QLab 4

Reference Manual

Last updated on March 10, 2021 for QLab 4.6.9

Table of Contents

Chapter 1: General

Getting Started

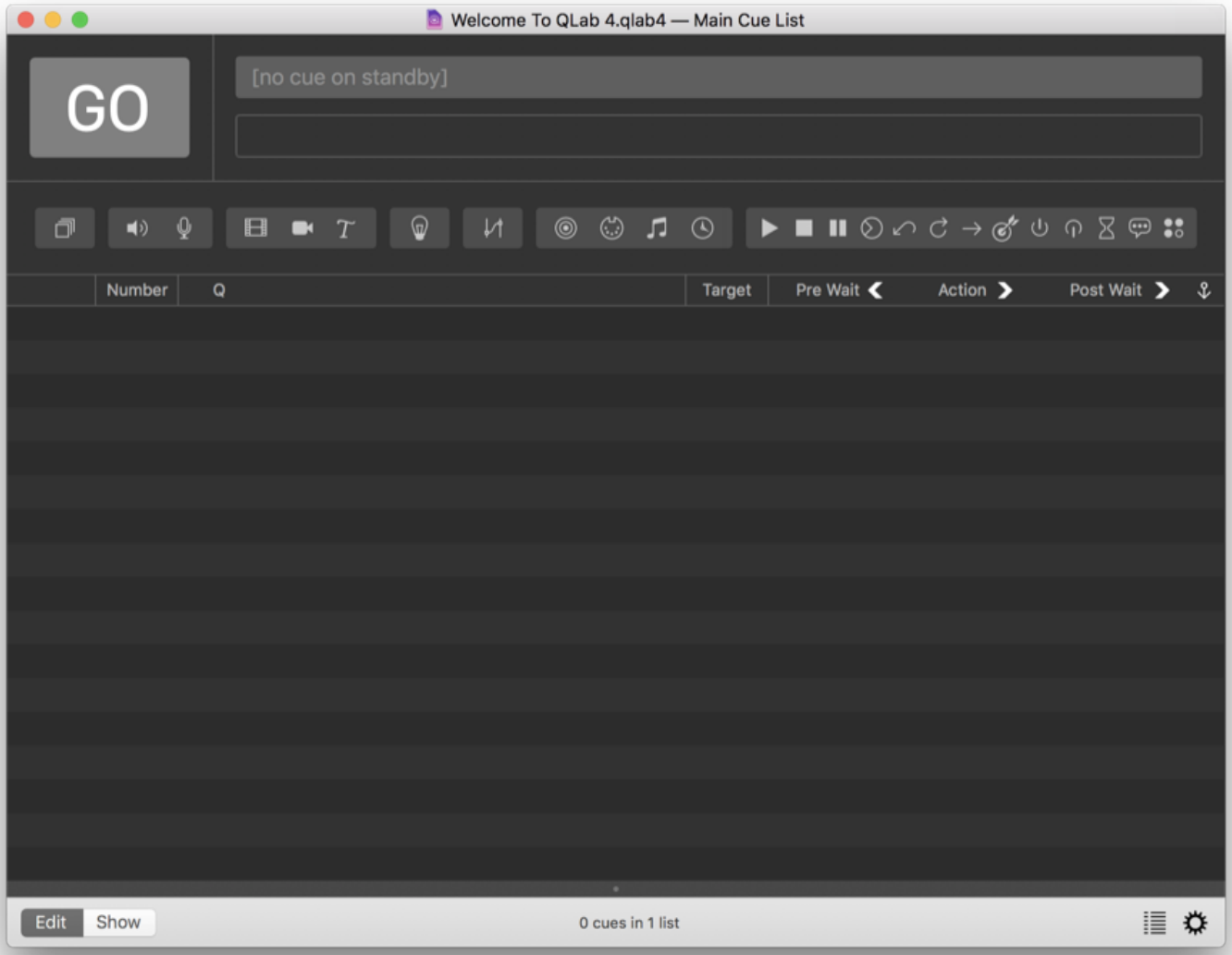
Welcome to the workspace

When you first open QLab, you're presented with a new, fresh document. QLab documents are referred to as **workspaces.** A workspace contains one or more **cue lists** and/or **carts** which each contain **cues.**

Detailed descriptions of the various cue types and cue lists can be found in their respective sections of the documentation; for now we will focus on what you see when you first look at a workspace, and how to get started building your show.

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 for emphasis, to highlight keyboard shortcuts (like **⌘S**), and for indicating a menu name (such as the **File** menu.)



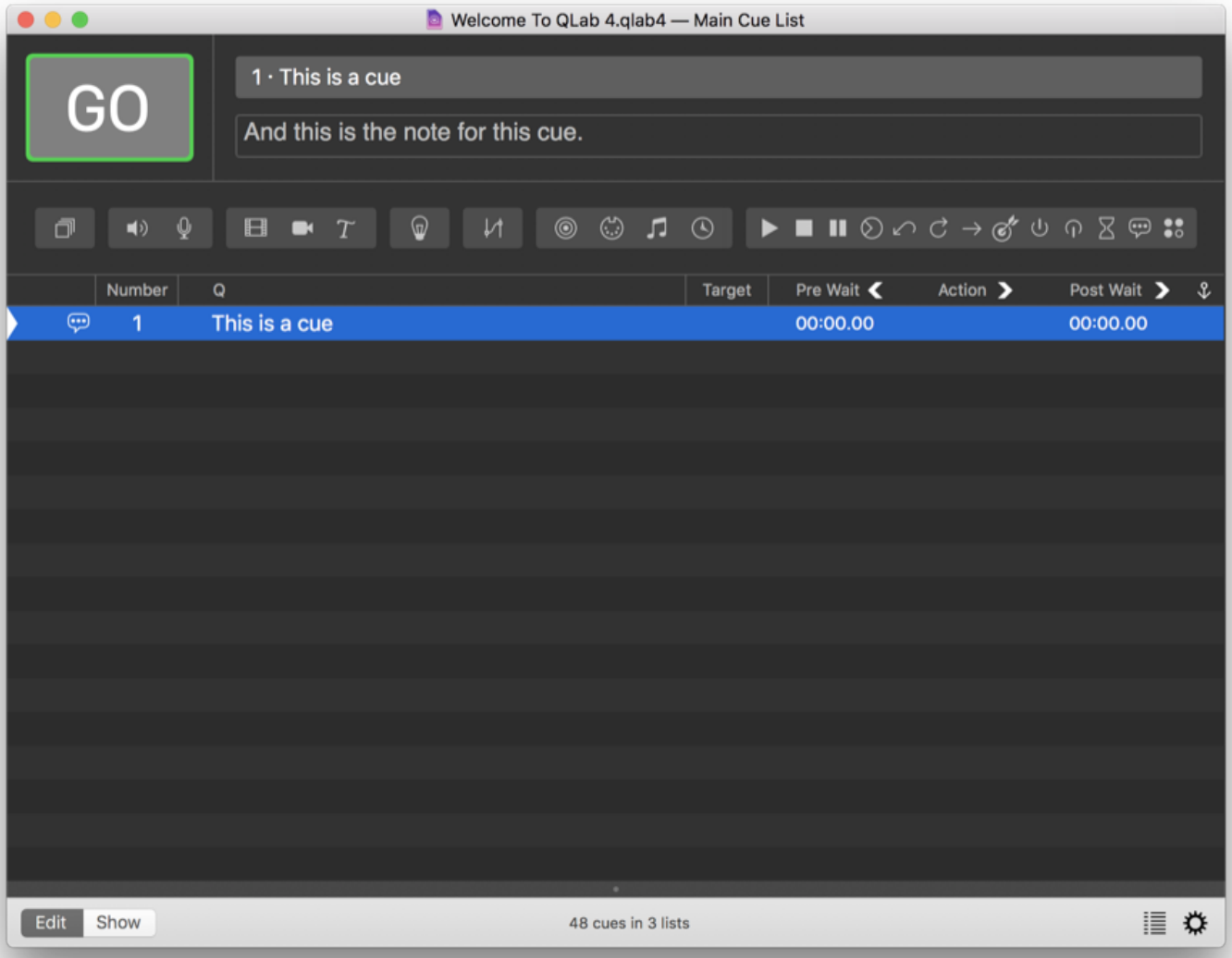
The Masthead

The GO Button

Prominently located in the top left corner of the workspace, the **GO button** starts, or **triggers**, the cue which is currently **standing by** at the **playhead.** The playhead then advances to the next cue, which will then be standing by, displayed in the **standby indicator** and ready to GO. The default keyboard shortcut for the GO button is the **space bar**. You can change this shortcut in Workspace Settings.

The Standby Indicator

Located at the very top of the workspace across most of the width of the window, the standby indicator displays the **cue number** and **cue name** of the cue at the playhead. In other words, it tells you what cue will play the next time the cue list is triggered. When a cue is standing by, it will also appear highlighted in the cue list, with a small indicator arrow against the left edge of the workspace.



The Notes Field

Beneath the standby indicator and to the right of the GO button is the **Notes field.** Text entered in this field is connected to the currently standing by cue, and is visible whenever that 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** feature. You can also edit a cue's notes in the Basics tab of the inspector.

The Toolbar

The **toolbar,** found underneath the Notes field, is a ribbon of icons for each of the different cue types available in QLab. Clicking any of these icons will create a new cue of that type. Explanations of the different cue types can be found later in this documentation.

The Cue List

Beneath the toolbar is the **cue list,** which is the heart of your workspace. Cues are listed here in the order that they will play during your show, top to bottom. The columns in the cue list show essential information about each cue.

Cue Status

The left-most column displays the status of the cue.

The playhead (a right-facing triangular pointer) shows you which cue is standing by.

Each cue might also display any of the following icons:

- ▶ A green triangle means the cue is **active.**
- ⦸ A yellow triangle within a circle means the cue is **loaded** and is ready to be triggered.
- ◺ A grey slope means the cue has been stopped, but has an effect that is **tailing out**.
- ✕ A red X means the cue is **broken** and cannot be played. Hovering the mouse over the red X will show you a tooltip with a brief explanation of the problem.
- ⌀ A red circle with a slash through it means that an **override** is blocking the cue's output. You can learn more about overrides in the Override Controls section of the documentation.
- ⚑ A flag indicates that the cue is flagged.
- No icon means... well, it means that none of the above is true.

This column also displays an icon depicting the cue's type; these icons match the cue icons in the toolbar.

Cue Number

A cue number may be any text string, 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. Acceptable cue numbers could be `1`, `1.5`, `A`, `AA`, `A.5`, `Preshow Music`, or `Steve`. 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 **N**.

It's important to realize that since cue numbers are text strings, `1`, `1.0`, and `1.00` qualify as three different unique cue numbers in QLab.

Reordering cues in the list does not automatically renumber them.

Renumbering Selected Cues

You can automatically assign new cue number to each currently selected cue by selecting *Renumber Selected Cues* in the **Tools** menu, or using the keyboard shortcut ⌘**R**. 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 string, or may be empty. The name of a cue will default to reflect the name of its **target**; for Audio and Video cues, that is the name of the target file. For Fade cues, it will be the word "fade" plus the name of the cue that the Fade targets. Other types of cues have their own, hopefully logical, default names. 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 **Q**. Unlike cue numbers, cue names do not have to be unique.

Note that changing the name of a cue will also change the name of any cues that target it (for example, Fade cues), if those cues are using their default names.

Target

A key concept in QLab is that some types of cues have a **target** which is the recipient of the action of that cue. For example, Audio and Video cues have targets which are media files. When an Audio cue is triggered, the target file is played. Fade cues have targets, which are other cues which have fade-able parameters, such as audio levels or video display geometry. When a Fade cue is triggered, the parameters of the target cue are faded. For cues that require a target, they *must* have one and only one target.

Since different types of cues have different types of targets, the target column can show different information for different cues.

For Audio and Video cues, the Target column displays a round button that looks like an upwards-pointing arrow inside a circle. Clicking this arrow will open a Finder window in which you can select the file you wish to target. You can also set the target of an Audio or Video cue by dragging and dropping an appropriate file onto the cue from the Finder.

Cues which target other cues, such as Fade cues and Stop cues, will display the cue number of their target cue. If the target cue has no number, the cue 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 its intended target, or by dragging and dropping the intended target cue onto 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 the amount of time that QLab waits between receiving a trigger for a cue and starting the action of that cue. For example, an Audio cue with a pre-wait of 3 would start playing sound three seconds after being triggered.

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 **E**.

Action

The **action** of a cue tells you how long it takes for the cue to complete, not counting pre-wait. Action is often used interchangeably with "duration" conversationally. They are one and the same. The action of some cues cannot be edited directly, but for those that can, they can be edited by double clicking and typing in the action column, or by using the keyboard shortcut **D**.

Post-wait

The **post-wait** of a cue is meaningful only in combination with an as-yet-un-discussed feature: **auto-continue.** When a cue is set to auto-continue, it triggers the next cue in the cue list when it is itself triggered. If the first cue has a post-wait time, QLab waits for the post-wait to elapse and then triggers 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 **W**.

Notes about pre-wait, action, and post-wait.

The pre-wait, action, and post-wait columns display seconds with two decimal places, but QLab is actually accurate to three decimal places.

When a cue is not playing, these columns will always display the total duration of the cue or its wait. When a cue is playing, you can toggle between viewing time elapsed or time remaining by clicking the arrows in the column headers.

It's also important to remember that in the interest of keeping QLab's interface from using too much processing power, QLab sometimes updates times as infrequently as once per second, which means that the times displayed may appear to be wrong by as much as one second. This does not indicate that the actual behavior of QLab is off by a second, only the display. A paused cue will always display exact times.

Auto-follow and Auto-continue

The final column, labeled with a downwards-pointing arrow in the column header, displays icons indicating whether a cue has been set to **auto-continue** or **auto-follow.**

If a cue is set to auto-continue ( ), as soon as the cue is triggered the next cue in the cue list will trigger as well. If the cue has a post-wait as well as an auto-continue, the post-wait will be honored before the next cue is triggered.

If a cue is set to auto-follow ( ), then the next cue will be triggered 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 action of the cue. This cannot be edited, and serves as a visual reminder of the auto-follow.

Cues which are connected with auto-follows or auto-continues, or with a combination of both, are called **cue sequences.** You can learn more about creating cue sequences in the appropriately named cue sequences section of the documentation.

The Toolbox

The **toolbox** can be shown or hidden by selecting *Toolbox* from the **View** menu or by using the keyboard shortcut ⌘**K**. 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.

The Inspector

The **inspector** is located beneath the cue list and and can be shown or hidden by selecting *Inspector* from the **View** menu or by using the keyboard shortcut ⌘**I**. The inspector is a set of tabs which let you see and edit the attributes of the selected cue or cues.

More information about the inspector can be found here.

The Workspace Footer

Edit Mode and Show Mode

On the left side of the workspace window footer, two buttons allow you to toggle between **Edit mode** and **Show mode**. When a workspace is in show mode, the following functions of QLab are disabled:

- The inspector
- The toolbar
- Load-to-time
- Find
- Adding, deleting, or reordering cues
- Editing any cue properties (name, number, target, times, etc.)
- Copy and paste levels
- Copy and paste geometry
- Copy and paste fade shape

Just as important is the list of things which are *not* disabled in show mode:

- The Audition window
- Opening, closing, and saving workspaces
- Use of the **ESC** key
- Showing or hiding the sidebar
- Viewing and changing workspace settings

Show mode is a safety mechanism designed to prevent accidental changes to a workspace, not a security mechanism to prevent deliberate changes.

When a workspace is in show mode, QLab will ask for confirmation before closing the workspace or quitting.

Cue and Cue List Count

The center of the footer displays the number of cues in the workspace and the number of cue lists into which they are divided. Useful for bragging about the complexity of your show.

Warnings

The Warnings icon () will appear in the bottom right corner of the workspace window when (and only when) the workspace contains flagged or broken cues. Clicking this icon opens the Warnings tab of the Status Window, which shows a list of the cues in question and explains why they're listed there. You can learn more about this in the Workflow Tools section of this documentation.

Sidebar

Click the Sidebar button () in the bottom right corner of the workspace window to show or hide the sidebar. You can learn more about this in the Workflow Tools section of this documentation.

The Status Window

Click the Status Window button () in the bottom right corner of the workspace window to show or hide the Status Window. You can learn more about this in the Workflow Tools section of this documentation.

Workspace Settings

Click the Workspace Settings button () in the bottom right corner of the workspace window to open the Workspace Settings window. You can learn more about this in the Workspace Settings section of this documentation.

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 a compatible audio or video file from the Finder into the cue list.
8. (new in 4.2) 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 documentation.

Targeting Audio and Video cues

Audio and Video cues require a target which is a file containing the relevant type of media. There are several ways to assign a target to an Audio or Video cue:

1. Click the *target button* for the cue in the cue list.
2. Select the cue and type the target hotkey, *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.

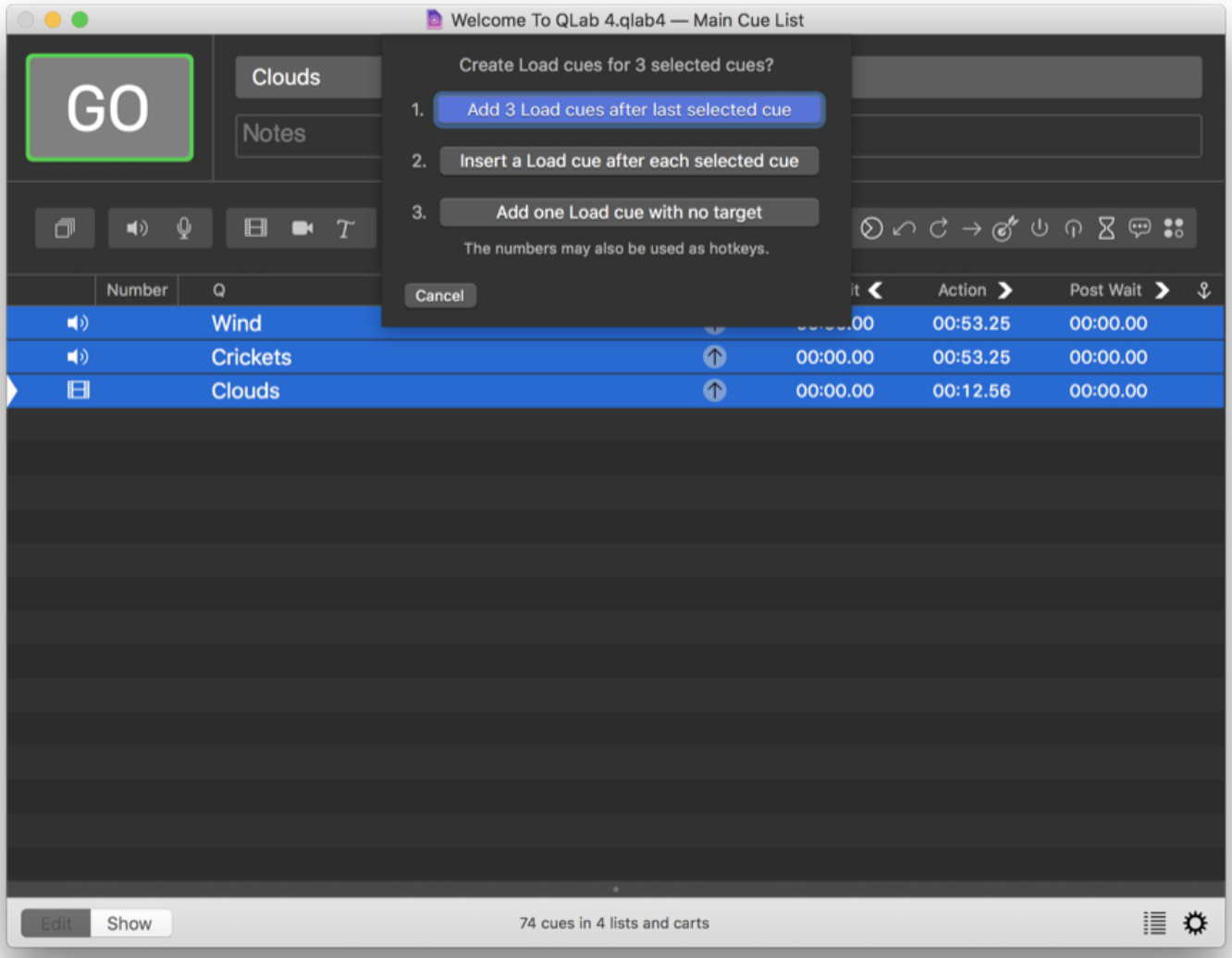
Targeting other cues

Cues which target another cue include Fade, Start, Stop, Pause, Load, Reset, Devamp, GoTo, Target, Arm and Disarm. There are several ways to assign a target to these cues:

1. Double-click in the target column in the cue list.
2. Select the cue and type the target hotkey, *T* by default.
3. Double-click in the target field in the *Basics* tab of the inspector.
4. Drag and drop the intended target cue onto the cue in the cue list.

Additionally, starting with QLab 4.4, QLab behaves slightly differently when you create cues of any of these types depending upon what other cues are selected. When you create a new Fade, Start, Stop, Pause, Load, Reset, Devamp, GoTo, Target, Arm, or Disarm cue:

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.



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 a Fade cue, the default duration is five seconds, although you can change the default in the Fade cue's cue template.

When you start a cue, the playback position will advance to the next cue. On the next press of the GO button or space bar, the next cue will start. If you're accustomed to using a non-theatrical playback program such as iTunes, this behavior can be disorienting at first. Fear not, for this is how QLab is meant to work.

Rules for control

QLab can be told to GO, which is to say, to trigger the cue or cue sequence that is currently standing by, and advance the playhead to the next cue or cue sequence, in a number of ways. You can:

1. Click on the GO button with the mouse.
2. Press `space` (or whichever keyboard shortcut you assigned to GO.)
3. Send an OSC `/go` command to QLab from another program or an external device.
4. Send another OSC command that you assigned to GO in OSC Controls.
5. Send an MSC `Go` command to QLab from another program or an external device.
6. Send a MIDI message that you assigned to GO in MIDI Controls.
7. Run an AppleScript which sends the `go` command to QLab.

There are similar options for other commands such as stop, panic, pause, and so forth. The purpose of this plurality is to accommodate the myriad situations and possibilities that QLab might encounter in the world, and make it easier for you to use QLab in the way that suits your needs or your style.

New in v4

This is a more or less fully inclusive list of the differences between QLab 4 and QLab 3. This list includes changes that were introduced over the lifetime of QLab 4, not just changes that debuted with QLab 4.0.

General Improvements

Lights (in addition to Camera and Action)

We are pleased to bring lighting control to QLab. Starting with QLab 4.1, we are also pleased to offer compatibility with select USB DMX interfaces.

Keeping It Fresh

QLab 4 sports a snappy new interface with font and design changes to stay aligned with Mac OS 10.11 and onwards.

QCart Included

You can now add Carts to your workspaces, bringing the functionality of QCart into QLab.

Timeline View

- **4.3:** Start-all-children Group cues are now Timeline Group cues, with a fast new visual editor that lets you drag, trim, and slip cues to adjust their timing within the Group.
- **4.4:** you can now pin cues to the top of the timeline, making it easier to compare them with cues lower down in the timeline.

d&b Soundscape (4.2)

The Network cue now includes support for directly controlling the d&b Soundscape DS100 processor.

Inspector Gadgetry

- **4.1:** The cue inspector can now be "popped out" into its own window to allow for more control over how screen space is used while programming.
- **4.4:** Multiple independent inspector windows can be opened to show properties of selected cues.

Windows, Windows Everywhere (4.1)

Cue lists and cue carts can now be opened in their own windows.

Top Audition (4.1)

The Audition Window now has the option to always float on top of all other windows.

Record Sequence

QLab can record your timing as you manually trigger a series of cues, so you can play back that sequence of cues later with exactly the same timing.

Highlight Related Cues

Find all cues that share the same target as the selected cue.

Fancy Paste

You can now paste some or all of the properties of the cue on the clipboard to one or more selected cues by choosing *Paste Cue Properties...* from the **Edit menu** or by using the keyboard shortcut ⇧⌘**V**.

Cue Templates

You can now fully customize the default values for new cues of every cue type.

4.1: workspace template files (with the file type `qlab4template`) can now be opened directly by QLab, including by double-clicking on them in the Finder.

Template Management (4.1)

In the workspace template management window, you can right-click to get a menu of new actions for templates: Set as Default, Reveal in Finder, Export, Rename, Delete.

New Cue Trigger Options

Cues can now be set to fade and stop their peers (cues within the same Group, Cue List, or Cart), or all other cues in the workspace, using a per-cue customizable fade time.

You can now set cues to respond to hotkey and MIDI triggers received while they're playing: do nothing, panic, stop, hard stop, or hard stop & restart.

Cue triggers are now displayed in their own inspector tab.

Batch Editing

- **4.0:** When multiple cues are selected, the inspector will allow you simultaneously adjust parameters in the Basics tab, the Triggers tab, and the Light cue's Levels tab.
- **4.1:** you can also batch edit all properties of Timecode, Load, Devamp, and Target cues, as well as some properties of Audio, Mic, Video, Camera, Text, and Fade cues. Also new to **4.1** is the ability to select multiple cart cues, and batch edit them.
- **4.2:** you can also batch edit the Audio Levels and Audio Trim tab of Audio and Video cues.
- **4.3:** you can also batch edit the Audio Levels and Audio Trim tab of Mic cues.
- **4.5:** you can also batch edit Network cues.
- **4.6:** you can also batch edit MIDI cues.

Negative Post-wait

You can now specify negative post-wait times, to trigger the following cue "this many seconds before the current one ends."

It's The Network (cue)

The cue formerly known as the OSC cue is now called the Network cue, and it's gained quite a few new powers...

- Network cue messages can insert the current value of (almost) anything in QLab that can be queried by OSC.
- Network cues can be given a duration, over which they will continuously re-send their message. This is especially useful in conjunction with the aforementioned inserted live values.
- Network messages can now fade single parameters or pairs of parameters.
- Network cues can now optionally be directed at a specific network interface (such as ethernet or wifi.)
- Network cues' 2D fade paths can now be edited, scaled, and dragged.
- **4.6:** Workspaces can now have as many (or as few) Network patches as you need.

Custom OSC Replies

You can now specify custom OSC reply formats from QLab.

Check Your Status

The new Status Window provides a variety of information about your workspace, so that you can see what condition your condition is in.

