, return the vertical offset of the text drop shadow of the specified cue.

Write: If `height` is specified, set the vertical offset of the text drop shadow of the specified cue. `height` must be a number.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`, although it only works if you index the first word or character in a line. It's also important to remember that the invisible "return" character at the end of a line (created by pressing the "return" key) counts as a character.

Note: If using both increment/decrement and live at the same time, the `/live` part of the address must come last, after the `/+` or `/-` part.

/cue/{cue_number}/text/format/strikethroughStyle {style}

/cue/{cue_number}/text/format/strikethroughStyle/live {style}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the strikethrough style for the text of the specified cue.

Write: If `style` is given, set the strikethrough style of the text of the specified cue to `style`. `style` may be `none`, `single`, or `double`.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`.

/cue/{cue_number}/text/format/underlineStyle {style}

/cue/{cue_number}/text/format/underlineStyle/live {style}

view	edit	control	query	+/-?	live
read	read/write	read	✓	✗	control

Read: If no argument is given, return the underline style for the text of the specified cue.

Write: If `style` is given, set the underline style of the text of the specified cue to `style`. `style` may be `none`, `single`, or `double`.

This command may also use the optional `/{index}/{length}` and `/word/{word_index}` forms as described above in the entry for `/cue/{cue_number}/text/format/fontFamily`.

/cue/{cue_number}/text/outputSize

/cue/{cue_number}/text/outputSize/live

view	edit	control	query	+/-?	live
read only	read only	read only	✗	✗	control

Return a JSON dictionary describing the width and height of the text of the specified cue:

```
{
    "width": number,
    "height": number
}
```

/cue/{cue_number}/text/outputSize/height

/cue/{cue_number}/text/outputSize/height/live

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	control

Return the height of the text of the specified cue.

/cue/{cue_number}/text/outputSize/width

/cue/{cue_number}/text/outputSize/width/live

view	edit	control	query	+/-?	live
read only	read only	read only	✓	✗	control

Return the width of the text of the specified cue.

Light cue messages

Messages specific to Light cues.

/cue/{cue_number}/alwaysCollate {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the *Collate effects of previous light cues* checkbox of the specified cue checked, and `false` if it is not.

Write: Set the state of the *Collate effects of previous light cues* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/collateAndStart

view	edit	control	query	+/-?
✗	✗	✓	✗	✗

Collate and start the specified cue.

/cue/{cue_number}/lightCommandText {string}

view	edit	control	query	+/-?
read	read/write	read	✗	✗

Read: If no argument is given, return the full command text of the specified cue.

Write: If `string` is given, set the full command text of the specified cue to `string`.

/cue/{cue_number}/prune

/cue/{cue_number}/pruneCommands

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Prune the command text of the specified cue. Pruning removes any commands which have no effect.

Example

Take for example a Light cue containing these commands:

```
myLight.intensity = 50
myLight.intensity = 100
```

Running this light cue would result in `myLight.intensity` being set to `100`, since light commands are interpreted sequentially. Pruning this Light cue would remove the first command, since it is obviated by the second command.

/cue/{cue_number}/removeLightCommandsMatching {string}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

If the specified cue contains a command that matches `string`, remove that command. Otherwise, do nothing.

/cue/{cue_number}/replaceLightCommand {old_command} {new_command}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

If the specified cue contains a light command matching `old_command`, replace that command with `new_command`. Both `old_command` and `new_command` must be strings which are valid light commands. If the specified cue does not have a command matching `old_command`, this message has no effect.

/cue/{cue_number}/safeSort

/cue/{cue_number}/safeSortCommands

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Lexically (alphanumerically) sort the command text of the specified cue, as long as sorting doesn't change the cue's output.

/cue/{cue_number}/setLight {string} {setting}

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Add a command to the specified cue in the form `string = setting`. `string` can be the name of an instrument or group, with or without a parameter. `setting` must be an acceptable value for the specified parameter of the specified instrument or group.

/cue/{cue_number}/subcontroller {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Use as subcontroller in dashboard* checkbox for the specified cue.

Write: Set the the state of the *Use as subcontroller in dashboard* checkbox for the specified cue. See details on booleans at the beginning of this section.

Fade cue messages

Messages specific to Fade cues. Fade cues also respond to most of the OSC messages that their targets respond to. For example, a Fade cue which targets an Audio cue will respond to `/levels`, and a Fade cue which targets a Video cue will respond to `/opacity`.

/cue/{cue_number}/audioMapTargetID {id}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the ID of the audio map currently targeted by the specified cue.

Write: If `id` is given, set the target of the specified cue to the specified audio map.

/cue/{cue_number}/doLevel {row} {column} {boolean}

/cue/{cue_number}/doLevel/{row}/{column} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no arguments are given, return all the currently active crosspoints in the Levels tab of the specified cue. The levels are returned as an array of arrays, like so: `[row0Array, row1Array, row2Array, ...]`

If `row` and `column` are given, but not `boolean`, return the active state of crosspoint { `row`, `column` }.

`row` must be an integer from 0 to 24. 0 is the main column.

`column` must be either an integer from 0 to 128, or a string that is the name of a cue output. 0 is the main row.

Write: If `number`, `row` and `column` are all given, set the active state (i.e. yellow or grey) of crosspoint { `row`, `column` }. See details on booleans at the beginning of this section.

/cue/{cue_number}/doObjectLevel {row} {object} {boolean}

/cue/{cue_number}/doObjectIDLevel {row} {object} {boolean}

/cue/{cue_number}/doObjectLevel/{row}/{object} {boolean}

/cue/{cue_number}/doObjectIDLevel/{row}/{object} {boolean}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no arguments are given, return all the currently active crosspoints in the Objects tab of the specified cue. The levels are returned as an array of arrays, like so: `[row0Array, row1Array, row2Array, ...]`

If `row` and `object` are given, but not `boolean`, return the active state of crosspoint { `row`, `object` }.

`row` must be an integer from 0 to 24. 0 is the main column.

`{object}` must be the name or ID of an audio object in the cue.

Write: If `number`, `row` and `object` are all given, set the active state (i.e. yellow or grey) of crosspoint { `row`, `object` }. See details on booleans at the beginning of this section.

/cue/{cue_number}/doOpacity {boolean}

/cue/{cue_number}/doRate {boolean}

/cue/{cue_number}/doRotation {boolean}

/cue/{cue_number}/doScale {boolean}

/cue/{cue_number}/doTranslation {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the relevant geometry parameter checkbox of the specified cue.

Write: Set the state of the geometry parameter checkboxes of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/fadeType {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the fade type of the specified cue.

Write: If `number` is given, set the fade type of the specified cue. Valid types are:

```
1 - 1D fade (curve)
2 - 2D fade (path)
```

/cue/{cue_number}/geoMode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the geometry fade mode of the specified cue.

Write: If `number` is given, set the geometry fade mode of the specified cue. Valid modes are:

```
0 - absolute fade
1 - relative fade
```

/cue/{cue_number}/levelsMode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the audio fade mode of the specified cue.

Write: If `number` is given, set the audio fade mode of the specified cue. Valid modes are:

```
0 - absolute fade
1 - relative fade.
```

/cue/{cue_number}/mode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Deprecated in QLab 5.0. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/levelsMode`.

/cue/{cue_number}/patchTargetID {id}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the ID of the patch currently targeted by the specified cue.

Write: If `id` is given, set the target of the specified cue to the specified patch.

/cue/{cue_number}/pathHeight {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the path height of the specified cue. If the specified cue is not a 2D fade, this message has no effect.

Write: If `number` is given, set the path height of the specified cue. `number` can be any positive number.

/cue/{cue_number}/pathWidth {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the path width of the specified cue. If the specified cue is not a 2D fade, this message has no effect.

Write: If `number` is given, set the path width of the specified cue. `number` can be any positive number.

/cue/{cue_number}/rotation {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, and the specified cue is using single-axis rotation, return the current rotation of the specified cue. If the specified cue is not using single-axis rotation, return `0`.

Write: If `number` is given, and if the specified cue is using single-axis rotation, set the rotation in degrees to `number`. If the specified cue is not using single-axis rotation, this message has no effect.

To work with Fade cues using 3D orientation, use `/quaternion` or `/rotate{N}`.

/cue/{cue_number}/rotationType {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the rotation type of the specified cue.

Write: If `number` is given, set the rotation type of the specified cue to `number`. Valid rotation types are:

```
0 - 3D orientation
1 - X rotation
2 - Y rotation
3 - Z rotation
```

/cue/{cue_number}/setGeometryFromTarget

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Set the geometry of the specified cue to the geometry of its target cue. If the target cue has no geometry, this message has no effect.

/cue/{cue_number}/setLevelsFromTarget

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Set the audio levels, trim, and gangs of the specified cue to match those of its target. If the target cue has no audio levels, this message has no effect.

/cue/{cue_number}/stopTargetWhenDone {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Stop target when done* checkbox of the specified cue.

Write: Set the state of the *Stop target when done* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/targetMode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return 0 if the specified cue is set to target cues or return 1 if the specified cue is set to target patches.

Write: If `number` is given, set the target type of the specified cue. Valid types are:

```
0 - Cues
1 - Patches
```

/cue/{cue_number}/willFade {row} {column} {boolean}

/cue/{cue_number}/willFade/{row}/{column} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no arguments are given, return all the currently active crosspoints in the specified cue. The levels are returned as an array of arrays, like so: `[row0Array, row1Array, row2Array, ...]`

If `row` and `column` are given, but not `boolean`, return the active state of crosspoint { `row`, `column` }.

`row` must be an integer from 0 to 24. 0 is the main column.

`column` must be either an integer from 0 to 128, or a string that is the name of a cue output. 0 is the main row.

Write: If `number`, `row` and `column` are all given, set the active state (i.e. yellow or grey) of crosspoint { `row`, `column` }. See details on booleans at the beginning of this section.

Deprecated in QLab 5.5. This message works in QLab 5.5, but will be removed in a future version of QLab. Use `/doLevel` instead.

Network cue messages

Messages specific to Network cues.

/cue/{cue_number}/customString {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the message of the specified cue.

Write: If `string` is given, and the specified cue's patch is set to "OSC Message," set the OSC message of the specified cue to `string`. If `string` is given, and the specified cue's patch is set to "Plain Text," set the text of the specified cue to `string`. In all other cases, this message has no effect.

/cue/{cue_number}/fadeEntries {array_of_values}

/cue/{cue_number}/fadeEntries {json_string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return a JSON array of point dictionaries comprising the complete set of fade handles for the specified cue's fade shape or path.

```
[
    {"x": number, "y": number},
    {"x": number, "y": number},
    { ... }
]
```

Write: If `array_of_values` or `json_string` is given, define the fade curve (1D) or path (2D) of the fade, which in turn defines the values sent by the specified cue when it runs. If the cue has a preexisting set of point values, setting new ones with this message entirely replaces the previous set.

Points can be represented as a single string in the form `"{x,y}"`, as a two number array in the form `[x,y]`, or as a JSON dictionary with numeric values for keys `x` and `y`.

For 1D fades, `x` must be a value between 0 and the fade duration (in seconds), and `y` must be a value between the starting value and ending value specified in the cue. Any invalid point value returns an error status. The array of points must include both the starting and ending points of the fade, i.e. an entry where `x` is `0.0` and an entry where `x` is equal to the fade duration.

For 2D fades, `x` and `y` can be any number, decimals allowed.

/cue/{cue_number}/fadeFrom {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the starting value for the specified cue.

Write: If `number` is given, and the specified cue is set to 1D fade, set the starting value for the fade. If the specified cue is not set to 1D fade, this message has no effect. `number` can be any number within the appropriate range for the specified cue.

/cue/{cue_number}/fadeNumberType {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the number format of the specified cue.

Write: If `number` is given, and the specified cue is set to fade, set the format of the values sent by the specified cue. `number` can be one of the following:

```
0 - Integers
1 - Floats
```

If the specified cue is not set to fade, this message has no effect.

/cue/{cue_number}/fadeTo {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the ending value for the specified cue.

Write: If `number` is given, and the specified cue is set to 1D fade, set the ending value for the fade. If the specified cue is not set to 1D fade, this message has no effect. `number` can be any number within the appropriate range for the specified cue.

/cue/{cue_number}/fadeType {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the fade type of the specified cue.

Write: If `number` is given, and the specified cue can be set to fade, set the fade type of the specified cue to `number`. `number` can be one of the following:

```
0 - No Fade/Resend
1 - 1D Fade
2 - 2D Fade
```

If the specified cue cannot fade, this message has no effect.

/cue/{cue_number}/fps {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the frame rate of the specified cue.

Write: If `number` is given, set the frame rate at which messages are sent by the specified cue. `number` must be a whole number from 1 to 120.

/cue/{cue_number}/message

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return the completed message that will be sent by the specified cue, as displayed towards the bottom of the Settings tab of the inspector, next to the Copy button. This can be most helpful when examining Network cues whose patches use network device descriptions.

/cue/{cue_number}/messageError

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

If the specified cue is a Network cue with an invalid outgoing message, return the yellow error message found towards the bottom of the Settings tab of the inspector.

/cue/{cue_number}/networkPatchName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the network patch currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the name of a network patch in the workspace, set the network patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/networkPatchNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the index of the network patch currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first patch in the patch list in Workspace Settings, `2` means the second patch, and so on.

Write: If `number` is given, set the network patch of the specified cue to that patch. If `number` is `0` , un-patch the specified cue. If `number` is greater than the number of network patches in the workspace, this message has no effect. `number` must be a whole number.

/cue/{cue_number}/networkPatchID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the patch ID of the network patch currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of a network patch in the workspace, set the network patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/parameterFadeEnabled/{index} {boolean}

/cue/{cue_number}/parameterFadeEnabled/{key} {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return `true` if the specified parameter of the specified cue is set to fade, and `false` if it is not.

Write: Enabled or disable fading for the specified parameter of the specified cue. There are two ways to identify the parameter in question:

- `index` is a 0-based counter which counts the parameters in a Network cue from top to bottom, counting visible parameters only.
- `key` refers to the key name of the parameter, which is sometimes different from its display name.

If a parameter that is not fade-able is specified, this message will have no effect.

See details on booleans at the beginning of this section.

/cue/{cue_number}/parameterFadesEnabled {array_of_values}

view	edit	control	query	+/-?
read	read/write	read	✗	✗

Read: If no argument is given, return an array of boolean values representing the state of the *Fade* checkboxes for all parameters of the specified cue.

Write: If `array_of_values` is given, set the state of the *Fade* checkboxes for all parameters in the specified cue. The array must be booleans, and values are applied only to the fade-able parameters from top to bottom in the specified cue.

/cue/{cue_number}/parameterValue/{index} {value}

/cue/{cue_number}/parameterValue/{key} {value}

view	edit	control	query	+/-?
read	read/write	read	✓	(mixed)

Read: If no argument is given, return the value for the specified parameter of the specified cue.

Write: If `value` is given, set it as the value for the specified parameter of the specified cue. There are two ways to specify a parameter:

- `index` is a 0-based counter which counts the parameters in a Network cue from top to bottom, counting visible parameters only.
- `key` refers to the key name of the parameter, which is sometimes different from its display name.

`value` must be formatted according to the requirements of the parameter being set. That is to say, if the command specifies a parameter which requires string input, such as cue name, then `value` must be a string. This command can only use increment/decrement syntax when `value` is a number.

When setting a parameter with values of type `"point"`, `value` can be a string in the form `"{x,y}"`, a two number array in the form `[x,y]`, or a JSON dictionary with numeric values for keys `x` and `y`.

When setting a parameter with values of type `"menu"`, `value` can either be the display name or the key name of the menu item. For example, these two OSC messages are equivalent: `/cue/1/parameterValue/0 "Workspace Settings"` and `/cue/1/parameterValue/0 "settings"`.

NOTE: If fading is enabled for the specified parameter, the value returned for that parameter is `null`. The actual values are generated by the cue when it runs.

If `value` is improperly formatted or otherwise invalid, this message has no effect.

/cue/{cue_number}/parameterValues {array_of_values}

view	edit	control	query	+/-?
read	read/write	read	✗	✗

Read: If no argument is given, return a JSON-formatted array of values for the parameters of the specified cue.

Write: If `array_of_values` is given, set the values for all parameters of the specified cue. The array must be equal in size to the number of parameters of the specified cue, and each item in the array must be of the appropriate type and in the appropriate range of values for each given parameter. That is to say, if the specified cue is expecting two integers and a string, then `array_of_values` must be `[{integer}, {integer}, {string}]`.

If `array_of_values` is improperly formatted or otherwise invalid, this message has no effect.

/cue/{cue_number}/patch {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Deprecated in QLab 5.0. This message works in QLab 5, but incompletely, and will be removed in a future version of QLab. Use `/networkPatchNumber` or `/networkPatchID` instead.

/cue/{cue_number}/patchList

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Deprecated in QLab 5.0. This command is present for backwards compatibility and will be removed in a future version of QLab. For now, it is a synonym of `/settings/network/patchList`.

/cue/{cue_number}/pathHeight {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the maximum Y value for the 2D fade grid of the specified cue.

Write: If `number` is given, set the maximum Y value for the 2D fade grid for the specified cue. `number` can be any number greater than `0`, decimals allowed.

If the specified cue is not set to 2D fade, this message has no effect.

/cue/{cue_number}/pathWidth {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the maximum X value for the 2D fade grid of the specified cue.

Write: If `number` is given, set the maximum X value for the 2D fade grid for the specified cue. `number` can be any number greater than `0`, decimals allowed.

If the specified cue is not set to 2D fade, this message has no effect.

MIDI cue messages

Messages specific to MIDI cues. Most of these messages will only work if the specified MIDI cue is in the appropriate mode. For example, if a MIDI cue is not set to send MSC messages, then OSC messages which get or set parameters concerning MSC will not work with that cue.

/cue/{cue_number}/byte1 {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return byte 1 of the MIDI message of the specified cue.

Write: If `number` is given, set byte 1 of the MIDI message of the specified cue to `number`. `number` must be a whole number from 0 to 127.

/cue/{cue_number}/byte2 {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return byte 2 of the MIDI message of the specified cue.

Write: If `number` is given, set byte 2 of the MIDI message of the specified cue to `number`. `number` must be a whole number from 0 to 127.

/cue/{cue_number}/byteCombo {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the value of bytes 1 and 2 (as a single number) of the MIDI message of the specified cue.

Write: If `number` is given, set both bytes of the MIDI message of the specified cue based on `number`. `number` must be a whole number from 0 to 16383.

/cue/{cue_number}/channel {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the MIDI channel of the specified cue.

Write: If `number` is given, set the MIDI channel of the specified cue to `number`. `number` must be a whole number from 1 to 16.

/cue/{cue_number}/command {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the MSC command of the specified cue.

Write: If `number` is given, set the MSC command of the specified cue to `number`. `number` must be a whole number from 0 to 127 representing the index of an MSC command, a list of which can be found in the Parameter Reference page of this manual.

/cue/{cue_number}/commandFormat {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the MSC command format of the specified cue.

Write: If `number` is given, set the MSC command format of the specified cue to `number`. `number` must be a whole number from 0 to 127 representing the index of an MSC command format, a list of which can be found in the Parameter Reference page of this manual.

/cue/{cue_number}/controlNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the MSC control number of the specified cue.

Write: If `number` is given, set the MSC control number of the specified cue to `number`. `number` must be a whole number from 0 to 16383.

/cue/{cue_number}/controlValue {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the MSC control value of the specified cue.

Write: If `number` is given, set the MSC control value of the specified cue to `number`. `number` must be a whole number from 0 to 16383.

/cue/{cue_number}/deviceID {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the outgoing MSC device ID of the specified cue.

Write: If `number` is given, set the outgoing MSC device ID of the specified cue to `number`. `number` must be a whole number from 0 to 127.

/cue/{cue_number}/doFade {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Fade* checkbox of the specified cue.

Write: Set the state of the *Fade* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/endValue {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the fade ending value of the MIDI message of the specified cue.

Write: If `number` is given, set the fade ending value of the MIDI message of the specified cue to `number`. `number` must be a whole number from 0 to 127, unless the message type of the specified cue is pitch bend, in which case `number` must be a whole number between 0 and 16383.

/cue/{cue_number}/hours {number}

/cue/{cue_number}/minutes {number}

/cue/{cue_number}/seconds {number}

/cue/{cue_number}/frames {number}

/cue/{cue_number}/subframes {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the appropriate section of the MSC timecode message of the specified cue.

Write: If `number` is given, set the appropriate section of the MSC timecode message of the specified cue to `number`. `number` must be a whole number in the correct range for the appropriate section.

/cue/{cue_number}/macro {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the MSC macro of the specified cue.

**Write:* If `number` is given, set the MSC macro of the specified cue to `number`. `number` must be a whole number from 0 to 127.

/cue/{cue_number}/messageType {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the message type of the specified cue.

Write: If `number` is given, set the message type of the specified cue to `number`. Valid message types are:

```
1 - MIDI Voice Message ("Musical MIDI")
2 - MIDI Show Control Message (MSC)
3 - MIDI SysEx Message
```

/cue/{cue_number}/midiPatchName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the MIDI patch currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the name of a MIDI patch in the workspace, set the MIDI patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/midiPatchNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the index of the MIDI patch currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first patch in the patch list in Workspace Settings, `2` means the second patch, and so on.

Write: If `number` is given, set the MIDI patch of the specified cue to that patch. If `number` is `0`, un-patch the specified cue. If `number` is greater than the number of MIDI patches in the workspace, this message has no effect. `number` must be a whole number.

/cue/{cue_number}/midiPatchID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the patch ID of the MIDI patch currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of a MIDI patch in the workspace, set the MIDI patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/patch {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Deprecated in QLab 5.0. This message works in QLab 5, but incompletely, and will be removed in a future version of QLab. Use `/midiPatchNumber` or `/midiPatchID` instead.

/cue/{cue_number}/qList {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the outgoing MSC cue list number of the specified cue.

Write: If `number` is given, set the outgoing MSC cue list number of the specified cue to `number`.

/cue/{cue_number}/qNumber {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the outgoing MSC cue number of the specified cue.

Write: If `number` is given, set the outgoing MSC cue number of the specified cue to `number`.

/cue/{cue_number}/qPath {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the outgoing MSC cue path number of the specified cue.

Write: If `number` is given, set the outgoing MSC cue path number of the specified cue to `number`.

/cue/{cue_number}/rawString {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the MIDI SysEx string of the specified cue.

Write: If `string` is given, set the MIDI SysEx string of the specified cue to `string`. `string` must be a valid SysEx string, formatted in hexadecimal, and omitting the starting `F0` and ending `F7`.

/cue/{cue_number}/status {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the MIDI message type of the specified cue.

Write: If `number` is given, set the MIDI message type of the specified cue to `number`. Valid message types are:

```
0 - Note Off
1 - Note On
2 - Key Pressure (Aftertouch)
3 - Control Change
4 - Program Change
5 - Channel Pressure Change
6 - Pitch Bend Change
```

/cue/{cue_number}/timecodeFormat {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the MSC timecode format of the specified cue.

Write: If `number` is given, set the MSC timecode format of the specified cue to `number`. Valid formats are:

```
0 - 24 fps
1 - 25 fps
2 - 30 fps drop
3 - 30 fps non-drop
```

/cue/{cue_number}/timecodeString {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the MSC timecode string of the specified cue.

Write: If `number` is given, set the MSC timecode string of the specified cue to `number`.

MIDI file cue messages

Messages specific to MIDI File cues.

/cue/{cue_number}/midiPatchName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the name of the MIDI patch currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the name of a MIDI patch in the workspace, set the MIDI patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/midiPatchNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the index of the MIDI patch currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first patch in the patch list in Workspace Settings, `2` means the second patch, and so on.

Write: If `number` is given, set the MIDI patch of the specified cue to that patch. If `number` is `0`, un-patch the specified cue. If `number` is greater than the number of MIDI patches in the workspace, this message has no effect. `number` must be a whole number.

/cue/{cue_number}/midiPatchID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the patch ID of the MIDI patch currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of a MIDI patch in the workspace, set the MIDI patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

/cue/{cue_number}/patch {number}

view	edit	control	query	+/-?

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Deprecated in QLab 5.0. This message works in QLab 5, but incompletely, and will be removed in a future version of QLab. Use `/midiPatchNumber` or `/midiPatchID` instead.

/cue/{cue_number}/rate {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the rate of the specified cue.

Write: If `number` is given, set the rate of the specified cue. `number` can be any number greater than 0.01, although the range of numbers that is genuinely useful in practice is rather smaller than that.

Timecode cue messages

Messages specific to Timecode cues.

/cue/{cue_number}/audioOutputPatchName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given and the type of the specified cue is LTC, return the name of the audio output patch currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of an audio output patch in the workspace, and the type of the specified cue is LTC, set the audio output patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

If the type of the specified cue is not LTC, this message has no effect.

/cue/{cue_number}/audioOutputPatchNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given and the type of the specified cue is LTC, return the index of the audio output patch currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first patch in the patch list in Workspace Settings, `2` means the second patch, and so on.

Write: If `number` is given and the type of the specified cue is LTC, set the audio output patch of the specified cue to that patch. If `number` is `0`, un-patch the specified cue. If `number` is greater than the number of audio output patches in the workspace, this message has no effect. `number` must be a whole number.

If the type of the specified cue is not LTC, this message has no effect.

/cue/{cue_number}/audioOutputPatchID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given and the type of the specified cue is LTC, return the patch ID of the audio output patch currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of an audio output patch in the workspace, and the type of the specified cue is LTC, set the audio output patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

If the type of the specified cue is not LTC, this message has no effect.

/cue/{cue_number}/endTime {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the end time of the specified cue.

Write: If `string` is given, set the end time of the specified cue to `string` , which must be a valid timecode string (like `2:01:44:12`) or a total number of seconds (like `2344.3`).

/cue/{cue_number}/framerate {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the frame rate of the specified cue.

Write: If `number` is given, set the frame rate of the specified cue to `number` . Valid frame rates are:

```
0 - 24 fps
1 - 25 fps
2 - 30 fps non-drop
3 - 30 fps drop
4 - 23.976 fps
5 - 24.975 fps
6 - 29.97 fps non-drop
7 - 29.97 fps drop
```

/cue/{cue_number}/ltcChannel {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given and the type of the specified cue is LTC, return the output channel used by the cue.

Write: If `number` is given and the type of the specified cue is LTC, set the output channel of the specified cue to `number` , which must be a whole number.

If the type of the specified cue is not LTC, this message has no effect.

/cue/{cue_number}/outputType {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return `0` if the specified cue outputs MTC, or `1` if the specified cue outputs LTC.

Write: If `number` is given, set the output type of the specified cue to `number`. Valid frame rates are:

```
0 - MTC
1 - LTC
```

/cue/{cue_number}/startTime {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the start time of the specified cue.

Write: If `string` is given, set the start time of the specified cue to `string`, which must be a valid timecode string (like `2:01:44:12`) or a total number of seconds (like `2344.3`).

/cue/{cue_number}/midiPatchName {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given and the type of the specified cue is MTC, return the name of the MIDI patch currently in use by the specified cue. String `"none"` means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of a MIDI patch in the workspace, and the type of the specified cue is MTC, set the MIDI patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

If the type of the specified cue is not MTC, this message has no effect.

/cue/{cue_number}/midiPatchNumber {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given and the type of the specified cue is MTC, return the index of the MIDI patch currently in use by the specified cue. Index `0` means that the cue is un-patched, index `1` means the first patch in the patch list in Workspace Settings, `2` means the second patch, and so on.

Write: If `number` is given and the type of the specified cue is MTC, set the MIDI patch of the specified cue to that patch. If `number` is `0`, un-patch the specified cue. If `number` is greater than the number of MIDI patches in the workspace, this message has no effect. `number` must be a whole number.

If the type of the specified cue is not MTC, this message has no effect.

/cue/{cue_number}/midiPatchID {string}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given and the type of the specified cue is MTC, return the patch ID of the MIDI patch currently in use by the specified cue. Empty string (`""`) means that the cue is un-patched.

Write: If `string` is given and matches the patch ID of a MIDI patch in the workspace, and the type of the specified cue is MTC, set the MIDI patch of the specified cue to that patch. If `string` is `"none"` or empty (`""`), un-patch the specified cue. If `string` is anything else, this message has no effect.

If the type of the specified cue is not MTC, this message has no effect.

Reset cue messages

Messages specific to Reset cues.

/cue/{cue_number}/audioMapTargetID {id}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the ID of the audio map currently targeted by the specified cue.

Write: If `id` is given, set the target of the specified cue to the specified audio map.

/cue/{cue_number}/patchTargetID {id}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the ID of the patch currently targeted by the specified cue.

Write: If `id` is given, set the target of the specified cue to the specified patch.

/cue/{cue_number}/targetMode {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return 0 if the specified cue is set to target cues or return 1 if the specified cue is set to target patches.

Write: If `number` is given, set the target type of the specified cue. Valid types are:

```
0 - Cues
1 - Patches
```

Devamp cue messages

Messages specific to Devamp cues.

/cue/{cue_number}/devampType {number}

view	edit	control	query	+/-?
read	read/write	read	✓	✓

Read: If no argument is given, return the devamp type of the specified cue.

Write: If `number` is given, set the devamp type of the specified cue. Valid types are:

```
1 - Devamp currently looping slice
2 - Devamp looping cue
```

/cue/{cue_number}/startNextCueWhenSliceEnds {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Start next cue when slice ends* checkbox of the specified cue.

Write: Set the state of the *Start next cue when slice ends* checkbox of the specified cue. See details on booleans at the beginning of this section.

/cue/{cue_number}/stopTargetWhenSliceEnds {boolean}

view	edit	control	query	+/-?
read	read/write	read	✓	✗

Read: If no argument is given, return the state of the *Stop target when slice ends* checkbox of the specified cue.

Write: Set the state of the *Stop target when slice ends* checkbox of the specified cue. See details on booleans at the beginning of this section.

Script cue messages

Messages specific to Script cues.

/cue/{cue_number}/compileSource

view	edit	control	query	+/-?
✗	✓	✗	✗	✗

Compile the script of the specified cue.

/cue/{cue_number}/scriptSource

view	edit	control	query	+/-?
read only	read only	read only	✗	✗

Return the contents of the script of the specified cue.

OSC Queries

OSC querying is a powerful capability of the ❁ Network cue which allows you to capture the current, live value of nearly any property that can be accessed via OSC and include that value as a part of an OSC message. This is one of those concepts that is easiest to understand through examples.

Imagine a show which is using QLab and some other device on a network. The other device needs to be given a cue number via OSC, so in QLab we make a ❁ Network cue and build this message:

```
/device/standby 53
```

Sending that message sends the value `53` to the address `/device/standby`, and in our imaginary situation, that's the address that the receiving devices wants.