Listing Triggers (4.2)

A new Triggers tab in the Status Window lists all cue triggers and workspace controls used in your workspace.

Group Cue Improvements

Group cues in "start random child" mode now operate in a round-robin manner, which means that every child cue will be triggered once before a new round of random cues begins.

Group cues in "start all children simultaneously" mode now display the duration of their longest child.

New Fade Options

QLab 4 includes a new parametric fade curve option, and new controls over the shape of audio fades.

New Find Options (4.1)

The “Find” tool will now search the content of Text cues, the content of custom OSC messages, the content of UDP messages, and the content of Script cues.

My exit music, please!

You can optionally specify a cue for QLab to trigger when your workspace closes.

Places, please!

You can now choose how QLab behaves when it launches, either doing nothing, restoring the most recently open workspace, creating a new workspace from scratch, creating a new workspace from a specific template, or (starting with QLab 4.1) displaying the Launch Window which provides a variety of tools designed to help you get started.

Override Overload

QLab’s global override controls now include timecode, and are divided into separate categories for each type of incoming and outgoing message. Also, there are now OSC and AppleScript hooks for each override control.

When overrides are engaged, indicators appear in the workspace footer and/or the cue list to let you know what’s going on, and what won’t be going on.

Export Broken Cues and Warnings

You can now copy the contents of the warnings tab of the Status Window to paste into a text editor or an email to the support team.

Double-Go Protection Indicator

If you’ve set a minimum time required between each GO, the area around the GO button flashes red whenever the double-go protection is invoked.

Option Drag (4.2)

Hold down the option key while dragging cues within a list or cart to duplicate them, rather than move them.

Do More With Durations

AppleScript and OSC hooks have been added to support a temporary duration (*tempDuration*) for all cues where editing duration is allowed.

A new *currentDuration* read-only property, accessible via both AppleScript and OSC, returns the current duration of the cue, accounting for any *tempDuration* that has been set.

Moving Amongst Groups (4.1)

When moving the playhead to next or previous sequences, Group cues in “start all” mode are now considered a cue sequence, so you’ll jump over the whole group.

OSC = Oh, Super Controls!

QLab’s OSC dictionary has been greatly expanded, including a new special address to target currently active cues. Each major point release has expanded the dictionary even further.

Go Button 3 Import

Show files created in Go Button 3, our iOS show control app, can be imported into QLab.

Accessibility Support (4.2)

QLab now supports high-contrast mode, colorless UI differentiation, screen reader support, and other accessibility-related improvements.

Audio

This one goes to 64

QLab now offers a maximum of 64 outputs, up from 48, to better support Dante, MADI, and AVB infrastructure.

Smaller Slices

The minimum slice time has been reduced to 0.05 seconds (down from 0.1)

Integrated Fade Options

The integrated fade envelope can now be optionally locked to the start and end time of the cue instead of the file.

More Precise Mic Cues

Mic cues now use specific channels of their patched device, making it dramatically easier to work with AudioUnit effects that need specific numbers of inputs and outputs.

Always Watching, Always Listening

Audio cues now continuously watch their target files for changes and update automatically.

Duck and Boost

Cues can now be set to duck or boost the master audio level of all other cues in the same cue list while they play. The amount and fade time of the duck or boost is customizable per cue.

More Batch Editing (4.2)

You can now batch edit audio levels and trim in Audio cues and Video cues.

Loops Within Loops (4.1)

You can now loop an entire cue, even when it is sliced. The Devamp cue now allows you to specify whether you want to devamp the current slice or the entire cue.

Easier Slice Markers (4.1)

Pressing the “m” key (“m” for “marker”) will now add slice markers when inspecting the audio waveform of an Audio or Video cue. There are also new OSC commands and AppleScript commands for working with slice markers.

Video

Still Image Durations

You can now specify a duration for Video cues which target still images, as well as for Text cues.

Copy/Paste Surface Geometry

You can now copy and paste your carefully adjusted surface geometry between surfaces.

Color Me Perfect

The color accuracy of both Video cues and video effects has been improved.

Hap

QLab can now display videos using Hap and Hap Alpha video codecs, which offers excellent image quality and improved performance over ProRes.

Textual Improvements (4.1)

The Text cue now has line spacing controls, a ruler to help you determine the actual display size of your cue, improved Fancy Paste options, and substantially expanded OSC and AppleScript support.

This One Goes To Eleven (4.2)

Significant under-the-hood changes have led to major performance improvements in **video effects**, **geometry fades**, and **load-to-time** for Video cues.

Paint It Black (4.2)

Video surfaces can now be set to keep rendering a black background between cues, ensuring a smoother start for the next Video cue. Screens assigned to surfaces with this setting will continue rendering until a panic.

Lighting

Submasters (4.5)

You can now create proportional-control, highest-takes-precedence lighting submasters.

Virtual Controls (4.5)

QLab now offers on-screen color pickers for both additive and subtractive color-mixing fixtures, as well as a graphical pan/tilt controller for moving heads and mirrors. Rear more here.

Pull From Cue (4.1)

You can now instruct a Light cue to "pull" a value from another Light cue when it runs. Those familiar with the idea of palettes or presets on other lighting consoles will find this concept familiar. Read more here.

MIDI Channel Specificity (4.1)

Light settings now have a dedicated MIDI input channel, used when controlling the dashboard or light cue inspector via MIDI.

More MIDI, Better Programming (4.3)

MIDI controls can be assigned to any instrument parameters, not just the default parameter. MIDI controls can also be assigned to "selected" instruments & groups, allowing you to dynamically adjust selected parameters.

A More Dashing Dashboard (4.2 & 4.3)

- **4.2:** The Light Dashboard now supports undo and redo, decimal values for percentage-based lighting parameters, a Clear All button (similar to "Go To Cue Out", for those who recognize that phrase) and improved cue recording and updating controls.
- **4.3:** a "used" filter which shows only those instruments which are currently in use, much more AppleScript support, better visual distinction between instruments and groups, a cleaner and clearer layout for tile view, and arrow indicators on sliders to show the direction of the most recent change to each parameter.

More Lights

- **4.2** includes fixture definitions for most of the current instruments made by American DJ, Chauvet, Chroma-Q, Clay Paky, Elation, ETC, GLP, Martin, Robe, Vari*Lite, and a few Rosco and Wybron fixtures as well.
- **4.3** includes fixture definitions for most of the current instruments made by Altman, CLF Lighting, Philips Selecon, and SGM, as well as one from Showtec and more back-catalogue fixtures from Martin, Robe, Vari*Lite, and others.
- **4.4** includes more from Altman, Chauvet, K9, Panasonic, Robe, and SGM.
- **4.5** includes more from Altman, Cameo, Chauvet, CLF, and ETC.
- **4.6** includes more from Cameo, Chauvet, CLF, Elation, GLP, High End Systems, and Martin, and improves compatibility with ETC ColorSource, Selador, Source Four LED, and Source Four LED Series 2 fixtures.

Decimal Values: a good point! (4.2)

Light parameters that use percentages now allow decimal values, which allows higher precision. You'll notice this especially with 16-bit pan and tilt parameters.

More Universes (4.2)

The second universe of the DMXKing ultraDMX2 PRO and eDMX2 PRO is now supported.

Parking (4.3)

You can park individual parameters or entire lights, freezing them at a specific value until you unpark them.

Specific Changes From QLab 3

- Workspace files now use the extension ".qlab4" to make it easier for QLab 3 and QLab 4 to coexist on the same computer.
- Workspace settings now appear in their own window, rather than on the back of the cue list.
- If you start a cue that is showing the hollow play indicator (meaning it's panicking or that it has an audio effect that is tailing out) QLab will now hard-stop the cue and then restart it.
- In OSC Controls and MIDI Controls, *Select Previous Cue* and *Select Next Cue* have been replaced with *Playhead Previous* and *Playhead Next*.
- The Notes field is now shown in the Basics tab of the inspector as well as in the masthead of the main workspace window.
- The cue formerly known as the Titles cue is now called the Text cue, since not all text is a title, after all.
- In Video cues, Camera cues, and Text cues, the term "full screen" has been replaced with the more accurate "full surface".
- The default surface for Video cues, Camera cues, and Text cues can now be set the in Cue Templates section of Workspace Settings. When using QLab without a Video license, the Video cue template defines the one usable surface in a workspace.
- Cue unique IDs are now UUIDs, rather than a string representation of an integer.

QLab Remote

Updated and ready for QLab 4!

All-new flexible design

- Looks great on all devices and supports iPad Split View mode.
- Adds elapsed Pre Wait, Action, and Post Wait progress time for all cues.
- Show or hide the GO and transport buttons.
- See the notes for the current playback position cue on the main cue list screen.

New Features for QLab 4 only

- **(4.2)** Light Keypad for fast, easy Light cue programming. (optional in-app purchase.)
- **(4.2)** Instrument Check for quickly checking through all your lights and light groups.
- **(4.1)** QLab Remote can now optionally communicate with QLab on your Mac over a USB cable instead of wifi. This is dramatically faster and eliminates concerns about wifi flaking out.
- Adjust video surface control points.
- Create new cues and reorder existing cues in both cue lists and carts.
- Browse and select a new cue file target from available media files on your Mac.
- Edit audio input and output crosspoints.
- Full access to the Triggers inspector tab
- Full access to the Light Levels inspector tab.
- Long-press a cue to move it in the cue list.
- **(4.6)** When QLab Remote is connected to a workspace running in QLab 4.6 or later, audio output names are now displayed.

New Features for QLab 3 and 4

- *Read-Only Mode* lets you watch a workspace without worrying about accidentally triggering or editing a cue.
- Keyboard support lets you navigate with familiar QLab shortcuts.
- New playing, loaded, paused, broken, and panicking status indicators for cues and cue lists keep you more closely informed.
- Access QLab's *Load to time*, *Undo*, *Redo*, *Save Workspace* menu items.
- Access to the Audio Levels inspector tab.
- Access Devamp settings on iPhone.
- *Disconnect* now prompts for confirmation, with the option to cancel an accidental disconnect.

System Recommendations

Overview

QLab 4 is a Mac-only program. It requires macOS 10.10 or higher, and will work on any Mac that can run 10.10. Starting with version 4.6.8, QLab 4 is compatible with macOS 11 (Big Sur) and Macs which use Apple Silicon (M1) processors.

Because of QLab's great flexibility and the varied scenarios in which it is used, it can be difficult to determine ahead of time how much computer power a given QLab workspace will require. What follows is a discussion of general concepts surrounding processor, GPU, RAM, and hard disk use for QLab. Please take this information not as a firm set of instructions about what to do, but rather as a set of recommendations about what to consider.

QLab is not supported on "hackintoshes" at all. Please do yourself a big favor, and just keep away from those. QLab is likewise not supported in a virtual machine environment.

Processor

The more work QLab needs to do, the happier it will be with a more powerful processor. Large numbers of audio or video cues playing back simultaneously, for example, benefit from an i7, i9, or Xeon processor which have better handling of multi-threading tasks than i3 or i5 processors. Processors on Macs cannot be upgraded after they're purchased, but most Macs let you select from several processor options at the time of purchase.

GPU and VRAM

For audio-only users, GPU considerations are fairly negligible. For video folks, what you need depends entirely upon what you're trying to accomplish. Mac Minis, for example, can drive two displays simultaneously; one for your operator and one for your projector. Since those two displays share a single integrated GPU, you can improve overall performance by lowering the resolution on your operator's display; the computer will be doing less work for the operator's display, which means more power is available for video crunching. If you use a pre-2014 Mac Pro ("cheese grater") or a late-2019 Mac Pro ("insanely expensive"), dedicating one modest video card for your operator display and one higher-end card for each projector, or one higher-end card per two projectors, is a good strategy. For the 2013-2018 "Darth Vader's wastebasket" Mac Pro, all graphics connections are on the same GPU so you don't have any choices in the matter. Testing is, as always, important.

For those seeking best-possible video performance, a Mac with a discrete GPU is the way to go. The Mac Pro, iMac Pro, some MacBook Pro models, and a few other iMac models have discrete GPUs.

For Macs with an integrated GPU, which is all Mac Minis, the MacBook, all MacBook Airs, most iMac models, and most MacBook Pro models, the GPU uses a portion of system RAM as VRAM. The size of this portion is based on the total amount of system RAM installed, so the more RAM you have, the more of it will be used for VRAM. While we don't recommend using a Mac with an integrated GPU for video-intensive shows, if you do use such a Mac we strongly encourage you to install the maximum possible amount of RAM.

RAM

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

Disk

QLab is happiest with a solid state drive (SSD) because they are, to put it plainly, really really fast. The more data you're pulling off your disk and pushing out of your speakers or projectors, the more an SSD is a good idea. We do not recommend using a traditional spinning hard disk at all, but if you do use one, it ought to be rated at 7200 RPM.

Apple's Fusion Drive, though it technically includes an SSD, is not recommended for use on a show computer under any circumstances, as the user has no control over which data is stored on the SSD portion and which is stored on the hard disk, nor any say in when the Mac decides to shuffle data between the

two.

Video Output

The best way to output video from QLab is to use the built-in video connections on your Mac (including PCI cards on Mac Pros with PCI slots.) Using a Pro Video or Pro Bundle license, you can also output video directly to Blackmagic Design's DeckLink, Intensity, and UltraStudio devices, although this comes at the cost of increased CPU use and an increase in latency.

Macs with Thunderbolt 3 ports running macOS High Sierra and later can make use of devices called eGPUs, which are discrete GPUs in external cases connected via Thunderbolt 3. While we have limited hard data on these devices, our understanding is that they work well with QLab as long as your projector or display is connected to the video output on the eGPU itself.

We do not recommend, nor do we support, video output via USB-connected monitors and video adapters. While these devices can work, they are not GPU accelerated and their drivers do not have a solid history. Your mileage may vary, but Figure 53's position is to avoid using these devices entirely.

Ask Us

If you have specific questions about hardware requirements, please email the support team, and tell us about your show. We will be happy to advise you.

Preparing Your Mac

Note: This page was last updated on *May 29, 2020* to reflect our most up-to-date research on the subject, and to reflect changes in macOS Catalina (10.15).

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.

What follows here is a list of the programs or processes which we recommend disabling, and instructions for doing so. This section presupposes a basic understanding of macOS and at least a passing familiarity with the Terminal.

Click here to download our "Prep and Restore" workspace (current version: 5.2020). The workspace contains two Script cues. The first one executes all the prep steps below, not including the video-specific ones, that require Terminal commands. The other Script cue reverses these steps.

The current version of this workspace behaves slightly differently from earlier versions, mostly because macOS Catalina requires a different approach to achieve the desired results. The scripts will ask for your password two separate times because these commands require administrator privileges. Nothing sneaky is going on; nothing is stored or transmitted.

Disable Spotlight

Spotlight periodically updates its index of all files on all attached disks, and this updating can cause the disk to be momentarily unavailable to QLab. This can cause late cues or stuttering in playback. To prevent Spotlight from updating its index, open a Terminal window and enter this command:

```
sudo mdutil -a -i off
```

Disable Display Sleep, Disk Spindown, and System Sleep

Obviously we don't want our computer going to sleep during a show. macOS has independent sleep intervals for the display, the hard disk, and the whole system. To prevent all three kinds of sleeping, open a Terminal window and enter this command:

```
sudo pmset -a displaysleep 0 disksleep 0 sleep 0
```

The built-in disk on most modern Macs is an SSD, solid state drive, which does not have any moving parts. The moniker "spindown" therefore is no longer exactly appropriate; nothing is spinning, nothing spins down. However, this setting is still sometimes used for SSDs which have a low-power "idle" state.

Disable Screen Saver

Likewise, we don't want the screen saver coming up, particularly if QLab is running video. To prevent that from happening, open a Terminal window and enter this command:

```
defaults -currentHost write com.apple.screensaver idleTime 0
```

Disable Time Machine

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. To shut off Time Machine, open a Terminal window and enter this command:

```
sudo tmutil disable
```

Disable Software Update

You don't want your computer trying to update software in the middle of a run, let alone in the middle of a performance. To disable Software Update, open a Terminal window and enter this command:

```
sudo softwareupdate --schedule off
```

Disable Dashboard

The Dashboard was removed from macOS Catalina. If your Mac is running Catalina or later, skip this step.

If your Mac is running an earlier version of macOS, however, the Dashboard remains a pernicious little vampire of CPU time and network access. Also, if accidentally invoked, it takes over the screen of your Mac entirely, which can be surprising and confusing and lead to missed cues. To disable Dashboard entirely, open a Terminal window and enter this command:

```
defaults write com.apple.dashboard mcx-disabled -boolean YES
```

Alternately, you can disable Dashboard on macOS El Capitan (10.11) through Mojave (10.14) using System Preferences:

1. Open System Preferences;
2. Choose Mission Control;
3. Set Dashboard to “Off”

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 (a.k.a. during standby), 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 Photo Library”.

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.

1. Open System Preferences;
2. Choose iCloud;
3. Click “Sign Out”.

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.

1. Open System Preferences;
2. Choose Internet Accounts;
3. Choose an account;
4. Uncheck each service type;
5. Repeat for each account.

Restart the Dock

Oddly, the Dock is in control of several of the system components that we just adjusted. Restarting the Dock allows these changes to take effect. Open a Terminal window and enter this command:

```
killall Dock
```

Stay Off The Internet (sort of)

Many individual applications, including QLab, have their own internal scheme to check for updates. You can turn them off manually, and we recommend that. 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 can be a very wise thing to do unless you need internet access on your show Mac for some specific reason. 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 4 prevents use of **⌘-Tab**, Mission Control, and Exposé while in Show Mode. If you want QLab to *not* prevent these things in Show Mode, choose *QLab Preferences…* from the **QLab** menu and uncheck the box marked *Disable Disruptive Task Switching Features*.

And If You're Doing Video…

If you're using QLab for video, there are two more critical settings:

Disable *Mirror Displays*

When 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. Amazingly, *there is no Terminal command for this!* To turn off display mirroring:

1. Open System Preferences;
2. Choose Displays;
3. Choose Arrangement;
4. Uncheck Mirror 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.) If your displays are set to have separate spaces, the Menu bar also appears on all Displays, and that is visible to your audience when no cues are playing through QLab. To set your displays to share Spaces, and thus keep the menu bar out of your picture, open a Terminal window and enter this command:

```
defaults write com.apple.spaces spans-displays -bool TRUE
```

Important: you'll need to log out, then back in again for this to take effect.

Blackout the Desktop

When QLab is playing a Video cue, it places a black "backdrop" over any screen that the Video cue is playing on. 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 Preferences;
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.

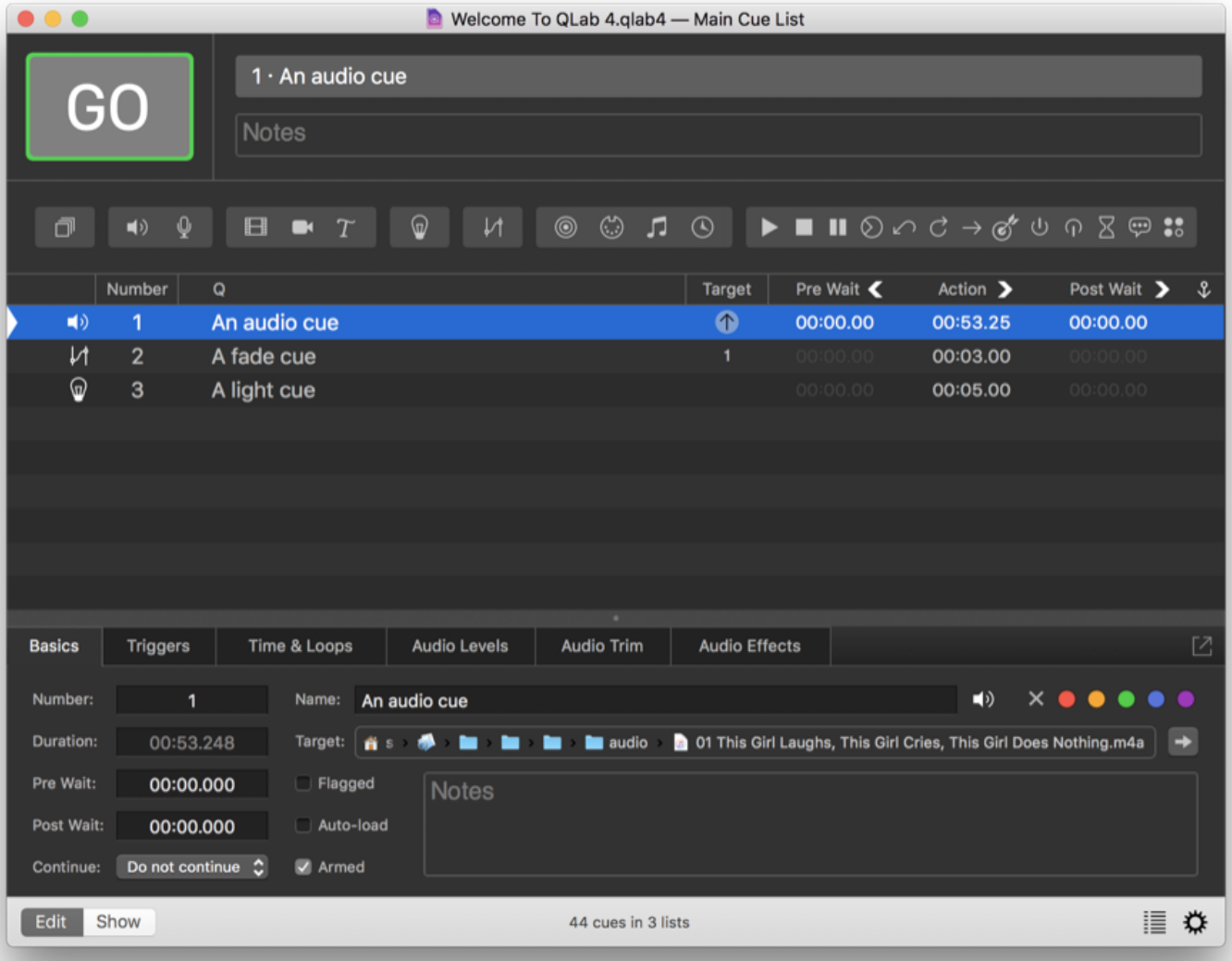
Alternately, QLab provides a quick and easy way to do the same thing. Simply choose *Black out desktop backgrounds* from the **Tools** menu, and all desktop backgrounds will be set to black. You can later choose *Restore saved desktop backgrounds*, also from the **Tools** menu, to restore the desktop backgrounds you had previously.

The Inspector

The inspector is a tabbed interface which lets you see 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 also 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 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 will be described in the sections of the documentation that discuss the various cue types.

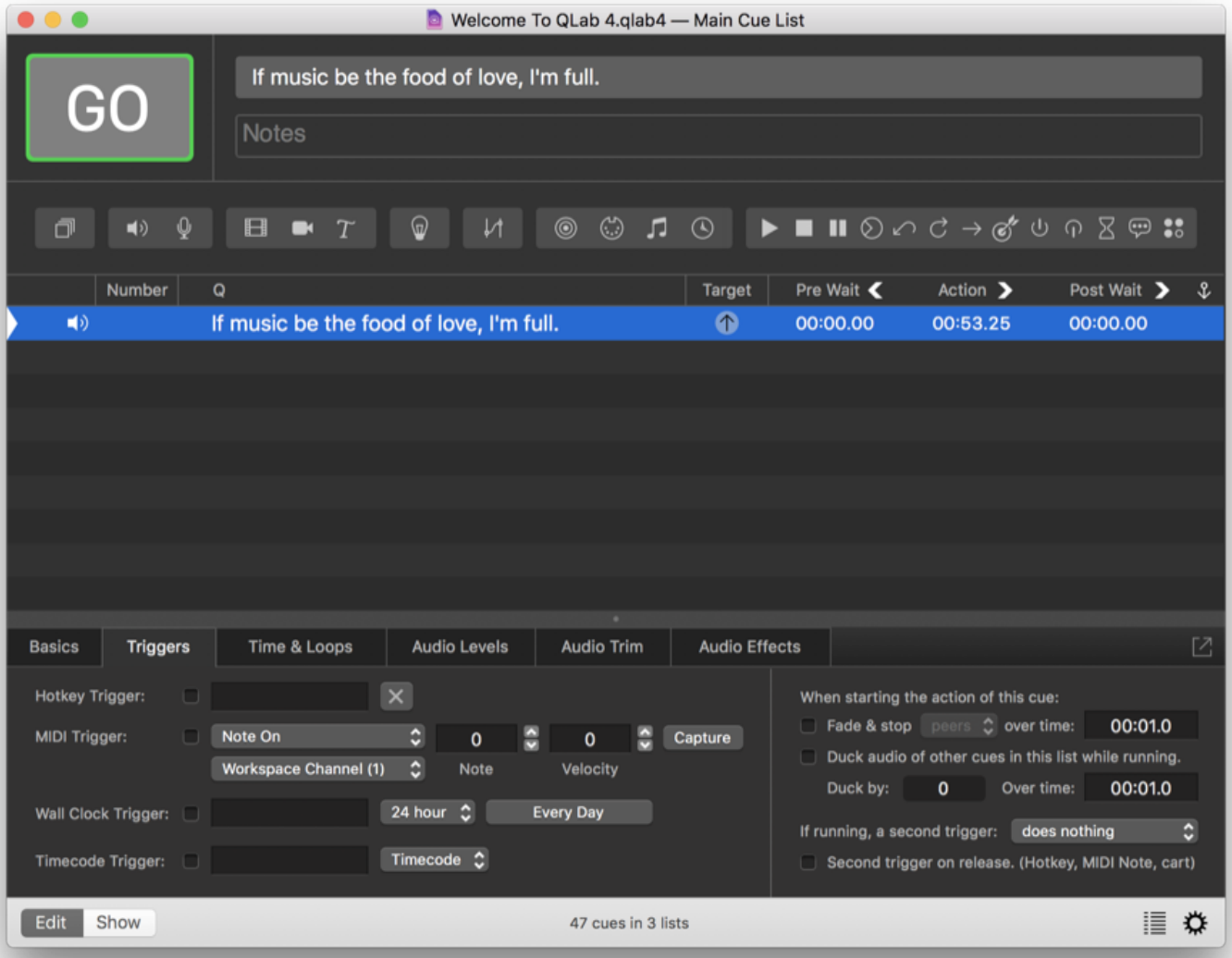
The Basics Tab



Perhaps unsurprisingly, the Basics tab shows basic properties of the cue. Many of these properties are also editable from the Cue list.