That's all well and good if we only need to send this one value, or maybe just a few, but if we want the device to simply follow QLab, and we have lots and lots of cues, it could get arduous to program. What we'd really like is for the device to just always know which cue is selected in QLab.

Enter OSC queries. By replacing the `53` with a query, we can build a single OSC message which inserts a value at the moment the message is sent. All we need to do is choose the right query, which in this case is:

```
#/cue/selected/number#
```

The query, which is enclosed in hashmarks[1], asks QLab for the cue number of the currently selected cue. When an OSC message including this query is actually sent by QLab, the query is replaced with the *result* of that query. The message, therefore, is built like this:

```
/device/standby #/cue/selected/number#
```

Which you can think of as:

```
/device/standby [the number of the selected cue in QLab]
```

So if cue `53` is selected when the OSC message is sent, it becomes:

```
/device/standby 53
```

But if cue `101` is selected when the OSC message is sent, it becomes:

```
/device/standby 101
```

Continuously Updating Queries

The above example works fine when you just need to extract a piece of information from QLab at a given moment, but you can also use queries to send a continuously updating value. When you use an OSC query in a ❁ Network cue, and you give that cue a duration, the query is continuously updated as long as the cue is running.

So, if you set a duration for the Network cue that sends the message:

```
/device/standby #/cue/selected/number#
```

then the output of that cue would dynamically update its query for as long as it was running, and therefore keep our mystery device updated about the currently selected cue.

Using Queries With Localhost

QLab's ability to route OSC to itself via the network address `localhost` allows you to use OSC queries to dynamically change QLab's behavior based on what's currently happening in your workspace. For example, you might like to use the loudness of an actor's voice to control the brightness of a lighting instrument. If you put a microphone in front of that actor and route it through a Mic cue, you can capture the level of the Mic using `liveAverageLevel`.

Imagine a workspace with a lighting instrument called “myLight” and a Mic cue with the cue number 10. You could create a Network cue with the OSC message:

```
/dashboard/setLight myLight #/cue/10/liveAverageLevel/1 0 100#
```

Let’s pull that apart into its individual pieces:

`/dashboard/setLight myLight {x}` is an OSC command to set `myLight` to level `x`. In this case, though, we replace `x` with a query which will return a numeric value:

```
#/cue/10/liveAverageLevel/1 0 100#
```

The hashmarks denote the query, and this particular message says “talk to cue 10, get the live average audio level of output 1, and re-scale it to a range of 0 to 100.”

If you wanted the loudness of the Mic to only vary the brightness of the light from 50% to 100%, you could change the message to:

```
/dashboard/setLight myLight #/cue/10/liveAverageLevel/1 50 100#
```

To make the most use of this, you’d probably also want to give the Network cue a duration so that it stays “alive” for as long as you need it to.

Reading the OSC Dictionary

OSC messages which can be queried are indicated in the QLab 5 OSC Dictionary with a checkmark under the heading “query”, like so:

query
✓

1. Also called pound signs or, no joke, *octothorpes*.↩

Script Cues

Script cues allow you to execute AppleScript from within QLab. AppleScript is a very old (and frankly pretty weird) scripting language which has deep integration into macOS and into many, many programs which run on macOS. A good, though fairly technical, introduction to AppleScript can be found here at macosxautomation.com. Script cues require a license of any kind.

The Inspector for Script Cues

When a Script cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as the Script tab.

The Script Tab

In a newly created Script cue, the text field in the Script tab will show the default AppleScript defined in the Script cue template.

Text entered here will follow Apple's standard formatting rules for AppleScript.

Click the **Compile Script** button to compile your script. If your script contains errors, QLab will display an error message next to this button in order to help you find and address the issue.

QLab also attempts to compile all Script cues in a workspace when it is opened. To prevent that from happening, you can hold down the ^ (control) and ⌥ (option) keys while opening the workspace.

The **Run in separate process** checkbox is checked by default. When it is checked, QLab spins off the script in a Script cue to an external invisible application which handles the execution of the AppleScript. This allows complex scripts to execute in the background, without monopolizing QLab and preventing you from interacting with QLab while the script runs. Uncheck this box if your script needs to execute from within QLab. Executing the script inside QLab improves the timing precision with which the script interacts with QLab.

Broken Script Cues

Script cues can become broken for the following reasons:

Script empty

Enter a script in the Script tab of the inspector to clear this warning.

AppleScript error

Correct the error in the Script tab of the inspector to clear this warning.

Script not compiled

The script cannot compile for some reason. Correct the error in the Script tab of the inspector to clear this warning.

License required

Script cues require a license of any kind.

AppleScript Dictionary

The list of commands, functions, properties, and so on that AppleScript can use to interact with an application is called that application's dictionary. You can find QLab's AppleScript dictionary here, or view it within the Script Editor application, which is found in /Applications/Utilities.

In Script Editor, choose *Open Dictionary...* from the **File** menu, and choose QLab from the list of applications.

AppleScript dictionaries are grouped by "suite"; all applications that use AppleScript must include the *Standard Suite*, and then any application-specific commands or properties are generally grouped together into another suite named after the application.

This documentation only describes the commands, classes, enumerations, and records from the *QLab Suite*. Items from the *Standard Suite* can also be used in QLab, such as the **save** command, which saves a specified workspace.

Readers are enthusiastically encouraged to use the navigation sidebar on this page, as AppleScript dictionaries are exceedingly verbose. Readers of the PDF version of this manual are encouraged to brace themselves accordingly.

Commands

---

audition go

(verb): make a workspace Audition GO.

Syntax

```
audition go {workspace}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace	The workspace to GO.

Classes

The following classes respond to the **audition go** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  audition go
end tell


tell application id "com.figure53.QLab.5"
  audition go workspace "hamlet.qlab5"
end tell
```

---

audition preview

(verb): Audition preview one or more cues. Previewing a cue starts the action of that cue, skipping pre-waits and ignoring auto-follows and auto-continues.

Syntax

```
preview {cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue(s) to preview.

Classes

The following classes respond to the **audition preview** command:

- any type of cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  audition preview cue "1"
end tell
```

---

capture timecode

(verb): Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.

Syntax

```
capture timecode {cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue whose timecode trigger is to be captured.

Classes

The following classes respond to the **capture timecode** command:

- any type of cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  capture timecode cue "1"
end tell
```

---

clear

(verb): clear the levels in the Light Dashboard.

Syntax

```
clear {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard you want to clear.

Classes

The following classes respond to the **clear** command:

- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  tell theDashboard to clear
end tell


tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  clear theDashboard
end tell
```

---

collapse

(verb): collapse a Group cue or cue list.

Syntax

```
collapse {group cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	group cue	The Group cue that will collapse.

Classes

The following classes respond to the **collapse** command:

- group cue
- cue list

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  collapse cue "1"
end tell
```

---

collateAndStart

(verb): collate and start a Light cue.

Syntax

```
collateAndStart {light cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light cue	The Light cue that you want to collate and start.

Classes

The following classes respond to the **collateAndStart** command:

- light cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  collateAndStart cue "1"
end tell
```

---

compile

(verb): verify and prepare the script for use.

Syntax

```
compile {script cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	script cue	The Script cue whose source you want to (re)compile.

Classes

The following classes respond to the **compile** command:

- script cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  compile cue "1"
end tell
```

---

delete

(verb): delete a cue or list of cues.

Syntax

```
delete {cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue or list	The cue or cues to delete.

Classes

The following classes respond to the **delete** command:

- cue

Examples

```
-- the cue whose cue number is "1"
delete cue "1"

-- one or several selected cues
delete selected


-- a list of cues based on position in the cue list
set listOfCues to cues 3 thru -1 of myCueList
delete listOfCues

-- an indivdual cue by position
delete last cue of myGroupCue
```

---

expand

(verb): expand a Group cue or cue list.

Syntax

```
expand {group cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	group cue	The Group cue that will expand.

Classes

The following classes respond to the **expand** command:

- group cue
- cue list

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  expand cue "1"
end tell
```

---

getGang

(verb): get the gang for a specified location in the cue's matrix.

Syntax

```
set theResult to getGang {cue} row {row_number} column {column_number}
```

Result

text: the value of the gang at the specified location in the cue's matrix.

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue from which you want to get the gang.
column	✓	integer	The column of the level matrix. Column 0 is the main and input levels column.
row	✓	integer	The row of the level matrix. Row 0 is the main and output levels row.

Classes

The following classes respond to the **getGang** command:

- audio cue
- mic cue
- video cue
- fade cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theGang to getGang 1 row 0 column 1
  display dialog "The gang for row 0, column 1 is: " & theGang
end tell
```

---

getInputChannelName

(verb): get the input name for a specified input channel.

Syntax

```
set theResult to getInputChannelName {cue} row {row_number}
```

Result

text: the name for the input channel.

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue from which you want to get the audio input name.
row	✓	integer	The row of the level matrix. Starts at 1.

Classes

The following classes respond to the **getGang** command:

- audio cue
- mic cue
- video cue
- camera cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theName to getInputChannelName 1 row 2
  display dialog "The input name for input 2 of cue 1 is: " & theName
end tell
```

---

getLevel

(verb): get the level for a specified location in the cue's matrix.

Syntax

```
set theResult to getLevel {cue} row {row_number} column {column_number}
```

Result

real number: the level in decibels of the specified location in the cue's matrix.

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue from which you want to get the level.
column	✓	integer	The column of the level matrix. Column 0 is the main and input levels column.
row	✓	integer	The row of the level matrix. Row 0 is the main and output levels row.

Classes

The following classes respond to the **getLevel** command:

- audio cue
- mic cue
- video cue
- fade cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theLevel to getLevel 1 row 0 column 1
  display dialog "The level for row 0, column 1 of cue 1 is: " & theLevel
end tell
```

---

getMute

(verb): get the status of the mute button for a specified output of the cue.

Syntax

```
set theResult to getMute {cue} output {output_number}
```

Result

boolean: the state of the specified mute control.

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue from which you want to get the mute state.
output	✓	integer	The output of the level matrix. Output 0 is the main level.

Classes

The following classes respond to the **getMute** command:

- audio cue
- mic cue
- video cue
- camera cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theMute to cue "6" getMute output 1
  display dialog "The mute state of output 1 is: " & theMute
end tell
```

---

getSolo

(verb): get the status of the solo button for a specified output of the cue.

Syntax

```
set theResult to getSolo {cue} output {output_number}
```

Result

boolean: the state of the specified solo control.

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue from which you want to get the solo state.
output	✓	integer	The output of the level matrix. Output 0 is the main level.

Classes

The following classes respond to the **getMute** command:

- audio cue
- mic cue
- video cue
- camera cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theSolo to cue "6" getMute output 1
  display dialog "The solo state of output 1 is: " & theMute
end tell
```

---

go

(verb): make a workspace GO.

Syntax

```
go {workspace}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace	The workspace to GO.

Classes

The following classes respond to the **go** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5"
  go front workspace
end tell
```

```
tell application id "com.figure53.QLab.5"
  go workspace "hamlet.qlab5"
end tell
```

---

hardStop

(verb): hardStop one or more cues or workspaces.

Syntax

```
hardStop {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference	The cue(s) or workspace(s) to hardStop.

Classes

The following classes respond to the **hardStop** command:

- workspace
- cue list
- any type of cue

Examples

```
tell application id "com.figure53.QLab.5"
  hardStop cue "1"
end tell


tell application id "com.figure53.QLab.5"
  hardStop front workspace
end tell
```

---

load

(verb): load one or more cues or workspaces to a given time. A negative value loads that many seconds back from the end of the cue.

Syntax

```
load {reference} time {real number}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference	The cue(s) or workspace(s) to load.
time	✗	real number	Load time.

Note: Because "load" is both a noun and a verb in QLab (load vs. Load cue), you need to place the reference within parentheses; see examples below.

Classes

The following classes respond to the **load** command:

- workspace
- cue list
- any type of cue

Examples

```
tell application id "com.figure53.QLab.5"
  -- load the cue whose cue number is "1"
  load (cue "1")
end tell


tell application id "com.figure53.QLab.5"
  -- load the cue whose cue number is "2" to 15 seconds
  load (cue "2") time 15
end tell


tell application id "com.figure53.QLab.5"
  -- load the cue whose cue number is "3" to 20 seconds before the end of the cue
```

```
  load (cue "3") time -20
end tell


tell application id "com.figure53.QLab.5"
  -- load the cue whose cue id is 4
  load (cue 4)
end tell


tell application id "com.figure53.QLab.5"
  -- this does not work!
  -- AppleScript thinks you're talking about a Load cue whose cue number is "5"
  load cue "5"
end tell
```

---

make

(verb): make a new cue in a workspace. The cue will be created below the currently selected cue.

Syntax

```
make {workspace} type {text}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace	The workspace in which the cue will be made.
type	✓	text	The name of the kind of cue you want to create (audio, video, camera, etc.) To create a new cue list, use “cue list”. To create a new cue cart, use “cue cart”.

Classes

The following classes respond to the **make** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5"
  make front workspace type "audio"
end tell


tell application id "com.figure53.QLab.5"
  make front workspace type "cue list"
end tell


tell application id "com.figure53.QLab.5"
  make workspace "hamlet.qlab5" type "midi"
end tell
```

---

move

(verb): move a cue or list of cues to a new location.

Syntax

```
move {reference to cue or list} to {location specifier}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference to cue or list of cues	The cue or list of cues to move.
to	✓	location specifier	The new location for the cue or cues.

Examples

```
-- moving a cue
set theCue to cue "FROMCUE"
set theDestination to cue "DESTCUE"
move theCue to {after|before|beginning of|end of} theDestination


-- moving a list of cues
set theCues to {list_of_cues}
set theDestination to cue "DESTCUE"
move theCues to {after|before|beginning of|end of} theDestination


-- by position within a group or list
set theGroupCue to cue "MYGROUP"
set theDestination to cue "DESTCUE"
move last cue of theGroupCue to {after|before|beginning of|end of} theDestination
```

---

movePlayheadDown

(verb): move the playhead to the next cue.

Syntax

```
movePlayheadDown {specifier}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace or cue number	The workspace or cue list whose playhead will change.

Classes

The following classes respond to the **movePlayheadDown** command:

- workspace
- cue list

Examples

```
-- move the playhead to the next cue in the active cue list of the front workspace
tell application id "com.figure53.QLab.5" to tell front workspace
```

```
  movePlayheadDown
end tell


-- move the playhead to the next cue in the active cue list of a specific workspace
tell application id "com.figure53.QLab.5"
  movePlayheadDown workspace "hamlet.qlab5"
end tell


-- move the playhead to the next cue in a cue list numbered "101" in the front workspace
tell application id "com.figure53.QLab.5" to tell front workspace
  movePlayheadDown cue "101"
end tell


-- move the playhead to the next cue in a cue list numbered "101" in a specific workspace
tell application id "com.figure53.QLab.5" to tell cue "101" of workspace "ophelia"
  movePlayheadDown
end tell
```

---

movePlayheadDownASequence

(verb): move the playhead to the top of the next cue sequence.

Syntax

```
movePlayheadDownASequence {workspace}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace	The workspace whose playhead will change.

Classes

The following classes respond to the **movePlayheadDownASequence** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  movePlayheadDownASequence
end tell


tell application id "com.figure53.QLab.5"
  movePlayheadDownASequence workspace "hamlet.qlab5"
end tell
```

---

movePlayheadUp

(verb): move the playhead to the previous cue.

Syntax

```
movePlayheadUp {specifier}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace or cue number	The workspace or cue list whose playhead will change.

Classes

The following classes respond to the **movePlayheadUp** command:

- workspace

Examples

```
-- move the playhead to the next cue in the active cue list of the front workspace
tell application id "com.figure53.QLab.5" to tell front workspace
  movePlayheadUp
end tell


-- move the playhead to the next cue in the active cue list of a specific workspace
tell application id "com.figure53.QLab.5"
  movePlayheadUp workspace "hamlet.qlab5"
end tell


-- move the playhead to the next cue in a cue list numbered "101" in the front workspace
tell application id "com.figure53.QLab.5" to tell front workspace
  movePlayheadUp cue "101"
end tell


-- move the playhead to the next cue in a cue list numbered "101" in a specific workspace
tell application id "com.figure53.QLab.5" to tell cue "101" of workspace "ophelia"
  movePlayheadUp
end tell
```

---

movePlayheadUpASequence

(verb): move the playhead to the top of the previous cue sequence.

Syntax

```
movePlayheadUpASequence {workspace}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace	The workspace whose playhead will change.

Classes

The following classes respond to the **movePlayheadUpASequence** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  movePlayheadUpASequence
end tell


tell application id "com.figure53.QLab.5"
  movePlayheadUpASequence workspace "hamlet.qlab5"
end tell
```

---

moveSelectionDown

(verb): select the next cue.

Syntax

```
moveSelectionDown {workspace}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace	The workspace whose selection will change.

Classes

The following classes respond to the **moveSelectionDown** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  moveSelectionDown
end tell


tell application id "com.figure53.QLab.5"
  moveSelectionDown workspace "hamlet.qlab5"
end tell
```

---

moveSelectionUp

(verb): select the previous cue.

Syntax

```
moveSelectionUp {workspace}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	workspace	The workspace whose selection will change.

Classes

The following classes respond to the **moveSelectionUp** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  moveSelectionUp
end tell


tell application id "com.figure53.QLab.5"
  moveSelectionUp workspace "hamlet.qlab5"
end tell
```

---

newCueWithAll

(verb): create a new Light cue containing levels for all parameters of all instruments.

Syntax

```
newCueWithAll {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard from which you want to create a new Light cue.

Classes

The following classes respond to the **newCueWithAll** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  newCueWithAll theDashboard
end tell
```

---

newCueWithChanges

(verb): create a new Light cue containing levels for all manually adjusted parameters in the Light Dashboard.

Syntax

```
newCueWithChanges {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard from which you want to create a new Light cue.

Classes

The following classes respond to the **newCueWithChanges** command:

- workspace

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  newCueWithChanges theDashboard
end tell
```

---

panic

(verb): panic one or more cues or workspaces.

Syntax

```
panic {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference	The cue(s) or workspace(s) to panic.

Classes

The following classes respond to the **panic** command:

- workspace
- any type of cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  panic
end tell
```

```
tell application id "com.figure53.QLab.5"
  panic front workspace
end tell
```

```
tell application id "com.figure53.QLab.5" to tell front workspace
  panic cue "1"
end tell
```

---

pause

(verb): pause one or more cues or workspaces.

Syntax

```
pause {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference	The cue(s) or workspace(s) to pause.

Classes

The following classes respond to the **pause** command:

- workspace
- any type of cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  pause
end tell
```

```
tell application id "com.figure53.QLab.5"
  pause front workspace
end tell
```

```
tell application id "com.figure53.QLab.5" to tell front workspace
  pause cue "1"
end tell
```

---

preview

(verb): preview one or more cues. Previewing a cue starts the action of that cue, skipping pre-waits and ignoring auto-follows and auto-continues.

Syntax

```
preview {cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue(s) to preview.

Classes

The following classes respond to the **preview** command:

- any type of cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  preview cue "1"
end tell
```

---

prune

(verb): remove light commands that have no effect from a Light cue.

Syntax

```
prune {light cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light cue	The Light cue(s) whose command text you want to prune.

Classes

The following classes respond to the **prune** command:

- light cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  prune cue "1"
end tell
```

---

recordAllToLatest

(verb): record all levels for all parameters of all instruments into the latest run Light cue.

Syntax

```
recordAllToLatest {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard from which you want to record all to the latest cue.

Classes

The following classes respond to the **recordAllToLatest** command:

- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  recordAllToLatest theDashboard
end tell
```

---

recordAllToSelected

(verb): record all levels for all parameters of all instruments into the selected Light cue(s).

Syntax

```
recordAllToSelected {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard from which you want to record all to the selected cue(s).

Classes

The following classes respond to the **recordAllToSelected** command:

- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  recordAllToSelected theDashboard
end tell
```

---

redo

(verb): redo the last undone action.

Syntax

```
redo {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference	The workspace or Light Dashboard in which you want to redo the last undone action.

Classes

The following classes respond to the **redo** command:

- workspace
- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  redo
end tell
```

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  redo theDashboard
end tell
```

---

removeLightCommandsMatching

(verb): remove existing light commands in the specified cue matching the command provided.

Syntax

```
removeLightCommandsMatching {cue} command {text}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue from which you want to remove light commands.
command	✓	text	The full text of the light command you want to remove.

Classes

The following classes respond to the **removeLightCommandsMatching** command:

- light cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  removeLightCommandsMatching cue "1" command "myLight.intensity = 100"
end tell
```

---

replaceLightCommand

(verb): replace a specified light command in the specified cue with a new light command.

Syntax

```
replaceLightCommand {cue} oldCommandText {text} newCommandText {text}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue in which you want to replace light commands.
newCommandText	✓	text	The full text of the light command that will replaced the old.
oldCommandText	✓	text	The full text of the light command that will be replaced.

Classes

The following classes respond to the **replaceLightCommand** command:

- light cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  replaceLightCommand cue "1" oldCommandText "myLight.intensity = 100" newCommandText "myLight.intensity = 80"
end tell
```

---

reset

(verb): reset one or more cues or workspaces.

Syntax

```
reset {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference	The cue(s) or workspace(s) to reset.

Classes

The following classes respond to the **reset** command:

- workspace
- any type of cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  reset cue "1"
end tell
```

```
tell application id "com.figure53.QLab.5" to tell front workspace
  reset
end tell
```

```
tell application id "com.figure53.QLab.5"
  reset workspace "hamlet.qlab5"
end tell
```

---

revert

(verb): revert changes in the specified Light Dashboard.

Syntax

```
revert {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard in which you want to revert changes.

Classes

The following classes respond to the **revert** command:

- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
```

```
    revert theDashboard
end tell
```

---

save

(verb): save the last undone action.

Syntax

```
save {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference	The workspace or Light Dashboard in which you want to save the last undone action.

Classes

The following classes respond to the **save** command:

- workspace
- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
    save
end tell
```

---

setGang

(verb): set the gang for a specified location in the cue's matrix.

Syntax

```
setGang {cue} row {row_number} column {column_number} gang {text}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue for which you want to set the gang.
column	✓	integer	The column of the level matrix. Column 0 is the main and input levels column.
gang	✓	text	The gang to set.
row	✓	integer	The row of the level matrix. Row 0 is the main and output levels row.

Classes

The following classes respond to the **setGang** command:

- audio cue
- mic cue
- video cue
- fade cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set cue "1" row 0 column 1 gang "a"
end tell
```

---

setInputChannelName

(verb): set the input name for a specified row of a cue.

Syntax

```
setInputChannelName {cue} row {row_number} name {text}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue for which you want to set the audio input name.
name	✓	text	The name for the input channel.
row	✓	integer	The row of the level matrix. Starts at 1.

Classes

The following classes respond to the **setInputChannelName** command:

- audio cue
- mic cue
- video cue
- camera cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set cue "1" row 0 input "vocals"
end tell
```

---

setLevel

(verb): set the level in decibels for a specified location in the cue's matrix.

Syntax

```
setLevel {cue} row {row_number} column {column_number} db {real number}
```

Parameters

Parameter	Required?	Type	Description

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue for which you want to set the level.
column	✓	integer	The column of the level matrix. Column 0 is the main and input levels column.
db	✓	real number	The level in decibels to set.
row	✓	integer	The row of the level matrix. Row 0 is the main and output levels row.

Classes

The following classes respond to the **setLevel** command:

- audio cue
- mic cue
- video cue
- fade cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  setLevel cue "1" row 0 column 1 db "-6"
end tell
```

---

setLight

(verb): add a light command to the specified cue or to the Light Dashboard.

Syntax

```
setLight {reference} selector {text} value {real number or text}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	reference	The Light cue or Light Dashboard for which you want to add a light command.
selector	✓	text	The instrument or group name for the command you want to add. Using a parameter name as well is optional.
value	✗	real number or text	Optional parameter value to set.

Classes

The following classes respond to the **setLight** command:

- light dashboard
- light cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  setLight cue "1" selector "myLight"
end tell


tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
```

```
  setLight theDashboard selector "myLight.red" value "50"
end tell
```

---

setMute

(verb): set the status of the mute button for a specified output of the cue.

Syntax

```
setMute {cue} output {output_number} mute {boolean}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue for which you want to set the mute.
output	✓	integer	The output number. Output 0 is the main level.
mute	✓	boolean	True = mute. False = unmute.

Classes

The following classes respond to the **setMute** command:

- audio cue
- mic cue
- video cue
- camera cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  setMute cue "1" output 0 mute true
end tell
```

---

setSolo

(verb): set the status of the solo button for a specified output of the cue.

Syntax

```
setSolo {cue} output {output_number} solo {boolean}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue for which you want to set the mute.
output	✓	integer	The output number. Output 0 is the main level.
solo	✓	boolean	True = solo. False = unsolo.

Classes

The following classes respond to the **setSolo** command:

- audio cue
- mic cue
- video cue
- camera cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  setSolo cue "1" output 0 mute true
end tell
```

---

shuffle

(verb): shuffle the order of child cues in a Group cue.

Syntax

```
shuffle {group cue}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	group cue	The Group cue(s) whose child cues you want to shuffle.

Classes

The following classes respond to the **start** command:

- group cue
- cue list

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  start
end tell
```

---

start

(verb): start one or more cues or workspaces.

Syntax

```
start {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue(s) or workspace(s) to start. Starting a workspace unpauses all paused cues; it does not start cues which are not paused.

Classes

The following classes respond to the **start** command:

- workspace
- any type of cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  start
end tell


tell application id "com.figure53.QLab.5" to tell front workspace
  start cue "1"
end tell
```

---

stop

(verb): stop one or more cues or workspaces.

Syntax

```
stop {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The cue(s) or workspace(s) to stop.

Classes

The following classes respond to the **stop** command:

- workspace
- any type of cue

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  stop
end tell


tell application id "com.figure53.QLab.5" to tell front workspace
  stop cue "1"
end tell
```

---

undo

(verb): undo the last action.

Syntax

```
undo {reference}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	cue	The workspace or Light Dashboard in which you want to undo the last action.

Classes

The following classes respond to the **undo** command:

- workspace
- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  undo
end tell
```

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  undo theDashboard
end tell
```

---

updateLatestCue

(verb): copy all manually adjusted levels into the latest run Light cue.

Syntax

```
updateLatestCue {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard from which you want to update the latest cue.

Classes

The following classes respond to the **updateLatestCue** command:

- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  updateLatestCue theDashboard
end tell
```

---

updateOriginatingCues

(verb): copy all manually adjusted levels into their originating Light cue(s).

Syntax

```
updateOriginatingCues {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard from which you want to update the originating cue(s).

Classes

The following classes respond to the **updateOriginatingCues** command:

- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  updateOriginatingCues theDashboard
end tell
```

---

updateSelectedCues

(verb): copy all manually adjusted levels into the selected Light cue(s).

Syntax

```
updateSelectedCues {light dashboard}
```

Parameters

Parameter	Required?	Type	Description
direct parameter	✓	light dashboard	The Light Dashboard from which you want to update the selected cue(s).