- **Number.** See the Cue List section of the Getting Started page for details.
- **Duration.** This field is not editable for all cue types, since some cues have no effective duration and others have durations that rely on targeted media files. If the duration of the selected cue is editable, you can do that here. If it is not editable, the duration will appear dimmed.
- **Pre Wait.** See the Cue List section of the Getting Started page for details.
- **Post Wait.** See the Cue List section of the Getting Started page for details.
- **Continue.** See the Cue List section of the Getting Started page for details.
- **Name.** See the Cue List section of the Getting Started page for details.
- **Target.** This field, only relevant when the selected cue has a target, shows either the name of the target cue, or the full path to the target file depending on what type of cue is selected. Clicking the arrow button next to the target field will show you the target, either by jumping the cue list to the target cue, or opening a Finder window to show the target file.
- **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. Flagged cues are listed in the Warnings tab of the Status Window so you can quickly see all the flagged cues in your workspace. Flagged cues also show a flag icon on the left side of the cue list.
- **Auto-load.** If this box is checked, this cue will automatically be loaded after the previous cue is played. A global setting is available to set all new cues to auto-load by default.
- **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.
- **Color.** The cue can be given a highlight color to make it stand out in the cue list. There are five colors to choose from; the *X* sets the cue to no highlight color. 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.
- **Notes.** Text entered in this field is shown in the notes field up in the top portion 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 can be searched using the Find feature.

The Triggers Tab



When a cue is given the signal to start, we say that the cue is *triggered*. The most straightforward way to trigger a cue is to press the GO button (or use the space bar) when the cue is standing by, but there are also several ways to directly trigger a cue even when it is not standing by.

The left side of the Triggers tab gives you access to four of those ways to directly trigger a cue. 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 for the cue when it is triggered. Note that these right side options apply to the cue no matter how the cue is triggered, including when it's triggered by the GO button.

Hotkey Trigger

Nearly any key on the keyboard that's not being used for something else can be assigned to trigger a cue when pressed. The **Esc** key is permanently assigned to *Panic/Stop all*, so that's out, and any keys that have already been assigned functions in Key Map Settings cannot be used as cue triggers. Beyond that, however, you're free to choose any key you like.

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 *X* button to the right of the text field.

Hotkey triggers behave just like regular keyboard shortcuts for menu items; for example, by assigning the Hotkey J to a cue, any time you press **J** on the keyboard, that cue will start even if it is not currently standing by. Hotkeys may include modifier keys (option, control, function, and shift). Hotkeys are particularly useful for triggering scripts that act on selected cues.

MIDI Trigger

Cues can be triggered with external hardware or software using MIDI voice messages here.

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.

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 trigger 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.

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 triggered from incoming LTC (SMPTE) or MTC timecode, but only if timecode has been enabled for the enclosing cue list. You can enable timecode for a Cue List in the **Timecode** tab of the inspector when the cue list 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 `reel:minutes:seconds:frames` or the real-time format of `hours:minutes:seconds:decimals`.

Fade & Stop Others Over Time

When this box is checked, triggering 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 amongst 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. So, for a cue within a Group, the other cues within that same Group are its peers. For a cue within a cue list or cart, the other cues in that same list or cart 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.

If this cue has a pre-wait time, the duck or boost begins after the pre-wait has elapsed.

Second Triggers

When a cue is triggered while it is already running, it can be set to behave in one of five ways:

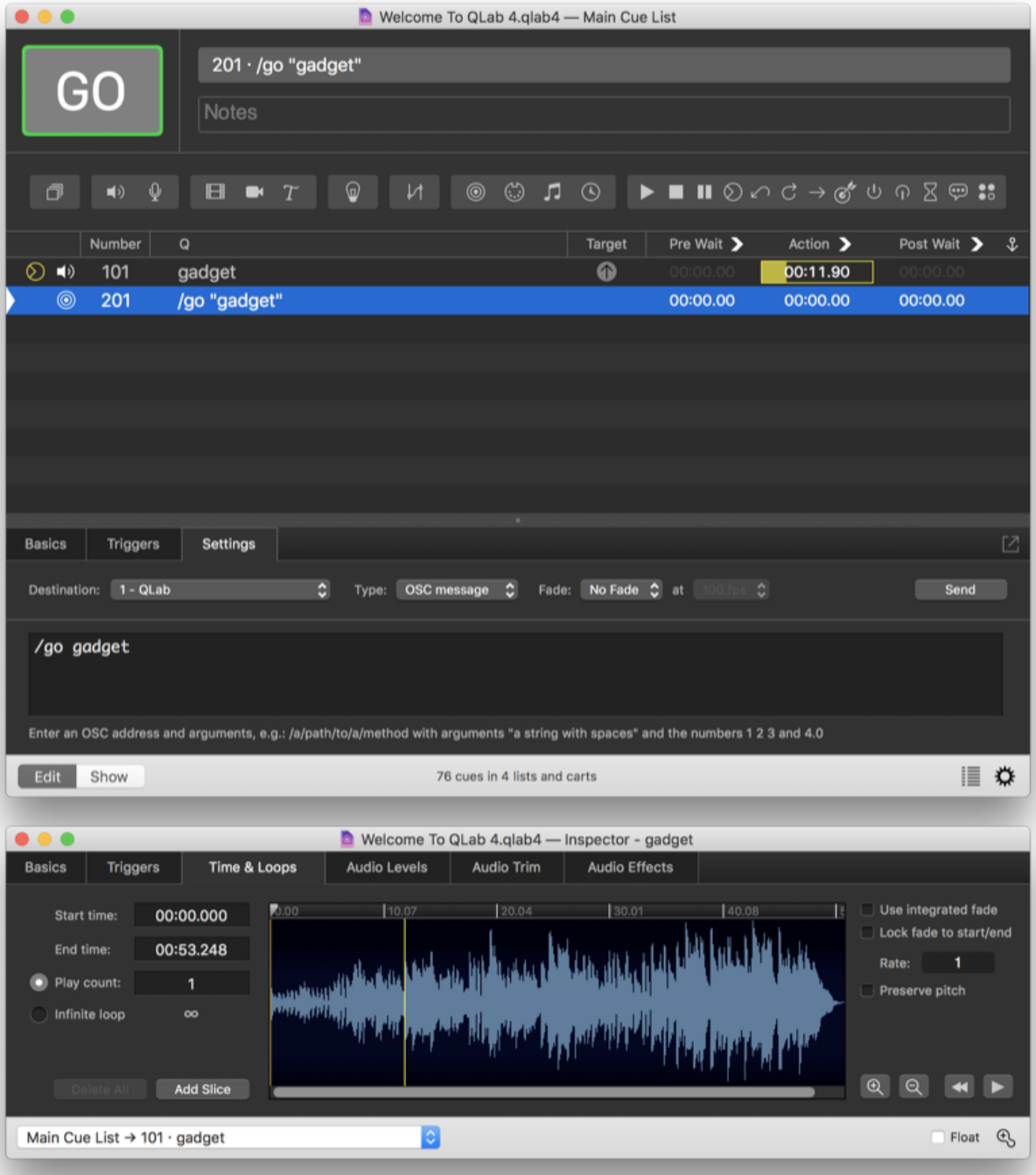
1. **Second trigger does nothing.** The cue will ignore all triggers while it's running. This is the default setting, and it's the way previous versions of QLab always behaved.
2. **Second trigger panics.** The cue will panic (fade out and stop) when it receives a second trigger.
3. **Second trigger stops.** The cue will stop when it receives a second trigger.
4. **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.
5. **Second trigger hard stops & restarts.** The cue will hard stop and then immediately restart when it receives a second trigger.

The checkbox marked *Also perform second trigger when releasing trigger* pertains only when a hotkey trigger, a MIDI trigger, or both are assigned to the cue. 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 4.4 and later 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 properties of the selected cue, cue 201, and a secondary inspector has been opened which is showing the Time & Loops tab of cue 101.

You can open secondary inspector windows by choosing *Inspector for Selected Cue* from the **View** menu, or by using the keyboard shortcut ⇧⌘**I**.

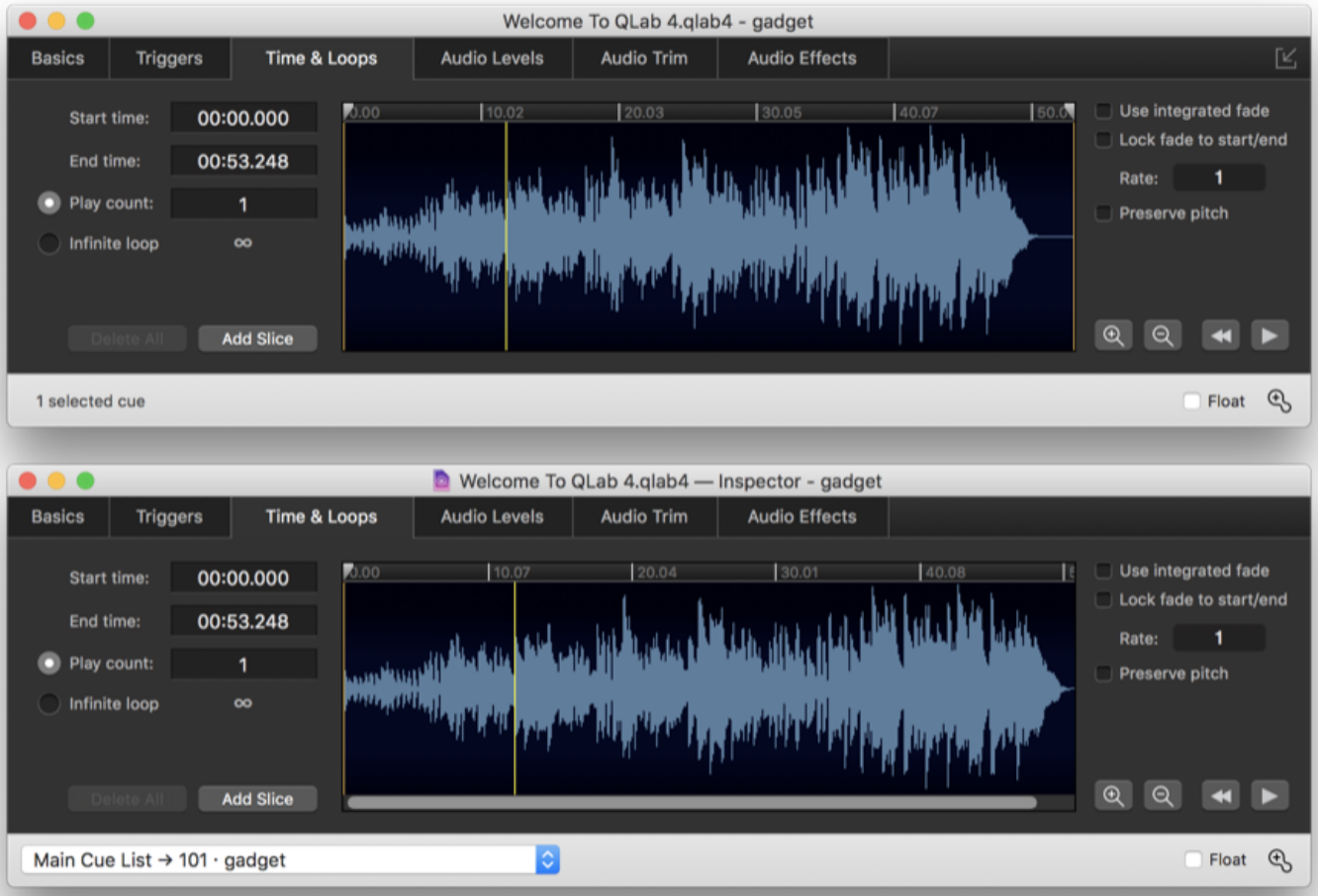
If you have multiple cues selected, QLab will open one inspector window for each cue.

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. You can manually select which cue a secondary inspector is inspecting by using the drop-down menu in the footer of the window. You can also open a new secondary inspector window for the same cue by clicking the button.

To keep a secondary inspector window floating on top of all other windows, check the box marked *Float*.

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 cue selection drop-down menu.

Group Cues

A Group cue is a type of cue which contains other cues. The cues within the 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 is ⌘**0**. If you create a Group cue while one or more other cues are selected, those cues will be placed inside the newly created Group. Once a group is created, it can be collapsed or expanded for visual simplicity using the gray disclosure arrow in the upper-left corner. Group cues behave exactly the same way whether they are collapsed or expanded.

When a Group cue is selected, the inspector shows three tabs: Basics, Triggers and Mode. Group cues set to Timeline mode also show a “Timeline” tab. Please refer to the section on the inspector to learn about the Basics and Triggers tabs of the inspector.

The Mode Tab

A Group cue can be set to one of four modes:

Timeline - Start all children simultaneously.

A Group in this mode has a green outline with square corners.

When a Group set to this mode is triggered, all child cues will start simultaneously and the playhead will advance to the next cue after the Group. Since child cues start simultaneously, the order of the child cues does not matter at all.

You can learn more about Timeline mode in the Timeline tab section below.

Start first child and enter into group.

A Group in this mode has a blue outline with rounded corners.

When a Group set to this mode is triggered, the first child cue will start and the playhead will advance to the next child cue within the Group. When the last child cue is triggered, the playhead will advance to the first cue after the Group

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 disclosure arrow in the upper-left corner of the box).

Start first child and go to next cue.

A Group in this mode has a blue outline with square corners.

When a Group set to this mode is triggered, the first child cue will start, and the playhead will advance to the next cue *after* the Group. Therefore, other cues within this type of group will get skipped over unless they are connected with auto-continues or auto-follows. By using auto-follows and/or auto-continues, the child cues within the Group can be made into a cue sequence, which will progress independently of the playhead.

Since the playhead advances to the cue after the Group, you can continue to press GO, triggering cues and advancing the playhead, while the cues within the Group are running.

Start random child and go to next cue.

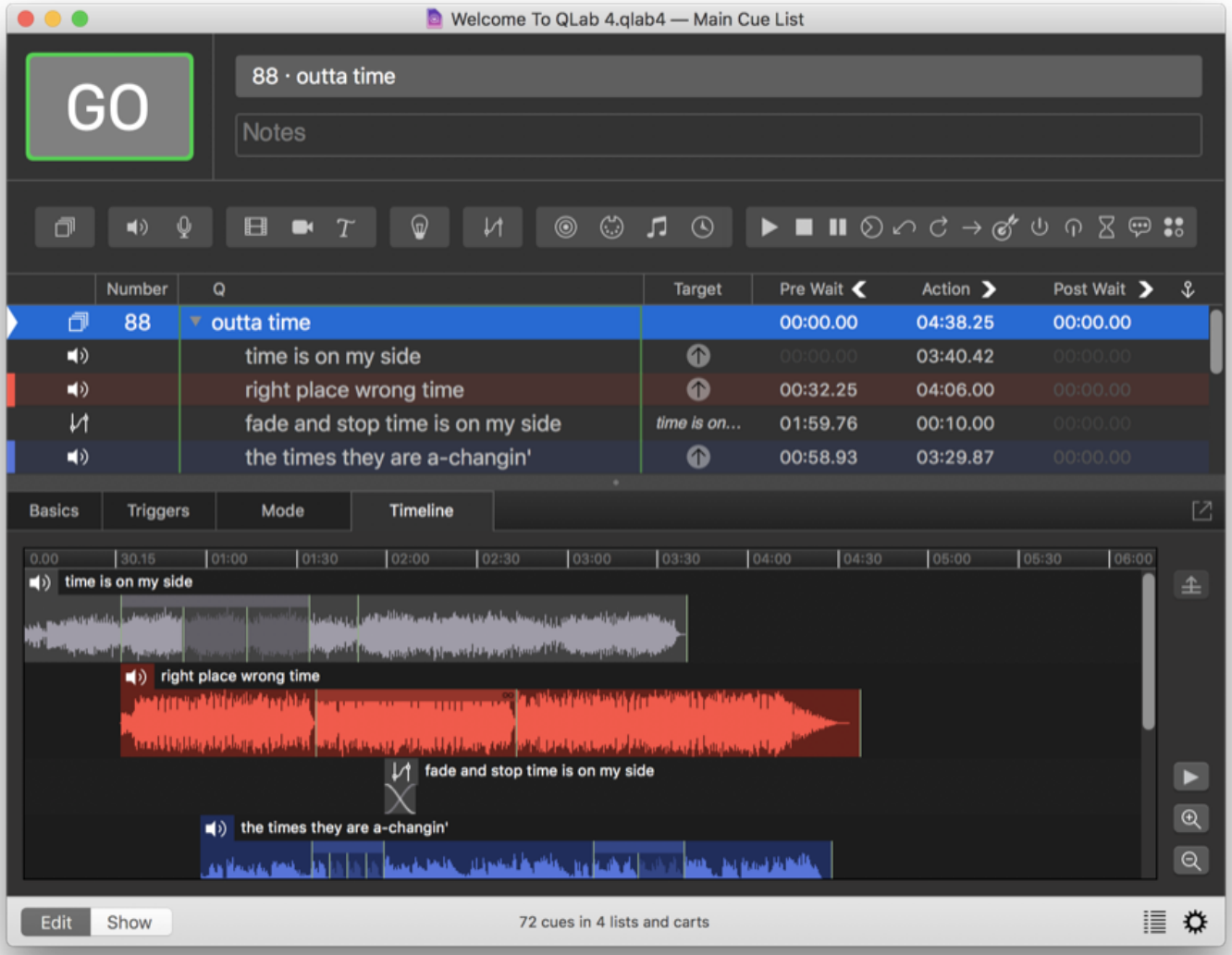
A Group in this mode has a purple outline with square corners.

When a Group set to this mode is triggered, a randomly selected child cue that is both armed and not currently playing will be triggered, and the playhead will advance to the first cue after the Group. Each armed cue within the Group will be triggered once before any of the cues is triggered a second time. This form of cueing is often referred to as “round robin” triggering.

Note: this is fairly different from the behavior of start-random Groups in QLab 3. If you have any questions about it, please don't hesitate to contact the support team and ask.

The Timeline Tab

Starting with QLab 4.3, Groups set to start all children simultaneously have a Timeline tab in the inspector, which display the Group's children in a visual timeline editor similar to the timeline found in many audio and video editing programs.



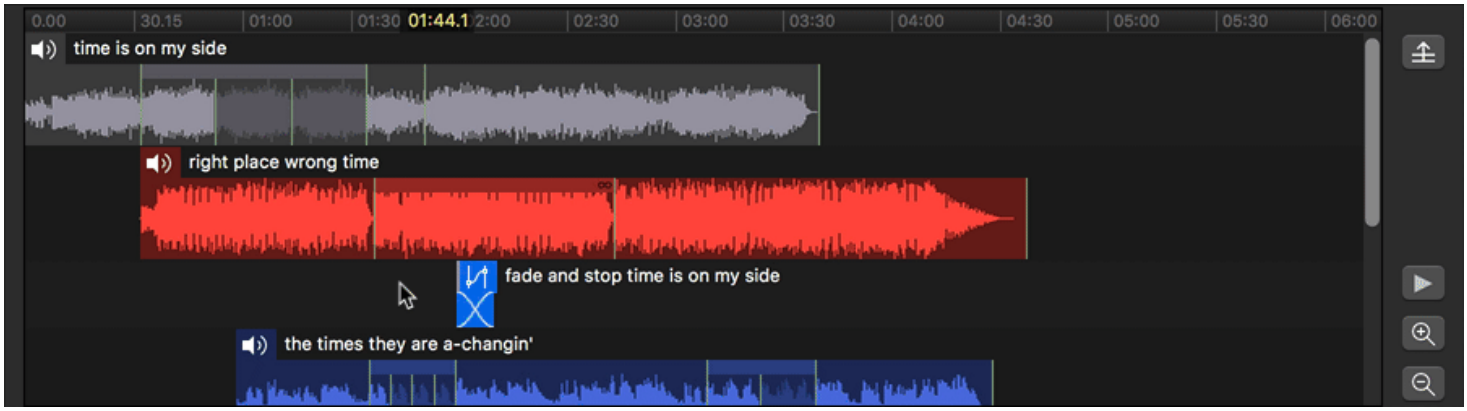
Each child cue in the Group appears in its own row in the timeline, in the same order as the cues are arranged in the Group. Cues show their type icon, their cue number, and their cue name. If a cue has a color, that color is also reflected in the timeline.

If an Audio cue or Video cue in the Group 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 (∞).

Cues can be adjusted in the timeline in several ways:

Drag

When the mouse 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.



Starting with QLab 4.4, 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.

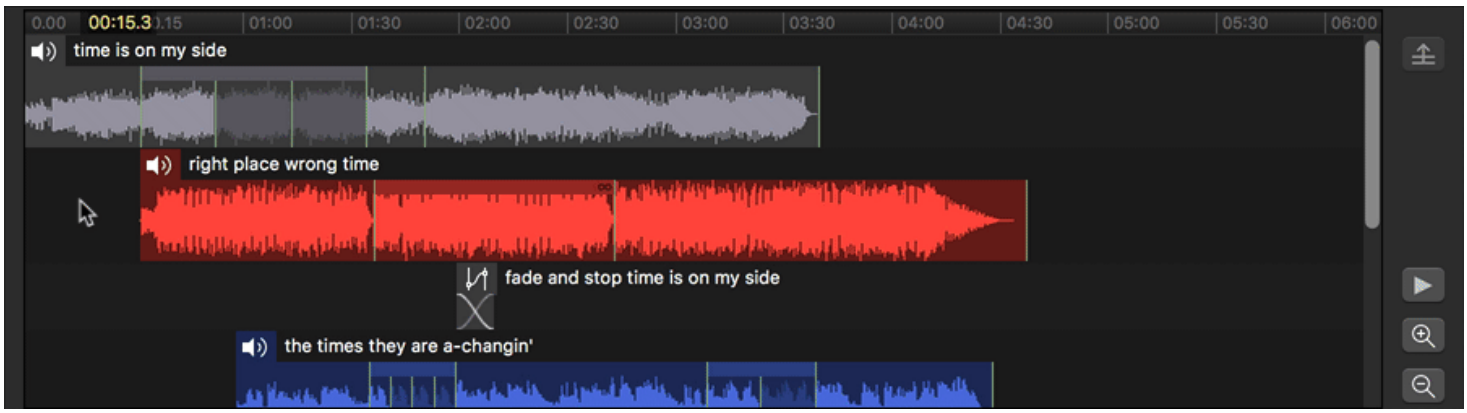
Nudge

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 or subtracting to 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.

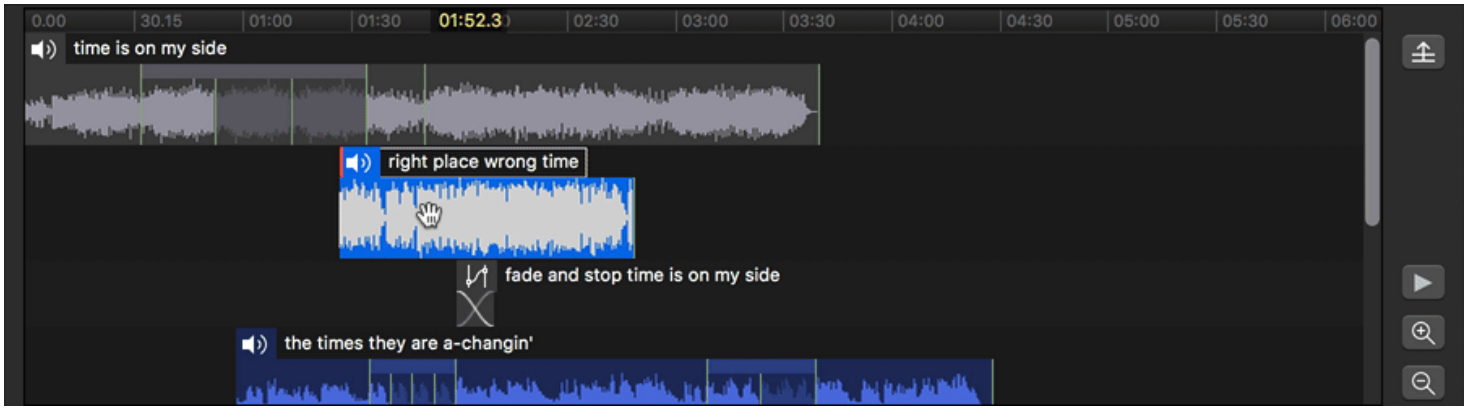
Trim

When the mouse 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

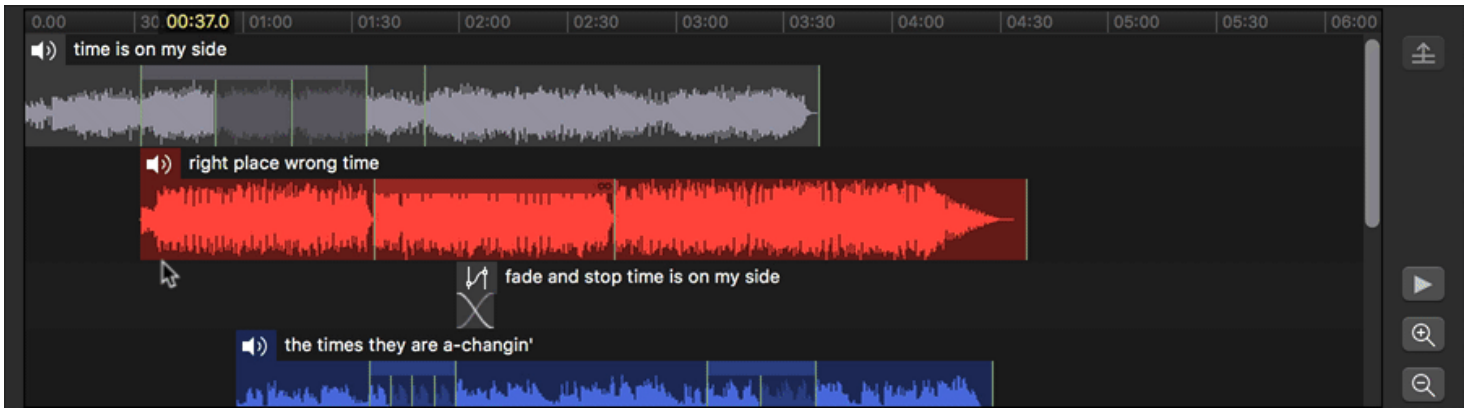
When the mouse is hovering over an Audio or Video cue, hold down the Option key to *slip* the cue, adjusting its start and end time without adjusting its pre-wait time or duration. Note that only cues without slices can be slipped.