Classes

The following classes respond to the **updateSelectedCues** command:

- light dashboard

Examples

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set theDashboard to current light dashboard
  updateSelectedCues theDashboard
end tell
```

Classes

---

application

(noun): the top-level scripting object of QLab.

Properties

Property	Access	Type	Description
frontmost	get	boolean	Is this the frontmost (active) application?
name	get	text	The name of the application.
overrides	get	override controller	Application-wide communication overrides.
preferences	get	preferences controller	Application-wide preferences and settings.
version	get	text	The version of the application.

Elements

Element	Access	Key Forms	Description
document	get	by name, by index, by range, relative to others, by whose/where, by unique ID	
window	get	by name, by index, by range, relative to others, by whose/where, by unique ID	
workspace	get	by name, by index, by range, relative to others, by whose/where, by unique ID	

Commands

The **application** class responds to the following commands:

Command	Description
open	Open QLab.
print	This command has no effect in QLab.
quit	Quit QLab.

---

audio cue

*(noun), pl. **audio cues** *

Properties

In addition to the properties listed here, **audio cue** inherits properties from [cue](#).

Property	Access	Type	Description
audio input channels	get	integer	The number of audio input channels for this cue (i.e. the number of distinct channels in the target audio file.)
audio output patch name	get/set	text	The name of this cue's audio output patch. `none` means "unpatched."
audio output patch number	get/set	integer	The 1-indexed number of this cue's audio output patch. `0` means "unpatched."
audio output patch id	get/set	text	The unique ID of this cue's audio output patch. Empty string or `none` means "unpatched."
end time	get/set	real number	Time in the target file where playback ends.
infinite loop	get/set	boolean	Does this cue loop infinitely?
integrated fade	get/set	enabled or disabled	State of the integrated fade checkbox.
last slice infinite loop	get/set	boolean	Does the last slice of this cue loop infinitely?
last slice play count	get/set	integer	Number of times the last slice of this cue plays. Always >= `1`.
lock fade to cue	get/set	enabled or disabled	State of the lock fade to start/end checkbox.
patch	get/set	integer	The 1-indexed number of this cue's audio output patch. *Deprecated in QLab 5.0 - use `audio output patch number` instead.*
play count	get/set	boolean	Number of times this cue plays. Always >= `1`.

Property	Access	Type	Description
preserve pitch	get/set	enabled disabled	State of the preserve pitch checkbox.
rate	get/set	real number	Playback rate of this cue.
slice markers	get/set	list of slice marker record	List of slice markers in this cue.
start time	get/set	real number	Time in the target file where playback begins.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **audio cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
getGang	Get the gang for a specified location in the cue's matrix.
getInputChannelName	Get the input name for a specified row in the cue's matrix.
getLevel	Get the level for a specified location in the cue's matrix.
getMute	Get the status of the mute button of a specified output.
getSolo	Get the status of the solo button of a specified output.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
setGang	Set the gang for a specified location in the cue's matrix.
setInputChannelName	Set the intput name for a specified row in the cue's matrix.
setLevel	Set the level for a specified location in the cue's matrix.
setMute	Set the status of the mute button of a specified output.
setSolo	Set the status of the solo button of a specified output.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **audio cue** class inherits elements and properties from the **cue** class.

---

camera cue

*(noun), pl. **camera cues** *

Properties

In addition to the properties listed here, **camera cue** inherits properties from **cue**.

Property	Access	Type	Description
anchor x	get/set	real number	Anchor along the x axis.
anchor y	get/set	real number	Anchor along the y axis.
audio input channels	get	integer	The number of audio input channels for this cue (i.e. the number of distinct channels in the target audio file.)
audio input patch name	get/set	text	The name of this cue's audio input patch. `none` means "unpatched."
audio input patch number	get/set	integer	The 1-indexed number of this cue's audio input patch. `0` equals "unpatched."
audio input patch id	get/set	text	The unique ID of this cue's audio input patch. Empty string or `none` equals "unpatched."
audio output patch name	get/set	text	The name of this cue's audio output patch. `none` means "unpatched."
audio output patch number	get/set	integer	The 1-indexed number of this cue's audio output patch. `0` means "unpatched."
audio output patch id	get/set	text	The unique ID of this cue's audio output patch. Empty string or `none` means "unpatched."
blend mode	get/set	text	Display name of the video blend mode.
camera patch	get/set	integer	The 1-indexed number of this cue's camera patch. *Deprecated in QLab 5.0 - use `video input patch number` instead.*
fill stage	get/set	boolean	Is the cue displaying in fill stage mode?
fill style	get/set	fill styles	How does the cue fill the stage?
full screen	get/set	boolean	Is the cue displaying in full-stage mode? *Deprecated in QLab 5.0 - use 'fill stage' instead.*
full surface	get/set	boolean	Is the cue displaying in full-stage mode? *Deprecated in QLab 5.0 - use 'fill stage' instead.*
layer	get/set	integer	The display layer of this cue. `0` is the bottom layer; `1000` is the top layer.
opacity	get/set	real number	The opacity of this cue. `0` = 0%; `0.5` = 50%; `1` = 100%
scale x	get/set	real number	The X-axis scale of this cue.
scale y	get/set	real number	The Y-axis scale of this cue.
smooth	get/set	boolean	Should the cue be scaled using smoothing interpolation?
stage name	get/set	text	Video output stage name. Empty string or `none` means "unpatched."
stage number	get/set	integer	Video output stage number. `0` means "unpatched."
stage id	get/set	text	Video output stage unique ID. Empty string or `none` means "unpatched."
translation x	get/set	real number	The X-axis translation (position) of this cue.
translation y	get/set	real number	The Y-axis translation (position) of this cue.
video input patch name	get/set	text	The name of this cue's video input patch. `none` means "unpatched."
video input patch number	get/set	integer	The 1-indexed number of this cue's video input patch. `0` means "unpatched."
video input patch id	get/set	text	The unique ID of this cue's video input patch. Empty string or `none` means "unpatched."

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **camera cue** class responds to the following commands:

Command	Description

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
getGang	Get the gang for a specified location in the cue's matrix.
getInputChannelName	Get the input name for a specified row in the cue's matrix.
getLevel	Get the level for a specified location in the cue's matrix.
getMute	Get the status of the mute button of a specified output.
getSolo	Get the status of the solo button of a specified output.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
setGang	Set the gang for a specified location in the cue's matrix.
setInputChannelName	Set the intput name for a specified row in the cue's matrix.
setLevel	Set the level for a specified location in the cue's matrix.
setMute	Set the status of the mute button of a specified output.
setSolo	Set the status of the solo button of a specified output.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **camera cue** class inherits elements and properties from the **cue** class.

---

cue

*(noun), pl. **cues** *

Properties

All cues have properties from this list, but not every type of cue has every property. For example, only cue type which accept a file target (Audio, Video, and MIDI file) have the *file target* property.

Property	Access	Type	Description
action elapsed	get	real number	The time in seconds that have elapsed in the action of this cue.
armed	get/set	boolean	Is this cue armed?
autoload	get/set	boolean	Does this cue auto-load?
broken	get	boolean	Is this cue broken?
cart position	get	row column record	The row and column numbers for the position of this cue within a cart. A cue that is not contained within a cart will return row and column `0`.
continue mode	get/set	contine modes	The continue mode of this cue.

Property	Access	Type	Description
cue target	get/set	cue	The cue this cue targets, if any.
current duration	get	real number	The current duration of this cue's action in seconds. This property reflects the temporary duration, if it has been set. Otherwise, it returns this cue's duration.
duck level	get/set	real number	The "duck or boost others" audio level. Accepts a range of `-120` to `12`.
duck others	get/set	enabled or disabled	The "duck or boost others" setting of this cue.
duck time	get/set	real number	The "duck or boost others" fade time.
duration	get	real number	The duration of this cue's action in seconds. Not editable for all cue types.
fade and stop others	get/set	integer	The "fade and stop others" setting of the cue: 0 = disabled 1 = peers 2 = list 3 = all
fade and stop others time	get/set	real number	The "fade and stop others" time in seconds.
file target	get/set	file	The file this cue targets, if any.
flagged	get/set	boolean	Is this cue flagged?
hotkey trigger	get/set	enabled or disabled	State of the hotkey trigger checkbox.
loaded	get	boolean	Is this cue loaded?
midi byte one	get/set	string	Byte 1 of this cue's MIDI trigger, if any.
midi byte one string	get/set	string	Display string of byte 1 of this cue's MIDI trigger, if any.
midi byte two	get/set	integer	Byte 2 of this cue's MIDI trigger, if any.
midi byte two string	get/set	string	Display string of byte 2 of this cue's MIDI trigger, if any.
midi command	get/set	midi commands	Type of MIDI command used for this cue's MIDI trigger, if any. NOTE: pitch bend messages cannot be used for MIDI triggers.
midi trigger	get/set	enabled or disabled	State of the MIDI trigger checkbox.
midi trigger channel	get/set	integer	MIDI channel used for the MIDI trigger of the cue. `0` is the workspace channel, `-1` is "any channel".
notes	get/set	text	The notes for this cue.
parent	get	group cue	The parent cue of this cue.
parent list	get	cue	The cue list or cue cart that contains this cue.
paused	get	boolean	Is this cue paused?
percent action elapsed	get	real number	The percentage of this cue's action that has elapsed.
percent post wait elapsed	get	real number	The percentage of this cue's post-wait that has elapsed.
percent pre wait elapsed	get	real number	The percentage of this cue's pre-wait that has elapsed.
post wait	get/set	real number	The time in seconds before this cue auto-continues, if this cue is set to auto-continue.
post wait elapsed	get	real number	The time in seconds of this cue's post-wait that has elapsed.

Property	Access	Type	Description
pre wait	get/set	real number	The time in seconds that this cue's action will delay after being started.
pre wait elapsed	get	real number	The time in seconds of this cue's pre-wait that has elapsed.
q color	get/set	text	The name of this cue's color, or "none" if no color is set.
q color 2	get/set	text	The name of this cue's second color, or "none" if no color is set.
q default name	get	text	The name that QLab would give to this cue by default.
q display name	get	text	The name of this cue as displayed in the standby view. Never empty.
q list name	get	text	The name of this cue as displayed in the cue list or cart. Might be a default name.
q name	get/set	text	The name of this cue.
q number	get/set	text	The number of this cue. Unique if present. May be empty.
q type	get	text	The name of this type of cue (i.e. "Audio", "Video", etc.)
running	get	boolean	Is this cue running?

second trigger action	get/set	integer	The second trigger action of the cue:
			`0 = do nothing`
			`1 = panic`
			`2 = stop`
			`3 = hard stop`
			`4 = hard stop and restart`
			`5 = devamp`
			`6 = (playlist only) play next`
			`7 = (playlist only) play previous`

second trigger on release	get/set	enabled or disabled	State of the "second trigger on release" checkbox.
skip if disarmed	get/set	boolean	Skip this cue if it is not armed?
temp duration	get/set	real number	The temporary duration of this cue's action in seconds. Not all cues support temporary durations. Setting the temporary duration does not mark the document as edited. Reset the cue to restore its original, saved duration.
timecode bits	get/set	integer	The bits field of the timecode trigger of this cue.
timecode frames	get/set	integer	The frames field of the timecode trigger of this cue.
timecode hours	get/set	integer	The hours field of the timecode trigger of this cue.
timecode minutes	get/set	integer	The minutes field of the timecode trigger of this cue.
timecode seconds	get/set	integer	The seconds field of the timecode trigger of this cue.
timecode show as timecode	get/set	boolean	True if the timecode trigger is shown as timecode; false if shown as real time.
timecode text	get/set	text	Text representation of the timecode trigger.
timecode trigger	get/set	enabled or disabled	State of the timecode trigger checkbox.
uniqueID	get	text	The unique ID of this cue.
use q color 2	get/set	boolean	Whether this cue will use the second cue color after it starts.
wall clock hours	get/set	integer	The hours field of the wall clock trigger of this cue.
wall clock minutes	get/set	integer	The minutes field of the wall clock trigger of this cue.
wall clock seconds	get/set	integer	The seconds field of the wall clock trigger of this cue.
wall clock trigger	get/set	enabled or disabled	State of the wall clock trigger checkbox.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	Cues contained by this cue, if any.

Commands

The **cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Subclasses

All cue classes inherit the elements and properties of the **cue** class.

Where Used

The **cue** class is used in the following ways:

- element of workspace class.
- direct parameter to the getLevel command.
- direct parameter to the setLevel command.
- direct parameter to the getGang command.
- direct parameter to the setGang command.
- direct parameter to the getInputChannelName command.
- direct parameter to the setInputChannelName command.
- direct parameter to the getMute command.
- direct parameter to the setMute command.
- direct parameter to the getSolo command.
- direct parameter to the setSolo command.
- direct parameter to the replaceLightCommand command.
- direct parameter to the removeLightCommandsMatching command.
- **active cues** property of the workspace class.
- **cue target** property of the cue class.
- **parent** property of the cue class.
- **parent list** property of the cue class.
- **playback position** property of the cue list class.
- **playhead** property of the cue list class.
- **selected** property of the workspace class.

---

cue list

*(noun), pl. **cue lists** *

Properties

In addition to the properties listed here, **cue list** inherits properties from **cue** and from **group cue**.

Property	Access	Type	Description
current timecode text	get	text	The timecode currently being received.
ltc sync channel	get/set	integer	Audio channel supplying an LTC sync signal.
mtc sync source name	get/set	text	Name of the MIDI device supplying an MTC sync signal.
playback position	get/set	cue	The playback position of a cue list is the cue which is standing by and which will start at the next GO.
playhead	get/set	cue	Synonym for **playback position**.
smpte format	get/set	timecode smpte format	The SMPTE format of the incoming timecode.
sync mode	get/set	sync mode	Which type of incoming timecode this cue list listens for.
sync to timecode	get/set	enabled or disabled	State of the sync to timecode checkbox.
timecode start behavior	get/set	timecode start	How cues in this list will behave when timecode starts.
timecode stop behavior	get/set	timecode stop	How cues in this list will behave when timecode stops.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	Cues contained by this cue list, if any.

Commands

The **cue list** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
collapse	Collapse the cue list in the sidebar.
expand	Expand the cue list in the sidebar.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
movePlayheadDown	Move the playhead in the active cue list to the next cue.
movePlayheadDownASequence	Move the playhead in the active cue list to top of the next cue sequence.
movePlayheadUp	Move the playhead in the active cue list to the previous cue.
movePlayheadUpASequnce	Move the playhead in the active cue list to top of the previous cue sequence.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **cue list** class inherits elements and properties from the **group cue** class.

Where Used

The **cue list** class is used in the following ways:

- element of the workspace class.
- **current cue list** property of the workspace class.

---

devamp cue

*(noun), pl. **devamp cues** *

Properties

In addition to the properties listed here, **devamp cue** inherits properties from **cue**.

Property	Access	Type	Description
start next cue when slice ends	get/set	boolean	Start the next cue at the moment the target slice ends?
stop target when slice ends	get/set	boolean	Stop the target at the moment the target slice ends?

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **devamp cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **devamp cue** class inherits elements and properties from the **cue** class.

---

fade cue

*(noun), pl. **fade cues** *

Properties

In addition to the properties listed here, **fade cue** inherits properties from **cue**.

Property	Access	Type	Description
audio fade mode	get/set	absolute or relative	Set absolute or relative mode for fading audio levels.
fade mode	get/set	absolute or relative	*Deprecated in QLab 5.0 - use* `audio fade mode` *instead.*
do opacity	get/set	boolean	Does this cue animate opacity?
do rotation	get/set	boolean	Does this cue animate rotation?
do scale	get/set	boolean	Does this cue animate scale?
do translation	get/set	boolean	Does this cue animate translation?
opacity	get/set	real number	Video opacity to fade to. `0` = 0%; `0.5` = 50%; `1` = 100%
patch target id	get/set	text	The patch this cue targets, if any.
preserve aspect ratio	get/set	boolean	Does this cue preserve aspect ratio?
rotation	get/set	real number	Rotation in degrees when this cue's *rotation type* is set to a single-axis mode (modes 1, 2, or 3). When *rotation type* is 3D orientation (mode 0), this cannot be used to set and returns 0.0 when used to get.
rotation type	get/set	integer	0 = 3D orientation 1 = X axis 2 = Y axis 3 = Z axis
scale x	get/set	real number	X-axis scale to fade to.
scale y	get/set	real number	Y-axis scale to fade to.

Property	Access	Type	Description
stop target when done	get/set	boolean	Stop the target cue when this cue completes?
target mode	get/set	target modes	The target mode of this cue.
video fade mode	get/set	absolute or relative	Set absolute or relative mode for fading video geometry.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **fade cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
getGang	Get the gang for a specified location in the cue's matrix.
getLevel	Get the level for a specified location in the cue's matrix.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
setGang	Set the gang for a specified location in the cue's matrix.
setLevel	Set the level for a specified location in the cue's matrix.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **fade cue** class inherits elements and properties from the **cue** class.

---

group cue

*(noun), pl. **group cues** *

Properties

In addition to the properties listed here, **group cue** inherits properties from **cue**.

Property	Access	Type	Description
mode	get/set	group modes	The playback behavior of this group.
playlist crossfade	get/set	boolean	Does this playlist group crossfade between its child cues?

Property	Access	Type	Description
playlist crossfade duration	get/set	real number	The duration of the cue's playlist crossfade in seconds.
playlist loop	get/set	boolean	Does this playlist group loop?
playlist shuffle	get/set	boolean	Does this playlist group shuffle the order of its child cues?

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	Cues contained by this cue, if any.

Commands

The **group cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
collapse	Collapse the cue list in the cue list.
expand	Expand the cue list in the cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
shuffle	Shuffle the order of child cues in a group cue.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **group cue** class inherits elements and properties from the **cue** class.

Subclasses

The **cue list** class inherits elements and properties from the **group cue** class.

Where Used

The **group cue** class is used in the following ways:

- direct parameter to the collapse command.
- direct parameter to the expand command.

---

light cue

*(noun), pl. **light cues** *

Properties

In addition to the properties listed here, **light cue** inherits properties from **cue**.

Property	Access	Type	Description
always collate	get/set	boolean	Flag for whether this cue should always collate the effects of previous light cues in the same list when it runs.
command text	get/set	text	The light command text of this cue.
subcontroller	get/set	boolean	Is this cue used as a subcontroller in the Light Dashboard?

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **light cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
collateAndsStart	Collate and start the light cue.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
prune	Remove light commands that have no effect from a light cue.
removeLightCommandsMatching	Remove existing light commands in the specified cue matching the command provided.
replaceLightCommand	Replace a specified light command in the specified cue with a new light command.
reset	Reset one or more cues or workspaces.
setLight	Add a light command to the specified cue.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **light cue** class inherits elements and properties from the **cue** class.

Where Used

The **light cue** class is used in the following ways:

- direct parameter to the prune command.
- direct parameter to the collateAndStart command.

---

light dashboard

(noun)

Property	Access	Type	Description
dashboard fade time	get/set	real number	The duration in seconds over which the next light command entered will fade from current to new level(s). Resets to 0.0 after each use.
dashboard mode	get/set	light dashboard view mode	The view mode of the Light Dashboard.
dashboard visibility	get/set	boolean	Is the Light Dashboard currently visible?
properties	get/set	record	All of the Light Dashboard's properties.

Commands

The **light dashboard** class responds to the following commands:

Command	Description
clear	Clear the levels in the light dashboard
newCueWithAll	Create a new Light cue containing levels for all parameters of all instruments.
newCueWithChanges	Create a new Light cue containing levels for all manually adjusted parameters in the light dashboard.
recordAllToLatest	Record all levels for all parameters of all instruments into the latest run Light cue.
recordAllToSelected	Record all levels for all parameters of all instruments into the selected Light cue(s).
redo	Redo the last undone action.
revert	Revert changes in the light dashboard.
setLight	add a light command to the light dashboard.
undo	Undo the last action.
updateLatestCue	Copy all manually adjusted levels into the latest run Light cue.
updateOriginatingCues	Copy all manually adjusted levels into their originating Light cue(s).
updateSelectedCues	Copy all manually adjusted levels into the selected Light cue(s).

Where Used

The **light dashboard** class is used in the following ways:

- direct parameter to the clear command.
- direct parameter to the updateLatestCue command.
- direct parameter to the updateOriginatingCues command.
- direct parameter to the updateSelectedCues command.
- direct parameter to the newCueWithAll command.
- direct parameter to the newCueWithChanges command.
- direct parameter to the recordAllToLatest command.
- direct parameter to the recordAllToSelected command.
- direct parameter to the revert command.
- **current light dashboard** property of the workspace class.

---

load cue

*(noun), pl. **load cues** *

Properties

In addition to the properties listed here, **load cue** inherits properties from **cue**.

Property	Access	Type	Description
load time	get/set	real number	Load target cue to this time.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **load cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **load cue** class inherits elements and properties from the **cue** class.

---

mic cue

*(noun), pl. **mic cues** *

Properties

In addition to the properties listed here, **mic cue** inherits properties from **cue**.

Property	Access	Type	Description
audio input channels	get	integer	The number of audio input channels for this cue.
audio input patch name	get/set	text	The name of this cue's audio input patch. `none` means "unpatched."
audio input patch number	get/set	integer	The 1-indexed number of this cue's audio input patch. `0` equals "unpatched."
audio input patch id	get/set	text	The unique ID of this cue's audio input patch. Empty string or `none` equals "unpatched."
audio output patch name	get/set	text	The name of this cue's audio output patch. `none` means "unpatched."
audio output patch number	get/set	integer	The 1-indexed number of this cue's audio output patch. `0` means "unpatched."

Property	Access	Type	Description
audio output patch id	get/set	text	The unique ID of this cue's audio output patch. Empty string or `none` equals "unpatched."
patch	get/set	integer	The 1-indexed number of this cue's audio output patch. *Deprecated in QLab 5.0 - use `audio input patch number` or `audio output patch number` instead.*

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **mic cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
getGang	Get the gang for a specified location in the cue's matrix.
getInputChannelName	Get the input name for a specified row in the cue's matrix.
getLevel	Get the level for a specified location in the cue's matrix.
getMute	Get the status of the mute button of a specified output.
getSolo	Get the status of the solo button of a specified output.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
setGang	Set the gang for a specified location in the cue's matrix.
setInputChannelName	Set the intput name for a specified row in the cue's matrix.
setLevel	Set the level for a specified location in the cue's matrix.
setMute	Set the status of the mute button of a specified output.
setSolo	Set the status of the solo button of a specified output.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **mic cue** class inherits elements and properties from the **cue** class.

---

midi cue

*(noun), pl. **midi cues** *

Properties

In addition to the properties listed here, **midi cue** inherits properties from **cue**.

Property	Access	Type	Description
byte combo	get/set	integer	(MIDI voice message type) Value when first and second bytes of the MIDI message are interpreted as parts of one number. Used for pitch bend messages.
byte one	get/set	integer	(MIDI voice message type) First byte of the MIDI message.
byte two	get/set	integer	(MIDI voice message type) Second byte of the MIDI message.
channel	get/set	integer	(MIDI voice message type) MIDI channel number.
command	get/set	midi command	(MIDI voice message type) MIDI command.
command format	get/set	integer	(MSC message type) MSC command format.
command number	get/set	integer	(MSC message type) MSC command.
control number	get/set	integer	(MSC message type) MSC control number.
control value	get/set	integer	(MSC message type) MSC control value.
deviceID	get/set	integer	(MSC message type) MSC device ID number.
end value	get/set	integer	(MIDI voice message type) The end value for a faded MIDI message.
fade	get/set	boolean	(MIDI voice message type) Does the MIDI message fade?
macro	get/set	integer	(MSC message type) MSC macro parameter.
message type	get/set	midi type	The type of MIDI message for this cue (msd, sysex, or voice.)
midi patch id	get/set	text	The unique ID of this cue's MIDI patch. Empty string or `none` means "unpatched."
midi patch name	get/set	text	The name of this cue's MIDI patch. `none` means "unpatched."
midi patch number	get/set	integer	The 1-indexed number of this cue's MIDI patch. `0` means "unpatched."
msc frames	get/set	integer	(MSC message type) MSC frames parameter.
msc hours	get/set	integer	(MSC message type) MSC hours parameter.
msc minutes	get/set	integer	(MSC message type) MSC minutes parameter.
msc seconds	get/set	integer	(MSC message type) MSC seconds parameter.
msc subframes	get/set	integer	(MSC message type) MSC subframes parameter.
patch	get/set	integer	The 1-indexed number of this cue's MIDI patch. *Deprecated in QLab 5.0 - use `midi patch number` instead.*
q_list	get/set	text	(MSC message type) Q List message parameter.
q_number	get/set	text	(MSC message type) Q Number message parameter.
q_path	get/set	text	(MSC message type) Q Path message parameter.
send time with set	get/set	boolean	(MSC message type) Send the timecode parameters with the SET command?
smpte format	get/set	smpte format	(MSC message type) SMPTE format of the timecode parameters.
start value	get/set	integer	(MIDI voice message type) The start value for a faded MIDI message.
sysex message	get/set	text	(SysEx message type) The raw SysEx message. Use only hexadecimal characters and whitespace. Omit the starting `F0` and the ending `F7`.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **midi cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **midi cue** class inherits elements and properties from the **cue** class.

---

midi file cue

*(noun), pl. **midi file cues** *

Properties

In addition to the properties listed here, **midi file cue** inherits properties from **cue**.

Property	Access	Type	Description
midi patch name	get/set	text	The name of this cue's MIDI patch. `none` means "unpatched."
midi patch number	get/set	integer	The 1-indexed number of this cue's MIDI patch. `0` means "unpatched."
midi patch id	get/set	text	The unique ID of this cue's MIDI patch. Empty string or `none` means "unpatched."
patch	get/set	integer	The 1-indexed number of this cue's MIDI patch. *Deprecated in QLab 5.0 - use `midi patch number` instead.*
rate	get/set	real number	Playback rate of the MIDI file.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **midi file cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **midi file cue** class inherits elements and properties from the **cue** class.

---

network cue

*(noun), pl. **network cues** *

Properties

In addition to the properties listed here, **network cue** inherits properties from **cue**.

Property	Access	Type	Description
custom message	get/set	text	The custom OSC message, for custom type messages. *Deprecated in QLab 5.0 - use* `parameter values` *instead.*)
fade entries	get/set	list of text	The list of {x,y} coordinates representing entries which define the shape (1D) or path (2D) of the current fade.
fade fps	get/set	integer	The rate in frames per second in which fade values are sent. Must be a positive integer between `1` and `120`.
fade from	get/set	real number	The starting value for a 1D fade.
fade number type	get/set	integer	Whether the fade sends integer (0) or float (1) values.
fade path height	get/set	real number	The maximum Y value for the grid displayed in the inspector for a 2D fade. Must not be a negative number.
fade path width	get/set	real number	The maximum X value for the grid displayed in the inspector for a 2D fade. Must not be a negative number.
fade to	get/set	real number	The ending value for a 1D fade.
fade type	get/set	integer	0 = no fade/resend 1 = 1D fade 2 = 2D fade Writeable only for `string` type parameters.
network patch name	get/set	text	The name of this cue's Network patch. `none` means "unpatched."
network patch number	get/set	integer	The 1-indexed number of this cue's Network patch. `0` means "unpatched."
network patch id	get/set	text	The unique ID of this cue's Network patch. Empty string or `none` means "unpatched."
parameter fades enabled	get/set	list of boolean	The list of boolean values that represent the fade-enabled states of all fade-able parameters in this Network cue.
parameter values	get/set	list of text, boolean, or number	The list of parameter values used to configure the current command.
patch	get/set	integer	The 1-indexed number of this cue's Network patch. *Deprecated in QLab 5.0 - use* `network patch number` *instead.*

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **network cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **network cue** class inherits elements and properties from the **cue** class.

---

override controller

(noun)

Properties

Property	Access	Type	Description
dmx output enabled	get/set	boolean	Allow DMX output (default is TRUE.)
midi input enabled	get/set	boolean	Allow MIDI Voice input (default is TRUE.)
midi output enabled	get/set	boolean	Allow MIDI Voice output (default is TRUE.)
msc input enabled	get/set	boolean	Allow MSC input (default is TRUE.)
msc output enabled	get/set	boolean	Allow MSC output (default is TRUE.)
network external input enabled	get/set	boolean	Allow external network input (default is TRUE.)
network external output enabled	get/set	boolean	Allow external network output (default is TRUE.)
network local input enabled	get/set	boolean	Allow local network input (default is TRUE.)

Property	Access	Type	Description
network local output enabled	get/set	boolean	Allow local network output (default is TRUE.)
osc input enabled	get/set	boolean	Allow OSC input (default is TRUE.)
osc output enabled	get/set	boolean	Allow OSC output (default is TRUE.)
overrides visibility	get/set	boolean	Is the Overrides Controls window visible?
sysex input enabled	get/set	boolean	Allow SysEx (other than MSC and MTC) input (default is TRUE.)
sysex output enabled	get/set	boolean	Allow SysEx (other than MSC and MTC) output (default is TRUE.)
timecode input enabled	get/set	boolean	Allow timecode input (default is TRUE.)
timecode output enabled	get/set	boolean	Allow timecode output (default is TRUE.)

Where Used

The *override controller* class is used in the following ways:

- *overrides* property of the application class.

Examples

```
-- override MIDI output (i.e. "don't output any MIDI")
tell application id "com.figure53.QLab.5"
  tell overrides to set midi output enabled to false
end tell