Scale

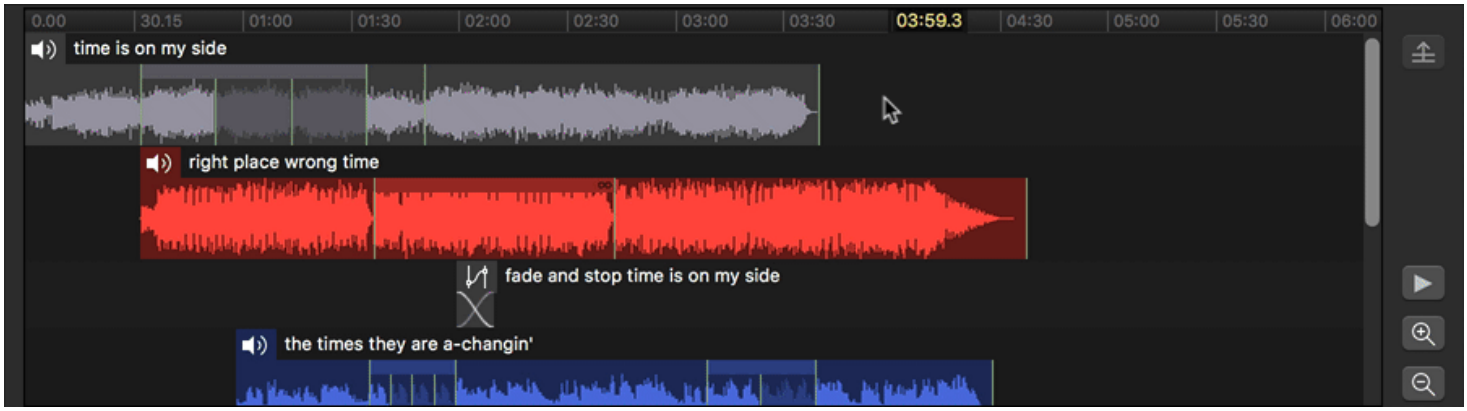
When the mouse is hovering over the bottom edge of a cue, it will turn into a bi-directional vertical arrow. Click and drag up or down to scale the display of the cue. This adjustment is purely visual and does not alter the behavior of the cue in any way.

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. Hold down the command key (⌘) 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 **⌘+/=** (that is to say, the **+/=** key with or without the shift key held down.)

Broken Cues

Group cues can become broken for the following reasons:

A cue in the group is broken.

Fix the cue inside the Group, and the Group will be fixed as well.

Timecode triggers require a Pro license.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the timecode trigger from this cue.

Cue Lists

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 by using the keyboard shortcut ⇧⌘**L**, or by clicking on the list 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.)

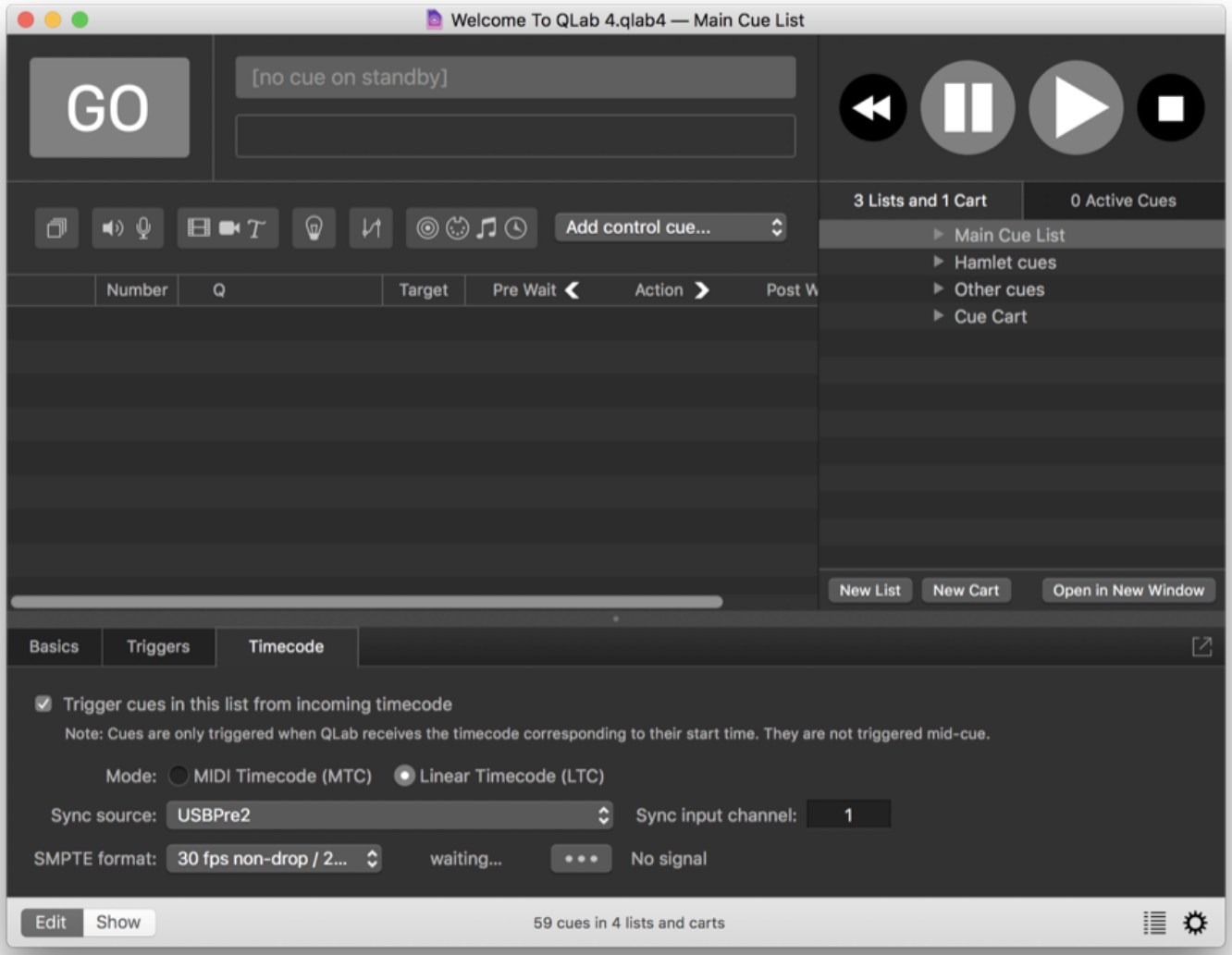
You can create a new cue list by clicking on the *New List* button. You can delete a list by selecting it and choosing *Delete* from the **Edit** menu or by using the keyboard shortcut ⌘⌫ (delete).

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.

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.

When a cue list is selected, the inspector shows three tabs: Basics, Triggers and Timecode. Please refer to the section on the inspector to learn about the Basics and Triggers tabs of the inspector.

The Timecode Tab



With a Pro Audio, Pro Video, or Pro Bundle license installed, QLab can listen to incoming timecode and start cues using timecode triggers, discussed in the Triggers section of the documentation. 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. Next, choose whether the cue list will listen for LTC (which stands for linear timecode or longitudinal timecode), or MTC (which stands for MIDI 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.

Once you've chosen LTC or MTC, choose an incoming MIDI device, or audio device and input channel, and the format (a.k.a. frame rate).

The [•••] button will open the Timecode window to allow you to watch the incoming timecode and verify that everything is working properly.

Rules for cue lists

1. 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. You can open additional windows to view separate cue lists and carts by selecting the list and clicking the *Open in New Window* button, or by right-clicking/control-clicking/two-finger-clicking on the name of the list or cart.
2. Triggering a cue list is the same thing as pressing GO for that list.
3. Cues can target cue lists, or cues in other lists. For example, a Start cue in your Main cue list could target a cue list called “Preshow music”, and a Fade cue in your Main cue list could fade out and stop that list. Alternately, the Start cue could trigger a specific cue in the list to start playing.

Cue Carts

New to QLab 4, a workspace can contain an unlimited number of cue carts alongside or in place of cue lists. A cart is a non-linear collection of cues meant to be operated without regard to sequence. Carts have no playhead, do not permit use of auto-follows or auto-continues, and have a grid-like interface designed for easy mouse clicking. While viewing a cart in Edit mode, the GO button becomes a Preview button to help reinforce the idea that there is nothing sequential about a cart.

QLab's cart powers come from a former sibling program called QCart. If you have any workspace files created in QCart, you can open them with QLab 4 and save them as QLab workspaces.

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 by using the keyboard shortcut ⇧⌘**L**, or by clicking on the list button on the right side of the workspace window footer. The sidebar has two tabs; the left tab shows cue lists and cue carts.

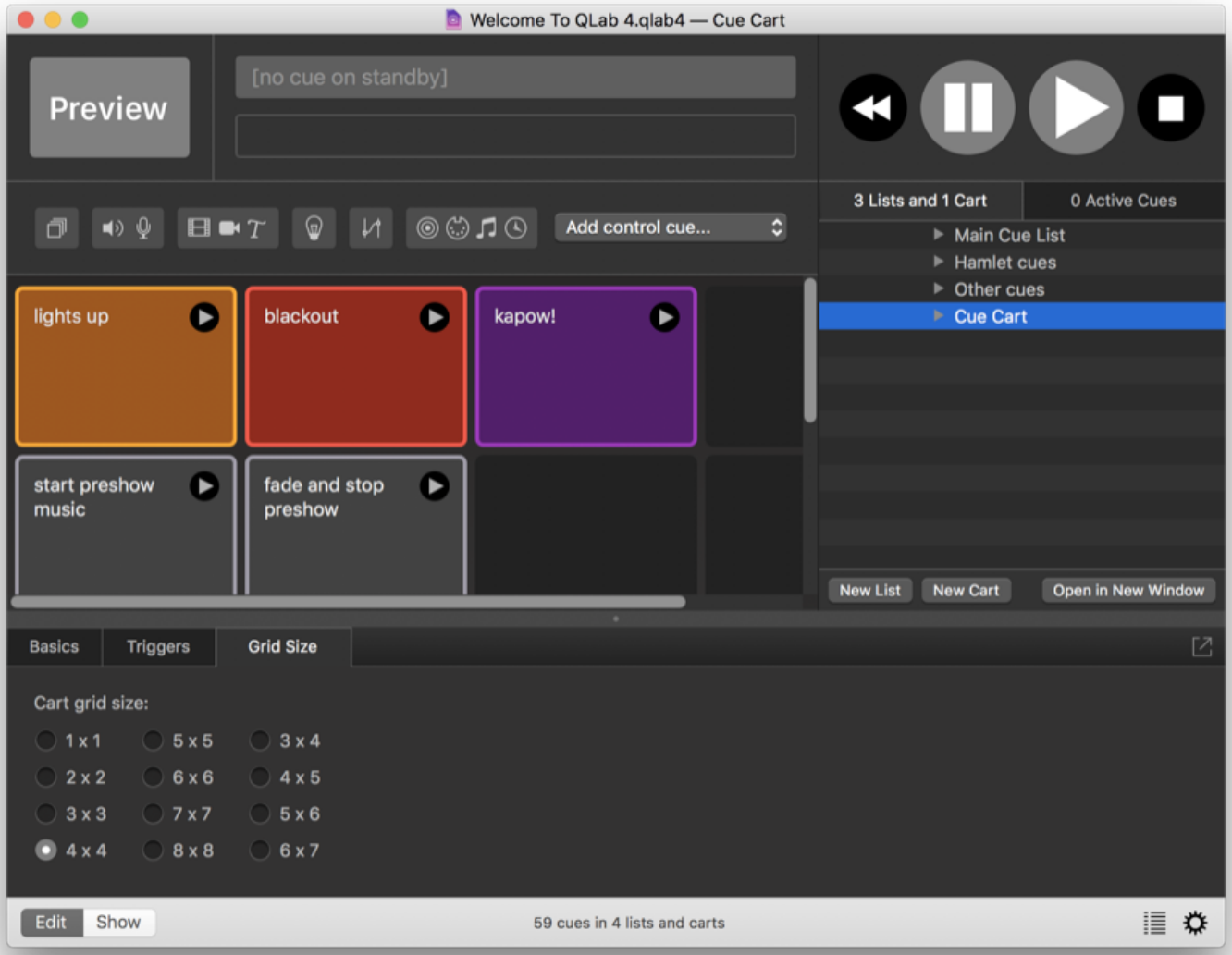
You can create a new cue cart by clicking on the *New Cart* button. You can delete a list by selecting it and choosing *Delete* from the **Edit** menu or by using the keyboard shortcut ⌘⌫ (delete).

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.

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.

When a cue cart is selected, the inspector shows three tabs: Basics, Triggers and Grid Size. Please refer to the section on the inspector to learn about the Basics and Triggers tabs of the inspector.

The Grid Size Tab



The Grid Size tab gives twelve options for the number of rows and columns of cells to display, from 1-by-1 up to 8-by-8. If you fill a cart with cues and then select a smaller grid size, the now-invisible cues will not be deleted, only hidden from view. If you choose a larger size once again, the cues will reappear.

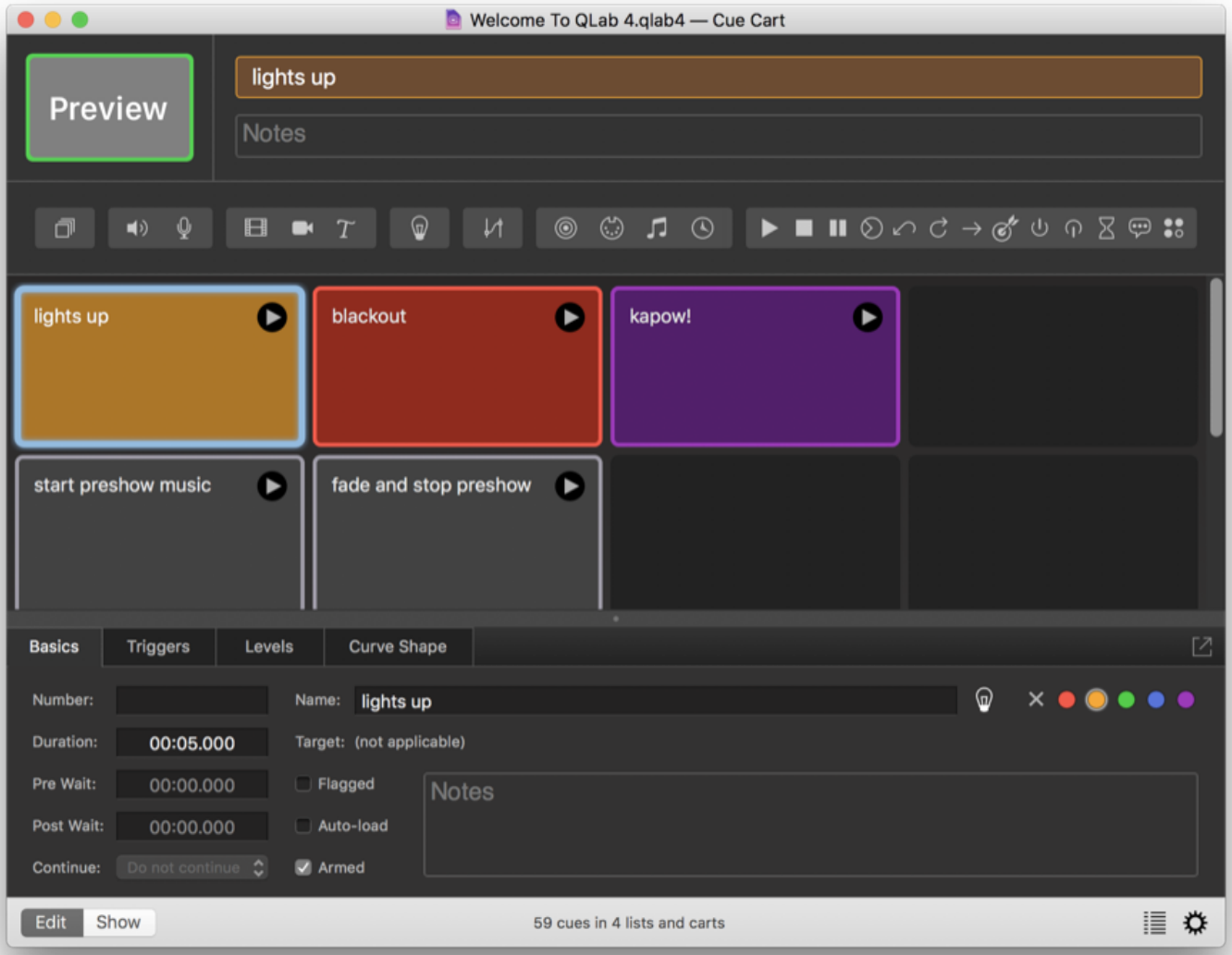
The cells themselves automatically resize to fit the available space, though they do have a minimum size.

Cues in carts

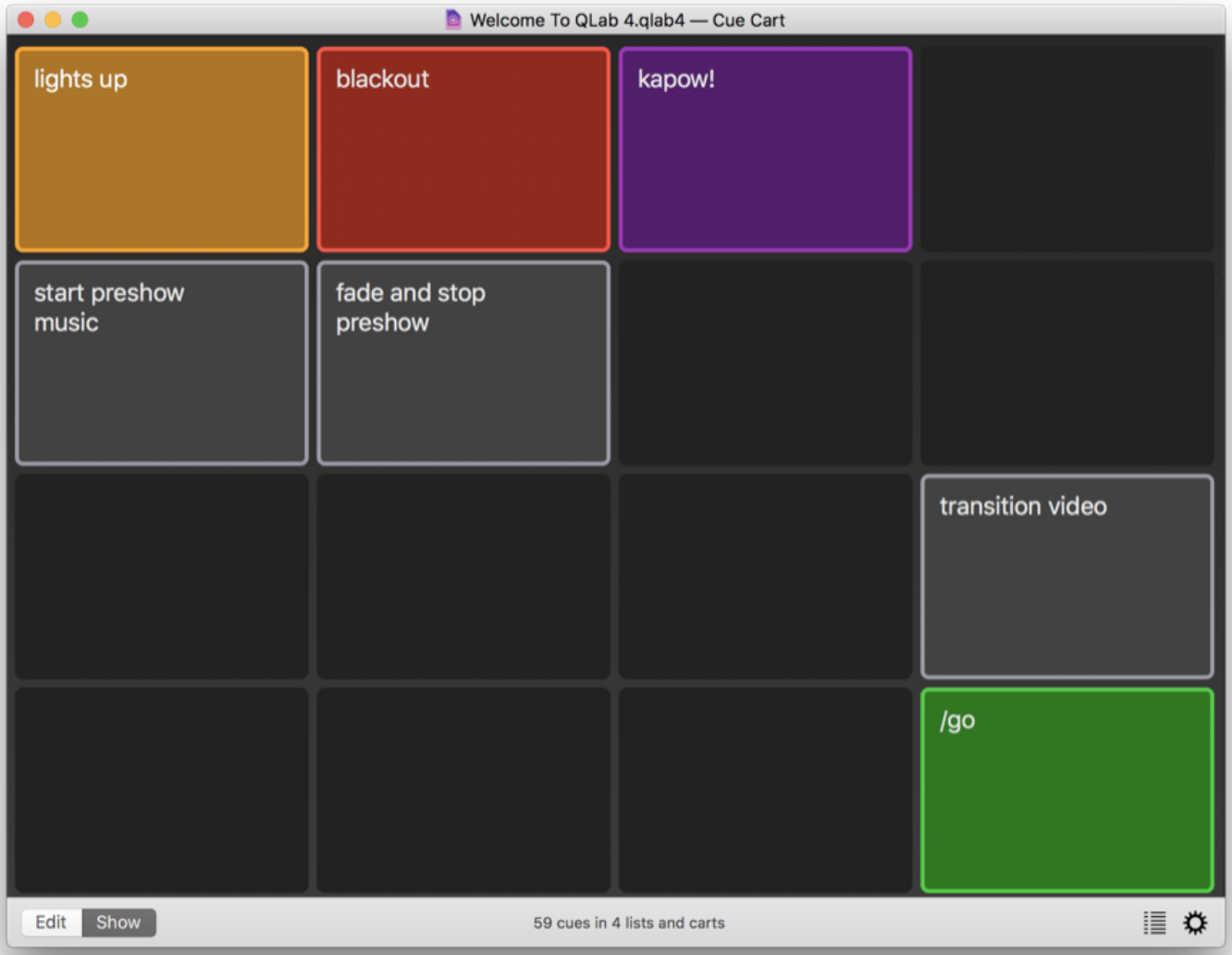
Cue carts can contain any type of cue except Group cues. You can create a cue and then drag it to whichever cell you like, or you can create a cue by dragging from the toolbar or toolbox directly to a cell. Once created, you can select the cue and edit it in much the same way as editing a cue in a cue list.

Fade cues in cue carts cannot be targeted by drag-and-drop. To set the target of a Fade cue, select the fade cue and type the cue number of its intended target into the Target field in the Basics tab of the inspector.

When the workspace is in Edit mode, cues can be triggered by clicking on the play button in the upper right corner of each cue.



When the workspace is in Show mode, the entire cell becomes a button which triggers the cue. Also, the masthead is hidden, since a cart in Show mode doesn't have a selected cue to show information about.



Rules for cue carts

1. 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. You can open additional windows to view separate cue lists and carts by selecting the list and clicking the *Open in New Window* button, or by right-clicking/control-clicking/two-finger-clicking on the name of the list or cart.
2. Triggering a cue cart in Edit mode will trigger the selected (or most recently selected) cue; triggering the cart in Show mode will load all cues in the cart.
3. Cues in carts can target lists, carts, or cues in other lists or carts. For example, you could create a cart containing a Start cue, Pause cue, and Stop cue all targeting another cue list, thereby creating a remote control panel for that list.

Cue Sequences

An essential concept in QLab is the *cue sequence*. A cue sequence is simply two or more cues that are triggered together from one single press of the GO button (or one single incoming MIDI command, MSC command, OSC command, hotkey press, etc.) Cue sequences can be built in three different ways:

- By connecting cues with auto-follows,
- By connecting cues with auto-continues,
- By putting cues into a Group cue set to "start all children simultaneously."

You can download an example workspace which explores cue sequences here.

Auto-follow

A cue set to auto-follow will trigger the next cue after it (the initial cue) has completed. For example, let's say we have two cues, cue 10 and cue 11. Cue 10 has a duration of three seconds and is set to auto-follow. When cue 10 is standing by and you press GO, QLab will start cue 10, wait three seconds, and then start cue 11 automatically. If you change the length of cue 10 to eight seconds and then run the cue sequence again, QLab will start cue 10, wait eight seconds, and then start cue 11. The following cue start after the prior cue finishes.

To set a cue to auto-follow, select *Auto-follow* from the drop-down menu in the bottom-left corner of the Basics tab in the inspector. An arrow with a circle on 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 drop-down menu.

You can also click in the rightmost column of the cue list to bring up a pop-up menu which lets you choose amongst the three continue modes.

> Note that this pop-up menu is new is QLab 4.6. In previous versions of QLab, clicking in the rightmost column cycles through continue modes without displaying the menu.

The default keyboard shortcut to cycle through continue modes of the selected cue or cues is **C**.

Auto-continue

A cue set to auto-continue will will trigger the next cue after its own 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 trigger simultaneously. If a post-wait is added, the post-wait will begin to elapse at the moment that the cue is triggered. Once that post-wait elapses, the second cue will trigger.

For example, say cue 12 is standing by, has a post-wait of 3 seconds, and cue 13 is next in the cue list. Pressing GO will start cue 12 immediately, wait 3 seconds, and then start cue 13, and it doesn't matter whether or not cue 12 has finished yet.

To create an auto-continue, select *Auto-continue* from the drop-down menu in the bottom-left corner of the Basics tab in the inspector. An arrow with a dotted line 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 drop-down menu.

Timeline Groups

Cues in a Group set to Timeline mode will start simultaneously when the Group cue is triggered.

To create a Timeline Group, you can either make a new Group cue and drag cues into it, or you can select two or more cues which you want to put into the Group, and *then* create the Group cue. The selected cues will automatically be placed within the Group.

The default keyboard shortcut to create a Group cue is ⌘**0**.

Once the Group cue is created, select it, then go to the Mode tab in the inspector and select *Timeline - Start all children simultaneously*.

For more information about Groups, including about other types of Groups, please refer to the Group Cues section of this documentation.

Pre-wait

If a cue has a pre-wait time set, triggering that cue will start the pre-wait counter, and the cue will start once the pre-wait time has elapsed. You can assign a pre-wait to a cue by double-clicking in the *pre-wait* column of the cue list and entering a time. You can also use the keyboard shortcut, **E** by default, or enter the pre-wait time in the Basics tab.

A pre-wait is a valuable tool to use in conjunction with Timeline Groups as a way to create a timeline of cues. For example, let's say you want a series of cues 20, 21, 22, and 23 to each start two seconds apart after pressing GO. You can put the cues in a Timeline Group, and assign a pre-wait of two seconds to cue 21, four seconds to cue 22, and 6 seconds to cue 23. When the Group cue is triggered, all four cues will start simultaneously, but the pre-wait times of cues 21, 22, and 23 will elapse before they actually begin.

You can then adjust the pre-wait times of each of the four cues without altering the timing of the other three.

Disarmed Cues

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.

Importing Go Button Shows

Shows exported from Go Button can be converted into QLab workspaces by either double-clicking on the Go Button document in the Finder, dragging and dropping the Go Button document onto the QLab icon, or by choosing *Open Workspace...* from the **File menu** and selecting a Go Button document.

QLab will automatically save the Go Button show as a new workspace to the same directory as the document being imported. QLab will create a unique name if there a workspace with the same name already exists in the import location.

When importing a `.gobutton` file, QLab will expect the media files to be located in same folder as the `.gobutton` file. When importing a `.gobundle` file, QLab will unpack the audio files from the bundle into a new "audio" folder inside the same folder as the `.gobundle` file.

Conversion Notes

- Go Button cues are imported into a cue list.
- Go Button hits are imported into a cue cart. Since cues within cue carts cannot have pre-wait times, any pre-waits on imported hits will be discarded.
- Basic cues and hits that have a media file assigned to them are imported as Audio cues. Basic "empty" cues and hits are imported as Wait cues.
- Cues and hits with complex settings are imported as Group cues containing the Audio or Wait cue along with the QLab cues needed to recreate similar Go Button behavior.
- Fade ins and fade outs in Go Button cues become Fade cues. Fade ins and fade outs in Go Button hits are converted to integrated fade envelopes in the Audio cues for each hit.
- Cues created from Go Button cues and hits that are "disabled" will be disarmed.
- Cues created from Go Button cues and hits can only "Duck Others" within their own cue list or cart in QLab.
- Triggers in Go Button cues and hits that are assigned to the "Start" action are converted into Hotkey, MIDI, or Wall Clock triggers. Triggers assigned to other actions are discarded. If a particular type of Go Button trigger has than one "Start" trigger, only one of those triggers is imported.
- Go actions "Fade & Stop Others" and "Stop Others" become Network cues. The "Destination" setting for these cues must remain set to `localhost` to behave correctly. Remember too that Network cues require a paid license to function.

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 same position as the listed key will work, but will simply have the wrong label.

Key Symbols

Symbol	Meaning
⌘	command
⇧	shift
^	control
⌥	option
⌫	delete

Default Key Map

Most of these keyboard shortcuts can be edited in the Key Map section of Workspace Settings.

Command	Key
GO	space
Panic All	escape
Hard Stop All	esc esc
Pause All	[
Resume All	]
Preview	V
Load Selected Cue	L
Pause/Resume Selected Cues	P
Panic Selected Cues	S
Hard Stop Selected Cues	SS
Edit Cue Number	N
Edit Cue Name	Q
Edit Cue Notes	O
Edit Cue Target	T
Edit Cue Pre-wait	E
Edit Cue Action (duration)	D
Edit Cue Post-wait	W
Cycle Cue Continue Mode	C
Flag/Unflag	F

Menu Keyboard Shortcuts

QLab

Command	Key
Hide QLab	⌘H
Hide Others	⌥⌘H
Quit QLab	⌘Q

File

Command	Key
New Workspace	⌘N
New From Template	⇧⌘N
Open Workspace...	⌘O
Close	⌘W
Save	⌘S
Save As...	⇧⌘S

Edit

Command	Key
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	⌘+
Smaller	⌘-
Show Colors	⌥⇧⌘C
Copy Style	⌥⌘C
Paste Style	⌥⌘V

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

Command	Key
Light	⌘6
Fade	⌘7
Network	⌘8
MIDI	⌘9

Tools

Command	Key
Load to time...	⌘T
Renumber selected cues...	⌘R
Delete numbers of selected cues...	⌘D
Jump to cue...	⌘J
Jump to selected cue's target	⇧⌘J
Turn on/off live fade preview	⇧⌘P

When a Fade cue is selected, the following Tools are also available:

Command	Key
Set Audio Levels From Target	⇧⌘T
Set Video Geometry From Target	^⌥⌘V
Revert Fade Action	⇧⌘R

View

Command	Key
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 Next Cue	⌘↓
Select Previous Cue	⌘↑
Move Playhead To Next Cue	⇧⌘↓
Move Playhead To Previous Cue	⇧⌘↑
Move Playhead To Next Cue Sequence	=
Move Playhead To Previous Cue Sequence	-
Select Next Tab	⌘→
Select Previous Tab	⌘←
Enter Edit Mode	⇧⌘[
Enter Show Mode	⇧⌘]

Window

Command	Key
Minimize	⌘M

Command	Key
Workspace Settings	⌘,
Workspace Status	⇧⌘W
Audition Window	⇧⌘A
Override Controls	⇧⌘O
Light Dashboard	⇧⌘D

Other keyboard shortcuts

Command	Key
Expand all Group cues	>
Collapse all Group cues	<
Set start time to current time in the Time & Loops waveform view	⇧I
Set end time to current time in the Time & Loops waveform view	⇧O
Add slice at current time in the Time & Loops waveform view	M
Zoom in on the Time & Loops waveform view	⌘+ / ⌘=
Zoom out on the Time & Loops waveform view	⌘-
Zoom in horizontally on the Group Timeline view	⌘+/=
Zoom out horizontally on the Group Timeline view	⌘-
Zoom in vertically on the Group Timeline view	⇧⌘+/=
Zoom out vertically on the Group Timeline view	⇧⌘-
Nudge selected cues +0.1 second in the Group Timeline view	⌥→
Nudge selected cues -0.1 second in the Group Timeline view	⌥←
Nudge selected cues +0.01 second in the Group Timeline view	⇧⌥→
Nudge selected cues -0.01 second in the Group Timeline view	⇧⌥←

Licenses

QLab 4 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 may also be installed on Mac which are not connected to the internet, and starting with QLab 4.1, licenses can also be installed onto a USB drive connected to your Mac.

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.

Rent-to-own Licenses

Rent-to-own licenses can be 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.

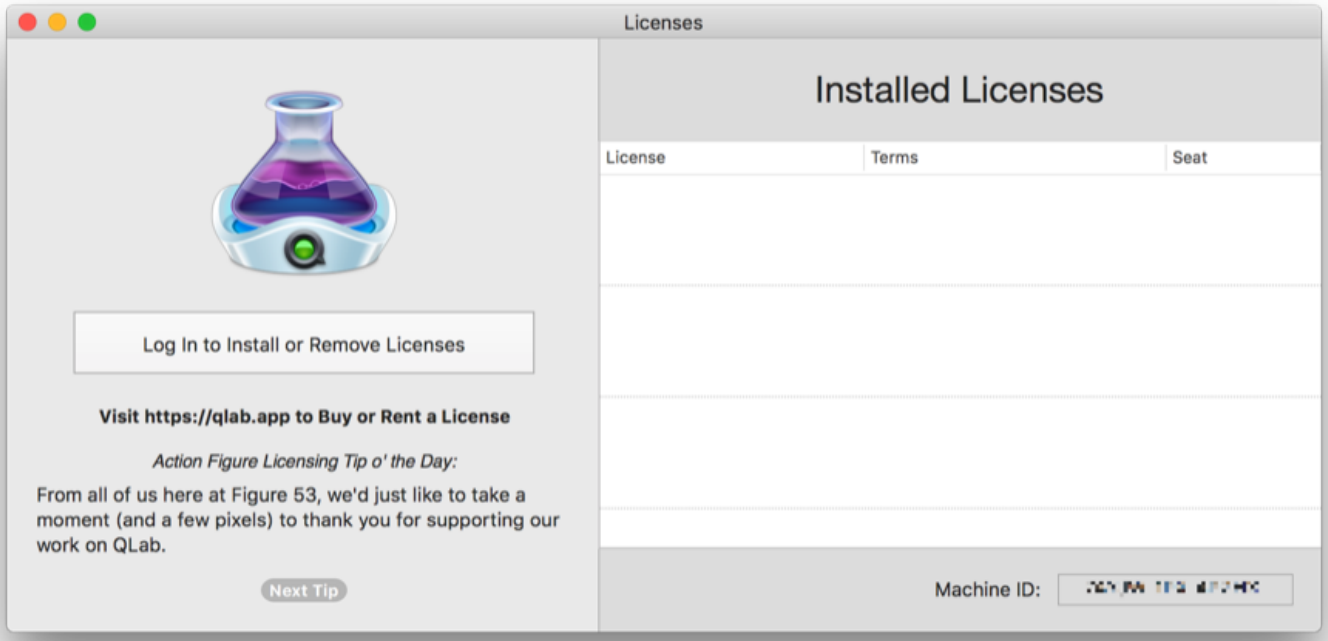
The Figure 53 shop uses your computer's clock to determine the timezone for your rental license.

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.

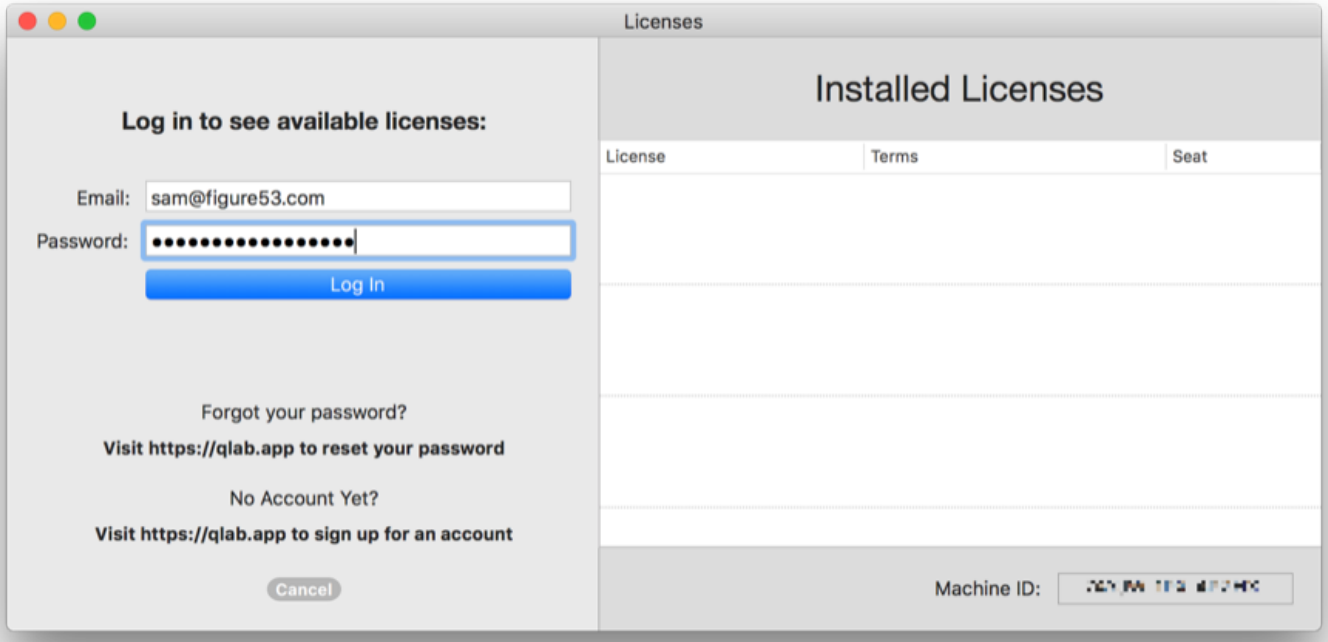
Once you have rented a license for 110 days, you will get a free standard license of the same type. The rental days do not need to be consecutive.

Installing Licenses

In QLab, choose *Manage Your Licenses...* from the **QLab menu** to open the License Manager:



Click *Log In to Install or Remove License* and enter your email and password.

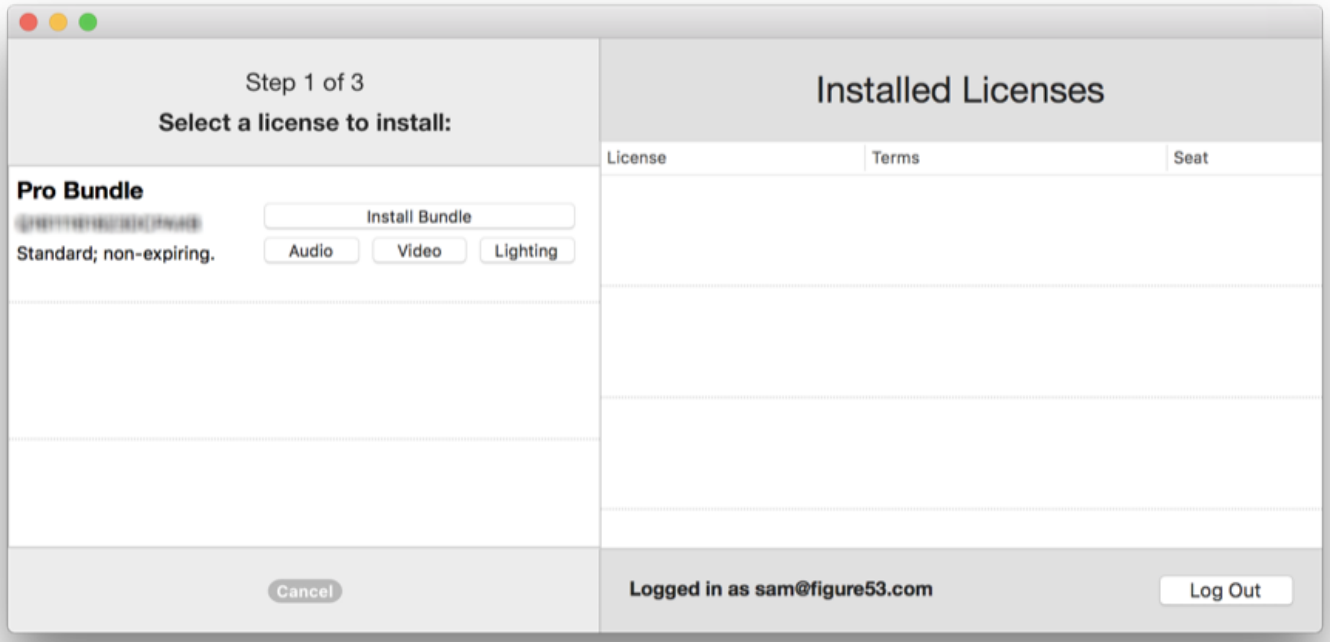


If you've forgotten your password, or if you have not set a password for your account, click *Visit Figure53.com to reset your password* which will open a web browser and allow you to reset your Figure 53 password.

If you do not yet have a QLab account, you'll need to create one in order to buy a license and install it.

Select a License

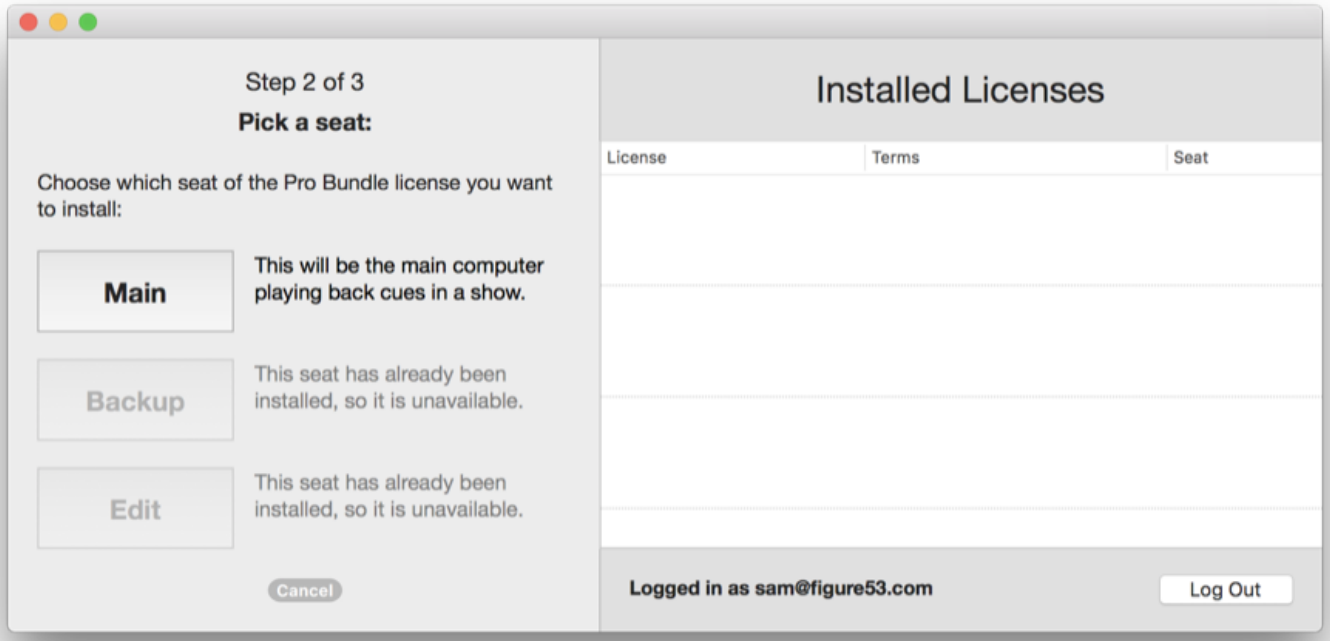
Once you've logged in, you'll be presented with a list of the licenses in your account.



Click the *Install...* button to select the license for installation. Pro Bundle licenses may be installed all together, or you can individually install the Audio, Video, or Lighting portion of the license.

Select a Seat

Once you choose a license, you'll be presented with a choice of which seat to install.

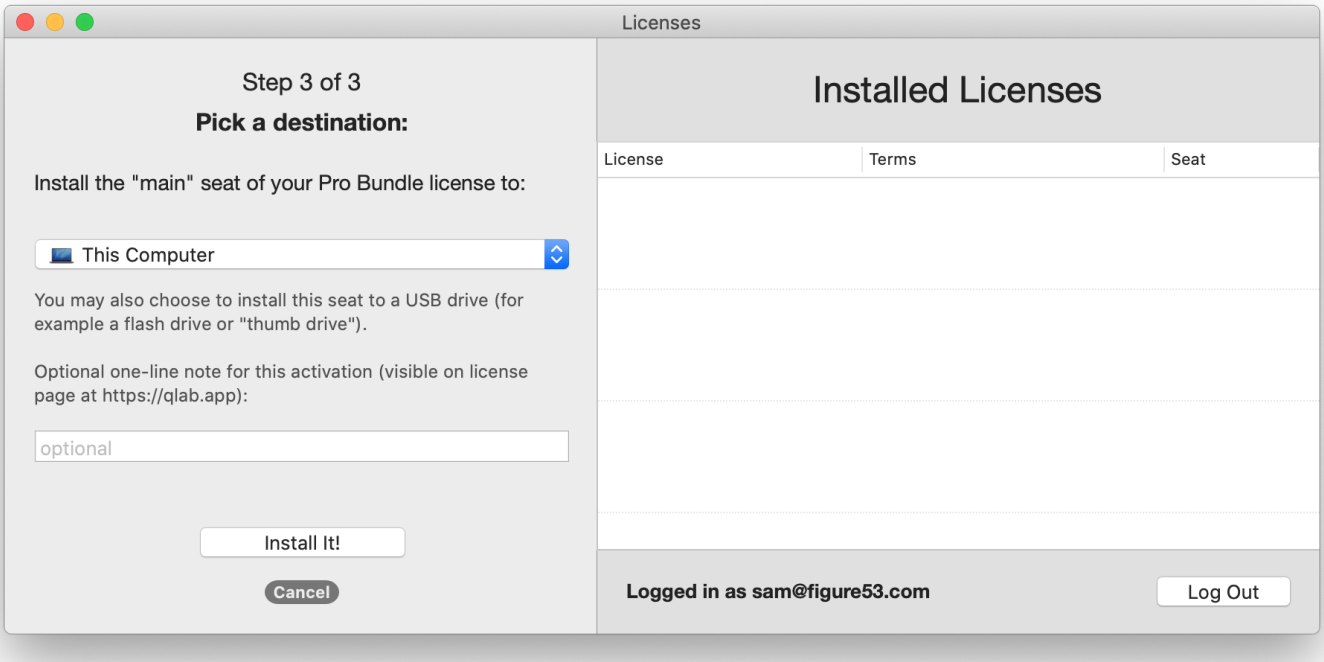


Each QLab license can be installed on up to three Macs at a time, with three specific uses:

1. **Main.** The Main computer is used to run cues during your show. A show that uses one Mac for playing back Audio and another Mac for playing back Video has two Main computers.
2. **Backup.** The Backup computer is there in case something goes wrong with the Main computer. You might have your Backup computer hooked up to your playback system, with a system in place to seamlessly switch over from the Main (this is often called a “hot spare” or “hot backup”), or you might have your Backup computer disconnected, but ready to plug in at a moment’s notice (called a “cold spare” or “cold backup.“) In either case, the Backup shouldn’t do anything that’s audible or visible to the audience during the show unless it takes over for the Main computer.
3. **Edit.** The Edit computer is used to work on your QLab workspaces in rehearsal, at your studio, or at home. If you have a visiting artist working on your show, you might loan your Edit seat to them so that they can work on their cues using their own laptop, rather than having to sit in the technical booth to work on cues. The Edit computer should never be used for anything that’s audible or visible to the audience.

Select a Destination

Once you choose a seat, you’ll be presented with a choice of destination:

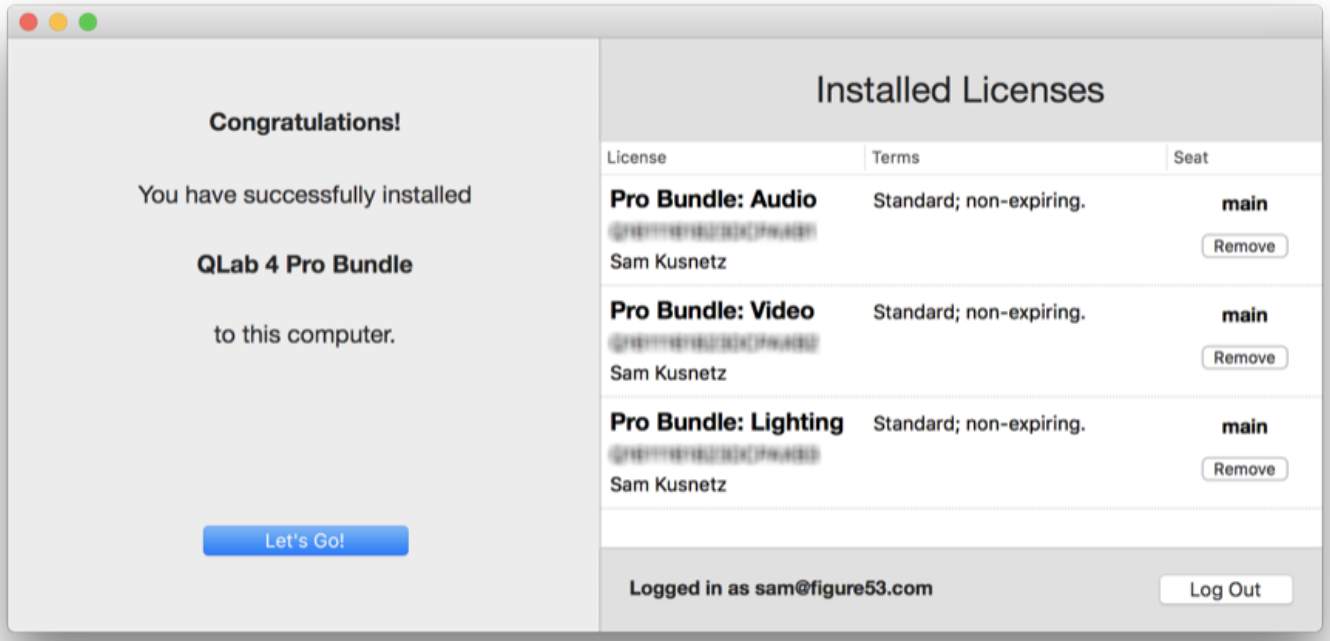


If you choose *This Computer*, the license seat will be installed onto your Mac, and will be accessible to all user accounts on that Mac. This is the most common way to install a license.

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.

Starting with QLab 4.6.5, 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 your licenses on your account page on qlab.app.

Click *Install it!* to complete the installation process. The License Manager will then show you the completed installation.



USB Licenses

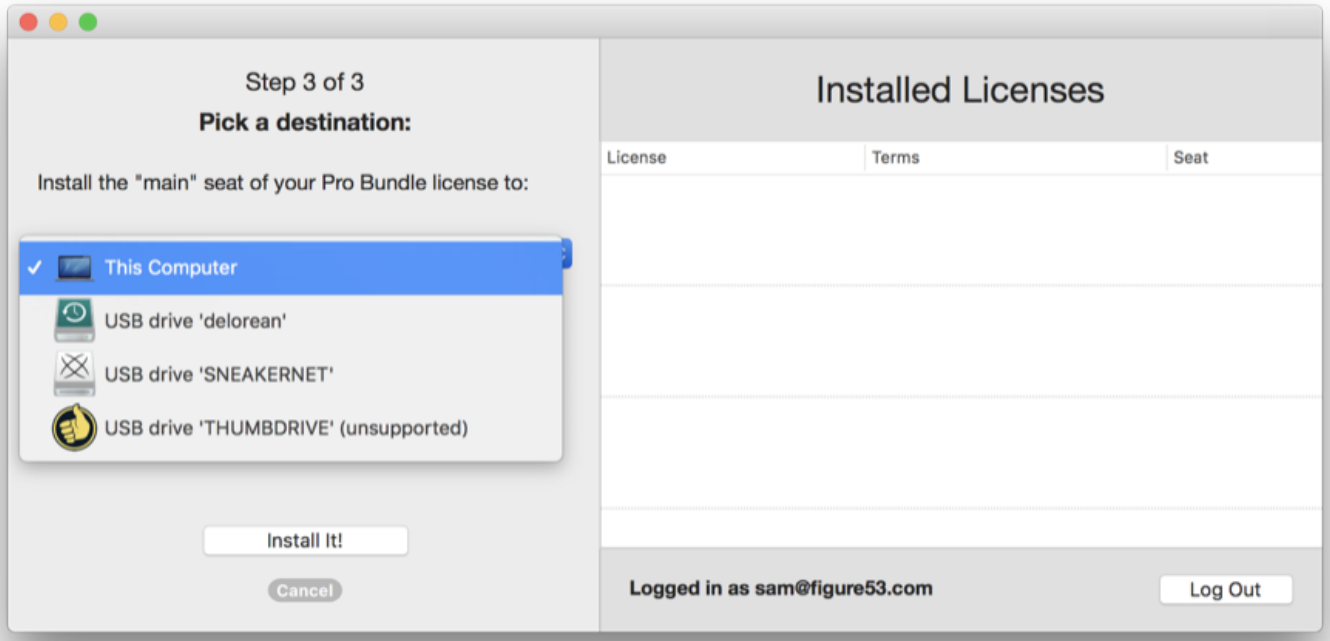
Starting with QLab 4.1, license seats can be installed onto USB drives. This makes it easy to loan a QLab seat to a short-term collaborator, use licenses with Macs that aren't connected to the internet, or be ready with a license when you're not sure what Mac you'll be using.

Please note that USB licenses can only be used with QLab 4.1 and later.