-- open the override controls window
tell application "QLab" to tell overrides to set overrides visibility to true
```

---

reset cue

*(noun), pl. **reset cues** *

Properties

In addition to the properties listed here, **reset cue** inherits properties from **cue**.

Property	Access	Type	Description
patch target id	get/set	text	The patch this cue targets, if any.
target mode	get/set	target modes	The target mode of this cue.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **devamp cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.

Command	Description
compile	Verify and prepare the script for use.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **devamp cue** class inherits elements and properties from the **cue** class.

---

script cue

*(noun), pl. **script cues** *

Properties

In addition to the properties listed here, **script cue** inherits properties from **cue**.

Property	Access	Type	Description
script source	get/set	text	AppleScript source for the cue. The script will be recompiled when set.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **script cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
compile	Verify and prepare the script for use.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.

Command	Description
stop	Stop one or more cues or workspaces.

Superclass

The **script cue** class inherits elements and properties from the **cue** class.

Where Used

The **script cue** class is used in the following ways:

- direct parameter to the compile command.

---

target cue

*(noun), pl. **target cues** *

Properties

In addition to the properties listed here, **target cue** inherits properties from **cue**.

Property	Access	Type	Description
assigned number	get/set	text	The cue number of the cue to assign. The cue with this number will be assigned as the new target of the cue which this cue targets.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **target cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **target cue** class inherits elements and properties from the **cue** class.

---

text cue

*(noun), pl. **text cues** *

Properties

In addition to the properties listed here, **text cue** inherits properties from **cue**.

Property	Access	Type	Description
anchor x	get/set	real number	Anchor along the x axis.
anchor y	get/set	real number	Anchor along the y axis.
blend mode	get/set	text	Display name of the video blend mode.
fill stage	get/set	boolean	Is the cue displaying in fill stage mode?
fill style	get/set	fill styles	How does the cue fill the stage?
full screen	get/set	boolean	Is the cue displaying in full-stage mode? *Deprecated in QLab 5.0 - use 'fill stage' instead.*
full surface	get/set	boolean	Is the cue displaying in full-stage mode? *Deprecated in QLab 5.0 - use 'fill stage' instead.*
fixed width	get/set	number	Fixed width of the text cue. Setting this to `0` specifies "auto" width.
layer	get/set	integer	The display layer of this cue. `0` is the bottom layer; `1000` is the top layer.
live text	get/set	text	Live text of this cue. Setting this does not mark the workspace as edited.
live text alignment	get/set	text	Text alignment of the live text of this cue. Possible values are "left", "center", "right", and "justify". Setting this does not mark the workspace as edited.
live text format	get/set	list of text format record	The list of text formats in the live text of this cue. Setting this does not mark the workspace as edited.
live text output size	get	list of number	A 2-item list representing the width and height of the live text of this cue.
opacity	get/set	real number	The opacity of this cue. `0` = 0%; `0.5` = 50%; `1` = 100%
preserve aspect ratio	get/set	boolean	Does this cue preserve aspect ratio?
scale x	get/set	real number	The X-axis scale of this cue.
scale y	get/set	real number	The Y-axis scale of this cue.
smooth	get/set	Should the cue be scaled using smoothing interpolation?	
stage name	get/set	text	Video output stage name. Empty string or `none` means "unpatched."
stage number	get/set	integer	Video output stage number. `0` means "unpatched."
stage id	get/set	text	Video output stage unique ID. Empty string or `none` means "unpatched."
text	get/set	text	Text of this cue.
text alignment	get/set	text	Text alignment of this cue. Possible values are "left", "center", "right", and "justify".
text format	get/set	list of text format record	The list of text formats in the text of this cue.
text output size	get	list of number	A 2-item list representing the width and height of the text of this cue.
translation x	get/set	real number	The X-axis translation (position) of this cue.
translation y	get/set	real number	The Y-axis translation (position) of this cue.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **text cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **text cue** class inherits elements and properties from the **cue** class.

---

timecode cue

*(noun), pl. **timecode cues** *

Properties

In addition to the properties listed here, **timecode cue** inherits properties from **cue**.

Property	Access	Type	Description
audio output patch name	get/set	text	(For cues in LTC mode.) The name of this cue's audio output patch. `none` means "unpatched."
audio output patch number	get/set	integer	(For cues in LTC mode.) The 1-indexed number of this cue's audio output patch. `0` means "unpatched."
audio output patch id	get/set	text	(For cues in LTC mode.) The unique ID of this cue's audio output patch. Empty string or `none` means "unpatched."
midi patch name	get/set	text	(For cues in MTC mode.) The name of this cue's MIDI patch. `none` means "unpatched."
midi patch number	get/set	integer	(For cues in MTC mode.) The 1-indexed number of this cue's MIDI patch. `0` means "unpatched."
midi patch id	get/set	text	(For cues in MTC mode.) The unique ID of this cue's MIDI patch. Empty string or `none` means "unpatched."
patch	get/set	integer	(For cues in LTC mode.) The 1-indexed number of this cue's audio output patch. *Deprecated in QLab 5.0 - use `audio output patch number` instead.*
patch	get/set	integer	(For cues in MTC mode.) The 1-indexed number of this cue's MIDI patch. *Deprecated in QLab 5.0 - use `midi patch number` instead.*
smpte format	get/set	smpte format	SMPTE format of the outgoing timecode.
start time offset	get/set	real number	Time in seconds where the timecode clock begins counting.
timecode end time	get/set	text	The timecode clock end time.
timecode start time	get/set	text	The timecode clock start time.
type	get/set	mtc or ltc	The type of timecode used by this cue.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **timecode cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **timecode cue** class inherits elements and properties from the **cue** class.

---

video cue

*(noun), pl. **video cues** *

Properties

In addition to the properties listed here, **video cue** inherits properties from **cue**.

Property	Access	Type	Description
anchor x	get/set	real number	Anchor along the x axis.
anchor y	get/set	real number	Anchor along the y axis.
audio input channels	get	integer	The number of audio input channels for this cue (i.e. the number of distinct channels in the target audio file.)
audio output patch name	get/set	text	The name of this cue's audio output patch. `none` means "unpatched."
audio output patch number	get/set	integer	The 1-indexed number of this cue's audio output patch. `0` means "unpatched."
audio output patch id	get/set	text	The unique ID of this cue's audio output patch. Empty string or `none` means "unpatched."
blend mode	get/set	text	Display name of the video blend mode.
clock type	get/set	audio or video	The clock type of the cue.
end time	get/set	real number	Time in the target file where playback ends.
fill stage	get/set	boolean	Is the cue displaying in fill stage mode?
fill style	get/set	fill styles	How does the cue fill the stage?
full screen	get/set	boolean	Is the cue displaying in full-stage mode? *Deprecated in QLab 5.0 - use 'fill stage' instead.*

Property	Access	Type	Description
full surface	get/set	boolean	Is the cue displaying in full-stage mode? *Deprecated in QLab 5.0 - use 'fill stage' instead.*
hold at end	get/set	boolean	Should the final frame of the video be left visible when playback reaches the end of the file?
infinite loop	get/set	boolean	Does this cue loop infinitely?
integrated fade	get/set	enabled or disabled	State of the integrated fade checkbox.
last slice infinite loop	get/set	boolean	Does the last slice of this cue loop infinitely?
last slice play count	get/set	integer	Number of times the last slice of this cue plays. Always >= `1`.
layer	get/set	integer	The display layer of this cue. `0` is the bottom layer; `1000` is the top layer.
lock fade to cue	get/set	enabled or disabled	State of the lock fade to start/end checkbox.
opacity	get/set	real number	The opacity of this cue. `0` = 0%; `0.5` = 50%; `1` = 100%
patch	get/set	integer	The 1-indexed number of this cue's audio output patch. *Deprecated in QLab 5.0 - use `audio output patch number` instead.*
play count	get/set	boolean	Number of times this cue plays. Always >= `1`.
preserve aspect ratio	get/set	boolean	Does this cue preserve aspect ratio?
preserve pitch	get/set	enabled or disabled	State of the preserve pitch checkbox.
rate	get/set	real number	Playback rate of this cue.
scale x	get/set	real number	The X-axis scale of this cue.
scale y	get/set	real number	The Y-axis scale of this cue.
slice markers	get/set	list of slice marker record	List of slice markers in this cue.
smooth	get/set	Should the cue be scaled using smoothing interpolation?	
stage name	get/set	text	Video output stage name. Empty string or `none` means "unpatched."
stage number	get/set	integer	Video output stage number. `0` means "unpatched."
stage id	get/set	text	Video output stage unique ID. Empty string or `none` means "unpatched."
start time	get/set	real number	Time in the target file where playback begins.
translation x	get/set	real number	The X-axis translation (position) of this cue.
translation y	get/set	real number	The Y-axis translation (position) of this cue.

Elements

Element	Access	Key Forms	Description
cue	get/make/delete	by name, by index, by uniqueID	

Commands

The **video cue** class responds to the following commands:

Command	Description
audition preview	Audition preview one or more cues. Previewing starts only the action of the cue, skipping any prewait and not continuing to other cues.
capture timecode	Set the cue's timecode trigger to the current incoming timecode received by its parent cue list.
getGang	Get the gang for a specified location in the cue's matrix.
getInputChannelName	Get the input name for a specified row in the cue's matrix.
getLevel	Get the level for a specified location in the cue's matrix.
getMute	Get the status of the mute button of a specified output.

Command	Description
getSolo	Get the status of the solo button of a specified output.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
preview	Preview one or more cues.
reset	Reset one or more cues or workspaces.
setGang	Set the gang for a specified location in the cue's matrix.
setInputChannelName	Set the intput name for a specified row in the cue's matrix.
setLevel	Set the level for a specified location in the cue's matrix.
setMute	Set the status of the mute button of a specified output.
setSolo	Set the status of the solo button of a specified output.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.

Superclass

The **video cue** class inherits elements and properties from the **cue** class.

---

workspace

*(noun), pl. **workspaces** *

Properties

Property	Access	Type	Description
active cues	get	list of cue	The list of active cues (running or paused) in this workspace.
always audition	get/set	boolean	Is the workspace currently set to always audition?
current cue list	get/set	cue list	The cue list that's currently visible in the main window of the workspace.
current light dashboard	get	light dashboard	The current Light Dashboard for the workspace.
edit mode	get/set	boolean	Is the workspace currently in edit mode?
inspector visibility	get/set	boolean	Is the inspector visible?
live fade preview	get/set	boolean	Is live fade preview currently enabled for this workspace?
selected	get/set	list of cue	The currently selected cue(s) in the current cue list.
show mode	get/set	boolean	Is the workspace currently in show mode?
unique id	get	text	The unique ID of the workspace.

Elements

Element	Access	Key Forms	Description
cue	get	by name, by unique id	The complete list of cues in this workspace.
cue list	get/make/delete	by name, by index, by unique id	The list of cue lists in the workspace.

Commands

The **workspace** class responds to the following commands:

Command	Description
audition go	Make a workspace Audition GO.
go	Make a workspace GO.
hardStop	hardStop one or more cues or workspaces.
load	Load one or more cues or workspaces to a given time.
make	Create a new cue.
movePlayheadDown	Move the playhead in the active cue list to the next cue.
movePlayheadDownASequence	Move the playhead in the active cue list to top of the next cue sequence.
movePlayheadUp	Move the playhead in the active cue list to the previous cue.
movePlayheadUpASequnce	Move the playhead in the active cue list to top of the previous cue sequence.
moveSelectionDown	Select the next cue.
moveSelectionUp	Select the previous cue.
panic	Panic one or more cues or workspaces.
pause	Pause one or more cues or workspaces.
redo	Redo the last undone action.
reset	Reset one or more cues or workspaces.
start	Start one or more cues or workspaces.
stop	Stop one or more cues or workspaces.
undo	Undo the last action.

Where Used

The workspace class is used in the following ways:

- element of application class.
- direct parameter to the make command.

Enumerations

---

absolute relative

Constants

Constant	Description
absolute	
relative	

Where Used

The **absolute relative** enumeration is used in the following ways:

- **audio fade mode** property of the fade cue class.
- **fade mode** property of the fade cue class.
- **video fade mode** property of the fade cue class.

---

clock types

Constants

Constant	Description
audio	
video	

Where Used

The **clock types** enumeration is used in the following ways:

- **clock type** property of the video cue class.

---

continue modes

Constants

Constant	Description
auto_continue	Automatically continue to the next cue after completing the post-wait.
auto_follow	Automatically continue to the next cue after completing the action of the cue.
do_not_continue	Do not automatically continue to the next cue.

Where Used

The **continue modes** enumeration is used in the following ways:

- **continue mode** property of the cue class.

---

enabled disabled

Constants

Constant	Description
disabled	
enabled	

Where Used

The **enabled disabled** enumeration is used in the following ways:

- **duck others** property of the cue class.
- **fade** property of the midi cue class.
- **hotkey trigger** property of the cue class.
- **integrated fade** property of the audio cue class.
- **integrated fade** property of the video cue class.
- **lock fade to cue** property of the audio cue class.
- **lock fade to cue** property of the video cue class.
- **midi trigger** property of the cue class.
- **preserve pitch** property of the audio cue class.
- **preserve pitch** property of the video cue class.
- **second trigger on release** property of the cue class.
- **sync to timecode** property of the cue list class.
- **timecode trigger** property of the cue class.
- **wall clock trigger** property of the cue class.

---

fill styles

Constants

Constant	Description
fill	Fill the entire stage with the cue, preserving aspect ratio. Some portion of the cue may be cut off.
fit	Fit the cue inside the stage, preserving aspect ratio. Some empty space may be left to the sides or top and bottom of the cue.
stretch	Fill the stage by stretching the cue height and width to match the stage.

Where Used

The **fill styles** enumeration is used in the following ways:

- **fill style** property of the video cue class.
- **fill style** property of the text cue class.
- **fill style** property of the camera cue class.

---

group modes

Constants

Constant	Description
cue_list	The group is a cue list.
playlist	Playlist - one cue at a time.
start_first	Start first child and go to next cue.
start_first_and_enter	Start first child and enter into group.
start_random	Start a random child and then go to the next cue.
timeline	Timeline - start all children simultaneously.

Where Used

The **group modes** enumeration is used in the following ways:

- **mode** property of the group cue class.

---

light dashboard view mode

Constants

Constant	Description
sliders	
tiles	

Where Used

The **light dashboard view mode** enumeration is used in the following ways:

- **dashboard mode** property of the light dashboard class.

---

midi command

Constants

Constant	Description
channel_pressure	
control_change	
key_pressure	a.k.a. aftertouch
note_off	
note_on	
pitch_bend	a.k.a. pitch wheel
program_change	

Where Used

The **midi command** enumeration is used in the following ways:

- **command** property of the midi cue class.
- **midi command** property of the cue class.

---

midi type

Constants

Constant	Description
msc	MIDI Show Control message.
sysex	MIDI System Exclusive message.
voice	MIDI Voice message.

Where Used

The **midi type** enumeration is used in the following ways:

- **message type** property of the midi cue class.

---

mtc ltc

Constants

Constant	Description
ltc	Linear/Longitudinal Timecode.
mtc	MIDI Timecode.

Where Used

The **mtc ltc** enumeration is used in the following ways:

- **sync mode** property of the cue list class.

---

smpte format

Constants

Constant	Description
fps_24	24 frames per second.
fps_25	25 frames per second.
fps_30_drop	30 frames per second, drop frame.
fps_30_non_drop	30 frames per second, non-drop frame.

Where Used

The **smpte format** enumeration is used in the following ways:

- **smpte format** property of the midi cue class.
- **smpte format** property of the timecode cue class.

---

target modes

Constants

Constant	Description
target mode cue	
target mode patch	

Where Used

The **target modes** enumeration is used in the following ways:

- **target mode** property of the fade cue class.
- **target mode** property of the reset cue class.

---

timecode smpte format

Constants

Constant	Description
fps_23_976	23.976 frames per second.
fps_24	24 frames per second.
fps_24_975	24.975 frames per second.
fps_25	25 frames per second.
fps_29_97	29.97 frames per second, drop frame.
fps_29_97_non_drop	29.97 frames per second, non-drop frame.
fps_30_drop	30 frames per second, drop frame.
fps_30_non_drop	30 frames per second, non-drop frame.

Where Used

The **timecode smpte format** enumeration is used in the following ways:

- **smpte format** property of the cue list class.

---

timecode start

Constants

Constant	Description
lookback time	start cues whose timecode triggers precede the incoming timecode by the lookback window.
recent hour	start cues whose timecode triggers fall within the most recent hour of incoming timecode.
recent minute	start cues whose timecode triggers fall within the most recent minute of incoming timecode.
skip	do not start cues whose timecode triggers precede the incoming timecode.
start all	start all cues whose timecode triggers precede the incoming timecode.

Where Used

The **timecode start** enumeration is used in the following ways:

- **timecode start behavior** property of the cue list class.

---

timecode stop

Constants

Constant	Description
hard pause	hard pause timecode triggered cues when incoming timecode stops.
hard stop	hard stop timecode triggered cues when incoming timecode stops.
none	do nothing when incoming timecode stops.

Where Used

The **timecode stop** enumeration is used in the following ways:

- **timecode stop behavior** property of the cue list class.

Records

---

range record

A 2-item record representing the offset and length of a substring.

Properties

Property	Access	Type	Description
rangeLength	get/set	integer or text	The length of the substring range.
rangeOffset	get/set	integer or text	The 1-indexed location of the starting character of a substring range.

Where Used

The **range record** is used in the following ways:

- **range** property of the text format record.

---

rgba color record

A 4-item record representing red, green, blue, and alpha percentage values of a color.

Properties

Property	Access	Type	Description
red	get/set	real number	
green	get/set	real number	
blue	get/set	real number	
alpha	get/set	real number	

Where Used

The rgba color record record is used in the following ways:

- **backgroundRgbaColor** property of the text format record.
- **rgbaColor** property of the text format record.
- **shadowRgbaColor** property of the text format record.
- **strikethroughRgbaColor** property of the text format record.
- **underlineRgbaColor** property of the text format record.

Examples

This script will set the color of all the text in cue `2` to a nice purple-y color:

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set aNiceColor to {rgbaColor:{red:0.5, green:0.2, blue:0.6, alpha:1}}
```

```
    set text format of cue "2" to aNiceColor
  end tell
```

This script will set the color of the underline of all the text in cue `2` to primary blue:

```
tell application id "com.figure53.QLab.5" to tell front workspace
  set aNiceColor to {underlineRgbaColor:{red:0, green:0, blue:1, alpha:1}}
  set text format of cue "2" to aNiceColor
end tell
```

---

row column record

A 2-item record representing a position defined by a numeric row and column value.

Properties

Property	Access	Type	Description
column	get/set	integer	
row	get/set	integer	

Where Used

The **row column record** is used in the following ways:

- **cart position** property of the cue class.

---

size record

A 2-item record representing width and height values.

Properties

Property	Access	Type	Description
height	get/set	real number	
width	get/set	real number	

Where Used

The slice marker record record is used in the following ways:

- **shadowOffset** property of the text format record.

---

slice marker record

A 2-item record representing the play count and end time of a slice.

Properties

Property	Access	Type	Description
playCount	get/set	integer	The number of times a slice will play. Play count -1 = infinite loop.

Property	Access	Type	Description
time	get/set	real number	The end time of a slice.

Where Used

The slice marker record record is used in the following ways:

- **slice markers** property of the audio cue class.
- **slice markers** property of the video cue class.

---

text format record

A record representing the formatting aspects of a text string.

Properties

Property	Access	Type	Description
backgroundRgbaColor	get/set	rgba color record	An RGBA color record representing the percentage values for the red, green, blue, and alpha components of the background color of this format.
fontFamily	get/set	text	The font family for this format. (e.g. “Helvetica”, “Courier New”)
fontName	get/set	text	The font name for this format. (e.g. “CourierNewPS-BoldItalicMT”)
fontSize	get/set	real	The font size for this format.
fontStyle	get/set	text	The font style (face) for this format. (e.g. “Regular”, “Light Oblique”)
lineSpacing	get/set	real number	The line spacing for this format.
range	get/set	range record	A range record representing the index and length for the substring that has this format.
rgbaColor	get/set	rgba color record	An RGBA color record representing the percentage values for the red, green, blue, and alpha components of the text color of this format.
shadowBlurRadius	get/set	real number	The shadow blur radius for this format.
shadowOffset	get/set	size record	A size record representing the width and height components of the shadow offset of this format.
shadowRgbaColor	get/set	rgba color record	An RGBA color record representing the percentage values for the red, green, blue, and alpha components of the shadow color of this format.
strikethroughRgbaColor	get/set	rgba color record	An RGBA color record representing the percentage values for the red, green, blue, and alpha components of the strikethrough color of this format.
strikethroughStyle	get/set	text	The strikethrough style of this format. Possible values are “none”, “single”, and “double”.
underlineRgbaColor	get/set	rgba color record	An RGBA color record representing the percentage values for the red, green, blue, and alpha components of the underline color of this format.
underlineStyle	get/set	text	The underline style of this format. Possible values are “none”, “single”, and “double”.
wordIndex	get/set	integer	An optional 1-indexed word number to which this format should be applied. When used, the “range” property will be ignored. (setting only)

Where Used

The text format record record is used in the following ways:

- **live text format** property of the text cue class.
- **text format** property of the text cue class.

Parameter Reference

QLab contains a number of OSC- and AppleScript-controllable parameters which must be referred to in a specific and (in some cases) difficult-to-remember way.

This section provides a straightforward reference for the scriptable parameters which come from long lists and their scripting name or index if applicable.

MIDI Show Control Commands

MSC commands are scriptable using their index number.

Command	Index
GO	1
STOP	2
RESUME	3
TIMED_GO	4
LOAD	5
SET	6
FIRE	7
ALL_OFF	8
RESTORE	9
RESET	10
GO_OFF	11
GO/JAM_CLOCK	16
STANDBY_+	17
STANDBY_-	18
SEQUENCE_+	19
SEQUENCE_-	20
START_CLOCK	21
STOP_CLOCK	22
ZERO_CLOCK	23
SET_CLOCK	24
MTC_CHASE_ON	25
MTC_CHASE_OFF	26
OPEN_CUE_LIST	27
CLOSE_CUE_LIST	28
OPEN_CUE_PATH	29
CLOSE_CUE_PATH	30

MIDI Show Control Command Format Types

MSC command format types are scriptable using their index number.

Command Format Type	Index
All Types	127
Lighting (General)	1
Moving Lights	2
Color Changers	3
Strobes	4
Lasers	5
Chasers	6
Sound (General)	16
Music	17
CD Players	18
EPROM Playback	19
Audio Tape Machines	20
Intercoms	21
Amplifiers	22
Audio Effects Devices	23
Equalizers	24
Machinery (General)	32
Rigging	33
Flys	34
Lifts	35
Turntables	36
Trusses	37
Robots	38
Animation	39
Floats	40
Breakaways	41
Barges	42
Video (General)	48
Video Tape Machines	49
Video Cassette Machines	50
Video Disc Players	51
Video Switchers	52
Video Effects	53
Video Character Generators	54
Video Still Stores	55

Command Format Type	Index
Video Monitors	56
Projection (General)	64
Film Projectors	65
Slide Projectors	66
Video Projectors	67
Dissolvers	68
Shutter Controls	69
Process Control (General)	80
Hydraulic Oil	81
H2O	82
CO2	83
Compressed Air	84
Natural Gas	85
Fog	86
Smoke	87
Cracked Haze	88
Pyrotechnics (General)	96
Fireworks	97
Explosions	98
Flame	99
Smoke Pots	100

Video Blend Modes

Blend modes are scriptable using their full names as strings.

Blend Mode Name
Normal
Darken
Multiply
Color Burn
Linear Burn
Lighten
Screen
Color Dodge
Linear Dodge
Overlay
Soft Light
Hard Light

Blend Mode Name
Pin Light
Difference
Exclusion
Subtract
Divide
Hue
Saturation
Color
Luminosity
Addition Compositing
Maximum Compositing
Source Atop Compositing

Video Effects

Video effects and their parameters are scriptable using their scripting name. Some parameters can accept values beyond those listed in the “Allowed Values” column, so feel free to try other values.

Effect Name	Parameter Name	Scripting Name	Allowed Values
Color Controls		**ColorControls**	
	Brightness	inputBrightness	-1.0 - 1.0
	Contrast	inputContrast	0.25 - 4.0
	Hue Angle	inputAngle	-180.0 - 180.0
	Saturation	inputSaturation	0.0 - 2.0
Exposure		**Exposure**	
	Exposure Value	inputEV	-10.0 - 10.0
Gamma		**Gamma**	
	Power	inputPower	-0.1 - 3.0
Sepia / Monochrome		**SepiaMonochrome**	
	Choose Effect	Choose_Effect	0 (Sepia) 1 (Monochrome)
	Color	inputColor	Four floats, each 0.0 - 1.0
	Intensity	inputIntensity	0.0 - 1.0
Min / Max / Invert		**MinMaxInvert**	
	Choose Effect	Choose_Effect	0 (Minimum Component) 1 (Maximum Component) 2 (Color Invert)
White Point		**WhitePoint**	
	Color	inputColor	Four floats, each 0.0 - 1.0
Highlights and Shadows		**HighlightShadow**	
	Highlight	inputHighlightAmount	0.0 - 1.0
	Shadow	inputShadowAmount	-1.0 - 1.0

Effect Name	Parameter Name	Scripting Name	Allowed Values
	Radius	inputRadius	0.0 - 100.0
Temperature and Tint		**TemperatureTint**	
	Source Temperature	Source_Temperature	2000.0 - 10000.0
	Source Tint	Source_Tint	-500.0 - 500.0
	Target Temperature	Target_Temperature	2000.0 - 10000.0
	Target Tint	Target_Tint	-500.0 - 500.0
Vibrance		**Vibrance**	
	Amount	inputAmount	-1.0 - 1.0
Tone Curve		**ToneCurve**	
	Point 1 X	Point_0_X	0.0 - 1.0
	Point 1 Y	Point_0_Y	0.0 - 1.0
	Point 2 X	Point_1_X	0.0 - 1.0
	Point 2 Y	Point_1_Y	0.0 - 1.0
	Point 3 X	Point_2_X	0.0 - 1.0
	Point 3 Y	Point_2_Y	0.0 - 1.0
	Point 4 X	Point_3_X	0.0 - 1.0
	Point 4 Y	Point_3_Y	0.0 - 1.0
	Point 5 X	Point_4_X	0.0 - 1.0
	Point 5 Y	Point_4_Y	0.0 - 1.0
Color Clamp		**ColorClamp**	
	Minimum Red	Min_R	0.0 - 1.0
	Maximum Red	Max_R	0.0 - 1.0
	Minimum Green	Min_G	0.0 - 1.0
	Maximum Green	Max_G	0.0 - 1.0
	Minimum Blue	Min_B	0.0 - 1.0
	Maximum Blue	Max_B	0.0 - 1.0
	Minimum Alpha	Min_A	0.0 - 1.0
	Maximum Alpha	Max_A	0.0 - 1.0
Color Matrix		**ColorMatrix**	
	Red to Red	R_R	-1.0 - 1.0
	Green to Red	G_R	-1.0 - 1.0
	Blue to Red	B_R	-1.0 - 1.0
	Alpha to Red	A_R	-1.0 - 1.0
	Red Bias	R_Bias	-1.0 - 1.0
	Red to Green	R_G	-1.0 - 1.0
	Green to Green	G_G	-1.0 - 1.0
	Blue to Green	B_G	-1.0 - 1.0
	Alpha to Green	A_G	-1.0 - 1.0
	Green Bias	G_Bias	-1.0 - 1.0
	Red to Blue	R_B	-1.0 - 1.0

Effect Name	Parameter Name	Scripting Name	Allowed Values
	Green to Blue	G_B	-1.0 - 1.0
	Blue to Blue	B_B	-1.0 - 1.0
	Alpha to Blue	A_B	-1.0 - 1.0
	Blue Bias	B_Bias	-1.0 - 1.0
	Red to Alpha	R_A	-1.0 - 1.0
	Green to Alpha	G_A	-1.0 - 1.0
	Blue to Alpha	B_A	-1.0 - 1.0
	Alpha to Alpha	A_A	-1.0 - 1.0
	Alpha Bias	A_Bias	-1.0 - 1.0
Color Threshold		**ColorThreshold**	
	Threshold	inputThreshold	0.0 - 1.0
Spot Color		**SpotColor**	
	Center Color 1	inputCenterColor1	Four floats, each 0.0 - 1.0
	Replacement Color 1	inputReplacementColor1	Four floats, each 0.0 - 1.0
	Closeness 1	inputCloseness1	0.0 - 1.0
	Contrast 1	inputContrast1	0.0 - 10.0
	Center Color 2	inputCenterColor2	Four floats, each 0.0 - 1.0
	Replacement Color 2	inputReplacementColor2	Four floats, each 0.0 - 1.0
	Closeness 2	inputCloseness2	0.0 - 1.0
	Contrast 2	inputContrast2	0.0 - 10.0
	Center Color 3	inputCenterColor3	Four floats, each 0.0 - 1.0
	Replacement Color 3	inputReplacementColor3	Four floats, each 0.0 - 1.0
	Closeness 3	inputCloseness3	0.0 - 1.0
	Contrast 3	inputContrast3	0.0 - 10.0
Linear Key		**LinearKey**	
	Key Color	inputKeyColor	0 (Green) 1 (Blue)
	Cutoff	inputCutoff	0.0 - 4.0
	Contrast	inputContrast	0.0 - 10.0
Box / Disc / Gaussian Blur		**BoxDiscGaussianBlur**	
	Choose Blur	Choose_Blur	0 (Box Blur) 1 (Disc Blur) 2 (Gaussian Blur)
	Radius	inputRadius	0.0 - 400.0
Bokeh Blur		**BokehBlur**	
	Radius	inputRadius	0.0 - 500.0
	Ring Amount	inputRingAmount	0.0 - 1.0
	Ring Size	inputRingSize	0.0 - 500.0
	Softness	inputSoftness	0.0 - 10.0
Motion Blur		**MotionBlur**	
	Radius	inputRadius	0.0 - 400.0

Effect Name	Parameter Name	Scripting Name	Allowed Values
	Angle	inputAngle	-360.0 - 360.0
Sharpen Luminance		**SharpenLuminance**	
	Sharpness	inputSharpness	0.0 - 2.0
Unsharp Mask		**UnsharpMask**	
	Radius	inputRadius	0.0 - 100.0
	Intensity	inputIntensity	0.0 - 1.0
Zoom Blur		**ZoomBlur**	
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Amount	inputAmount	0.0 - 200.0
Morphology Gradient		**MorphologyGradient**	
	Radius	inputRadius	0.0 - 100.0
Morphology Min / Max		**MorphologyMinMax**	
	Choose Effect	Choose_Effect	0 (Morphology Minimum) 1 (Morphology Maximum)
	Width	inputWidth	1.0 - 101.0
	Height	inputHeight	1.0 - 101.0
Depth of Field		**DepthOfField**	
	Point 1 X	Point_0_X	0.0 - 1.0
	Point 1 Y	Point_0_Y	0.0 - 1.0
	Point 2 X	Point_1_X	0.0 - 1.0
	Point 2 Y	Point_1_Y	0.0 - 1.0
	Radius	inputRadius	0.0 - 100.0
	Saturation	inputSaturation	0.0 - 5.0
	Unsharp Mask Intensity	inputUnsharpMaskIntensity	0.0 - 10.0
	Unsharp Mask Radius	inputUnsharpMaskRadius	0.0 - 20.0
Pixellation		**Pixellation**	
	Choose Effect	Choose_Effect	0 (Pixellate) 1 (Hexagonal Pixellate)
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Scale	inputScale	1.0 - 100.0
Screen		**Screen**	
	Choose Effect	Choose_Effect	0 (Dot Screen) 1 (Line Screen) 2 (Hatched Screen) 3 (Circular Screen)
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Angle	inputAngle	-360.0 - 360.0
	Width	inputWidth	2.0 - 50.0
	Sharpness	inputSharpness	0.0 - 1.0

Effect Name	Parameter Name	Scripting Name	Allowed Values
Bloom and Gloom		**BloomGloom**	
	Choose Effect	Choose_Effect	0 (Bloom) 1 (Gloom)
	Radius	inputRadius	0.0 - 100.0
	Intensity	inputIntensity	0.0 - 1.0
CMYK Halftone		**CMYKHalftone**	
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Angle	inputAngle	-360.0 - 360.0
	Width	inputWidth	0.0 - 100.0
	Sharpness	inputSharpness	0.0 - 1.0
	Gray Component Replacement	inputGCR	0.0 - 1.0
	Under Color Removal	inputUCR	0.0 - 1.0
Photo Effects		**PhotoEffects**	
	Choose Effect	Choose_Effect	0 (Chrome) 1 (Fade) 2 (Instant) 3 (Mono) 4 (Noir) 5 (Process) 6 (Tonal) 7 (Transfer)
Color Posterize		**ColorPosterize**	
	Levels	inputLevels	2.0 - 30.0
Crystallize and Pointillize		**CrystallizePointillize**	
	Choose Effect	Choose_Effect	0 (Crystallize) 1 (Pointillize)
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Radius	inputRadius	0.0 - 100.0
Edge Work		**EdgeWork**	
	Choose Effect	Choose_Effect	0 (Edge Work) 1 (Edges)
	Intensity	inputIntensity	0.0 - 10.0
	Radius	inputRadius	0.0 - 20.0
Line Overlay		**LineOverlay**	
	Contrast	inputContrast	0.25 - 10.0
	Edge Intensity	inputEdgeIntensity	0.0 - 1.0
	Threshold	inputThreshold	0.0 - 1.0
	NR Noise Level	inputNRNoiseLevel	0.0 - 4.0
	NR Sharpness	inputNRSharpness	0.0 - 1.0
Kaleidoscope		**Kaleidoscope**	
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0

Effect Name	Parameter Name	Scripting Name	Allowed Values
	Angle	inputAngle	-360.0 - 360.0
	Count	inputCount	1.0 - 64.0
Triangle Kaleidoscope		**TriangleKaleidoscope**	
	Center X	Point_X	0.0 - 1.0
	Center Y	Point_Y	0.0 - 1.0
	Rotation	inputRotation	-360.0 - 360.0
	Size	inputSize	1.0 - 4000.0
	Decay	inputDecay	0.0 - 1.0
Droste		**Droste**	
	Inset Point 1 X	Point_0_X	0.0 - 1.0
	Inset Point 1 Y	Point_0_Y	0.0 - 1.0
	Inset Point 2 X	Point_1_X	0.0 - 1.0
	Inset Point 2 Y	Point_1_Y	0.0 - 1.0
	Periodicity	inputPeriodicity	1.0 - 400.0
	Strands	inputStrands	0.0 - 20.0
	Zoom	inputZoom	0.0 - 5.0
	Rotation	inputRotation	-360.0 - 360.0
Light Tunnel		**LightTunnel**	
	Center X	Center_X	0.0 - 1.0
	Center Y	Center_Y	0.0 - 1.0
	Radius	inputRadius	1.0 - 4000.0
	Rotation	inputRotation	-360.0 - 360.0
Median and Comic Effect		**MedianComicEffect**	
	Choose Effect	Choose_Effect	0 (Median) 1 (Comic Effect)
False Color		**FalseColor**	
	No scriptable parameters.		
Thermal / X-Ray		**ThermalXRay**	
	Choose Effect	Choose_Effect	0 (Thermal) 1 (X-Ray)
Gabor Gradients		**GaborGradients**	
	No parameters.		
Spotlight		**Spotlight**	
	Color	inputColor	Four floats, each 0.0 - 1.0
	Brightness	inputBrightness	0.0 - 2.0
	Concentration	inputConcentration	0.0 - 5.0
	Light Position X	Light_Pos_X	-1.0 - 2.0
	Light Position Y	Light_Pos_Y	-1.0 - 2.0
	Light Position Z	Light_Pos_Z	-1.0 - 2.0
	Points At X	Light_Points_X	-1.0 - 2.0
	Points At Y	Light_Points_Y	-1.0 - 2.0

Effect Name	Parameter Name	Scripting Name	Allowed Values
	Points At X	Light_Points_Z	-1.0 - 2.0
Dither		**Dither**	
	Intensity	inputIntensity	0.0 - 5.0
Noise Reduction		**NoiseReduction**	
	Noise Level	inputNoiseLevel	0.0 - 0.1
	Sharpness	inputSharpness	0.0 - 2.0
Circle Splash / Hole Distortion		**CircleSplashHoleDistortion**	
	Choose Effect	Choose_Effect	0 (Circle Splash) 1 (Hole)
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Radius	inputRadius	0.0 - 1000.0
Pinch / Bump Distortion		**PinchBumpDistortion**	
	Choose Effect	Choose_Effect	0 (Pinch) 1 (Bump) 2 (Bump Linear)
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Radius	inputRadius	0.0 - 1000.0
	Angle	inputAngle	-360.0 - 360.0
	Scale	inputScale	0.0 - 2.0
Torus / Lens Distortion		**TorusLensDistortion**	
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Radius	inputRadius	0.0 - 500.0
	Width	inputWidth	0.0 - 200.0
	Refraction	inputRefraction	-5.0 - 5.0
Twirl / Circular Wrap / Vortex		**TwirlVortexDistortion**	
	Choose Effect	Choose_Effect	0 (Twirl) 1 (Circular Wrap) 2 (Vortex)
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Radius	inputRadius	0.0 - 1000.0
	Angle	inputAngle	-360.0 - 360.0
Glass Lozenge		**GlassLozenge**	
	Point 1 X	Point_0_X	0.0 - 1.0
	Point 1 Y	Point_0_Y	0.0 - 1.0
	Point 2 X	Point_1_X	0.0 - 1.0
	Point 2 Y	Point_1_Y	0.0 - 1.0
	Radius	inputRadius	0.0 - 1000.0
	Refraction	inputRefraction	-5.0 - 5.0

Effect Name	Parameter Name	Scripting Name	Allowed Values
Keystone Correction		**KeystoneCorrection**	
	Correction Type	Choose_Effect	0 (Combined) 1 (Horizontal) 2 (Vertical)
	Upper Left X	UL_X	0.0 - 1.0
	Upper Left Y	UL_Y	0.0 - 1.0
	Upper Right X	UR_X	0.0 - 1.0
	Upper Right Y	UR_Y	0.0 - 1.0
	Lower Left X	LL_X	0.0 - 1.0
	Lower Left Y	LL_Y	0.0 - 1.0
	Lower Right X	LR_X	0.0 - 1.0
	Lower Right Y	LR_Y	0.0 - 1.0
	Focal Length	inputFocalLength	0.0 - 30.0
Op Tile		**OpTile**	
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Angle	inputAngle	-360.0 - 360.0
	Scale	inputScale	0.1 - 10.0
	Width	inputWidth	0.0 - 1000.0
Perspective Tile		**PerspectiveTile**	
	Bottom Left X	Bottom_Left_X	0.0 - 1.0
	Bottom Left Y	Bottom_Left_Y	0.0 - 1.0
	Bottom Right X	Bottom_Right_X	0.0 - 1.0
	Bottom Right Y	Bottom_Right_Y	0.0 - 1.0
	Top Left X	Top_Left_X	0.0 - 1.0
	Top Left Y	Top_Left_Y	0.0 - 1.0
	Top Right X	Top_Right_X	0.0 - 1.0
	Top Right Y	Top_Right_Y	0.0 - 1.0
Quad Tiles		**QuadTiles**	
	Choose Effect	Choose_Effect	0 (Fourfold Translated Tile) 1 (Fourfold Reflected Tile) 2 (Parallelogram Tile)
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Angle	inputAngle	-360.0 - 360.0
	Acute Angle	inputAcuteAngle	-360.0 - 360.0
	Width	inputWidth	1.0 - 200.0
Reflected Tiles		**ReflectedTiles**	
	Choose Effect	Choose_Effect	0 (Glide Tile) 1 (Sixfold Reflected Tile) 2 (Eightfold Reflected Tile) 3 (Twelvefold Reflected Tile)
	Input X	_protocolInput_X	0.0 - 1.0

Effect Name	Parameter Name	Scripting Name	Allowed Values
	Input Y	_protocolInput_Y	0.0 - 1.0
	Angle	inputAngle	-360.0 - 360.0
	Width	inputWidth	1.0 - 200.0
Rotated Tiles		**RotatedTiles**	
	Choose Effect	Choose_Effect	0 (Triangle Tile) 1 (Fourfold Reflected Tile) 2 (Sixfold Reflected Tile)
	Input X	_protocolInput_X	0.0 - 1.0
	Input Y	_protocolInput_Y	0.0 - 1.0
	Angle	inputAngle	-360.0 - 360.0
	Width	inputWidth	1.0 - 200.0
Nine-Part Stretched		**NinePartStretched**	
	Lower Left X	LL_X	0.0 - 1.0
	Lower Left Y	LL_Y	0.0 - 1.0
	Upper Right X	UR_X	0.0 - 1.0
	Upper Right Y	UR_Y	0.0 - 1.0
	Grow Amount X	Grow_X	0.0 - 1000.0
	Grow Amount Y	Grow_Y	0.0 - 1000.0
Nine-Part Tiled		**NinePartTiled**	
	Lower Left X	LL_X	0.0 - 1.0
	Lower Left Y	LL_Y	0.0 - 1.0
	Upper Right X	UR_X	0.0 - 1.0
	Upper Right Y	UR_Y	0.0 - 1.0
	Grow Amount X	Grow_X	0.0 - 5000.0
	Grow Amount Y	Grow_Y	0.0 - 5000.0
	Flip Vertically	inputFlipYTiles	BOOLEAN
Shutter		**Shutter**	
	Shape	Shape	0 - Rectangle 1 - Ellipse
	Left	Left	0.0 - 1.0
	Right	Right	0.0 - 1.0
	Bottom	Bottom	0.0 - 1.0
	Top	Top	0.0 - 1.0
	Feaather	Feather	0.0 - 800.0
Window		**Window**	
	Origin X	Origin_X	0.0 - 1.0
	Origin Y	Origin_Y	0.0 - 1.0
	Width	Size_W	0.0 - 1.0
	Height	Size_H	0.0 - 1.0
	Rectangle	inputRectangle	Four floats, each 0.0 - 1.0
Drop Shadow		**DropShadow**	

Effect Name	Parameter Name	Scripting Name	Allowed Values
	Distance	Distance	0.0 - 400.0
	Angle	Angle	-360.0 - 360.0
	Blur radius	BlurRadius	0.0 - 400.0
	Alpha	Alpha	0.0 - 1.0
	Color	inputColor	Four floats, each 0.0 - 1.0

OSC & Scripting Examples

While having a list of available OSC and AppleScript commands is all well and good, it can sometimes be difficult to put the pieces together into something complete write to support@figure53.com and send us your script-in-progress, and we'll do our best to get you going.

Remember, too, that there is usually more than one way to get something done, and the examples here are not meant to be authoritatively the "right" or even "best"

The AppleScript examples here can be run from within a Script cue or from another application such as Script Editor. The OSC examples can be run from a Network

Adjusting Audio Levels

The following AppleScript and OSC examples both set the main level of all selected cues to `-10` dB. You could easily replace `-10` with some other number, but rem

AppleScript

```
tell application id "com.figure53.qlab.5"
  repeat with theCue in (selected of front workspace as list)
    try
      theCue setLevel row 0 column 0 db -10
    end try
  end repeat
end tell
```

`tell` is how AppleScript starts a block of code. `com.figure53.qlab.5` is how you instruct the AppleScript interpreter to send the contents of this block of code to C

`repeat` starts another block of code which will be repeated until a particular condition is met, and `with` indicates that the condition has to do with a variable. `the list)` sets up the details of the condition: find all of the cues in the front workspace which are selected, make a list of those cues, and then repeat the following cod

`try` starts yet another block of code with a special condition: if, for some reason, the code inside this block doesn't work for a particular iteration of `theCue`, don't case wherein some of the selected cues don't have audio levels. We want to be able to select a bunch of cues and run the script without worrying about whether eve cues and ignore the 💬 Memo cue. The `try` block allows that to happen.

`setLevel` is a QLab-specific AppleScript command which sets the level of the specified matrix crosspoint in the specified cue. `theCue` is the variable which repres the main input column of the audio levels matrix mixer, and `db -10` states the level that you want to set.

`end try` closes the try block.

`end repeat` closes the repeat block.

`end tell` closes the tell block.

OSC

The OSC solution to this problem is rather simpler; it's just a single command:

```
/cue/selected/level/0/0 -10
```

`/cue/selected` directs the OSC message to all selected cues.

`/level` says that the value we're sending (`-10` in this case) should be used to adjust the audio level. `/0/0` represents the row and column of the crosspoint that sh

And finally `-10` is the value that gets sent.

QLab always ignores OSC messages when they're inapplicable, so AppleScript's concept of "try" is unnecessary.

Disarming A Specific Cue

The following AppleScript and OSC examples show how to disarm a specific cue. In this example, the cue is numbered “4”, but the same thing works with any cue tha
“panda” works too, if you have cue in your workspace numbered “panda”. Since cue numbers have to be unique across a workspace and are always human-readable

AppleScript

```
tell application id "com.figure53.qlab.5" to tell front workspace
  set armed of cue "4" to false
end tell
```

It seems simple enough, but there are a couple of important details in there.

First, notice that the tell block says “tell *something* to tell *something else*”. The reason for that is that QLab can have more than one workspace open at a time, and ea
wanted to address a specific workspace, whether or not it was in front, we could instead write:

```
tell application id "com.figure53.qlab.5" to tell workspace "Hamlet.qlab5"
```

That only works, obviously, if the workspace is saved with the name “Hamlet”.

The second important thing is that the cue number is in quotation marks. If you did not put quotes around the number, QLab would think you were trying to refer to
difference.

OSC

The OSC version of this operation is fairly similar:

```
/cue/4/armed 0
```

The OSC message `/armed` interprets “0” as “false” and any other number as true. You could also use `/F` in place of the `0`, because `/F` is the special OSC operato

Create And Move a New Cue

This example is not necessarily the most immediately practical, but it demonstrates several useful concepts which can be applied to many different kinds of situation

AppleScript

This script creates a new 💬 Memo cue and then moves the newly created cue into a [ ] Group cue. It asks the human to enter the cue number of the [ ] Group cue
the default choice.

```
1  tell application id "com.figure53.QLab.5" to tell front workspace
2
3      set theGroupNumber to ""
4
5      try
6          set theSelectedCue to last item of (selected as list)
7          if q type of theSelectedCue is "Group" then
8              set theGroupNumber to q number of theSelectedCue
9          end if
10     end try
11
12     display dialog "Enter the cue number of the Group cue that the new cue will be created in:" default answer theGroupNumber with icon
13
14     set theGroupNumber to text returned of result
15     set theGroup to the first cue whose q number is theGroupNumber
```

```
16
17      make type "Memo"
18      set newCue to last item of (selected as list)
19      set newCueId to uniqueID of newCue
20      set cueList to parent of newCue
21      move cue id newCueId of cueList to end of theGroup
22
23  end tell
```

First, the `tell` block directs the AppleScript at the front workspace in QLab.

Next, we `set theGroupNumber to ""` which creates an empty variable `theGroupNumber`. This is necessary because later on, the script is going to check the contents there will be something to check.

Next comes a `try` block, which means "if the code inside this block returns an error, just forget about it and move on." That's necessary because the first line in the and sets its contents to the selected cue. If more that one cue is selected, the last selected cue is the one that goes into the variable. This line assumes that there is a result, because the script doesn't require a selected cue to operate correctly, so we want to allow for the possibility that no cues are selected.

If a cue *is* selected, we want to save its cue number for future reference, but only if it's a [ ] Group cue. So we use an `if` statement to open another block: `if q ty` executed and the cue number of the cue is stored in `theGroupNumber : set theGroupNumber to q number of theSelectedCue`. If the selected cue is any other type

The next line displays a dialog box on screen which asks the human for input. This line has a lot of pieces:

- `display dialog "Some text"` is the basic form of this message. The text inside quotation marks will appear as a prompt to the human. This is the only necessar
- `default answer theGroupNumber` fills in a default answer using the contents of the variable `theGroupNumber`.
- `with icon note` shows an icon in the box. There are three options... `icon note` shows the application icon, `icon stop` shows a stop sign with the application
- `with title "New Cue In Group"` sets the text which appears in the title bar of the dialog box.
- `buttons {"Cancel", "OK"} default button "OK" cancel button "Cancel"` tells the box to show two buttons with the labels "Cancel" and "OK". Then, it tells A enter key for a keyboard shortcut, and that the button labeled "Cancel" should be the cancel button, which stops the rest of the script from running when it's cl

The first line after the `display dialog` line stores the text that the human entered in the variable called `theGroupNumber : set theGroupNumber to text returned`

Then, `set theGroup to the first cue whose q number is theGroupNumber` sets `theGroup` to refer to the actual cue that has that number.

Next, the script creates a new 💬 Memo cue with `make type "Memo"`. Newly created cues are automatically selected, which means they are necessarily the last selec `last item of (selected as list)`

Moving cues can be complicated because they have to be moved relative to the cue list that they are contained within, rather than to the workspace as a while, and t uniqueID of the new cue and saves it in a variable called `newCueId`

Next, we need to find out what cue list contains the new cue, and store that in another variable: `set cueList to parent of newCue`

At last, we know everything we need to know to move the new cue into the [ ] Group: `move cue id newCueId of cueList to end of theGroup`

And then we wrap it all up with `end tell` which closes the outermost block of the script.

OSC

The above can be approximated with a series of OSC messages...

```
/new memo
/cue/selected/name #/cue/selected/uniqueID#
/move/#/cue/selected/name# {1} {group_cue_id}
```

First, create a new 💬 Memo cue. Then set the name of the selected cue (which is that new cue) to its own uniqueID using an OSC query. Then move the cue to withi cue).

Create Fade-in Cues

This is the most complex of these examples: create fade-in cues for every selected cue, fading the main audio level to the level that the source cue is set to, and if th

While it's technically possible to achieve this via OSC, there's a lot of decision making in this script which is not what OSC is good at. OSC is about sending individual m
QLab, interprets it, and then makes decisions about how to proceed.

If you skipped over the first example, I recommend going back and reading it first. This example builds on those concepts.

```
tell application id "com.figure53.qlab.5"
  try
    repeat with sourceCue in (selected of front workspace as list)
      if q type of sourceCue is "Audio" or q type of sourceCue is "Mic" or q type of sourceCue is "Video" or q type of sourceCue is "Cam
        set sourceCueLevel to sourceCue getLevel row 0 column 0
        sourceCue setLevel row 0 column 0 db -120
        make front workspace type "Fade"
        set newCue to last item of (selected of front workspace as list)
        set cue target of newCue to sourceCue
        newCue setLevel row 0 column 0 db sourceCueLevel

        if q type of sourceCue is "Video" or q type of sourceCue is "Camera" then
          set sourceOpacity to opacity of sourceCue
          set opacity of sourceCue to 0
          set opacity of newCue to sourceOpacity
          set do opacity of newCue to true
        end if

      else if q type of sourceCue is "Text" then
        make front workspace type "Fade"
        set newCue to last item of (selected of front workspace as list)
        set cue target of newCue to sourceCue
        set sourceOpacity to opacity of sourceCue
        set opacity of sourceCue to 0
        set opacity of newCue to sourceOpacity
        set do opacity of newCue to true
      end if
    end repeat
  end try
end tell
```

We start with a `tell` block, as always, and immediately follow up with a `try` block. In this case, we're going to use other rules to narrow the scope of how the script

`repeat with sourceCue in (selected of front workspace as list)` tells AppleScript that we're going to repeat this block of code once for each selected cue, a

The next line is our first `if` statement, and it's a fairly complex one: `if q type of sourceCue is "Audio" or q type of sourceCue is "Mic" or q type of sourc`

AppleScript reads a lot like plain English, so the meaning of this line is actually fairly straightforward. Remember that each time the `repeat` block loops through, `sou`
Audio, Mic, Video, or Camera cue, then the code within the `if` block is executed. If the cue is any other type of cue, the code within the `if` block is s

Assuming the cue passes the test, we move into the `if` block.

`set sourceCueLevel to sourceCue getLevel row 0 column 0` creates a variable called `sourceCueLevel`, and then fills that variable with the main level of `sourceC`

Once we've stored the level, we can set `sourceCue` to silent by setting its main level to `-120`, which is the lowest possible audio level in QLab: `sourceCue setLevel`

Then, we make a new Fade cue with the command `make front workspace type "Fade"`. That "front workspace" part tells AppleScript which workspace the new

`set newCue to last item of (selected of front workspace as list)` creates a new variable, `newCue`, and fills it with the Fade cue we just created.

`set cue target of newCue to sourceCue` sets the target of the new Fade cue to the cue currently being represented by `sourceCue` for this loop.

`newCue setLevel row 0 column 0 db sourceCueLevel` sets the main audio level of the new Fade cue to the level we fetched from `sourceCue` and stored in the

Now, we dive one level deeper with another `if` block: `if q type of sourceCue is "Video" or q type of sourceCue is "Camera" then`

This isn't the most efficient thing, to check on the type of cue twice, but it's not so terrible and it makes the code easier to read. Here, we've already set the audio lev to be faded in as well.

`set sourceOpacity to opacity of sourceCue` creates a variable called `sourceOpacity`, and then fills that variable with the opacity of `sourceCue`. Now that we've

`set opacity of newCue to sourceOpacity` sets the opacity of the new Fade cue to the stored `sourceOpacity`, and `set do opacity of newCue to true` chec

`end if` closes the inner `if` block.

The next line is an `else` statement, which is a cousin to the `if` statement. This line is only evaluated when the answer to the first `if` question is "no". So the first `if` block executes. If the answer is *no*, then the script skips down to this `else` statement, which asks "OK, well is `sourceCue` a Text cue?" If the answer is still *no*, thi

The reason we need the `else` statement is that Text cues have no audio component. The code inside the `else` block is a copy of the opacity-related second ha

`end if` closes the main if block, `end repeat` closes the repeat block, `end try` closes the try block, and `end tell` closes the tell block.

Chapter 10: Other Cues

10.1 Transport Cues
10.2 Devamp Cues
10.3 GoTo Cues
10.4 Target Cues
10.5 Arm and Disarm Cues
10.6 Wait Cues
10.7 Memo Cues

Transport Cues

Transport cues are the five types of cues whose role is to direct the playback of other cues:

- Start cues tell their target to begin playing or, if they're currently paused, to un-pause.
- Stop cues tell their target to stop playing.
- Pause cues tell their target to pause.
- Load cues prepare their target to start playing.
- Reset cues tell their target to stop playing and discard any temporary changes they have accumulated.

Transport cues require a cue target, and other than the Load cue they have no unique settings; they simply do what their name says.

Load cues and Reset cues require some explanation.

It's important to keep in mind that Transport cues do *not* move the playhead. For example, a Start cue will start its target cue playing, but it will not move the playhead to the target cue, or to the cue after its target. To move the playhead to a specific spot, use a GoTo cue.

The Load Cue

The Load cue has a dual role, loading and loading-to-time. To understand both roles requires a discussion of loading in general.

Loading Cues

At the highest level, loading a cue prepares QLab to start that cue with a minimum of latency. Starting cues generally happens very, very fast from a human's perspective, but there are some things that can be done ahead of time to make them start even faster. When you need a cue to start absolutely as soon as possible after it receives the command to GO, loading that cue shortly before it's started can help.

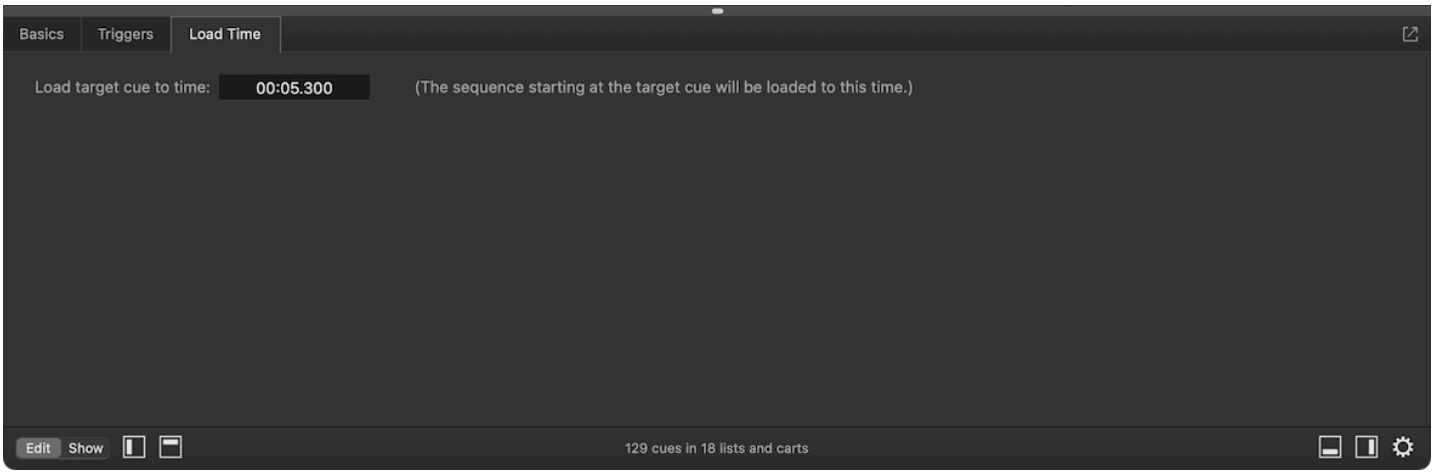
Loading can also help the situation of a cue sequence that involves either a large number of cues starting at exact same time, or a modest number of cues starting at the same time, all targeting "heavy" file targets such as multi-track audio files or high resolution video files. Because QLab prioritizes accuracy over speed, every cue in a cue sequence must be ready before the sequence can start. Since opening media files and decoding their contents takes time, it is possible to stack up a large enough number of these cues in a Timeline Group cue or a series of cues set to ↓ auto-continue that the delay between pressing GO and seeing or hearing the sequence begin can become noticeable. Loading the sequence, which means simply loading all the cues in the sequence, minimizes this delay.

The list of things that happen in response to loading varies depending upon the type of cue(s) being loaded, but can be conceptualized as "all the things that are involved in starting the cue, right up to and *not* including actually starting it." Loading cues which have file targets begins to load data from the file into memory, but the exact amount of data is not fixed and depends on a large number of details.

Most of the time, loading cues is absolutely not necessary. Modern Macs all use high quality SSDs for storage and high-speed RAM for memory. Apple Silicon Macs in particular have extraordinarily fast SSDs and even more remarkably fast memory. It is our enthusiastic recommendation to avoid loading cues unless you discover that doing so addresses an observed performance issue.

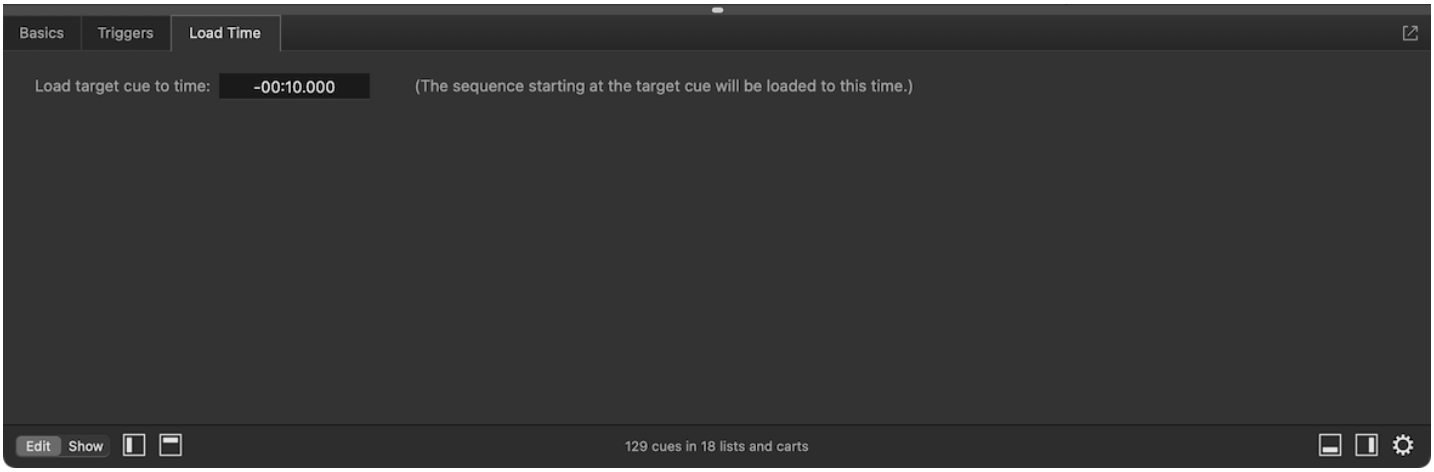
Load To Time

Loading to time means preparing a cue to start somewhere other than at its beginning. Cues can be loaded to time on demand by using the Load to Time tool, which you can read about in the section on the Tools Menu in this manual, or in a pre-arranged, cue-like way by using a Load cue.



The Load Time tab of the inspector contains a single text field in which you can enter a time to which the target cue will load when the Load cue is run. If the target cue is at the beginning or in the middle of a cue sequence, the following cue(s) in the sequence will follow this load time as well.

If you type a negative value into the field, QLab will load to that amount of time back from the end of the cue or sequence, if possible. For example, if you type `-10` into the field and press enter, QLab will load the selected cue to ten seconds before the end of the cue.



If you create a Load cue while another cue is loaded to time and selected, the new Load cue will start off targeting that cue *and* use the loaded time of the other cue as its load time.

The Reset Cue

Running a Reset cue will stop its target cue and reset any temporary changes that have been made to the cue since the workspace opened. Possible temporary changes include:

- A change of any parameter made by a Fade cue.
- A change of target made by a Target cue.
- A "before action" or "after action" cue color.
- Any cue parameter changed by an OSC message which includes "temp" in the message address.

Broken Transport Cues

Transport cues become broken if they have no cue target assigned.

Additionally, Pause cues require any type of license and will become broken if they are used without a license installed.

Devamp Cues

> The Devamp tutorial explores the capabilies of the ↻ Devamp cue.

↻ Devamp cues are named for the musical term “vamp” which means to play a specific section of music on loop until a signal is given, then exit the loop naturally at the end of the current iteration. ↻ Devamp cues require an 🔊 Audio or 🎞 Video cue as a target. When the target cue is running and the ↻ Devamp cue is started, it watches its target cue. When the playback of the target cue reaches the end of the currently-playing slice, or the end of the cue, the ↻ Devamp cue’s action is executed. That action is configurable in the cue’s Settings tab. ↻ Devamp cues require any type of license.

The Inspector for Devamp Cues

When a ↻ Devamp cue is selected, the inspector shows the Basics tab and Triggers tab, used by all cues, as well as the Settings tab.

The Settings tab

The Settings tab contains three controls which, used in combination, allow you to use the ↻ Devamp cue in several different ways.

Slice or Cue

The first control is a pop-up menu with two options.

The first option, **Devamp currently looping slice**, tells the ↻ Devamp cue to execute its action at the end of the current slice of its target cue. If that slice is looping, playback exits the loop and continues on to the next slice.

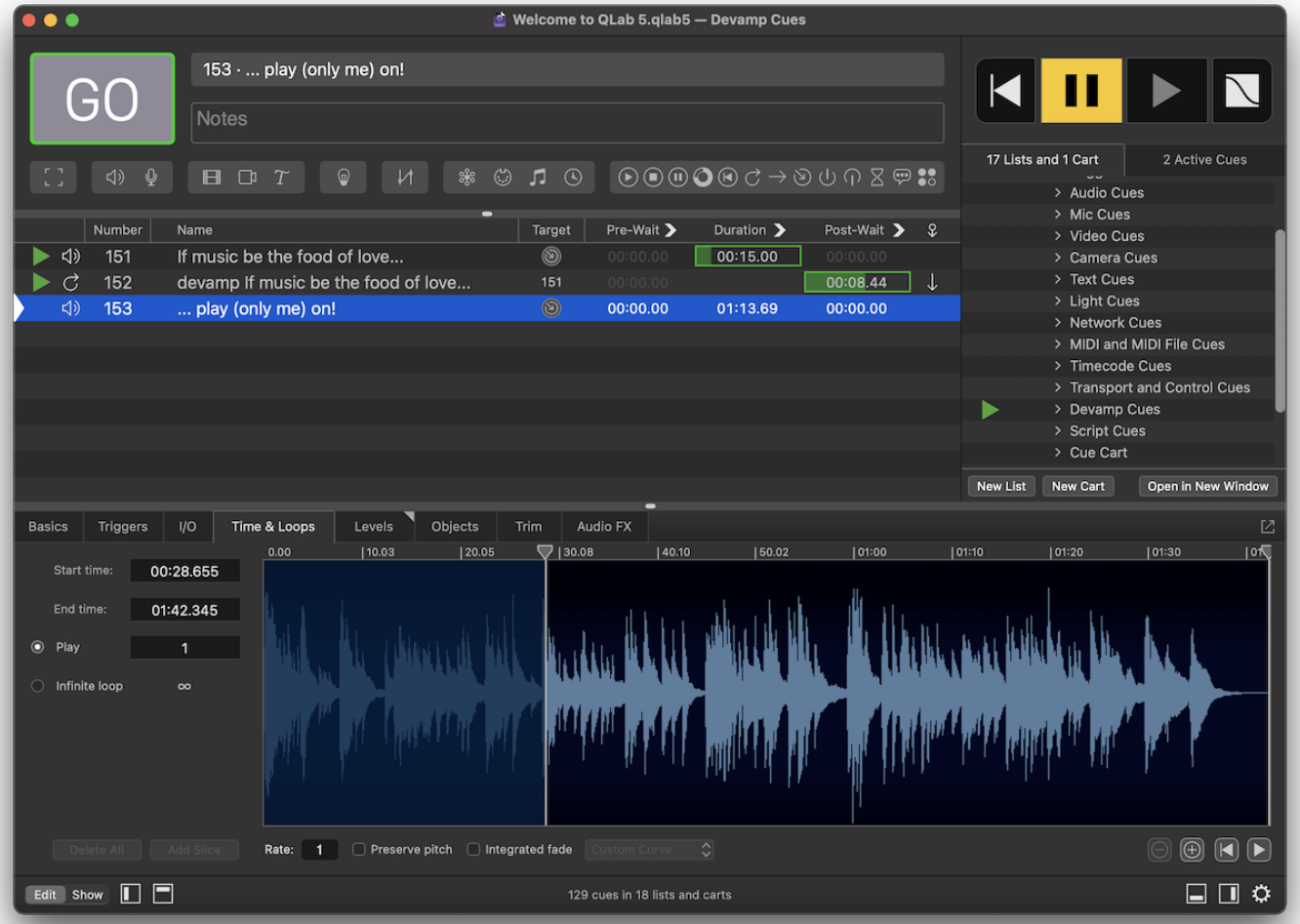


In this screen shot, an 🔊 Audio cue contains a slice set to loop infinitely from around `21` seconds to around `28` seconds. If a ↻ Devamp cue targets this cue and is run while the section in question is looping, QLab waits until the current loop is complete and then cleanly exits the loop.