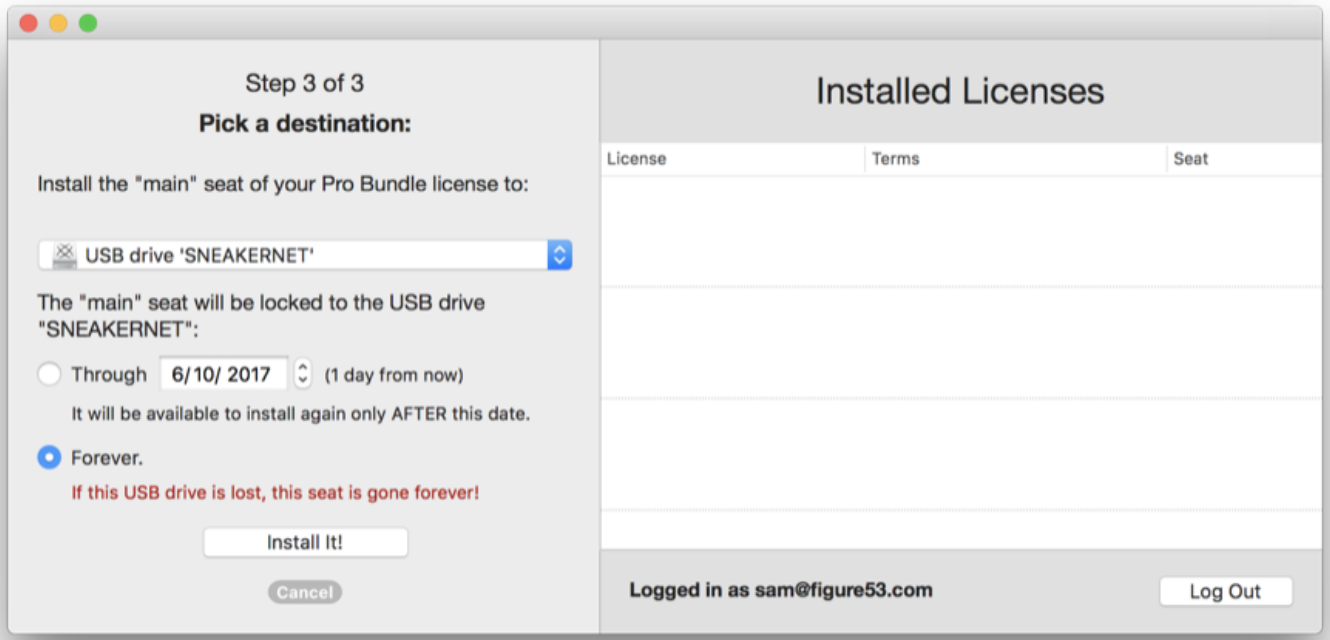
Because not every USB device manufacturer adheres to the same standards, only USB drives made by the following manufacturers can be used for QLab licenses. This restriction makes it possible for us to provide the highest possible guarantee of stability for USB licenses without resorting to using something like an iLok or Sentinel key.

- Corsair
- HGST
- HP
- Kingston
- LaCie
- Lexar
- OCZ
- SanDisk

To install a license to a USB drive, you'll need to use a Mac that's connected to the internet and has 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 to install 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.



Very Important Thing To Understand

If you choose *forever* then your license seat will be **permanently and irrevocably installed on the USB drive.** This is dramatically different from installing a license seat onto a Mac, which can be un-done at any time.

If the USB drive is lost, stolen, or erased, **so is the license seat installed onto it.**

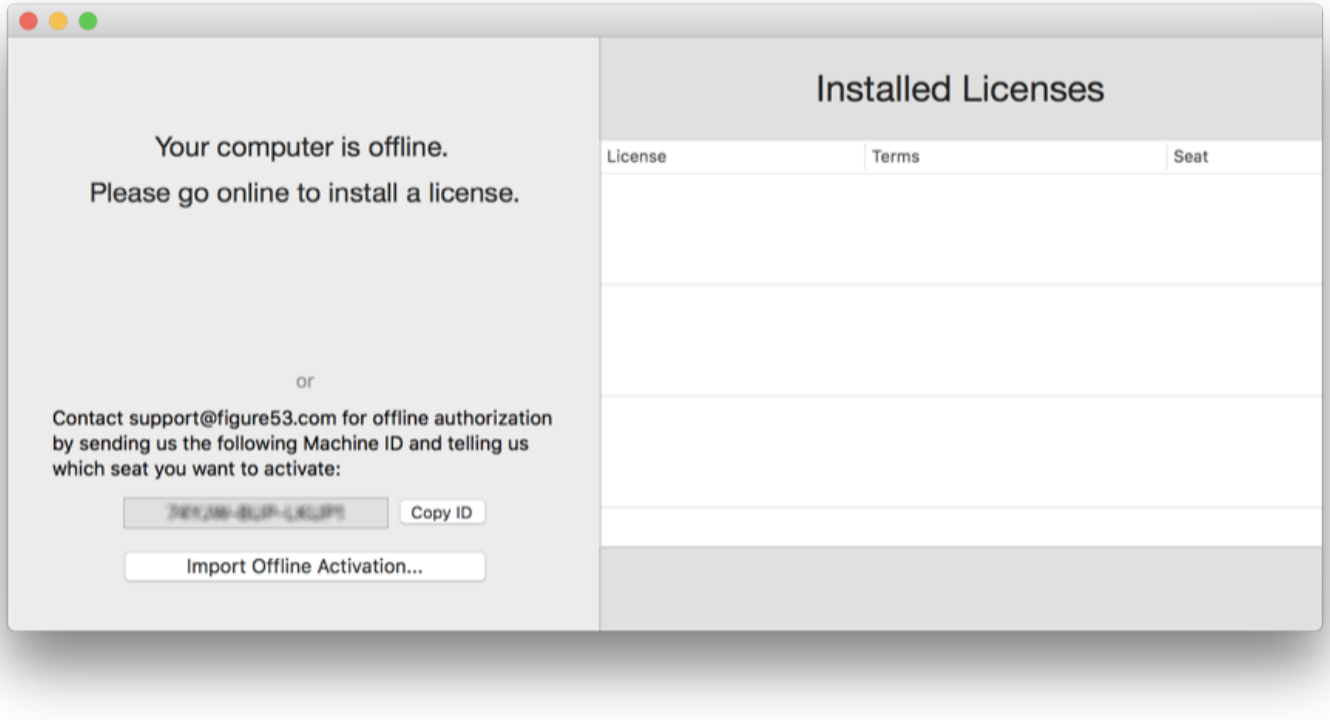
In the case of a damaged USB drive, please contact support@figure53.com, and we will tell you how to mail us the damaged drive and receive a replacement license.

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 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. When you open the License Manager on a Mac that's not connected to the internet, it will look like this:

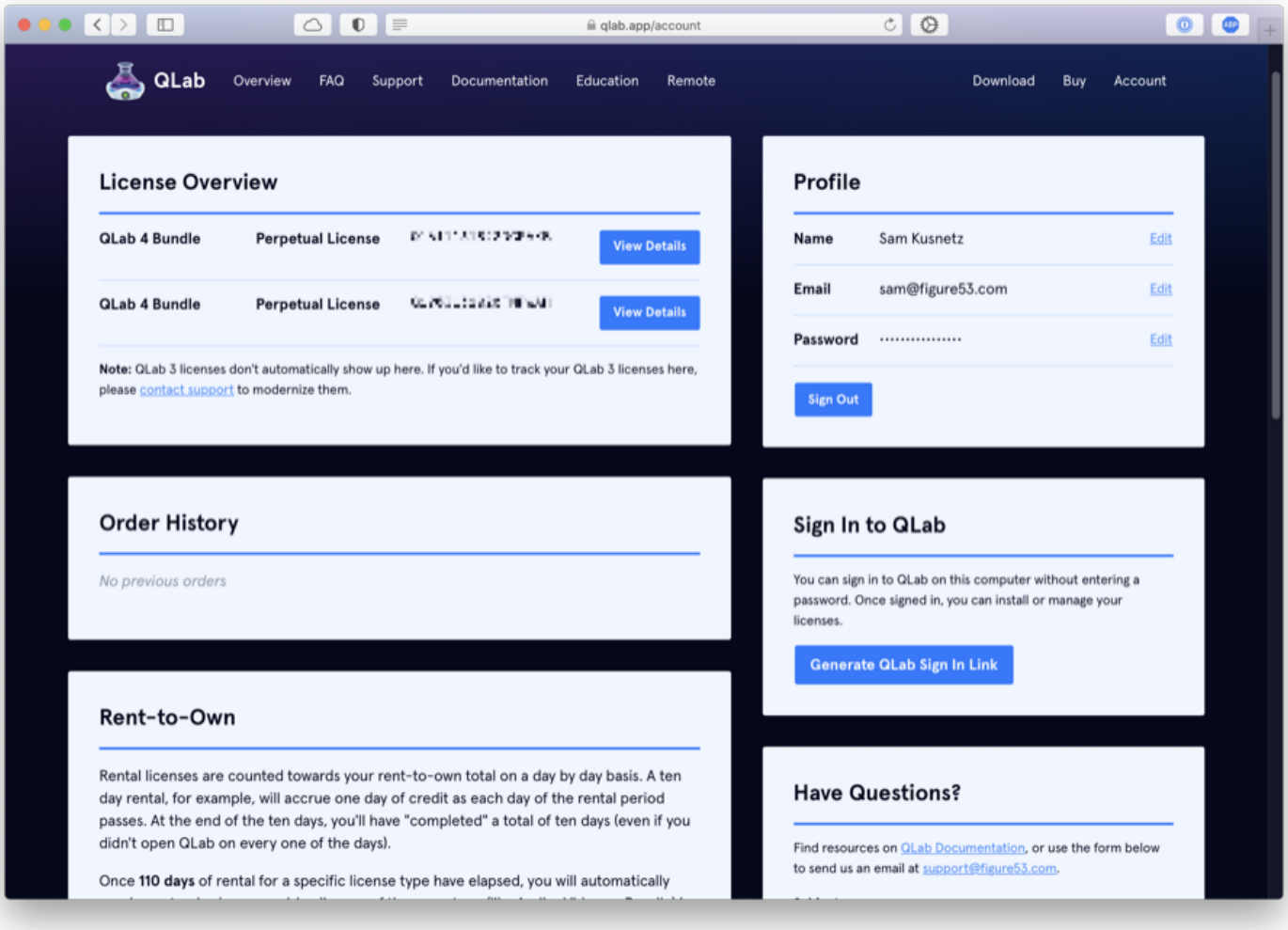


You can click *Copy ID* to copy that code for easy copy-pasting.

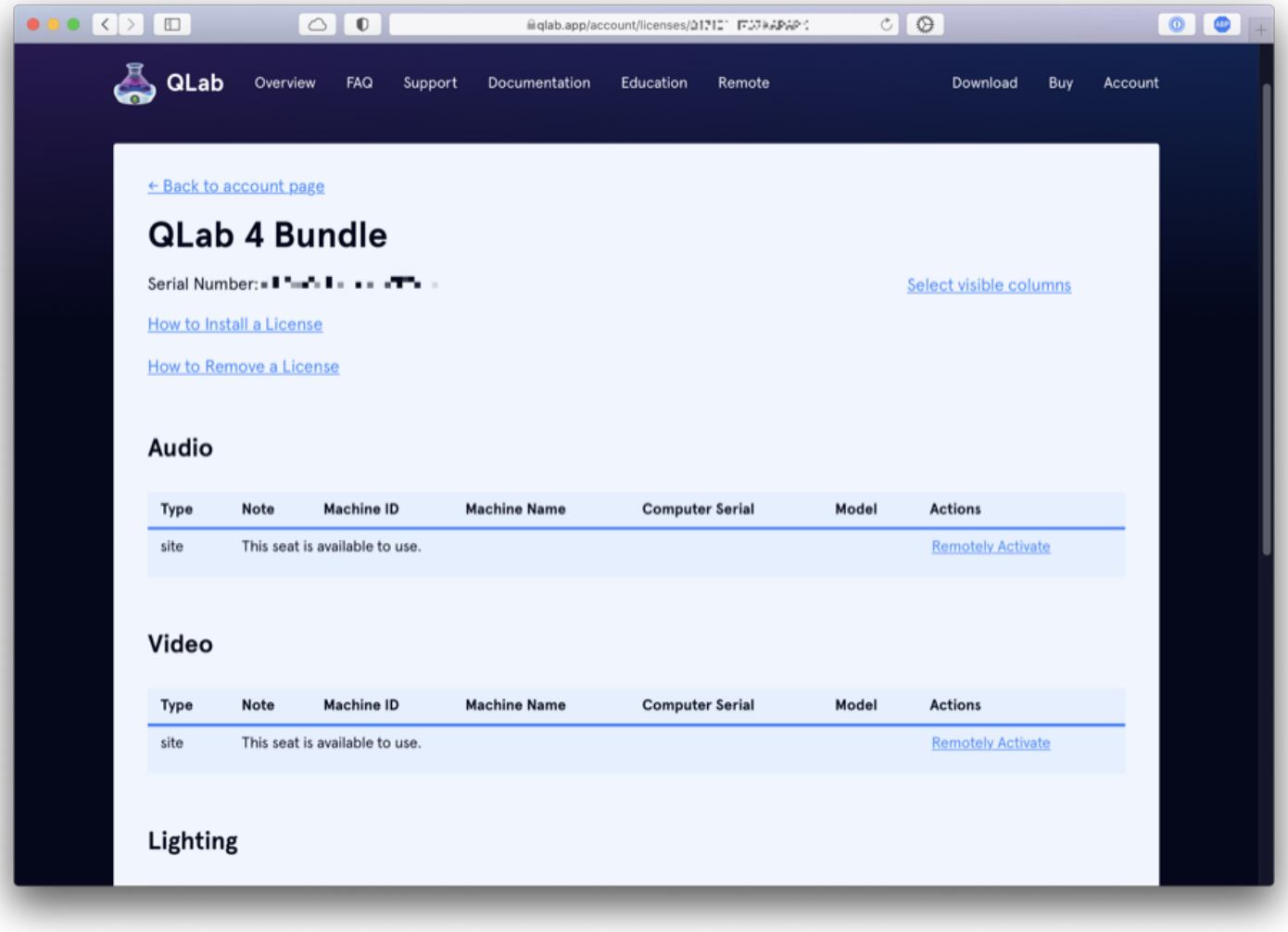
The alphanumeric code near the bottom of the window is the Machine ID, which is a QLab-only identification code unique to each Mac. You can also find your Mac's Machine ID by opening a workspace, choosing *Workspace Status* from the **Window menu**, and clicking on the *Info* tab.

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 **Your Licenses.**



When you've found the license you want to install, click *See details* under that license, and you'll be taken to a page showing the details of that license.



Choose the seat you wish to install, and click *Activate*. You'll be asked to enter the Machine ID of the Mac, then click *Activate* once more. The web page will refresh, and then you'll see a link next to the seat that says *Download Activation File*. Click to download the file, and then copy that file over to the offline 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 and which seat you want to install on this Mac. For example, you might say "I want to install the Main Audio seat of my Pro Bundle" or you might say "I want to install the Main Lighting and Backup Video seats of my Pro Bundle."

When we receive your message, we'll use the code to generate an offline activation file, which we will email back to you. When you receive that email, copy the file to the offline Mac using a local network or a USB key or some similar method. You can then double-click on it or drag it onto QLab's icon to install it.

Removing Licenses

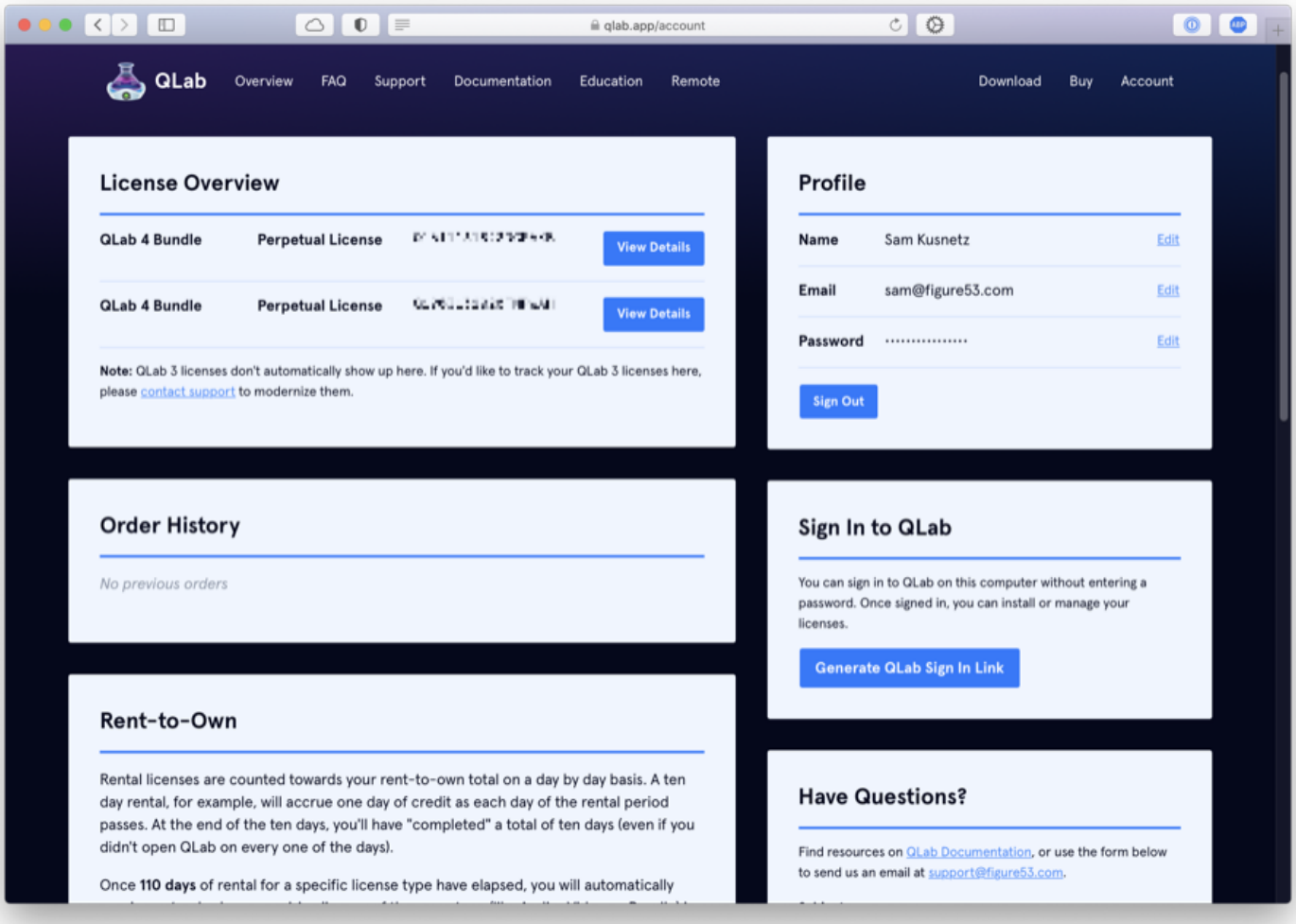
To remove a license installed on your Mac, open QLab, choose *Manage Your Licenses...* from the **QLab menu**, and click the *Remove* button next to the license you wish to remove. If the *Remove* button is not visible, log in to the account that owns that license to reveal it.

If you need to remove a license from a Mac which has been broken, lost, stolen, or erased, please write to support@figure53.com and we will help you out, no problem.

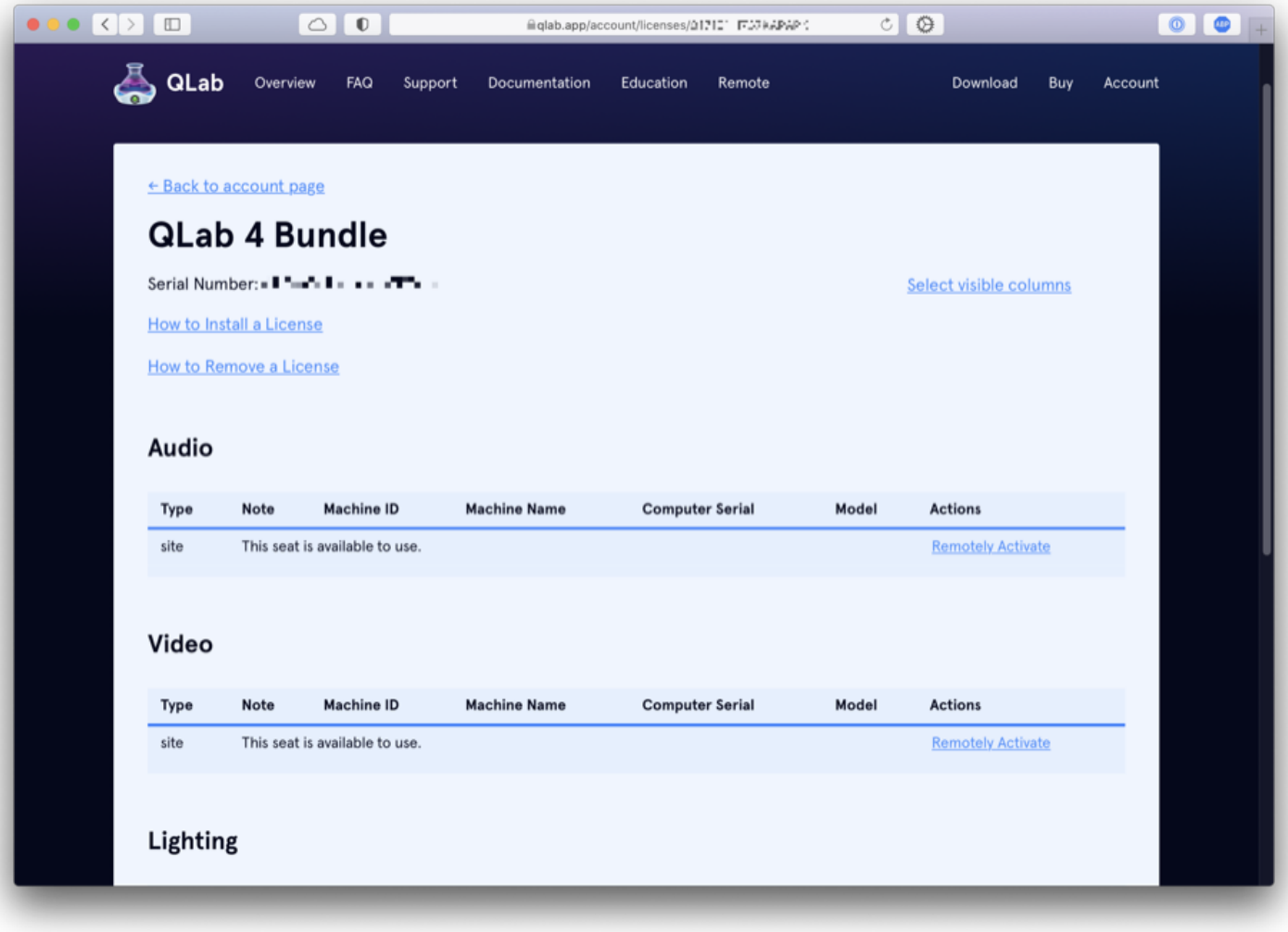
Remote Deactivation

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 **Your Licenses.**



When you've found the license you want to remotely deactivate, click *See details* under that license, and you'll be taken to a page showing the details of that license.



Choose the seat you wish to deactivate, and 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 originally had the license installed, the now-deactivated license seat will be automatically uninstalled. The Mac must have access to the internet in order to complete this uninstallation.

Using QLab 4 licenses with QLab 3

QLab 3.2 and later has the ability to communicate with the new account system, and you can use any QLab 4 license with QLab 3. License activation happens on a per-computer basis, so using both QLab 3.2 and QLab 4 on the same Mac only counts as a single activation. Please note that using a QLab 4 license with QLab 3.2 does not enable any of the 4.x features... you'll need to use QLab 4 to get access to those.

Modernizing QLab 3 licenses

If you own a QLab 3 license, you can "modernize" this license and bring it into the account system if you wish. If you want to modernize a QLab 3 license, you must remove it from every computer that it's been installed on. Then, contact us at support@figure53.com and we'll take care of it.

You do not need to modernize a license in order to use it for an upgrade discount.

Features by License

QLab 4 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.

Most of the advanced features can be used experimentally without a license installed, and will only require the license if you re-open a saved workspace that uses these features.

For those features which cannot be tested in this way, QLab has a demo mode which enables all features for twenty minutes. While demo mode is active, all of QLab's licensed features are available, but saving, copying, and pasting are disabled. All workspaces will automatically close after 20 minutes. Demo mode will exit when you quit QLab.

To activate demo mode, choose *Start Demo Mode...* from the **QLab** menu.

Feature	Free	Pro Audio	Pro Video	Pro Lighting
Channels of audio output	2	64	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	✓	✓	✓	✓
Unlimited cue lists	✓	✓	✓	✓
Unlimited cue carts	✓	✓	✓	✓
Audio fades	✓	✓	✓	✓
Live fade previews	✓	✓	✓	✓
Edit audio output routing		✓		
Edit audio output names		✓		
Audio effects on cues		✓		
Audio effects on cue outputs		✓		
Audio effects on device outputs		✓		
Audio effects fading		✓		
Audio playback rate fading		✓		
Mic cues		✓		
Single-screen video outputs	1	1	unlimited	1
Multi-screen video outputs	0	0	unlimited	0
1000 video layers	✓	✓	✓	✓
Full-surface Video cues	✓	✓	✓	✓
Custom geometry Video cues			✓	
Fade Video cues			✓	
Video playback rate fading			✓	
Masking & edge blending			✓	

Feature	Free	Pro Audio	Pro Video	Pro Lighting
Surface warping & keystone correction			✓	
Video effects			✓	
Syphon input & output			✓	
Blackmagic device input & output			✓	
Camera cues			✓	
Text cues			✓	
Patchable DMX addresses	16	16	16	unlimited
Remote control via OSC, MIDI, MSC, iOS app	✓	✓	✓	✓
Network cues		✓	✓	✓
MIDI cues		✓	✓	✓
MIDI File cues		✓	✓	✓
Pause cues		✓	✓	✓
Devamp cues		✓	✓	✓
Target cues		✓	✓	✓
Arm & Disarm cues		✓	✓	✓
Script cues		✓	✓	✓
Timecode cues		✓	✓	✓
Timecode triggers		✓	✓	✓

Templates

When you create a new workspace in QLab, you get a new window with an empty cue list, and all the settings of that workspace match QLab's default settings. A template allows you to create your own set of starting conditions for new workspaces.