The second option, **Devamp looping cue**, tells the ↻ Devamp cue to ignore slices within the target cue and execute its action when the cue reaches its end. If the cue is looping, it simply stops playing.

Start next

The second control is a checkbox labeled *Start next cue when target reaches the end of the current slice*. This box can be thought of as a special devamp-specific form of ↓ auto-continue. When this box is checked, the ↻ Devamp cue starts the cue which follows it exactly simultaneously with the moment that it devamps its target cue.



The post-wait time is dynamically calculated based on the exact playback time of the target cue. If the ↻ Devamp cue runs very early in the loop, the post-wait time will be longer. If it runs closer to the end of the loop, the post-wait time will be shorter.

Stop target

The third control is a checkbox labeled *Stop target when it reaches the end of the current slice.** This checkbox is only enabled when the *start next* checkbox is checked. If this box is checked, the ↻ Devamp cue stops its target instead of devamping. In this way, a ↻ Devamp cue can be used to precisely transition from one cue to another.

Using the Devamp Cue

There are three basic ways to use a Devamp cue, illustrated by the following three examples.

Devamp and Continue

Devamp and continue is the quintessential example of a devamp, musically speaking; the music loops until *some appropriate moment* happens, and then the music continues onwards in a musically appropriate way, not skipping ahead to the end of the loop but simply playing on.



When the target cue is started, it will play into the looping slice and then keep repeating that slice indefinitely... until you run the Devamp cue. Once you do, and playback reaches the end of the slice, the Devamp cue causes the target cue to exit the loop and proceed onwards.

If you have multiple looping slices in the target cue, you can use multiple Devamp cues to pop out of each loop. Each Devamp cue will “un-loop” whichever slice is currently looping at the time that the Devamp cue is run.

Devamp and Start Next

Devamp and start next starts the following cue at the instant of the devamp. You could use this technique with an Audio cue to layer in an additional instrument to a musical recording, or with a MIDI cue to send a MIDI message at a precise musical moment.



When the target cue is started, it will play into the looping slice and then keep repeating that slice indefinitely... until you run the Devamp cue. Once you do, and playback reaches the end of the slice, the Devamp cue causes the target cue to exit the loop and proceed onwards.

At the same moment that the original cue passes the slice marker and continues onwards, the Devamp cue starts the following cue.

Devamp, Start Next, and Stop Target

Devamp, start next, and stop target “hands off” playback from the target cue to the following cue.



When the target cue is started, it will play into the looping slice and then keep repeating that slice indefinitely... until you run the ↻ Devamp cue. Once you do, and playback reaches the end of the slice, the ↻ Devamp cue causes the target cue to stop.

At the same moment that the target cue stops, the ↻ Devamp cue starts the following cue.

Thinking In Bars and Beats

The ↻ Devamp cue enables you to communicate with QLab in terms of bars and beats by putting slice markers on each beat, and using ↻ Devamp cues to execute actions that line up perfectly with those beats. As with all things, giving yourself plenty of time to experiment is the key to success.

Broken Devamp Cues

↻ Devamp cues can become ✕ broken for the following reasons:

No target cue.

Select a target cue to clear this warning.

License required

↻ Devamp cues require a license of any type.

GoTo Cues

→ GoTo cues move the playhead to their target cue. As a result, the target of the → GoTo cue becomes the cue that's standing by. The next press of the GO button will start that cue. The → GoTo cue is a complement to the ▶ Start cue.

→ GoTo cues have only the usual Basics tab and Triggers tab used by all cues.

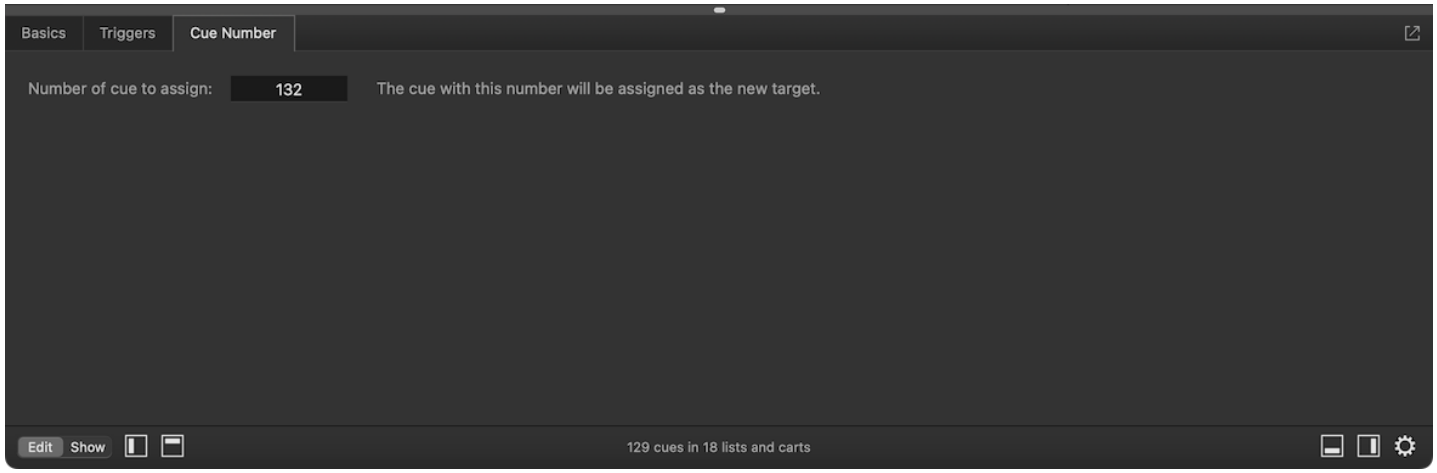
Broken GoTo Cues

→ GoTo cues will only become ✕ Broken if they do not have a target.

Target Cues

Target cues change the target of another cue in the workspace. They require any type of license.

Besides the usual Basics tab and Triggers tab, used by all cues, Target cues have a Cue Number tab.



Target cues require a cue target, which itself must be a cue type that accepts a cue target. When the Target cue runs, it sets the *temporary target* of its target cue to whatever cue is specified in the Cue Number tab. As you may have surmised, a cue must have a cue number in order to be set as a temporary target by a Target cue.

Target cues cannot target a Fade cue, however, because the way that Fade cues interact with their targets makes them difficult to accurately receive a temporary target.

Temporary Targets

Target cues do not permanently change their targets' target, nor do they cause a workspace to have an unsaved change when they run. This is because they set a *temporary target*. If a Target cue is used in a workspace and then that workspace is saved, closed, and reopened, the cue whose target was changed will have its original target, not the one that was set by the Target cue.

Temporary targets of cues are also reverted when either the cues or the workspace as a whole are reset.

Broken Target Cues

Target cues will become broken if they do not have a target, if they do not have a new target cue set in the Cue Number tab, or if there is no license installed.

Arm and Disarm Cues

Arm cues and Disarm cues simply arm or disarm their target cues. They require any type of license.

You can learn more about the function of arming or disarming a cue from the Basics tab section of the inspector section of this manual.

Arm and Disarm cues have only the usual Basics tab and Triggers tab used by all cues.

Broken Arm and Disarm Cues

Arm and Disarm cues can become broken if they do not have a target or if there is no license installed.

Wait Cues

⌛ Wait cues' only role is to elapse. Their post-wait time is automatically set to equal their duration. You can use a ⌛ Wait cue in combination with ⤵ auto-follows or ↓ auto-continues to control timing in cue sequences, or as a simple timer for tasks outside of QLab.

⌛ Wait cues have only the usual Basics tab and Triggers tab used by all cues.

Broken Wait Cues

⌛ Wait cues should never be ✕ broken.

Memo Cues

💬 Memo cues have no effect when started. You can use 💬 Memo cues as a place to store notes to your operator (as the name of the cue, or in the cue's notes field), as a visual separator between other cues, or for some similar reason.

💬 Memo cues have only the usual Basics tab and Triggers tab used by all cues.

Broken Memo Cues

💬 Memo cues should never be ❌ broken.

Chapter 11: Tutorials

11.1 The GO Button
11.2 How To Use A Mac
11.3 Basic Networking
11.4 Understanding USB-C
11.5 Sam's Toolbox

The GO Button

This tutorial has not yet been written.

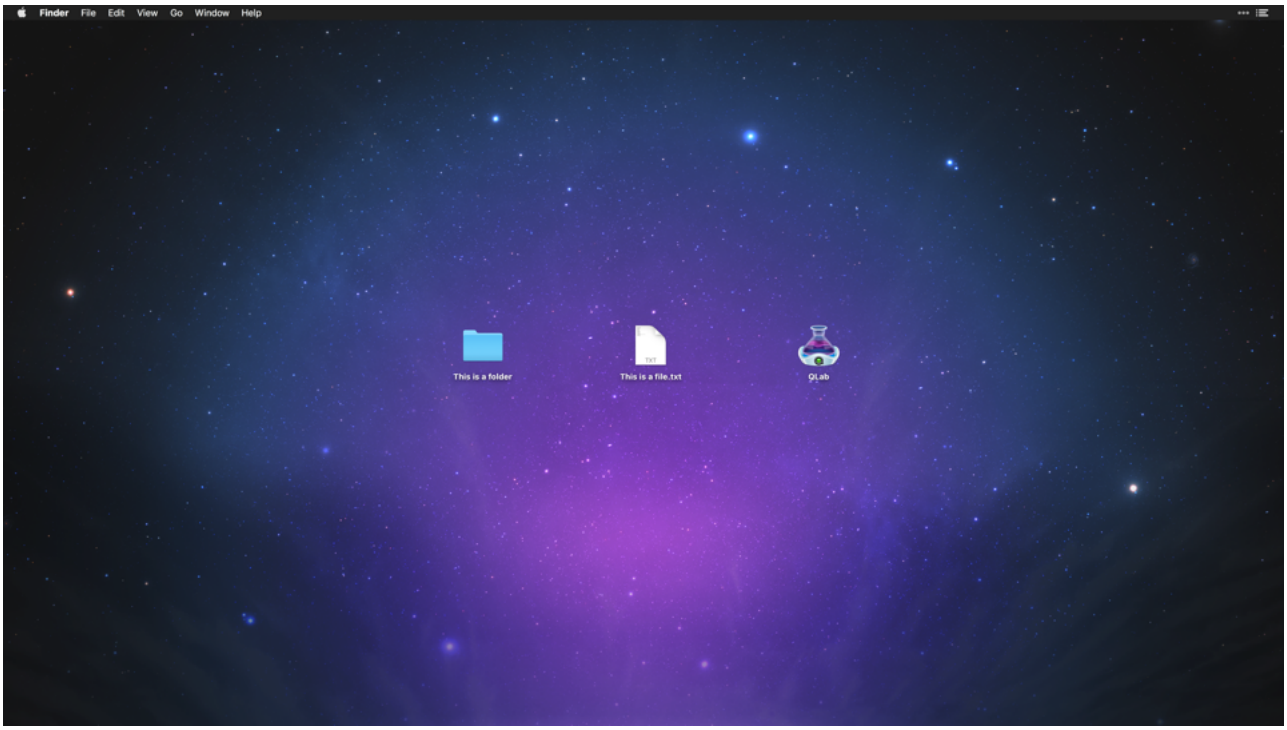
How To Use A Mac

This document is intended as a brief primer on using macOS computers. It is best viewed in a web browser on a Mac or a PC.

macOS is the name of the operating system that runs Mac computers. An operating system, or *OS*, is the base-level software that is always running on a computer. The OS defines how the computer feels to use, what the basic aesthetic is, and how the fundamental behaviors work. The OS also defines a great number of things which are invisible or at least obscure to you, the human using the computer, which dictate how other software running on the computer can operate. This is essentially why using a Mac, a Windows PC, a Linux PC, an iPhone, and an Android tablet all feel similar, yet somehow completely different.

The Desktop

The graphical interface of macOS, which is to say "the part you can look at" is designed around a series of metaphors. The main metaphor is the *desktop* which is what you're looking at when your Mac first starts up. This notion was devised in the 1980s when computers were almost exclusively used in business settings, so the metaphors all sprang from office culture, where you sit at a desk and do work.



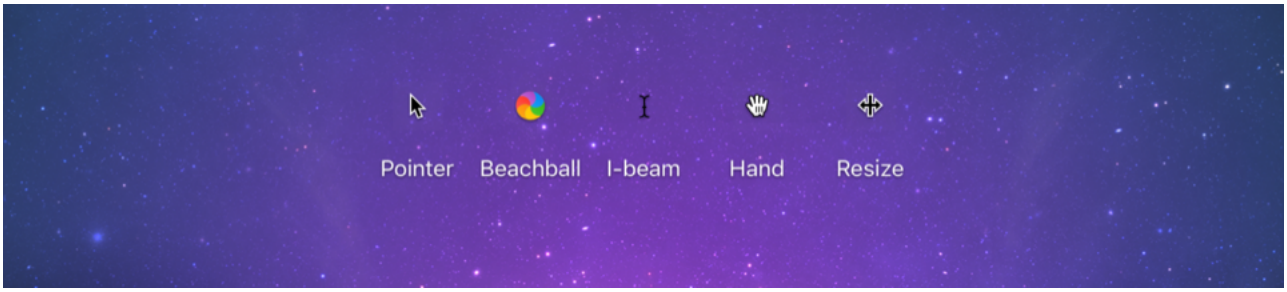
The desktop holds *folders*, *files*, and *applications*. The idea is that folders contain and organize files, so you *open* a folder to find the file you put there, then you work with that file for a while, then you *close* the file when you're done with it. Applications, also called programs, are like tools which can act upon files, just like a photocopier can act upon a printed document. Whether or not this metaphor works for you personally, it can be useful to understand it.

Unfortunately the metaphor breaks almost immediately, because opening a folder or file results in a *window* on your screen, which makes no physical sense at all. But there it is, so we have to get used to it. A window is a rectangular space on your screen which contains data. If you open a folder, the window which appears shows the contents of that folder. If you open a file, the window which appears shows the contents of that file. If you open an application, you generally see a window representing a file that the application is working with.

The Cursor

To use the Mac, you interact with it using a keyboard and a *pointing device*, which is typically a mouse, a trackpad, or a trackball. The pointing device allows you to move a *cursor* around on the screen by either moving the mouse across your desk, dragging your finger across the trackpad, or rolling the trackball.

The cursor usually takes the shape of an arrow, but can take other shapes too. The shape of the cursor is meant to give you information about what will happen when you click the button of your pointing device.



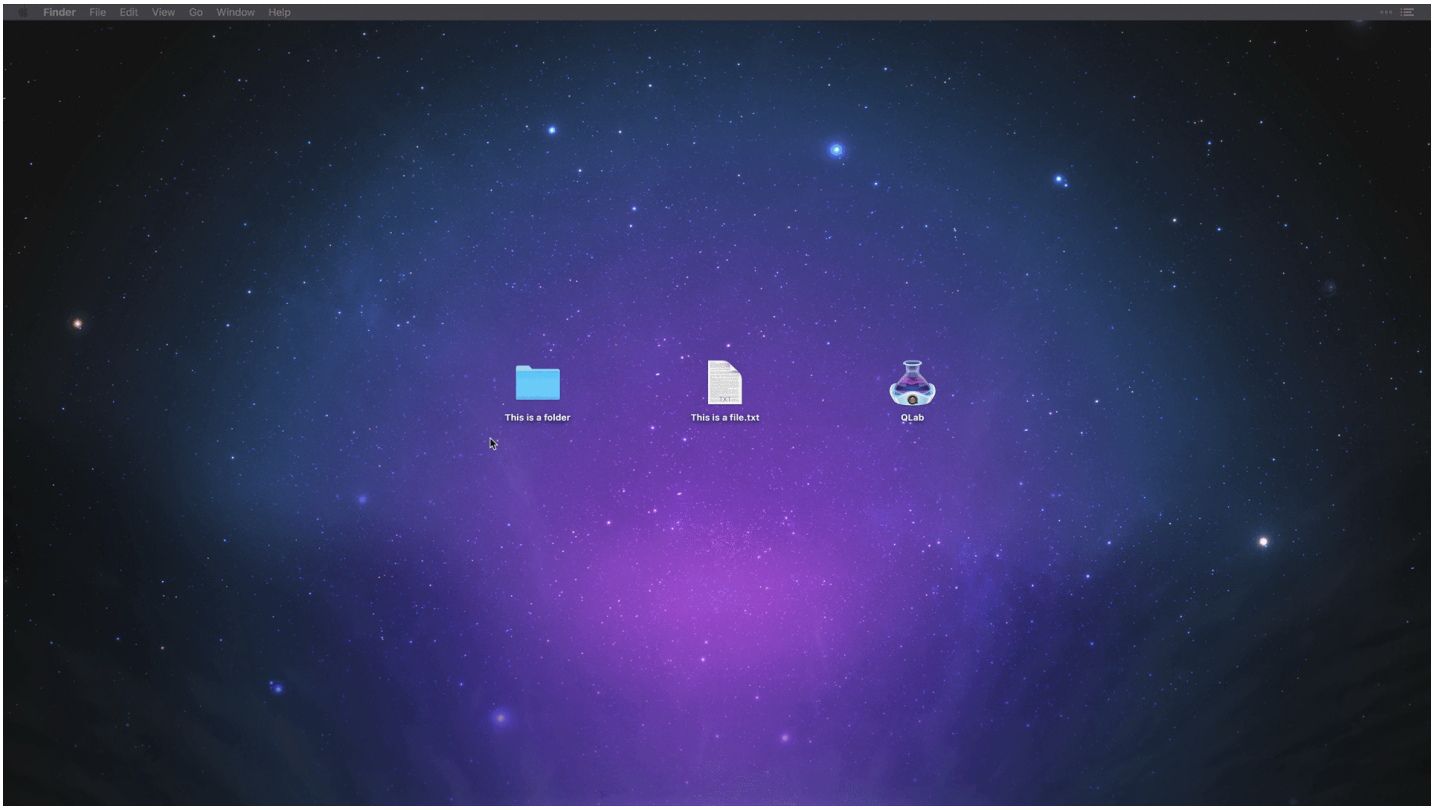
- The **pointer** is for selecting things.
- The **beachball** means the computer is busy, please wait.
- The **I-beam** is for selecting text.
- The **hand** is for moving things.
- The **resize** cursor is for, well, resizing things.

There are other cursors as well, and some programs have their own special cursor types for specific uses.

When you click the button on the pointing device, that initiates some kind of action relevant to the location of the cursor on the screen. There are four basic actions you need to know about: click, double-click, right-click, and click-drag-and-drop.

Mousing Around

Click means to press the mouse button once. If your mouse has two or more buttons, a click is usually the left-most button. If you're using a trackpad, pressing down or tapping gently is usually the equivalent of a click. If the cursor is pointing at a metaphorical object, like a file or folder, clicking *selects* that item.

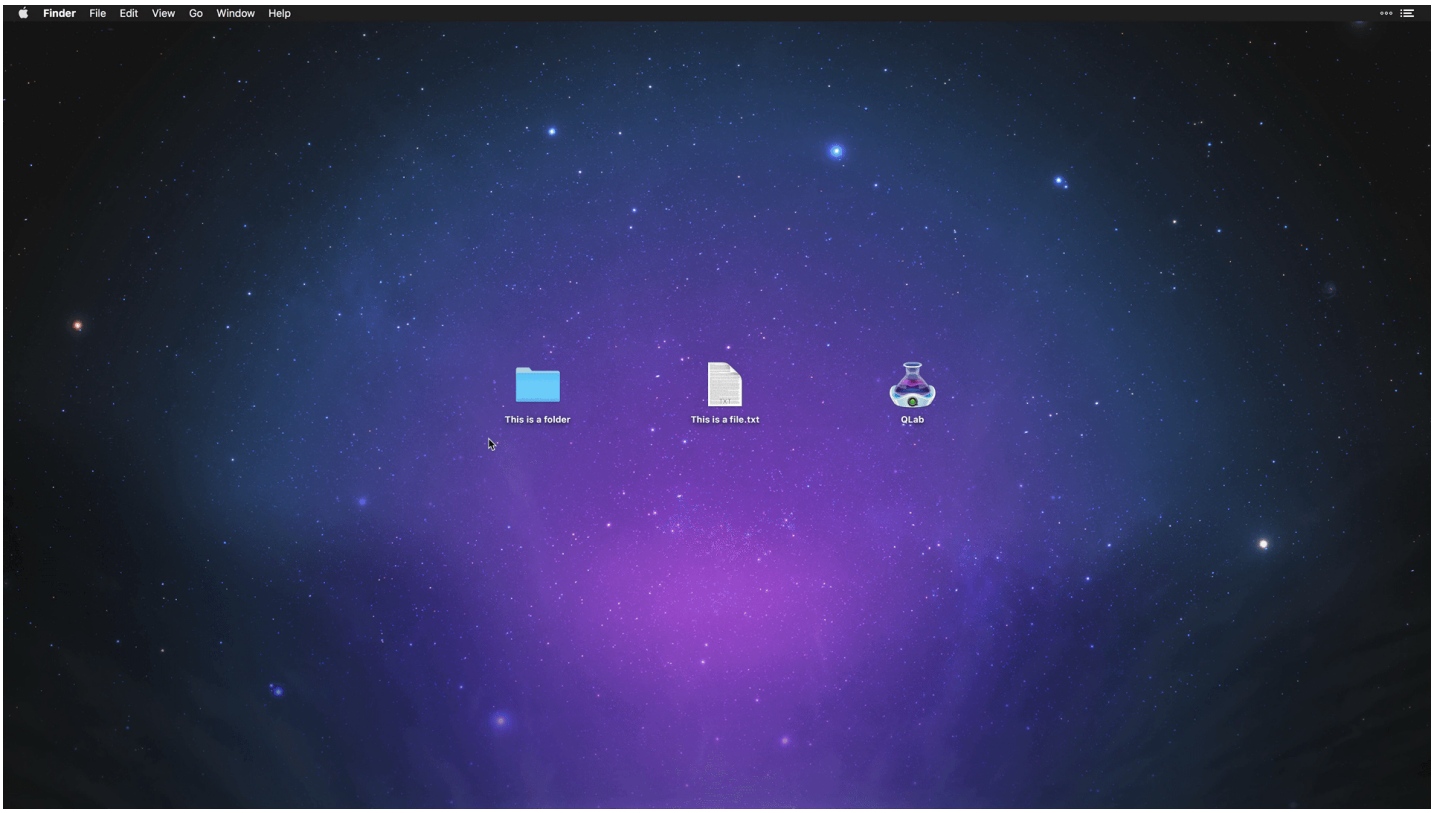


Note that the animation which appears around the cursor is added to these recordings to show when the mouse button is clicked. They don't appear in normal use.

The concept of a “selected” item is very important on the Mac. A common sequence of events is to select an item, and then instruct the Mac to do something to that item.

If the cursor is pointing at an on-screen button, clicking presses the button, causing it to do whatever it’s supposed to do. So, if you’re using a program which has a button labeled “copy,” you might click on an item to select it, then click the copy button to copy that item.

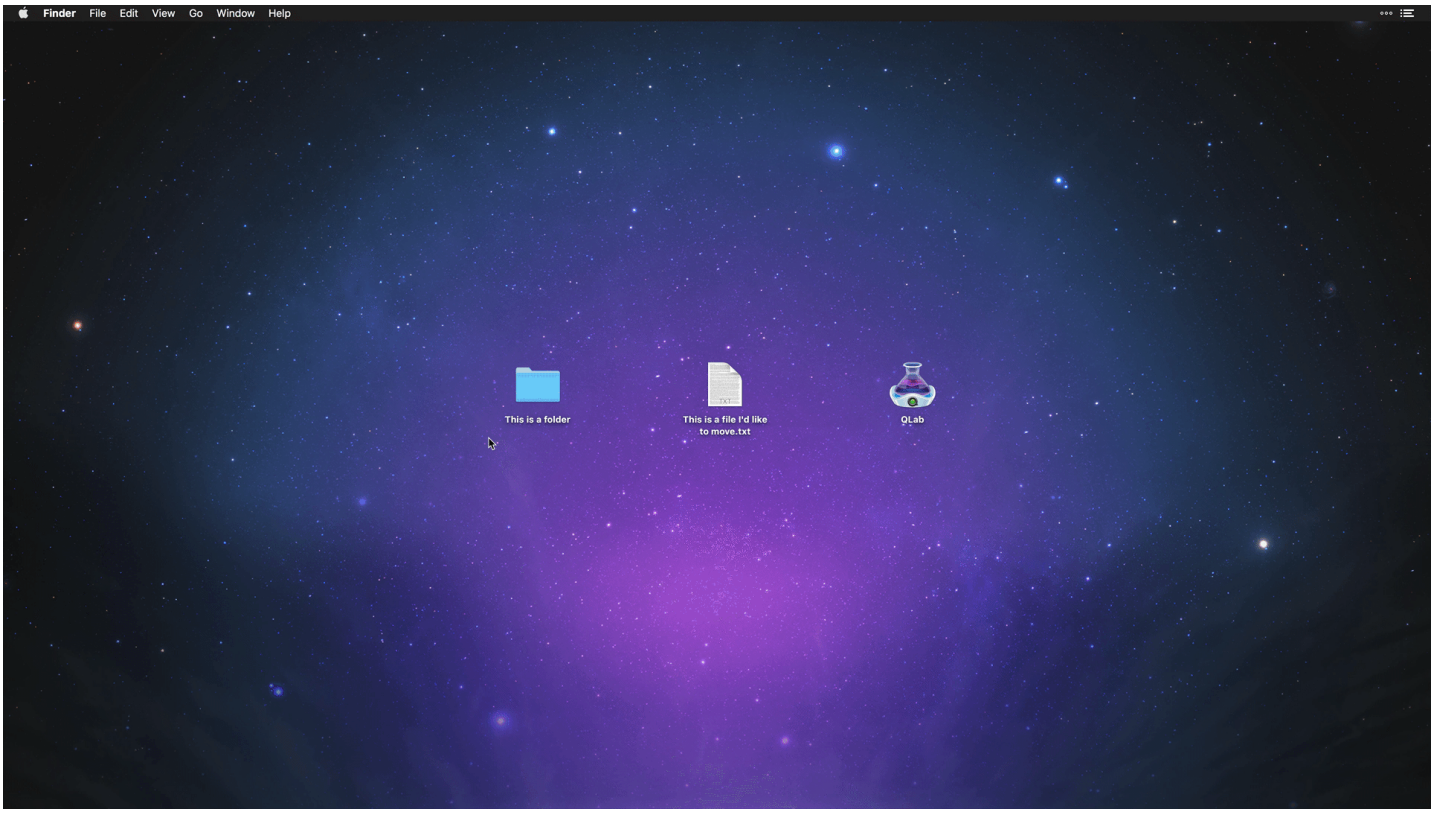
Double-click means to click the mouse button twice in rapid succession. Double-clicking on a file, folder, or application selects and opens it.



You can adjust the required time between clicks in *System Preferences* → *Mouse*.

Right-click means to press the *other* mouse button. If you’re using a trackpad, pressing or tapping with two fingers is the equivalent of a right-click. In general, you can also achieve a right-click by holding down the *control* key (often marked with this icon: ^) and clicking normally.

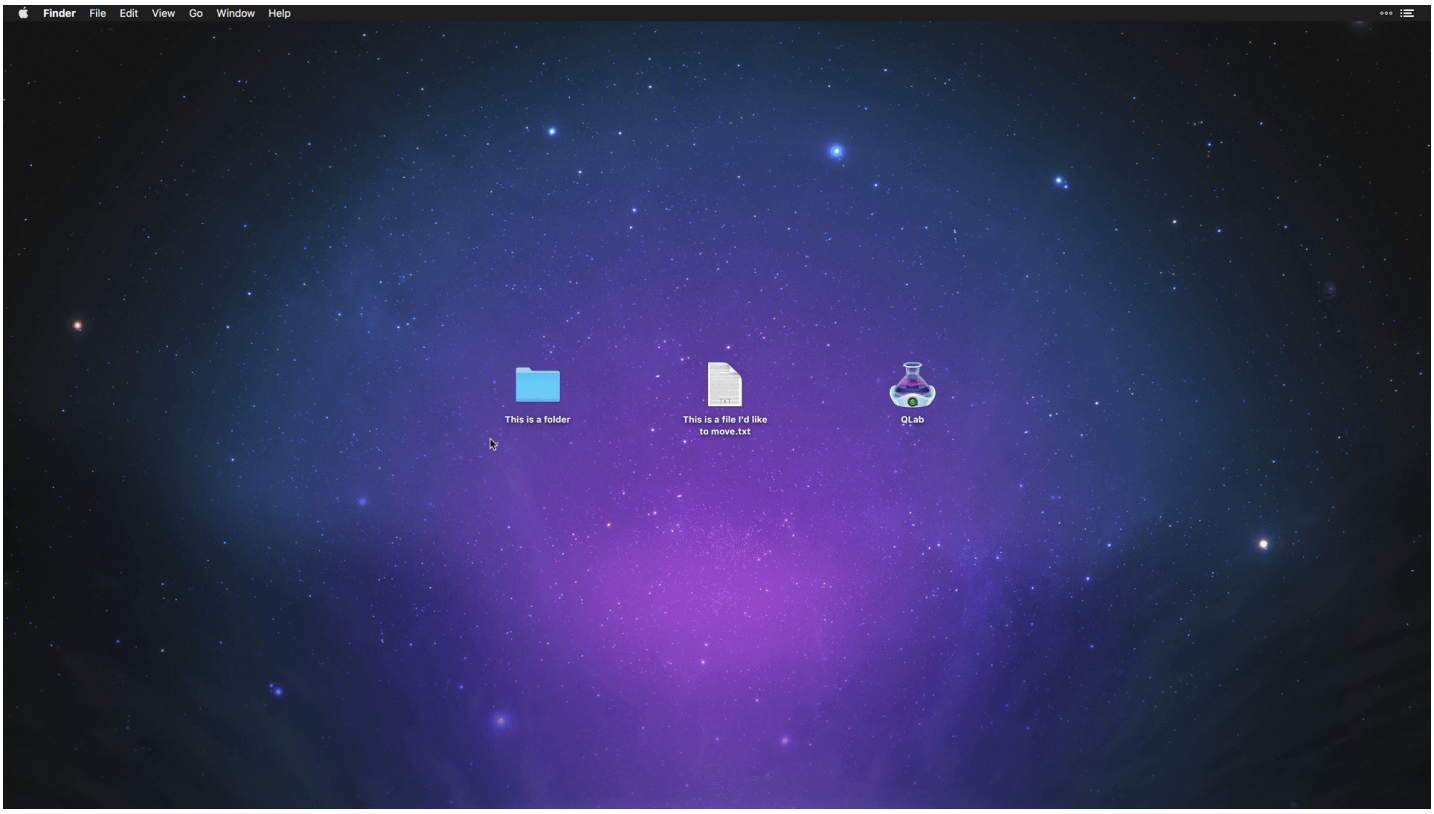
Right-clicking often opens up a *contextual menu*, which is a list of actions for you to choose from, which are relevant to the current context. For example, right-clicking on a misspelled word in a text editor often offers spelling options.