Creating Templates

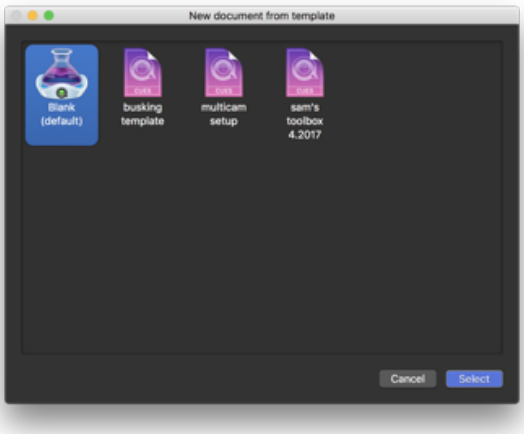
To create a template, make a new workspace and adjust all its settings to suit your preference. If, for example, you like Fade cues to have a default duration of eight seconds, you can visit Workspace Settings, select Cue Templates, highlight Fade, and set the duration to 8.

You can also create cues or cue lists 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 include those too.

Save this 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.

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 cue you select as the default template will be used whenever you create choose *New Workspace* from the **File** menu.

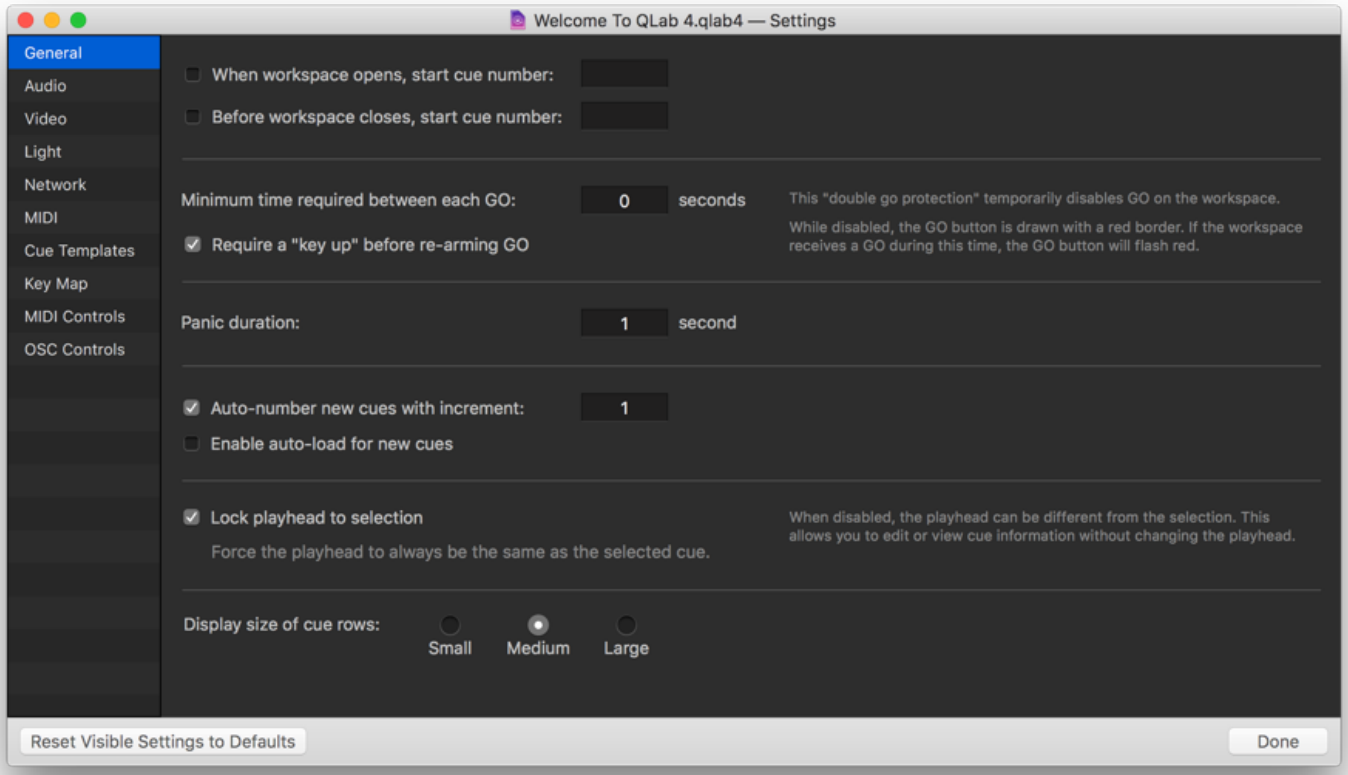
You can also right-click on a template to modify it or open a Finder window showing the folder that contains your templates.

Workspace Settings

The Workspace Settings window can be accessed by choosing *Workspace Settings* from the **Window** menu, or by using the keyboard shortcut ⌘,

Settings in this window belong to the workspace, not to QLab, so they travel with the workspace if you bring it to another computer.

General



When workspace opens, start cue number. Check this box and enter a cue number in the text field to have QLab automatically start that cue when the workspace is opened. If you need to temporarily prevent this cue from starting, hold down the **control** and **option** keys (⌃⌥) while opening the workspace.

Before workspace closes, start cue number. Check this box and enter a cue number in the text field to have QLab automatically start that cue when the workspace is closed. A dialog box will appear to give you the opportunity to prevent the closing cue from running.

Minimum time required between each GO. This can be set to any duration (including 0) in order to help prevent accidental starting of cues due to a double-press of the GO button. This protection will apply to mouse clicks on the GO button, pressing **space** or any other key you've assigned to GO, and incoming MIDI, MSC, and OSC controls which directly trigger the "GO" action.

If a time is set, a red border will appear around the GO button to indicate that the GO button is temporarily disabled. 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.

Require a "key up" before re-arming GO. This protection applies only to the keyboard shortcut assigned to GO; it forces QLab to wait for the key to be released before accepting another command to GO.

Panic duration. This is the time over which all running cues will fade out and stop when the escape key is pressed or the cue list receives a command to panic.

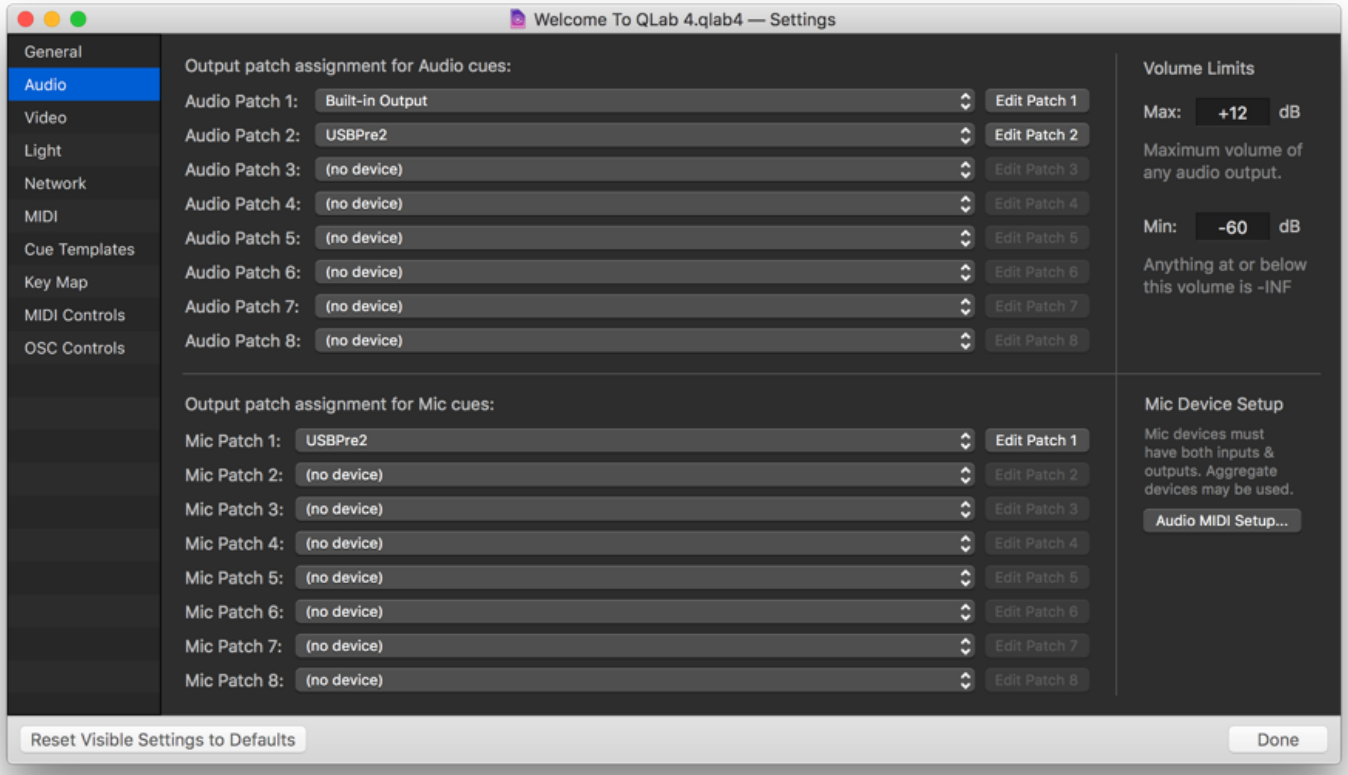
Auto-number new cues with increment. Check this box and enter a number into the text field to automatically assign cue numbers that increase by the desired increment to newly created cues. Cues will be numbered with the lowest available number according to this increment.

Enable auto-load for new cues. Check this box and newly created cues will have their "auto-load" option enabled by default.

Lock playback position to selection. Check this box to make the playback position always match the selected cue. When disabled, the playback position can be different from the selection. This allows you to edit or view cue information without changing the playback position.

Display size of cue rows. Choose a size to display your cues: small, medium, or large. Small allows more information to fit on screen, and large is convenient for easier viewing from a distance.

Audio



Output patch assignment for Audio cues. QLab workspaces can connect to eight individual audio devices, which are assigned to one of eight audio patches. Use the drop-down menus to assign an audio device to a patch. Click the *Edit Patch* buttons to the right of each drop-down menu to edit the details of that audio patch.

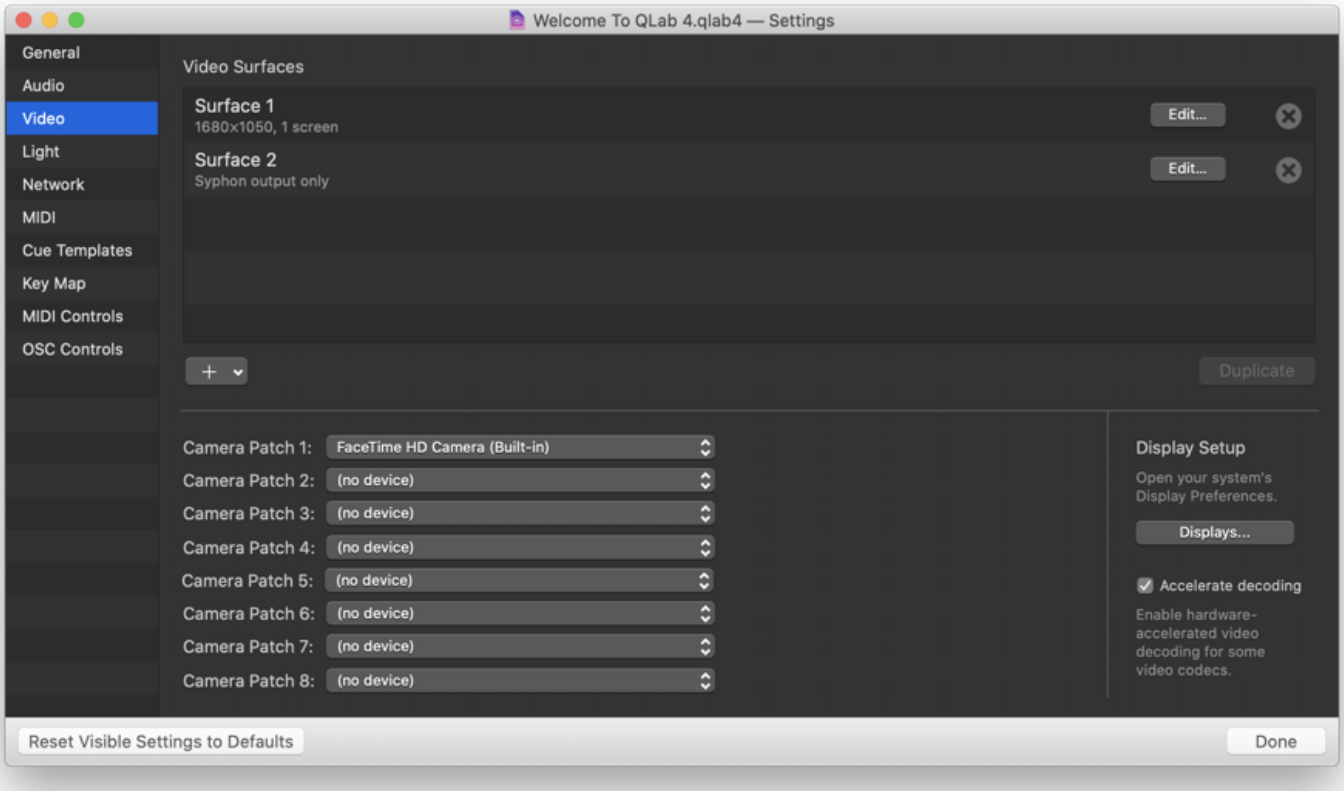
Output patch assignment for Mic cues. Output patches for Mic cues need to be set up separately from output patches for Audio cues. They can be the same, similar, or different from Audio cue output patches, depending upon your needs.

You can read more about editing audio patches here.

Volume Limits. Set an upper limit to prevent a sound from being played too loudly. Specify a lower limit, and QLab will treat anything at or below that level as `-INF` (silent).

Mic Device Setup. Output patches for Mic cues can only use audio devices which have both inputs and outputs, including aggregate audio devices. You can click the **Audio MIDI Setup...** button to open the Audio MIDI Setup application, which allows you to define aggregate audio devices and explore the details of the audio devices connected to your Mac.

Video



Video Surfaces. The surfaces defined for this workspace are listed here to be edited or removed. You can learn more about editing surfaces here.

By default, in a new workspace, QLab creates one surface for each display connected to your Mac.

The + button below the list of surfaces gives you three ways to create a new surface.

- *New Empty Surface* creates a new surface with no screens assigned to it.
- *New With Display* creates a new surface with the selected screen or partial screen pre-assigned, and with dimensions equal to that screen's.
- *New Multi-Screen Surface* creates a new surface via a helper tool which lets you enter the resolution of each projector, the physical layout of the projectors, and the percentage overlap between each projector.

Duplicate makes a copy of the selected surface.

Camera Patches. QLab workspaces can connect to eight individual video input devices, which are assigned to one of eight Camera Patches. Use the drop-down menus to assign a device to a patch.

Camera cues in QLab can use the iSight or FaceTime camera built into many Macs, most USB- or FireWire-connected webcams, and Blackmagic video capture devices belonging to the UltraStudio, Intensity, DeckLink, and MultiBridge product lines.

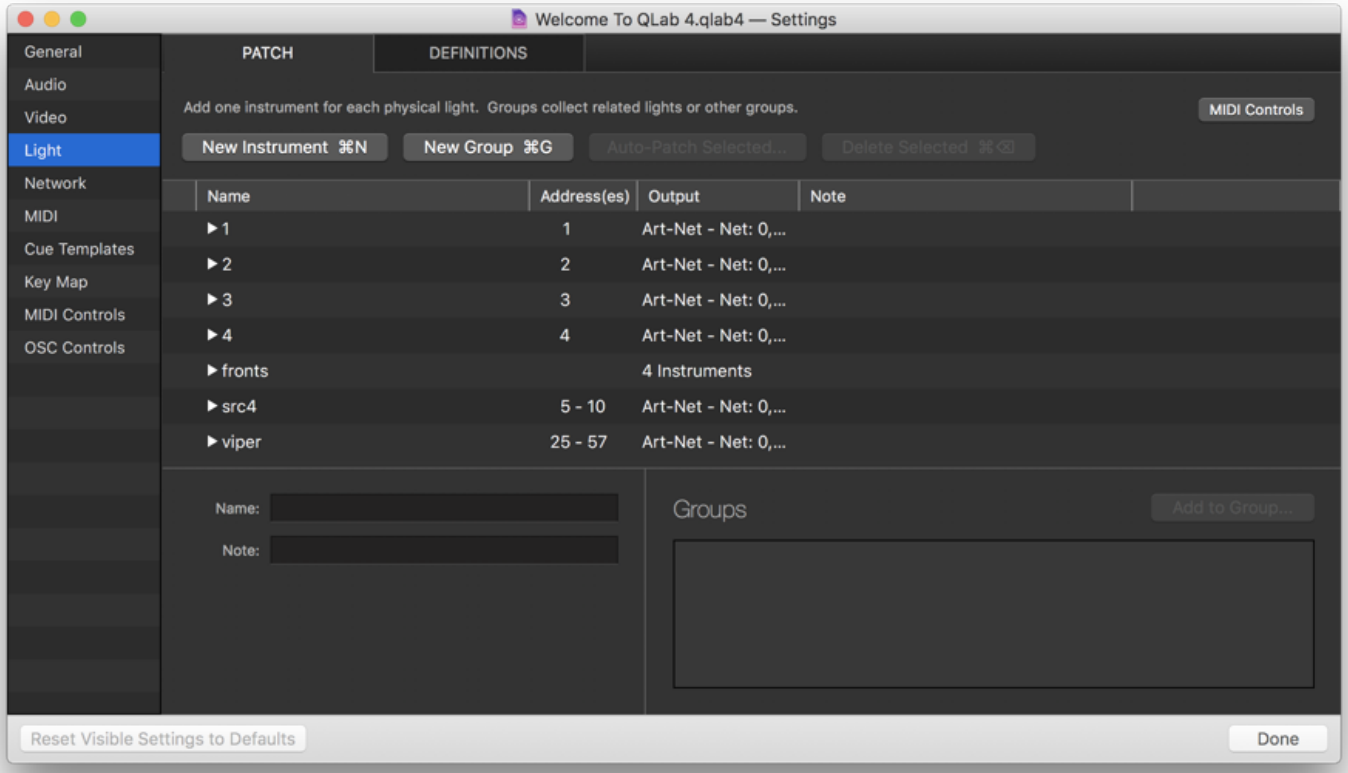
Display Setup. Click this button to open the Displays section of System Preferences.

Accelerate decoding. When this box is checked, QLab uses Apple's GPU-accelerated video decoding framework, which lets your Mac's GPU help decode some types of video, including H.264 and H.265. Apple does not offer very much information about the intricacies of this framework, so we're sorry to say we cannot either. If your Mac has a very fast CPU but a lackluster GPU, it's possible that you'll see better video performance with this box unchecked. If you're experiencing choppy video playback, experiment with checking or unchecking this box.

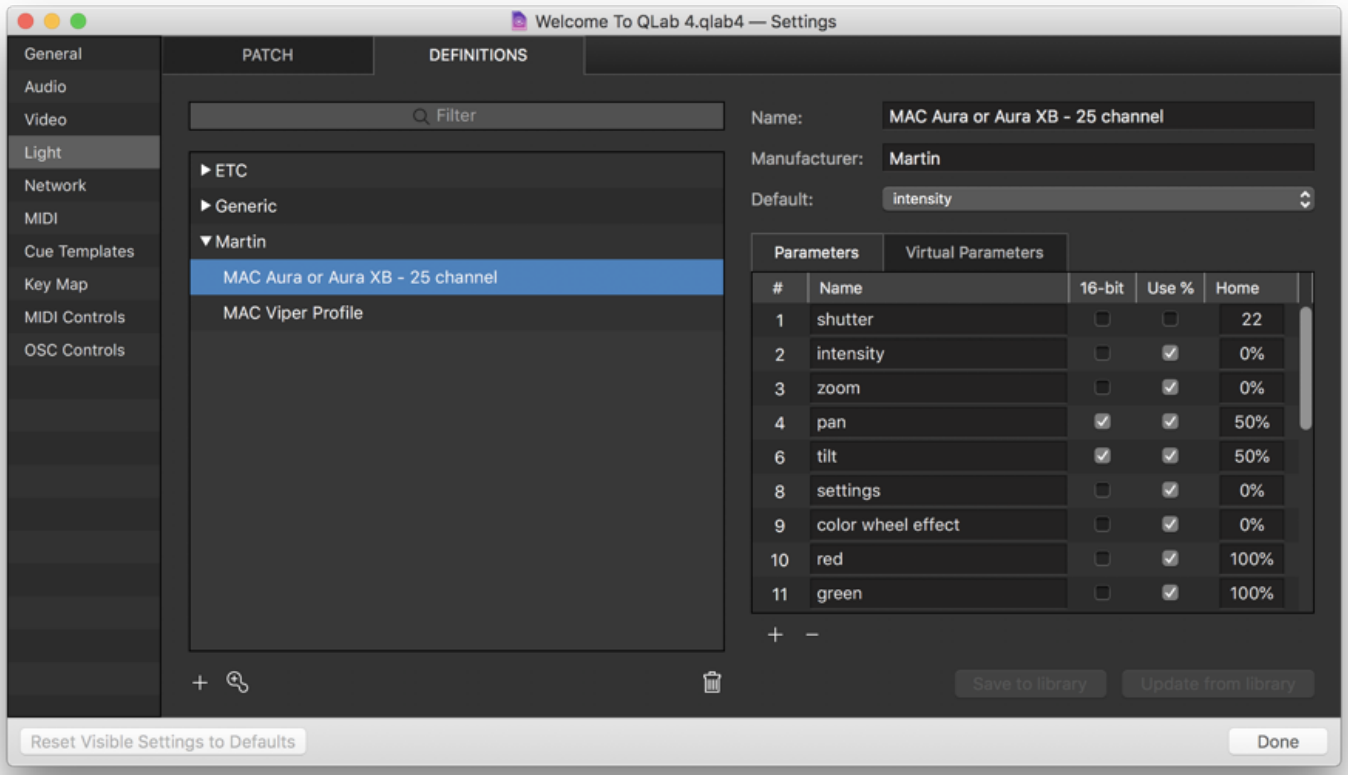
Light

Light settings are divided into two tabs: Patch and Definitions.

The Patch Tab allows you to view and edit the instruments and light groups in the workspace.

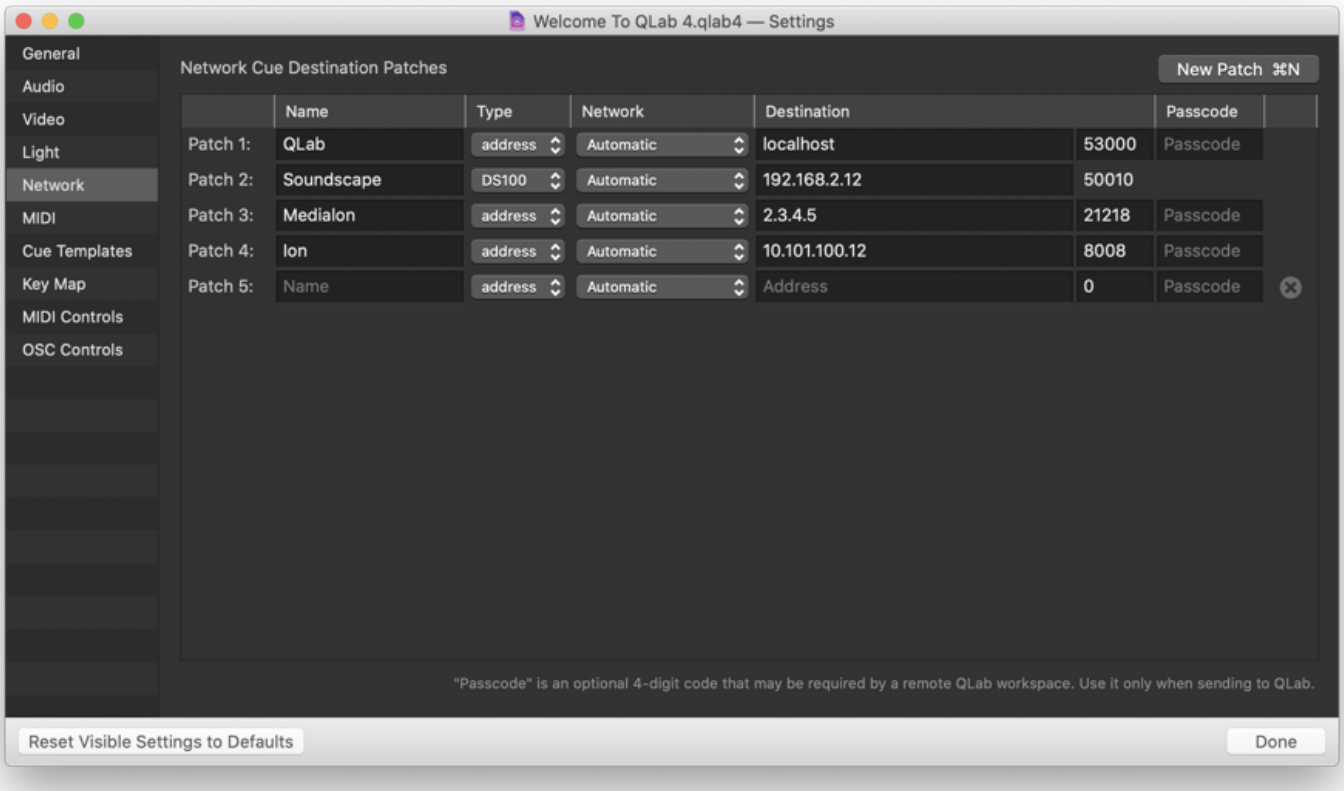


The Definitions tab lists every light definition used by at least one instrument in the workspace.



You can learn more about the Lighting Patch Editor in its section of the documentation.

Network



Network cues send OSC or UDP text messages to destinations which are configured in the Network section of Workspace Preferences. Starting with QLab 4.6, workspaces can have any number of destination patches. By default, new QLab workspaces start with a single patch, and you can add more by clicking the *New Patch* button or by using the keyboard shortcut ⌘**N** while this screen is frontmost.

Name. This is for setting an easy-to-read label, and has no bearing on the functionality of the patch.