Right-clicking is handled differently by different applications and in different contexts. Right-clicking is seldom essential; it's usually a shortcut to get something done quickly, but rarely is it the only way to get something done.

If you use your pointing device with your left hand, you may prefer to reverse the behavior of your mouse buttons, so that the button under your index finger or thumb is the primary button. This can be done in *System Preferences → Mouse*, but do remember that most folks will refer to clicking the primary mouse button as "left click" and you'll have to reverse that in your mind.

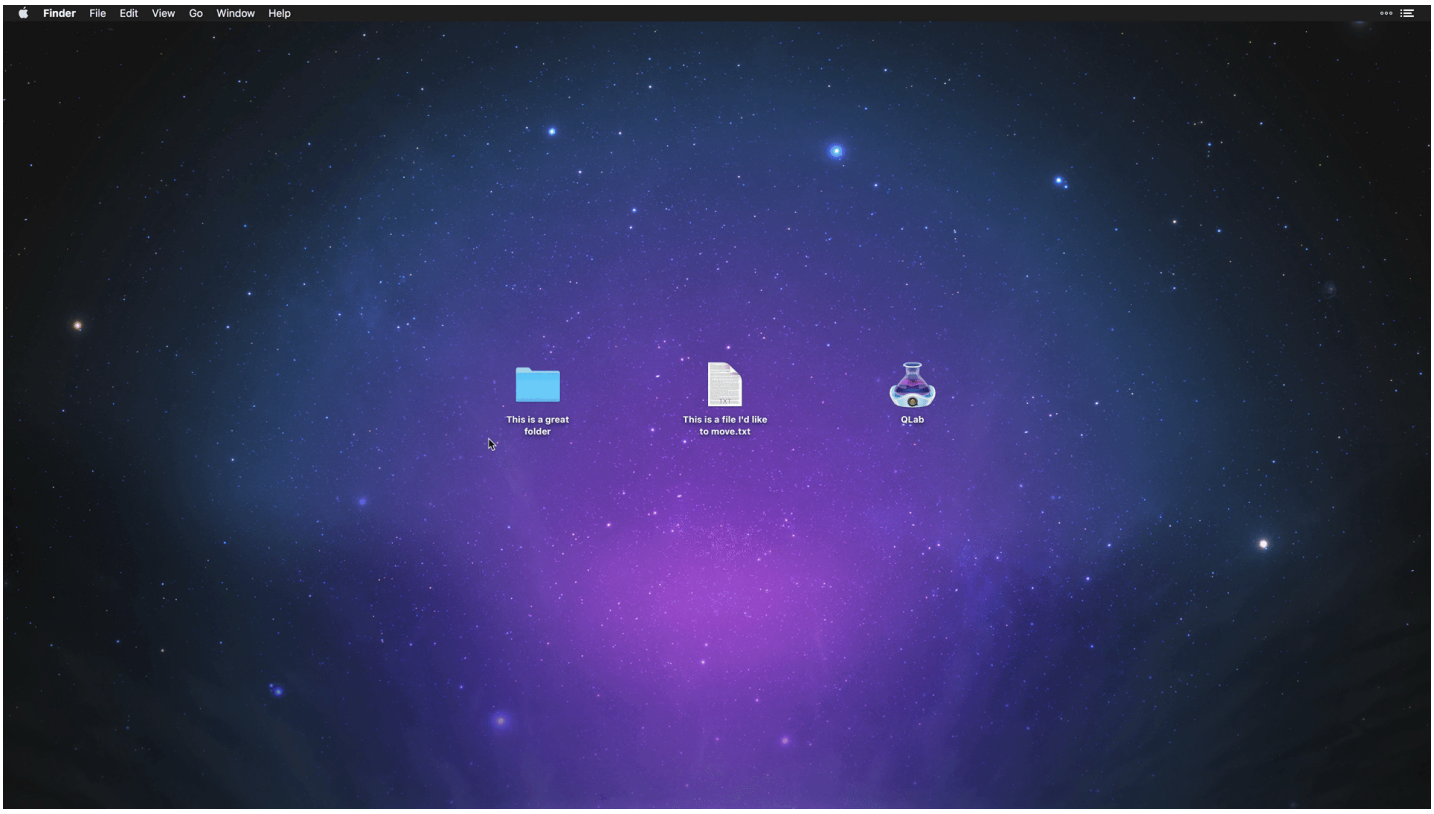
Click-drag-and-drop is a minor feat of acrobatics; you point at an item like a file, then click. Then, without letting go of the click, *drag* the mouse across the screen and notice that the item you clicked on comes with you. When you arrive at a destination, you release the click and thus *drop* the thing you've been dragging.



This is how you put a file into a folder, for example. In QLab, this is a very important operation because it's the simplest way to target a cue or place a cue inside a [ ] Group.

Menus

At the top of the screen is a thin area which stretches the full width of the screen. This area is called the *Menu Bar* and it contains a number of *menus* which vary depending on context. Menu are so named because, like their real-world counterparts, they contain a list of options, loosely grouped by category, from which you can make a selection. What happens next depends on the menu item chosen, and possibly also depends on what item or items you have selected.



- The **Apple** menu is always the left-most menu and should be available at all times. It gives you access to basic computer-wide tools and functions, like accessing System Preferences or shutting down the Mac.
- The **Application** menu is always next, and it displays the name of the application that you're currently using. When you're not using an application, this menu is named *Finder* because the Finder is the name of the always-on application that you use to interact with your Mac.
- The **File** menu is almost always next. It usually contains items relevant to opening, closing, and saving files.
- The **Edit** menu is almost always next. It's home to *undo*, *cut*, *copy*, *paste*, and often other items pertinent to the minute-to-minute work you're doing in a file.

Other menus vary widely from application to application, but there are common conventions which most applications try to adhere to and which, hopefully, make it easy to move between different applications as you work.

Further Reading

While this primer is deliberately brief, the overall subject of "how to use a Mac" is both broad and deep. If you're interested in more, the tutorial materials that Apple has made available on their website are excellent, and cover a wide range of topics.

As always, we encourage you to contact support@figure53.com with any questions, large or small.

Basic Networking

The purpose of this document is to provide a basic understanding of networking as it is used in the theater industry. While a fair amount of background is needed to really get up to speed, the goal here is to get you to the relevant parts as quickly as possible. The topic of computer networking is both broad and deep, and this document will omit some details that are not directly relevant in the interest of getting your show up on its feet as quickly as possible. Please assume that any such omissions are both deliberate and considered.

Understanding IP Addresses

Every device connected to the internet (or to a local network which *could* connect to the internet) is required to have a unique address called an IP[1] address. There are, confusingly, several different types of IP addresses and as of the writing of this document; the world is engaged in an excruciatingly slow-motion crossfade from IPv4 to IPv6. Internet experts have been declaring for many years that IPv4 is dangerously close to collapse and IPv6 is just about to become the dominant system of addressing. While it is technically true that there are no more unallocated IPv4 addresses, it's also true that IPv4 remains totally functional for most folks in most situations. So, this document will focus on IPv4 addresses and how to use them.

IPv4 addresses consist of four numbers, each between 0 and 255, separated by periods. Each of these numbers is referred to as a "octet". As an example, consider the address `142.250.217.78`, which is the address for google.com. If you type or paste that address into a web browser, you will find yourself looking at the familiar Google search page.

Just like physical addresses, IP addresses represent a hierarchy. Consider this address of a more familiar sort:

```
Sherlock Holmes
221b Baker Street
London
United Kingdom
```

If you read that address from bottom to top, each segment of the address represents a progressively more specific area, with progressively fewer potential addressees. The United Kingdom contains about 69 million people, London contains about 9 million, Baker Street contains around 165 people, 221b contains maybe eight people, and exactly one of them is Sherlock Holmes[2].

IP addresses work much the same way: there are 16.7 million addresses which start with `142`, then 65,534 addresses which start with `142.250`, then 254 addresses that start with `142.250.217`, and exactly one internet-connected device whose address is `142.250.217.78`, and that internet-connected device is a computer which hosts www.google.com.

> Internet-savvy readers will note that the websites of very large companies are actually hosted on multiple computers spread around the world, so that whenever a person browses to that website, some clever software directs them to a server that is physically close by. In those cases, the IP address points not to the web server, but to another computer which does the calculating and re-routing. There are a lot of layers here, and it can get very complicated very quickly. Mostly, if you're just trying to make a network in your theater, don't worry about it.

Large organizations have the rights to certain blocks of addresses. Examples include Apple (17.x.x.x), the US Postal Service (56.x.x.x), and CERN (128.142.x.x). This isn't specifically important, per se, but it is useful to help understand that in many cases, the IP address of a device can tell you something about the network it belongs to, the organization that is administering that network, and its location in real space.

Understanding Networking Hardware

Putting a full glossary of networking hardware here would be an act of cruelty. Here are the really essential bits:

Network switch. A device that lets you connect several devices together with networking cables. Switches have a small amount of "smarts" in that they can be configured to prioritize certain types of network traffic, restrict traffic speed in certain situations, etc. But in general, they are just there to pass data between other devices.

Network router. A router is the focal point of a network. All traffic which comes into the network from outside, or goes out of the network from inside, must go through the router. The router "knows" what's connected to it, and can direct traffic accordingly. Most routers have DHCP servers built in (discussed below.) Many routers have additional features. Many routers have switches built in. If your router has more than two ethernet ports on it, it has a switch built in.

Wifi access point. A radio transceiver which sends and receives network traffic wirelessly. Wifi can be thought of as simply ethernet-without-wires. Many WiFi access points are built into routers, but not all of them, and it's important to know which you're using. A WiFi access point with no built-in router has very little "smarts" and generally can be treated as a simple adapter that converts network traffic from electricity in a cable to radio waves in the air.

CAT-5, CAT-5e, CAT-6, CAT-6e, CAT-7. These are types of cables that are used in ethernet networking. They differ only in the speed and quantity of data that they can handle. You will often hear them colloquially called ethernet cables, which is technically incorrect in the same way that calling XLR-3 "microphone cable" is technically incorrect: everyone does it and nobody cares, and it's only important to make the distinction occasionally.

What Does This Have To Do With Theater?

At this point, you may be wondering why you're reading this arcane, mind-numbing nonsense when all you want to do is send OSC messages from QLab to your Ion so that your cues sync up really nicely. We absolutely promise you that all of this information is relevant to theater, but it's a very slow burn to get there.

This is a great moment to take a break, grab a drink, and stretch.

Too Few Addresses

You may have done a little recreational multiplication so far, and concluded that IPv4 only has about 4.3 billion addresses (256 × 256 × 256 × 256). You are quite right, and actually only about 3.7 billion of them are useable for boring technical reasons. Given that there are clearly more internet-connected devices in the world than that, what gives?

The answer lies in this trick: many devices which are connected to the internet are not, in fact, actually directly connected to the internet, thanks to something called **Network Address Translation**, or NAT.

When you go to a coffee shop and connect to their free WiFi network, your laptop or phone does not connect directly to the internet. Instead, it uses a protocol called DHCP (discussed below) to receive an IP address from the WiFi router. This address almost always starts with `192.168`, and this is a clue that will be useful later on.

What's going on here is that the WiFi router is connected (probably) straight to the coffee shop's cable modem or other source of internet, and has a normal public IP address. It's also running a bit of software called a DHCP server which assigns IP addresses to devices that connect to it. All those addresses are from a special pool, `192.168.x.x`, and the devices which have those addresses are essentially invisible to the rest of the internet. In order to communicate with the internet, devices on the network use the WiFi router as a sort of personal assistant. Your laptop asks the coffee shop router things like "hey, can I see google.com please?" and then the router goes and fetches whatever data google.com is serving up, and passes it back to your laptop, *translating* the request from the *address* of your laptop on the internal private network to the public internet and back again.

Think of it like a switchboard in a large office building. Calls within the building can go straight from office to office, but calls into and out of the building must go through the switchboard.

These private addresses are where we start to get really relevant to theatrical networking.

Local Networks and Private Networks

A local network is a collection of devices which use a common IP address scheme and which are generally physically close together. A private network is a local network that is isolated from the internet as a whole.

The coffee shop WiFi network discussed above is isolated from the internet by virtue of NAT; the router can be "seen" by the internet, but the computers and phones on the local network cannot. You could also isolate a network from the internet by simply not connecting it to the internet at all, and that is typical of networks used in theater.

There are three sets of IP addresses which are reserved for private networks:

- **192.168.0.0** through **192.168.255.255** for a total of 65,536 addresses.
- **172.16.0.0** through **172.31.255.255** for a total of 1,048,576 addresses.
- **10.0.0.0** through **10.255.255.255** for a total of 16,777,216 addresses.

When you set up a private network, you're generally going to want to use one of these three address schemes. If your private network never, ever connects to the internet, there's nothing stopping you from using any addresses you like, but it's good form to use these three schemes only.

This begs the question: what does it mean to choose an IP address scheme, practically speaking? If you've read this far, then you know that the answer to that question can only come after another little lesson.

Understanding DHCP vs. Static Addresses

In the context of QLab and a private network in a theater, there are two ways to assign IP addresses to devices on the network.

DHCP, which stands for Dynamic Host Control Protocol, is a system whereby any device that connects to a network announces itself as a new member of the network, and then the DHCP server says, "oh hello, here is your IP address, please hang onto it for four hours, then ask for a new one." This is very convenient, since it means that a given network doesn't need to have enough available addresses for every device that will ever connect to it, only enough for the likely number of devices that are connected at once. If device X disconnects from the network and then device Y connects after a while, the DHCP server may "recycle" the address that device X had and give it to device Y.

When devices on a network are set to use DHCP and there is no DHCP server to be found, they assign themselves an address that starts with `169.254`[3]. For some applications (notably Dante), this works perfectly well. For others, seeing that a device has a `169.254` address is a useful troubleshooting tool.

Static, in the context of IP addresses, means that the device's address is preemptively assigned and manually entered in. The most common real-world example of a static IP is the example above with Google's website. The address `172.217.11.174` has been assigned by Google to its web server. Google has the authority to assign all addresses beginning with `172.217`, and so they configure the computer hosting their web site to use `172.217.11.174`. Whenever that computer is online (which should be all the time), that address corresponds directly to google.com. A static IP address can be assumed to "belong" to the device that it's been assigned to, and should never be used for another device without first removing it from the first device.

> Again, this is glossing over a lot of irrelevant detail. Please don't consider this explanation to be authoritative on the topic of how IP addresses belonging to web services operate in general.

Choosing Between DHCP and Static Addresses

When you set up a private network, you'll need to choose whether your network will use DHCP, static addresses, or a mixture of both. How do you decide?

If your network is being used for any or all of the following activity, you probably don't need to use static IPs. Using DHCP will just be simpler and easier:

- File sharing
- Screen sharing
- QLab Remote
- Dante audio
- AirBeam video
- Art-net

On the other hand, if you're using the network to send OSC messages, it will be preferable to use static IP addresses because OSC messages need to know where they're going, and a static IP address is the simplest way to ensure that.

DHCP

If you decide to use DHCP, you'll need to set one device on your network to act as a DHCP server. Nearly every WiFi or ethernet router does this, but also you can run a DHCP server on a Mac or PC, or even on an ETC console. All that's important is that your network includes exactly one DHCP server, no more and no less.

There's no way for this document to be all-inclusive on the topic of setting up DHCP serving, so you'll need to consult the manual of whatever device or software you're using.

Static IPs

On a network which uses static IP address, you may not need a router. Or, if you do have one, its DHCP server will not be involved in the process of assigning IP addresses. Therefore, you need to assign them. Make a list of all the devices that will be connected to your network, and then give each of them an IP address. Skip addresses ending in 0, 1, and 255 for reasons that will be explained below.

An example list might look like this:

Device	Address
Sound Designer laptop	10.10.0.5
Associate laptop	10.10.0.6
QLab main	10.10.0.10
QLab backup	10.10.0.11
Meyer GALAXY primary	10.10.0.20
Meyer GALAXY secondary	10.10.0.21
ION	10.10.0.100
LD tech table Nomad	10.10.0.101

The choice of address for each device is not important, but each device must have a unique address. Here, devices that logically go together have sequential addresses, but that's for the benefit of the humans reading the list, not because the machines care about it. Do note, however, that for all these addresses, the first three octets are the same. More on that topic below.

Network setup on Macs

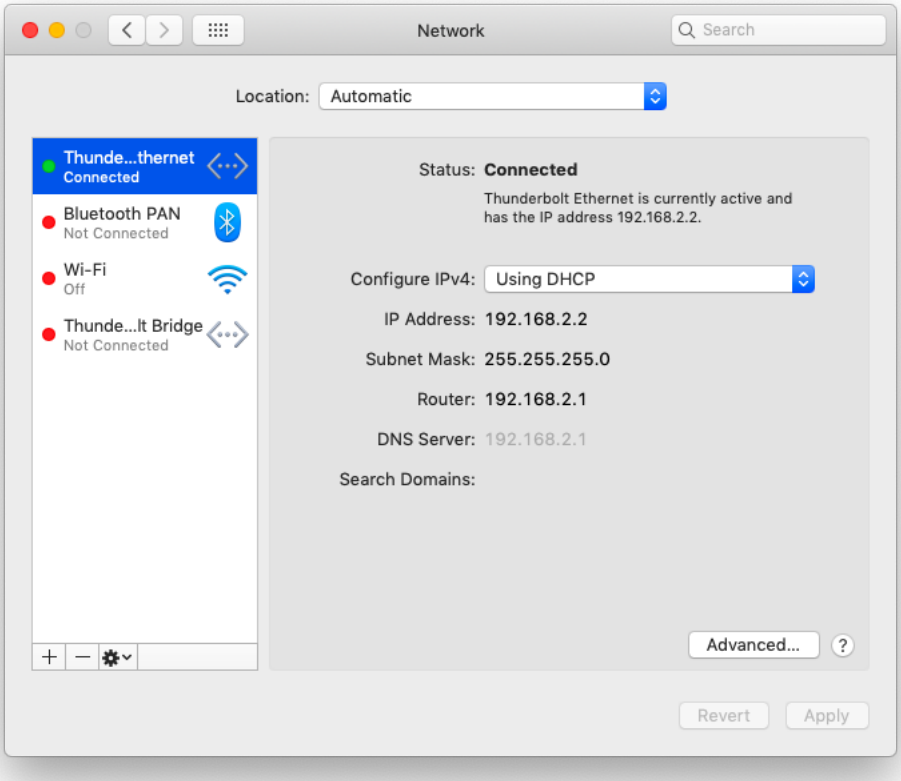
Open *System Preferences* → *Network* to adjust network settings on your Mac.

Ethernet

Ethernet is preferred over WiFi for a number of reasons:

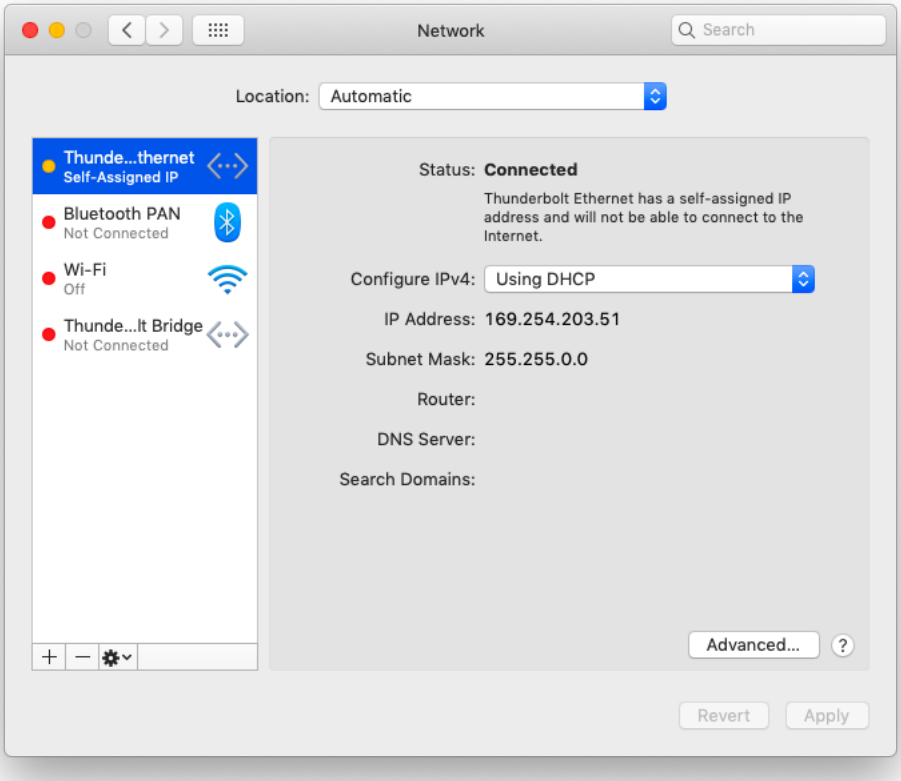
- It's dramatically more reliable
- It's dramatically faster
- It's harder for an interloper to connect to
- There are fewer pieces to configure
- It's cheaper to do well
- Dante only works over ethernet

If you're using ethernet and DHCP, the Network preference pane will look like this:



There's a green dot on the left next to "Thunderbolt Ethernet" (it's truncated in this screen shot; it says "Thunderbolt Ethernet" because this Mac is using a Thunderbolt adapter to connect to ethernet.) On the right it says **Connected**, and beneath that you can see that IPv4 is using DHCP, and you see the IP address and some other details discussed below. All the details are listed in a non-editable form, because DHCP is designed to work automatically.

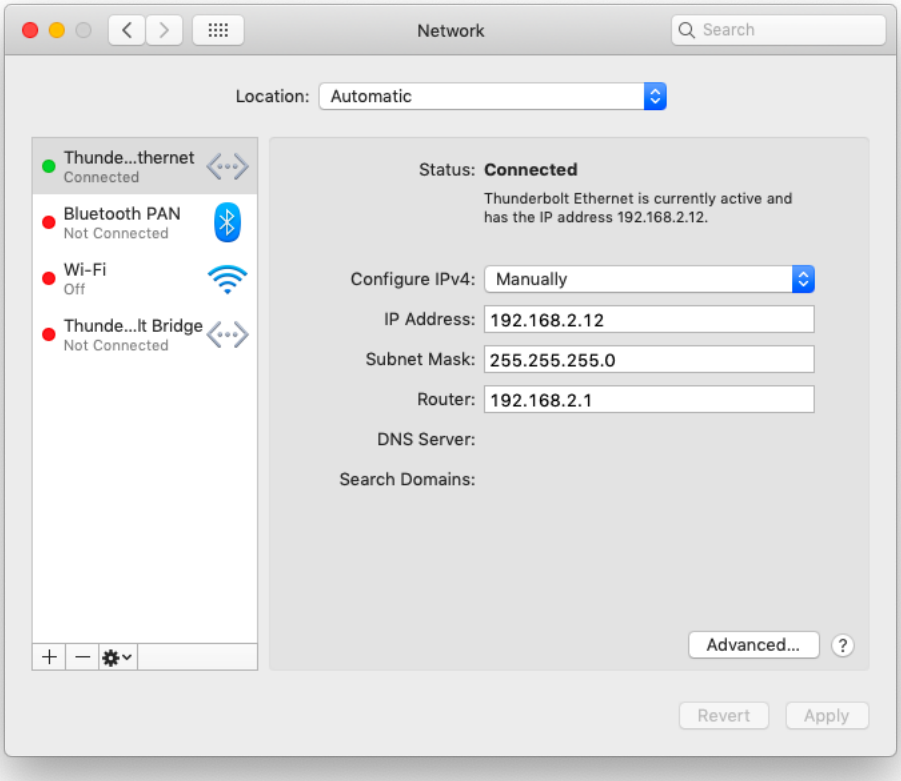
If your Mac is physically connected to your network but not working correctly, the window will look like this:



The green dot is instead yellow, and the IP address begins with `169.254`. The most common explanation for this scenario is that the DHCP server isn't working or cannot be reached. The first thing to try is rebooting the device that's doing the DHCP serving.

> If you're using the DHCP server built into an ETC console, it's important to hook up your physical network infrastructure before turning the console on. If an ETC console does not detect that it's connected to an ethernet network while it's booting up, it will deactivate its ethernet port.

If you're using a static IP address, the window will look like this:

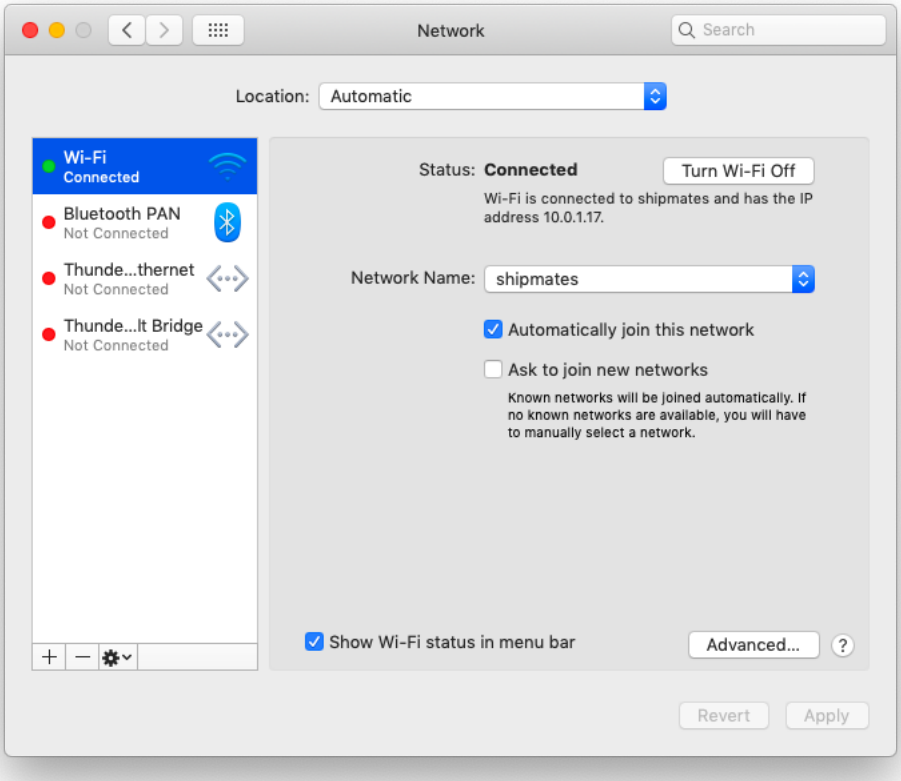


In this case, the green dot only indicates that you've entered settings which are technically valid. It does not necessarily mean that you entered the correct settings for your network, or that the Mac is successfully communicating with other devices on the network.

A discussion on entering static IP details can be found below.

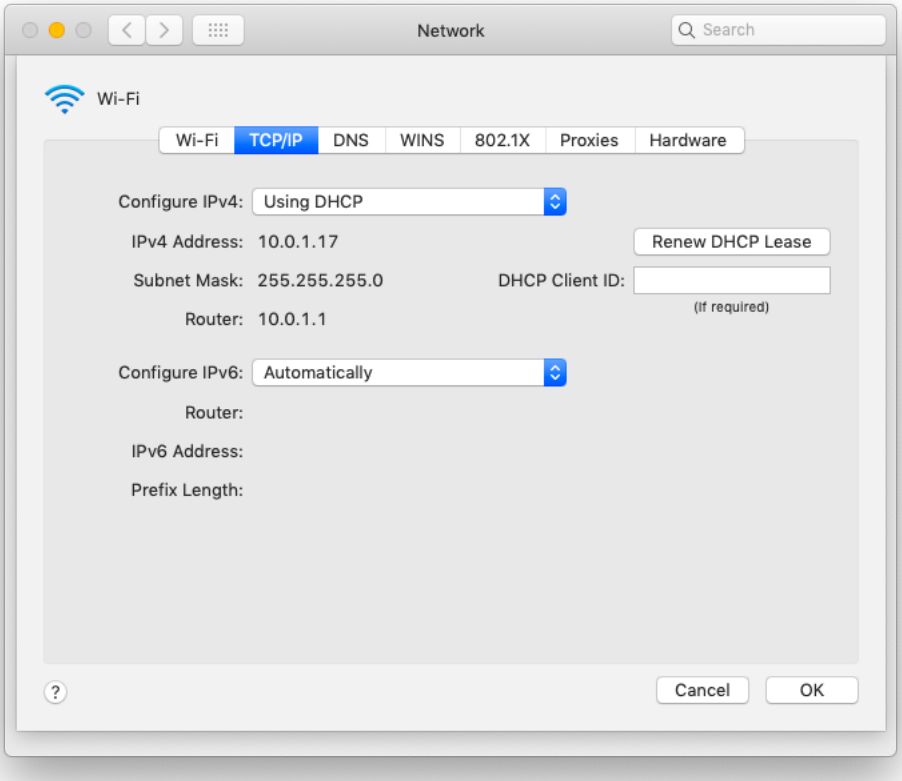
Wifi

If you're using WiFi to connect your Mac to the network, this is what you should see if everything is working:



The green dot next to "Wi-Fi"[4] on the left side is the first clue, and the small explanatory text on the right includes the IP address that the DHCP server has assigned.

If you click the *Advanced* button at the bottom right, you can see and change the IP address settings:



The *TCP/IP* tab shows you how your IPv4 address is being selected, DHCP in this case, then below that you see the address that's been assigned, the subnet mask which we'll discuss in a bit, and the IP address of the router that's doing the DHCP serving.

As long as you're connected to the right WiFi network and your router (or other DHCP server) is configured correctly, this is what you should see, and there's nothing to adjust or configure. Do not concern yourself with all the other tabs and buttons.

If you're using static IPs with a WiFi network, this is where you choose "Manually" from the *Configure IPv4* drop-down menu.

After making any changes to the Network preference pane, it's necessary to click on the *Apply* button in the lower right corner for those changes to take effect.

Setting Up Static IPs

There are three settings that you need to enter when your network is using static IP addresses: the **IP address** itself, the **subnet mask** and the **router address.**

IP Address

Referring to your list above, enter in the IP address that you've planned for this device. Remember these rules:

- Each octet can be in a range between `1` and `254`, inclusive. `0` and `255` are reserved for special use.
- Dots (periods) separate the four octets.
- There are customary address ranges for private networks which you should stick to using unless you have a good reason not to.
- The last octet usually shouldn't be `1` because that's usually the address of a router. See below.

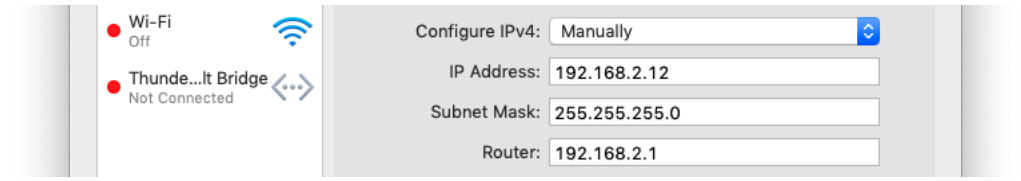
Subnet Mask

A subnet is, unsurprisingly, a subsection of a network. Devices which are on the same subnet are able to communicate to each other directly, without sending traffic to the router "above" them. Putting devices on the same subnet is a combination of the physical setup (there must be a physical path for

network traffic to move between them), IP address setup (the must have IP addresses in the same scheme), and subnet mask.

Subnet masks look a lot like IP addresses; four octets separated by periods. There are fancy ways to use the subnet mask, but they're not relevant to theatrical networking. The basic way to use a subnet mask involves using only two values for each octet: `255` or `0`.

Remember the hierarchical nature of the IP address? In the subnet mask, each octet is effectively telling the device whether that level of the hierarchy is part of its subnet. If the value is `255` then that means "this level of the hierarchy is not on your subnet", and if the value is `0`, that that means "this level *is* on your subnet."



In the screen shot above, the Mac's IP address is set to `192.168.2.12` and the subnet mask is set to `255.255.255.0`. This Mac is therefore able to send data straight to any device with an IP address beginning `192.168.2.`.

It cannot send data straight to `192.168.50.12`, because the third octet does not match and the subnet mask for that octet is `255`, and `255` means "has to match." That communication must go through a router that is available to both subnets.

A network full of devices configured like this can therefore contain a maximum of 254 devices (since `0` and `255` are not permissible in regular IP addresses.)

If the subnet mask were instead set to `255.255.0.0`, the Mac would be able to send data straight to any device whose address begins with `192.168`, allowing for a network with 64,516 devices on it.

The Mac would not be able to send data straight to `192.40.8.19` because the second octet does not match.

Router Address

As discussed above, a router behaves as the central traffic authority of a network, and behaves somewhat like a post office, dispatching incoming data to the appropriate neighborhoods and passing outgoing data to the outside world (assuming the network is connected to the outside world.)

All devices on a network must know the IP address of their router in order to properly communicate with the router. While a few network protocols can sidestep this, most cannot and so it's crucial to enter the IP address of the network's router identically on all devices on the network.

If your network is using only static IP addresses and is not connected to the internet, an actual router is often not required. In this situation, the correct thing to do is enter the IP address of an *imaginary router* as though it were really there. Odd as this may sound, it's a simple solution to a lot of vague networking problems.

Routers are traditionally given IP addresses that end in `.1` although this is not technically required. It is our advice to stick with tradition here.

So, if you do not have a real router, invent an IP address that is reachable by your other devices (according to their subnet masks) and ending in `1` and enter it on all your devices.

Some Specifics About ETC Consoles

If you're networking together with the lighting department, chances are good that they're using ETC gear, which comes with some very specific default IP settings:

Device	Address
Congo	10.101.80.x
Congo jr	10.101.81.x
ColorSource AV	10.101.20.x
Element	10.101.97.x
Eos	10.101.90.x

Device	Address
Eos Ti	10.101.92.x
ETCNet2 nodes	10.101.50.x
ETCNet3 nodes	10.101.50.x
Gio	10.101.91.x
Gio @5	10.101.91.x
Ion	10.101.100.x
Ion Xe	10.101.100.x
Ion RPU	10.101.96.x
Net3 RVI	10.101.85.x
Net3 RVI3	10.101.86.x
Sensor+	10.101.101.x

Also:

- Subnet mask: `255.255.0.0`
- Router: `10.101.1.1`

Since adjusting these settings is more time consuming on the ETC gear than on a Mac, it's often convenient and neighborly to follow their scheme.

Because ETC is so careful about their defaults, it's easy for you to work around their IP scheme and assign addresses to all your devices in a way which won't conflict. In short:

- Use an IP address scheme starting with `10.101`
- For the third octet of your addresses, use a number lower than `50`, but not `20`.
- For all devices that will communicate with the lighting network (i.e. the QLab mac), use a subnet of `255.255.0.0`
- For all devices that will *not* communicate with the lighting network, you can optionally use a more restrictive subnet mask of `255.255.255.0`, but you'll need to do some extra work to make sure that each device is either on the same subnet as the other devices it must communicate with, or is able to "see" a router that can see those other devices.

Closing Thoughts

As you can see, this is a very, very complex topic. This document barely scratches the surface of the entire topic of computer networking; we did not discuss physical networking considerations, software considerations, and we glossed over a lot of detail which is irrelevant to putting up a theatrical production, but which is nevertheless part of the general topic.

You are encouraged to use this document as a jumping-off point. The internet does a terrific job of documenting itself, and it is not difficult to find articles and tutorials which you can use to expand your knowledge of any of the topics discussed here.

As always, writing to support@figure53.com is the best and fastest way to get help using QLab.

---

1. **I**nternet **P**rotocol↩
2. Population figures according to Wikipedia as of April 2025.↩
3. This is referred to as a "link-local" address. That's not super important to remember, but you might come across it.↩
4. Reasonable people may differ on the spelling of wifi, WiFi, or Wi-Fi.↩

Understanding USB-C

Modern Macs make extensive use of the USB-C connector for interconnecting with external hardware like input devices, storage devices, and displays. This connector has a lot going for it; it's versatile, durable, symmetrical, and small. Unfortunately, with it comes a great deal of confusion and difficulty. The goal of this tutorial is to provide you with a full working knowledge of the USB-C connector and everything it can do.

This tutorial was updated in November of 2024.

Connectors and Protocols

Prior to the invention of USB-C, it was generally true that each type of connector found on a computer was used for one thing only, so you could just look at the connector and know what it was for. This wasn't always true, but it was true enough to prevent mass confusion. The USB-C connector broke from this tradition rather sharply.

Before USB-C, the term "USB" referred to both a set of physical connector types *and* to the type of signal that was being used by those connectors. USB-C, on the other hand, is only the name of a connector. There is no protocol or signal called "USB-C." Had they simply named it the "C" connector, or really any other name, a considerable amount of confusion could have been prevented.

On top of that, the USB-C connector is used for more than one communication protocol, unlike most of its predecessors, and the protocols themselves *contain other protocols*, like digital Matryoshka dolls.

To un-nest the dolls and make sense of things, we'll discuss the physical design of the USB-C connector, and the three roles that it fills: **USB**, **Thunderbolt**, and **USB Power Delivery**.

The USB-C Connector design

USB-C connectors have 24 electrical contacts, called "pins." These pins are connected to wires within the cable, although not all connections are mandatory:

- Four **ground** pins which connect to a pair of wires. These are mandatory.
- Four **bus power** pins which connect to a pair of wires. These are mandatory.
- Four **low-speed data** pins which connect to a pair of wires. These are mandatory.
- Eight **high-speed data** pins which connect to four pairs of separately shielded wires. These are optional.
- Two **configuration channel** pins which connect to two wires. These are optional.
- Two **sideband use** pins which connect to two wires. These are optional.

Any USB-C cable which includes all optional connections is considered a "full-featured" cable. Cables that do not include all optional connections do not have a special name, they are just *not* full-featured cables.

Passive and Active Cables

USB-C cables can be either passive, which is what most cables are, or *active*. Simply put, active cables have a little microchip inside their connectors which draws a little power from the device they're connected to and which amplify, re-time, or otherwise correct the signals that pass through them.

Passive USB-C cables can achieve full speed only at lengths below 0.8 meters or 2.6 feet[1]. Active USB-C cables can achieve full speed up to 2.0 meters or 6.6 feet.

USB

USB, which stands for Universal Serial Bus, is a set of closely-related standards that are all, basically, ways to send data back and forth between a computer and a device connected to that computer. Ever since it was invented in the late 90s, USB has always embraced the concept of different *classes* of device, because the essential premise of USB is "you just plug it in, and the machines will figure out what's supposed to happen next." This is a worthy goal, and a frankly ambitious one. It works well most of the time; you can plug a mouse, a printer, and a synthesizer into the same computer at the same time, using the exact same type of cable, and pretty much everything works without a hassle.

When the USB-C connector is used for USB, it's using either USB 3.2 or USB4 (note the deliberately missing space between "USB" and "4"... we don't make these rules.) These two versions of the USB protocol are similar but not identical.

USB 3.2

USB 3.2 was released in 2017 and supersedes all previous versions of USB. It defined four data transfer modes with confusing names:

- **USB 3.2 Gen 1**, called *SuperSpeed*, is a re-naming of the USB 3.1 Gen 1 protocol, which was itself a re-naming of the USB 3.0 protocol. It transfers data at 5 Gbps[2].
- **USB 3.2 Gen 2**, called *SuperSpeed+*, is a re-naming of the USB 3.1 Gen 2 protocol. It transfers data at 10 Gbps.
- **USB 3.2 Gen 1x2**, also called *SuperSpeed+* also transfers data at 10 Gbps, but uses a different technical process than USB 3.2 Gen 2. Since the human-observable behavior is the same, the "marketing" name is the same.
- **USB 3.2 Gen 2x2**, inexplicably *also* called *SuperSpeed+* transfers data at 20 Gbps.

It may be useful to know that while USB 3.2 Gen 1 and USB 3.2 Gen 2 can be used with various types of USB connectors, Gen 1x2 and Gen 2x2 can only be used with USB-C connectors.

Modes with the "x2" in their name use all four high-speed data connections, which means you'll usually need a full-featured USB-C cable to use them. Modes without the "x2" use only two of the high-speed data connections.

USB4

USB4 was released in 2019 and is much more powerful than USB 3.2, but with great power comes great confusion. The best way to wrap your head around USB4 is to think about it from the perspective of someone planning to build a device that uses USB4. Here is a list of things that person would learn by reading the USB4 specifications:

- It is based on Thunderbolt 3, but full compatibility with Thunderbolt 3 devices is optional.
- It supports data transfer at a minimum of 20 Gbps.
- It allows "tunneling" other protocols (discussed below), some of which must be supported and some of which are optional.
- It allows "alternate modes" (discussed below), some of which must be supported and some of which are optional.
- It can be single-lane per direction (requiring use of two of the high-speed data pairs) or dual-lane per direction (requiring all four high-speed data pairs.)
- It always and only uses USB-C connectors.

There are two data transfer modes that are mandatory for all USB4 devices:

- **Legacy USB** which has a maximum speed of 480 Mbps. Because of this, you can theoretically plug *any* USB device ever made into a USB4 port on a computer, and as long as the computer has the appropriate software to talk to the device, it will work.
- **Tunneled USB 3.2 Gen 2.** Again, we'll get to tunneling in a minute.

There are at least twelve other modes that USB4 devices *may* support, but don't have to:

- **USB4 20 Gbps**
- **USB4 40 Gbps**
- **Host-to-Host communication**
- **Tunneled USB 3.2 Gen 2x2**
- **Tunneled DisplayPort**
- **Tunneled PCI Express**
- **DisplayPort Alternate Mode**
- **Mobile High-Definition Link Alternate Mode**
- **HDMI Alternate Mode**
- **Thunderbolt Alternate Mode**
- **VirtualLink Alternate Mode**

USB4 hosts (i.e. computers) must support at least Host-to-Host communication and DisplayPort Alternate Mode. USB4 docks and hubs must support at least Tunneled PCI Express and Thunderbolt Alternate Mode.

Clear as mud, right?

Tunnels and Alternates

Tunneling, in this context, is the nesting protocol-within-protocol that was mentioned above. When one protocol allows tunneling of another, the data that belongs to the "inner" protocol is encapsulated into data packets belonging to the "outer" protocol and sent along with any other data that the outer protocol is sending. When the encapsulated packets get to the other end of the connection, they are de-capsulated and reassembled to re-form the stream of data that entered the tunnel. Tunneling can increase the *average* speed of transmission because it does not require switching the connection back and forth between modes, although tunneling usually has a lower *absolute* speed since it necessarily does not use the full bandwidth of the connection.

Alternate modes, by contrast, achieve the same end result as tunneling (allowing more than one type of communication along a single connection) but instead of the inner/outer protocol concept, they electrically switch one or more of the high-speed data pairs for dedicated use by the alternate mode. As a result, using an alternate mode necessarily reduces the amount of bandwidth available for a regular data connection, since some of that bandwidth is being dedicated to the alternate mode connection. In exchange, they have the highest possible potential speed.

Alternate Modes

The USB specification allows for any number of alternate modes in theory, but there are only five that exist as of the writing of this manual:

- **DisplayPort Alternate Mode** which can support DisplayPort 1.2 or higher. This mode can use one, two, or all four of the high-speed data pairs depending upon the version of DisplayPort being used and the resolution and frame rate of the display being connected. If all four pairs are used, then only the low-speed (USB 2-speed) data pair remains available for non-video data transfer. This mode also uses the sideband pins for the DisplayPort Aux channel, which transmits EDID and other metadata associated with the DisplayPort connection. Most good USB-C to HDMI adapters use DisplayPort Alternate Mode together with the necessary hardware components to convert the DisplayPort signal to HDMI.
- **Mobile High-Definition Link (MHL) Alternate Mode** which is a kind of awkward middle child of multimedia that can be thought of as basically "HDMI using different connectors." It behaves very similarly to DisplayPort Alternate Mode, using one, two, or four high-speed data pairs plus the two sideband pins, but the technical details of the video signal that it uses are different.
- **HDMI Alternate Mode** which behaves very similarly to the above two modes. HDMI and MHL use the same video signal but different metadata and sideband signals. HDMI Alternate Mode also co-opts one of the bus power pins to provide 5V power which is part of the HDMI spec.[3]
- **Thunderbolt Alternate Mode** which allows the USB-C port to use USB or Thunderbolt as needed. Additionally, a dock or hub which supports this mode can provide up to three downstream Thunderbolt 3 ports, each of which independently switches data modes and alternate modes based on the devices connected to that port. Needless to say, a dock or hub of this kind must be connected upstream to a USB-C device that supports this mode, and must use a full-featured cable or a Thunderbolt-branded cable.
- **VirtualLink Alternate Mode** which was developed to support virtual reality headsets but which, like most virtual reality technology, failed to catch on and is effectively dead. As of August 2020, the consortium that developed this standard disbanded. This mode is being listed here only in the interest of completeness, and you should not expect to ever need to know anything about it.

USB4 Version 2.0

As though the naming scheme so far wasn't confusing enough, USB4 Version 2.0 was released in October 2022. As of this writing, there are no Macs which support USB4 version 2.0, but if you've read this far into this tutorial, you're likely the sort of person who might enjoy some advance warning.

What's New

- **80 Gbps** data transfer. This uses all four data lanes at a speed of 40 Gbps per lane. That adds up to 80 Gbps total per direction.
- **120 Gbps** data transfer. This is optional and uses three data lanes in one direction, leaving 40 Gbps of bandwidth available for data transfer in the other direction.
- **Tunneled USB 3.2 Gen 2x2** (which provides 20 Gbps data transfer) is now mandatory, not optional.
- **Tunneled PCIe** now supports PCIe 4.0.
- **HDMI Alternate Mode**, as discussed in footnote #2, no longer exists because nobody was using it.
- **DisplayPort Alternate Mode** now supports the DisplayPort 2.1 spec at minimum.[4]

Device Compatibility

USB4 Version 2.0 is completely backwards-compatible with USB4. All USB4 devices will be able to interconnect with all USB4 Version 2.0 devices, though of course they will only be able to communicate using original USB4 protocols and modes.

Cable Compatibility

Interestingly, any full-featured USB-C cable will fully support USB4 Version 2.0. Non-full-featured cables will work, of course, but will be restricted to lower speeds just like with USB4.

A minor update to the USB-C connector design requirements was also released at the same time as USB4 Version 2.0. The changes are all internal and relate to improved longevity.

Thunderbolt

Thunderbolt is similar to USB on its face; it's a way to connect devices to a computer. Thunderbolt has a fundamental conceptual difference, however, which might feel technical and vague at first, but which is essential to unraveling the mysteries of the USB-C connector.

As you may be expecting at this point, Thunderbolt is not really a protocol in and of itself; rather, it's a set of hardware that combines two preexisting protocols plus a supply of electrical power into a single cable. The two protocols that it combines are **PCI Express** and **DisplayPort** which can each be described briefly, mercifully:

- **PCI Express (or PCIe)** is a very old but well-maintained standard for passing data to and from a computer's CPU at very high speeds. Cards in a desktop computer usually use PCIe, and the fundamental workings of PCIe are built into most modern CPUs at the hardware level. The main limitation of PCIe is that it only works over very short distances.
- **DisplayPort** is a modern digital video interconnection standard which provides a direct connection between a display and the GPU inside a computer, like VGA and DVI did before it. You can think of it as "HDMI, but specifically for computers." HDMI is more often seen built into computers because HDMI cables are cheap and ubiquitous, and HDMI inputs are found on TV and projectors everywhere, so the DisplayPort connector has become somewhat less common. Nevertheless, the digital signal used by that connector lives on within USB4 and Thunderbolt.

Thunderbolt allows high-bandwidth devices which previously could only be built as PCIe cards and installed in desktop computers to instead be built as portable stand-alone devices powered by the same cable that provides the data connection.

Thunderbolt 1 and 2 used the Mini DisplayPort connector. Thunderbolt 3, 4, and 5 use the USB-C connector only.

Thunderbolt 3

The Thunderbolt 3 specification requires the following:

- Up to four lanes of PCIe 3.0 data transfer totaling a maximum bandwidth of 32.4 Gbps
- Up to eight lanes of DisplayPort 1.2 or 1.4 data totaling a maximum bandwidth of 32.4 Gbps
- A total maximum bandwidth of 40 Gbps
- At least 15 watts of power delivery downstream from host to devices
- Prioritization of video data over general data. This means that if a mixture of displays and other devices are connected to the same Thunderbolt controller, the displays are allowed to take as much bandwidth as they need, and the other devices only get whatever is left over.

Thunderbolt ports on a device are connected to the internal circuitry of that device through a Thunderbolt controller chip. Thunderbolt 3 controllers come in three forms:

- a double port controller, which provides two physical USB-C ports that share four PCIe 3.0 lanes.
- a single port controller, which provides a single USB-C port with its own four PCIe 3.0 lanes.
- a low power controller, which provides a single USB-C port that uses two PCIe 3.0 lanes, and which therefore has half the amount of data transfer capacity.

Thunderbolt 4

Thunderbolt 4 has the same maximum total bandwidth as Thunderbolt 3, 40 Gbps, but requires four lanes of PCIe 3.0 (rather than allowing either two lanes or four) and support for either DisplayPort 1.4 or DisplayPort 2.0. This raises both the minimum and maximum available video performance of a Thunderbolt connection, although the specifics of that performance are too complex to summarize and range from two 4K@60 Hz displays to one 16K@60 Hz display.

Thunderbolt 4 also makes it possible to have a hub or dock which has more than one “downstream” Thunderbolt port. Previously, the only way to connect multiple devices to a single Thunderbolt port on a computer was to daisy-chain those devices in a line. With Thunderbolt 4, it’s possible to create a device that has a total of four Thunderbolt 4 ports, one that connects to the host computer and three which can receive connections from devices.

Thunderbolt 5

Thunderbolt 5 is a step up in terms of basic capabilities. Thunderbolt 5 requires:

- Four lanes of PCIe 4.0 data transfer totaling a bandwidth of 64 Gbps
- Video transfer bandwidth of 120 Gbps
- A total maximum bandwidth of 160 Gbps
- USB4 version 2.0
- DisplayPort 2.1

The 160 Gbps total bandwidth comes in the form of four lanes, which are typically configured to allow 80 Gbps of data transfer in each direction. If a high-bandwidth display is connected, though, the host can flip one lane around to send 120 Gbps of data downstream to the display, and accept “only” 40 Gbps of data upstream from other devices on the Thunderbolt bus. This is optional within the USB4 version 2.0 spec; it is mandatory within the Thunderbolt 5 spec.

Thunderbolt 5 is fully backwards-compatible with Thunderbolt 4, Thunderbolt 3, USB4, and USB 3.

Through the use of new signaling technology, Thunderbolt 5 works on any full-featured USB-C cable up to 1 meter in length, and on some 2 meter cables as well.

Thunderbolt 5 was adopted by Apple starting with some M4 family-based Macs.

USB Power Delivery

Finally, USB-C connectors are used by the USB Power Delivery standard, which requires hosts to supply a minimum of 1.5 amps at 5 volts DC per physical port.

Hosts can optionally (of course there are options) supply higher amperages and voltages as well, all the way up to 5 amps at 48 volts (240 watts).

Devices can optionally serve as power supplies for hosts; this is rather inelegantly named “PC charging.” Thunderbolt 4 PC charging is at least 100 watts and at most 140 watts. Thunderbolt 5 PC charging is at least 140 watts and at most 240 watts.

The Power Delivery specification includes a system of signaling and physical wiring which prevents devices from receiving more power than they are designed for. This is accomplished by always supplying the base amperage and voltage, and only raising the amount of supplied power if the device requests it.

Accessory Modes

Separately from the three main uses of the USB-C connector, there are also two “accessory modes” which are supported directly by the USB-C hardware design and which do not involve software at all:

- **Debug Accessory Mode** is employed by using specific resistor values to connect the configuration channel pins to either the power pins or ground pins. This is generally only used for testing and development purposes.
- **Audio Adapter Accessory Mode** is employed by grounding both configuration channel pins. In this mode, no data connection is possible at all, and the low speed data and sideband pins are used for headphone and microphone connections. This is how small, inexpensive USB-C to headset adapters work.

USB4 and Thunderbolt

Any device which supports *all* optional USB4 functionality is, officially, a Thunderbolt 4 device. Since one of the USB4 options is Thunderbolt Alternate Mode, which is Thunderbolt 3, the definition of Thunderbolt 4 is therefore: Thunderbolt 3 + USB4.

Any device which supports all optional USB4 version 2.0 functionality is a Thunderbolt 5 devices.

You can think of Thunderbolt as being simply USB4 with all the optional bits included.

One of the stated goals of the convergence of USB4 and Thunderbolt was to reduce confusion. It is left to the reader to decide whether this goal has been met, or even approached.

Conclusion

If your head is spinning by this point, you are not alone. One connector is used for multiple protocols, each of which contains other protocols, many of which include multiple different ways to achieve the exact same results using either identical protocols implemented in different ways, or different protocols whose differences are necessarily invisible to the end user. It is mind-boggling. In the hopes of assembling some order out of this chaos, the following recommendations can be used:

- If you’re using a Thunderbolt-branded device, only use a Thunderbolt-branded cable with it.
- Only use “full-featured” USB-C cables (ones which connect wires to every pin) when connecting devices that have substantial data needs like high-channel-count audio interfaces or anything to do with video.
- To keep life simple, consider *only* keeping Thunderbolt and/or full-featured USB4-branded USB-C cables around at all; they will always work with everything.
- Evenly distribute high-bandwidth devices across ports belonging to separate controllers.
- Avoid putting high-resolution displays on the same controller as non-video devices with high bandwidth requirements.

When shopping for docks or hubs:

- Prefer Thunderbolt 4 or Thunderbolt 5 branded docks or hubs since they’ll have the best feature set and performance.
- Accept USB4 or USB4 version 2 branded docks as long as they clearly state that they include the features you’re interested in.
- Accept Thunderbolt 3 docks or hubs as long as you don’t want or need to connect a wide variety or large number of USB-C devices downstream of the dock or hub.
- Accept other USB-C docks or hubs as long as you’re not using them for performance-critical applications.

When shopping for video adapters (DisplayPort, HDMI, MHL, DVI, or VGA):

- Prefer Thunderbolt video adapters, since these will necessarily properly support DisplayPort at high bandwidth.
- Accept video adapters that specifically mention using Alternate Modes, especially DisplayPort.
- Do not use video adapters that require drivers or other software to be installed.

When shopping for ethernet adapters:

- Prefer Thunderbolt ethernet adapters or USB4 adapters that specifically mention PCIe Alternate Mode, since these will connect via PCIe which is the same technology that a built-in ethernet port would use.
- Accept 2.5G or 10G USB4 ethernet adapters, since these higher-bandwidth adapters almost certainly require PCIe Alternate Mode.
- Thoroughly test any other ethernet adapters before using them in a show-critical situation.

---

1. Some versions of some protocols can achieve full speed on passive cables that are a little longer. You'll often see 1 meter referred to as the length limit instead of 0.8 meters.↩
2. Gbps stands for gigabits or billion (1,000,000,000 or 10^9) bits per second. Why is speed expressed in *bits* when file sizes are expressed in *bytes*? The reason is that while there are typically 8 bits to a byte in file storage, data transfer protocols can use different numbers of bits to make up a byte, and have different quantities of overhead that eat up portions of the total capacity. Measuring data transfer protocols in bits, not bytes, per second makes it easier to compare any two protocols accurately, even though it makes it harder to understand how long it will take to transfer a file of a known size.↩
3. Evidently, as of January 11, 2023, HDMI Alternate Mode has been discontinued. Apparently no actual USB-C cables or adapters using HDMI Alternate Mode were ever manufactured; all USB-C adapters which output HDMI use DisplayPort in some form. As such, we likely will never know just how useful HDMI Alternate Mode might have been.↩
4. This tutorial will *not* attempt to brave the depths of the depravity of the DisplayPort 2.1 spec. Suffice to say, it is at least as complicated as USB4.↩

Sam's Toolbox

One of the Action Figures, Sam Kusnetz, also works as a sound designer and associate sound designer in New York and around the US. Over the years, he's developed a set of AppleScripts which he uses in tech rehearsal to help program shows as quickly and cleanly as possible.

This section of the manual will walk you through these scripts and explain what they do and how to use them.

The workspace has three cue lists: *Main Cue List*, which is where your cues go, *Scripts*, which is where the script cues live, and *Script Variables* which is where a few settings for the scripts are stored.

Some of these scripts are built upon concepts or whole scripts created by Rich Walsh, Mic Pool, and Chris Ashworth.

Level scripts

Make Fade In

- Cue number: **FI**
- Hotkey: **^I**

This script creates fade-in cues for all selected Audio, Mic, Video, Camera, and Text cues. The duration of the fade is set using the name of cue "FItime" in the "Script Variables" list.

Make Fade Out

- Cue number: **FO**
- Hotkey: **^O**

This script creates fade-out cues for all selected Audio, Mic, Video, Camera, and Text cues. The duration of the fade is set using the name of cue "FOtime" in the "Script Variables" list.

Main level change

- Cue number: **L**
- Hotkey: **^L**

This script sets or changes the main audio level of selected Audio, Mic, Video, and Fade cues.

To set a specific level, enter an asterisk (*****) followed by a level, including a + or - sign. For example, if you enter ***** `-12` this script will set the master level of all selected cues to `-12`.

To add to or subtract from the current levels of the selected cues, enter either a + or - sign followed by a number. For example, if you have two cues selected whose master levels are currently `-20` and `-5`, and you enter `+5`, the master levels for those cues will become `-15` and `0`.

Time scripts

Pre-wait change

- Cue number: **EWC**
- Hotkey: **^E**

This script sets or changes the pre-wait of the selected cues. If you enter a number, the pre-wait of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the pre-wait of all selected cues.

Post-wait change

- Cue number: **PWC**
- Hotkey: **^W**

This script sets or changes the post-wait of the selected cues. If you enter a number, the post-wait of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the post-wait of all selected cues.

Duration change

- Cue number: **CD**
- Hotkey: **^D**

This script sets or changes the duration of the selected cues, assuming the selected cues have editable durations. If you enter a number, the duration of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the duration of all selected cues.

Rate change

- Cue number: **CR**
- Hotkey: **⇧^R**

This script sets or changes the rate of selected Audio, Video, and Fade cues. If you enter a number, the rate of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the rate of all selected cues.

Geometry Scripts

Opacity change

- Cue number: **OC**
- Hotkey: **^Y**

This script sets or changes the opacity of selected Video, Camera, Text, and Fade cues. If you enter a number, the opacity of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the opacity of all selected cues.

Layer change

- Cue number: **LC**
- Hotkey: **^R**

This script sets or changes the layer of selected Video, Camera, Text, and Fade cues. If you enter a number, the layer of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the layer of all selected cues.

Scale X change

- Cue number: **SXC**
- Hotkey: **^S**

This script sets or changes the X scale of selected Video, Camera, Text, and Fade cues. If you enter a number, the X scale of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the X scale of all selected cues.

If the selected cues' scale ratio is locked, setting the X scale will also set the Y scale and vice versa.

Scale Y change

- Cue number: **SYC**
- Hotkey: **⇧^S**

This script sets or changes the Y scale of selected Video, Camera, Text, and Fade cues. If you enter a number, the Y scale of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the Y scale of all selected cues.

If the selected cues' scale ratio is locked, setting the Y scale will also set the X scale and vice versa.

Translation X change

- Cue number: **TXC**
- Hotkey: **^T**

This script sets or changes the X translation of selected Video, Camera, Text, and Fade cues. If you enter a number, the X translation of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the X translation of all selected cues.

Translation Y change

- Cue number: **TYC**
- Hotkey: **⇧^T**

This script sets or changes the Y translation of selected Video, Camera, Text, and Fade cues. If you enter a number, the Y translation of all selected cues will be set to that many seconds. If you enter a + or - sign followed by a number, that many seconds will be added to or subtracted from the Y translation of all selected cues.

Next Blend Mode

- Cue number: **BMN**
- Hotkey: **^B**

This script sets the selected Video, Camera, and Text cues to use the next blend mode. You can learn more about blend modes from the Blend Mode Demo tutorial.

Previous Blend Mode

- Cue number: **BMP**
- Hotkey: **⇧^B**

This script sets the selected Video, Camera, and Text cues to use the previous blend mode. You can learn more about blend modes from the Blend Mode Demo tutorial.

Workflow Scripts

Finish all fades

- Cue number: **FF**
- Hotkey: **⇧F**

This script causes all running Fade cues and Light cues to rush to completion. The duration of the rush is set using the name of cue "FFtime" in the "Script Variables" list.

Sub-number cues in a group

- Cue number: **SNC**
- Hotkey: **^N**

This script numbers all selected cues within a Group cue using the format "x-y", where "x" is the cue number of the enclosing Group cue, and "y" is an integer starting from 1 and counting up to the number of selected cues.

Revised May 2023.