Type. This drop-down menu lets you choose either *address* or *DS100*. If the network patch will be used to communicate with a d&b DS100 Soundscape processor, then choose *DS100*. Otherwise, choose *address*.

Network. You can set a patch to automatically choose the network interface that it will use, or manually select a specific network interface. By default, this is set to *Automatic*. Unless your Mac is connected to multiple networks interfaces all using the same IP addressing scheme and subnet, using *Automatic* should work properly nearly all of the time.

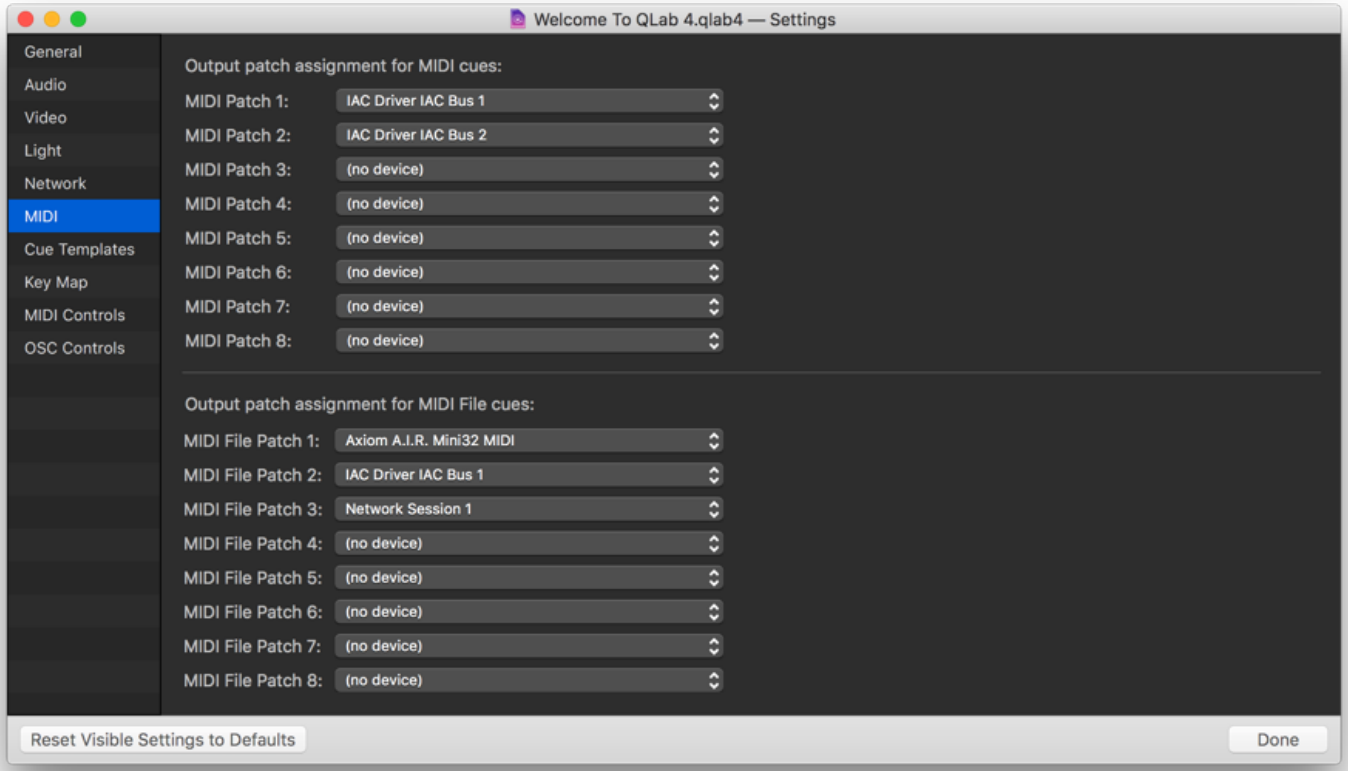
Destination. This is the IP address and port number of the destination. Typing *localhost* or *127.0.0.1* will set the destination to this computer, to allow QLab to send OSC messages to itself or to other software on the same machine. QLab listens for OSC on port 53000, and for plain text formatted as OSC on port 53535. Note that the port cannot be edited for DS100 patches; the DS100 processor has a hard-coded port number which cannot be changed.

Important: in the interest of speed, messages sent to *localhost* do not go through the network stack of macOS, and as such can only be used for sending messages from QLab to itself. Messages sent to *127.0.0.1* **do** traverse the network stack, and so can be used to send messages to other programs running on the same Mac.

Passcode. If the destination is a copy of QLab, and that copy has a passcode set (see OSC Controls, above) enter that passcode here. This code is QLab-specific and should only be set when necessary.

While there is no harm in having unused patches in your workspace, you can delete patches off the bottom of the list by clicking on the ⊗ button which appears at the right edge of the window.

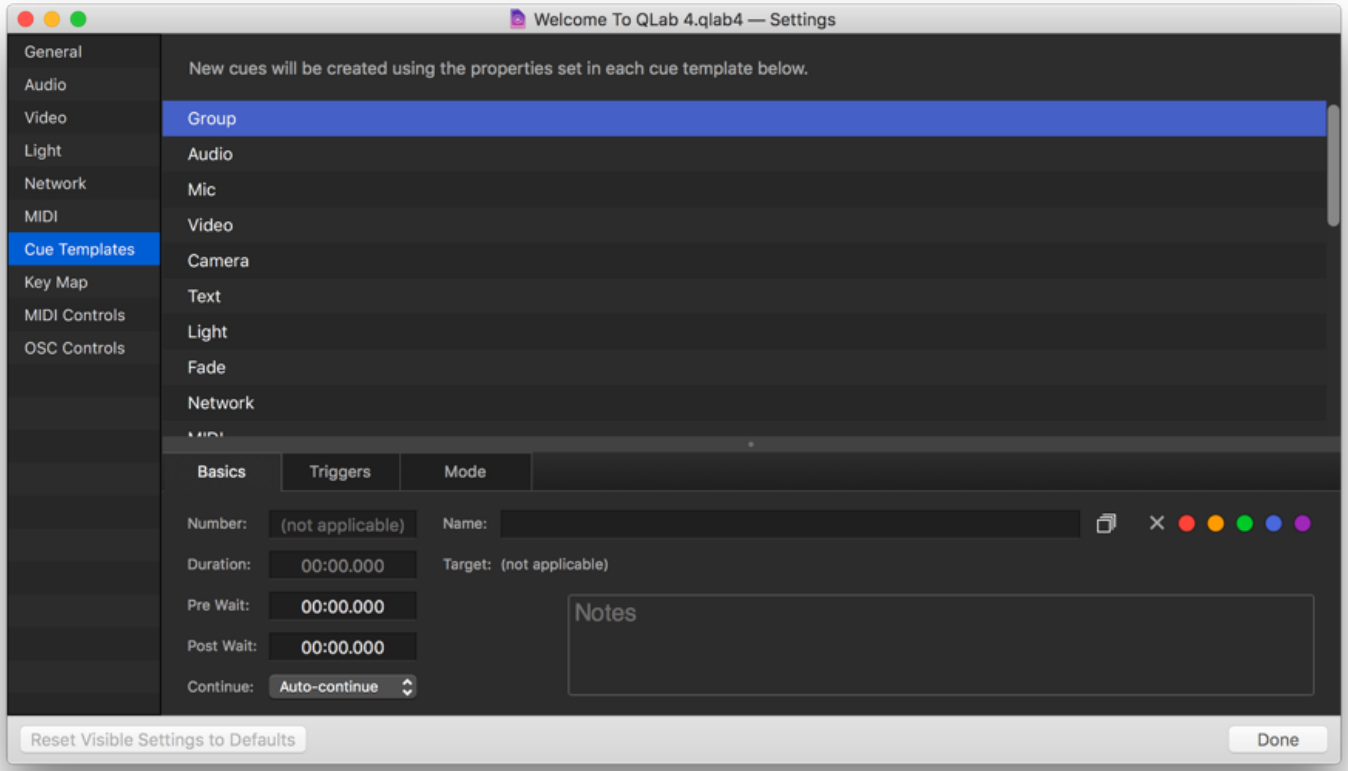
MIDI



Output patch assignment for MIDI cues. MIDI cues can output to any of eight individual MIDI devices, which are assigned to one of eight MIDI Patches. Use the drop-down menus to assign a MIDI device to a patch. Note that these patches are only relevant to outgoing MIDI from MIDI cues, and have no bearing on incoming MIDI for triggers or workspace controls.

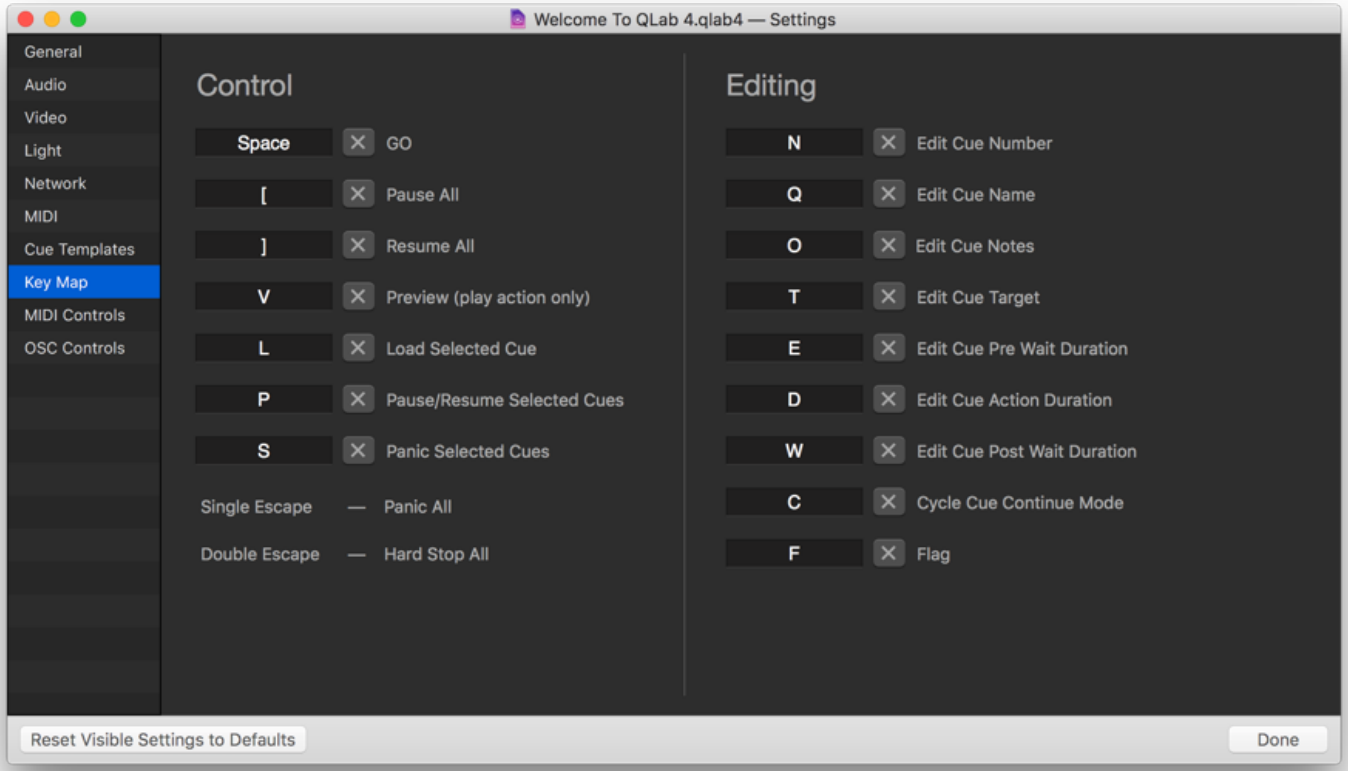
Output patch assignment for MIDI File cues. Output patches for MIDI File cues need to be set up separately from output patches for MIDI cues. They can be the same, similar, or different from MIDI cue output patches, depending upon your needs.

Cue Templates



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 the list. You can select one of the cue types and use the Inspector to make any adjustments you like, including cue names, notes, targets... everything. Newly created cues will use these settings as their default.

Key Map



The Key Map allows you customize keyboard shortcuts, or hotkeys, for the listed commands. Simply enter the desired hotkey in the text field next to the appropriate command. To disable a hotkey, press the X button next to the command.

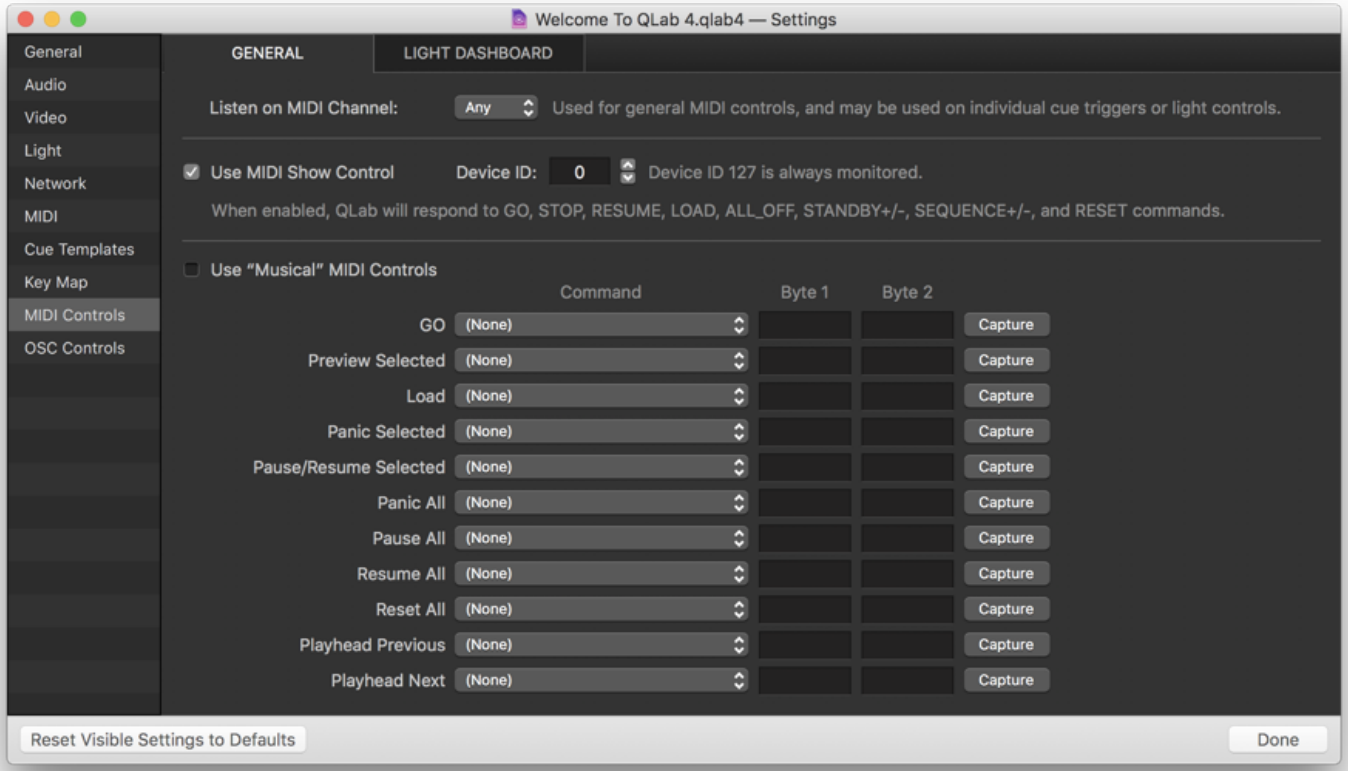
Note that a single press of the **escape** key is always assigned to panic, and a double press of the **escape** key is always assigned to hard stop.

MIDI Controls

MIDI Controls settings are divided into two tabs: General and Light Dashboard.

The General Tab

The **General** tab allows you to configure how your workspace responds overall to incoming MIDI messages and MIDI Show Control messages.



Listen on MIDI Channel. You can restrict incoming MIDI to a specific channel for transport commands, or set to "Any." Individual cue triggers will default to listening on that same channel, but can be individually set to a different channel.

Use MIDI Show Control. When enabled, QLab will respond to GO, STOP, RESUME, LOAD, ALL_OFF, STANDBY+/-, and RESET commands sent via MIDI Show Control. QLab listens for commands categorized as *Lighting (General)*, *Sound (General)*, and *Video (General)*.

Use Device ID. Device ID 127 is always monitored, along with whatever ID is entered here. All devices on an MSC network should have unique device IDs.

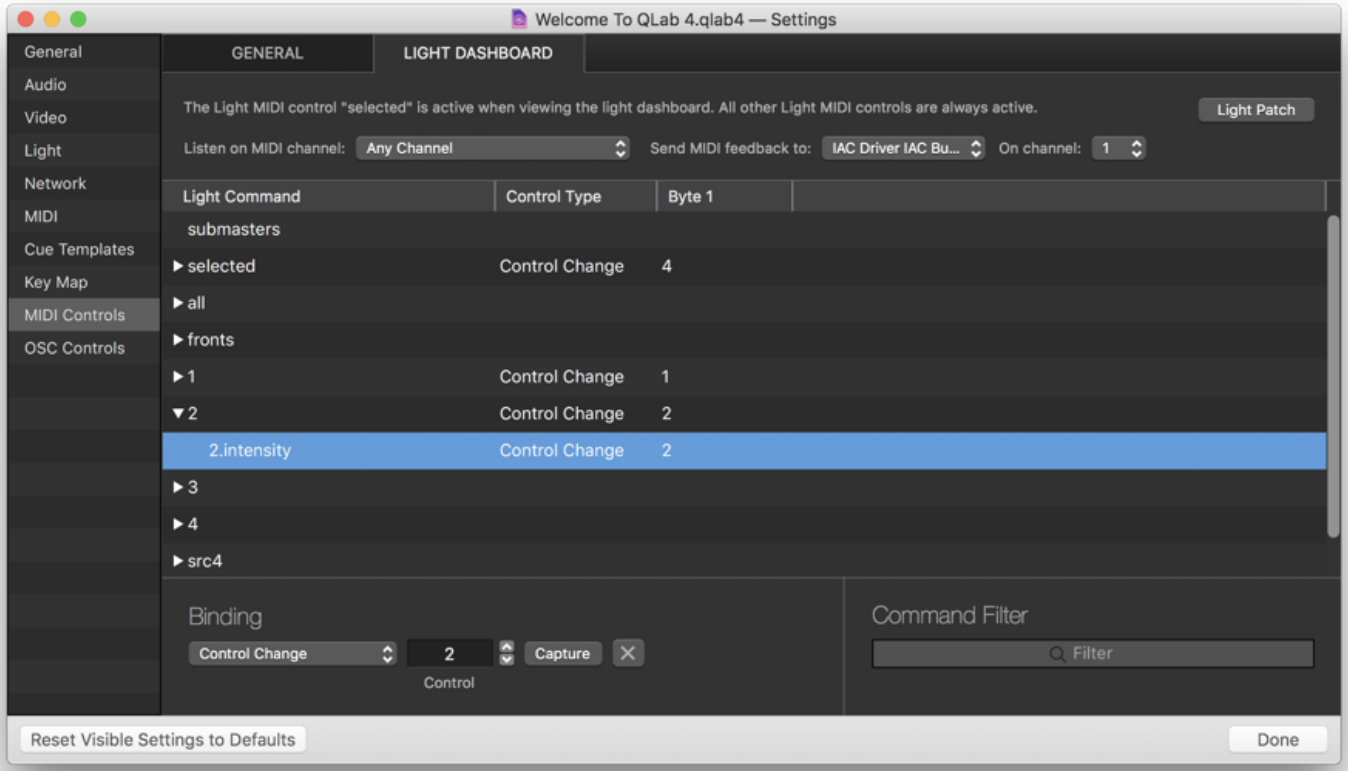
Use "Musical" MIDI Controls. When enabled, QLab listens for incoming MIDI messages which can be assigned to the transport controls listed below.

Command. For the listed transport controls, you can manually enter a MIDI command, or click "Capture" to have QLab listen for the next incoming MIDI command and assign that command to the selected control. Note that only Note On, Note Off, Control Change and Program Change messages can be used for transport controls.

The field marked "Byte 2" can contain greater-than (>) or less-than (<) signs, and can also contain the word "any."

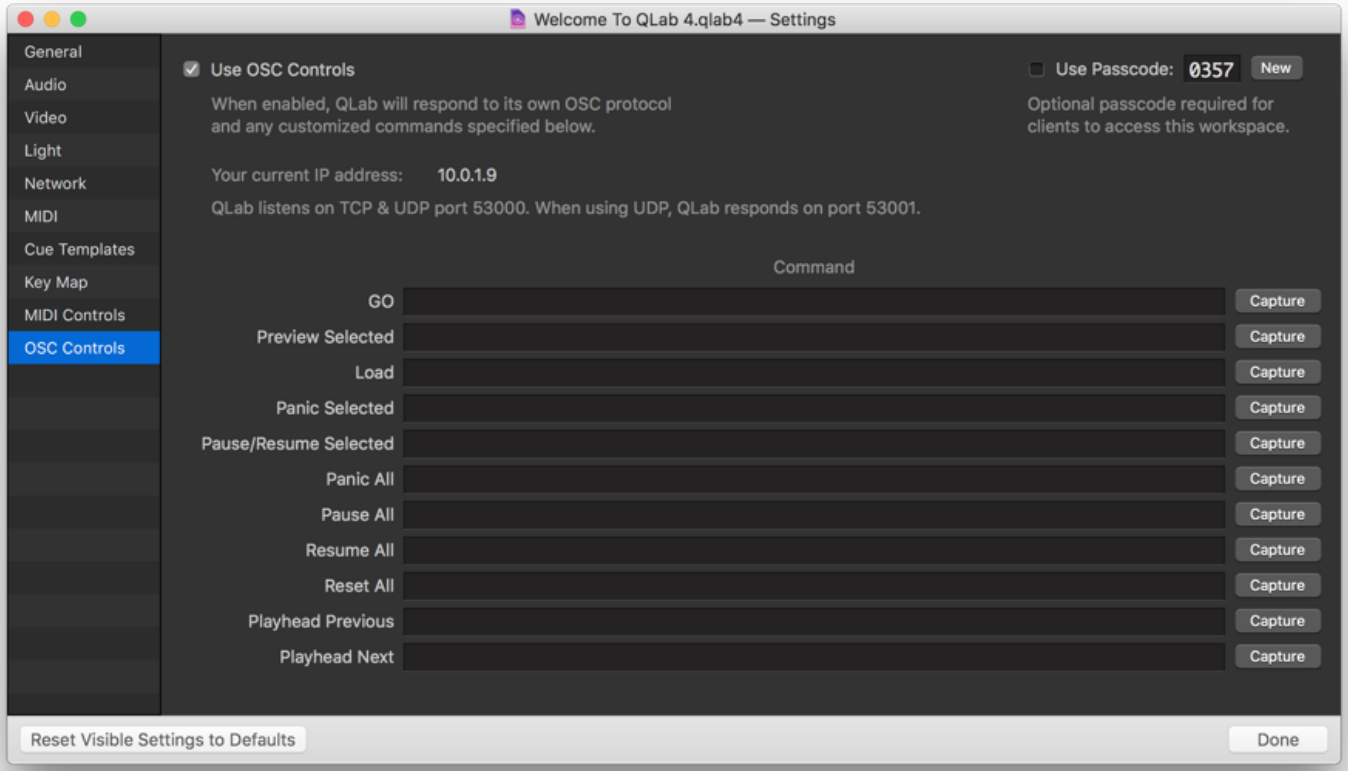
The Light Dashboard Tab

The **Light Dashboard** tab allows you to map MIDI controls to instrument parameters in the Light Dashboard.



You can learn more about the Light Dashboard tab in the Light Patch Editor section of this documentation.

OSC Controls



Use OSC Controls. When enabled, QLab will respond to incoming OSC (Open Sound Control) messages on all available network connections. QLab responds to messages defined in the QLab OSC dictionary, as well as any customized commands for the transport controls listed below.

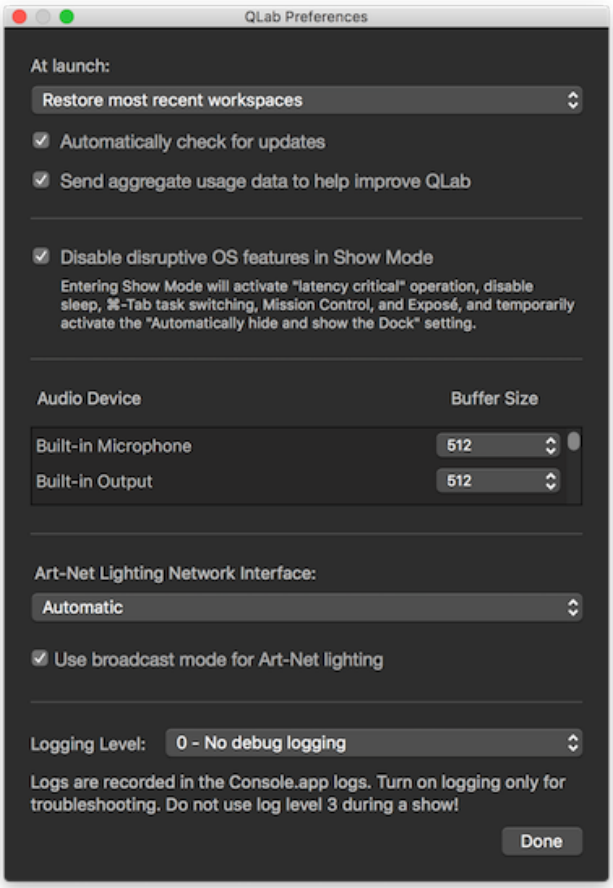
Use Passcode. You can check this box to require a four-digit passcode which clients must enter in order to send OSC commands to QLab.

Command. For the listed transport controls, you can manually enter any valid OSC command, or click “Capture” to have QLab listen for the next incoming OSC command and assign that command to the selected control. Note that QLab’s own OSC commands for these controls are not overridden by entries here; they remain active in addition to them.

QLab listens for OSC on port 53000. Additionally, QLab listens for plain text formatted as OSC commands on port 53535. When incoming commands are sent to port 53000 via UDP, QLab will send replies on port 53001.

QLab Preferences

You can set application-wide preferences for QLab by choosing *QLab Preferences...* from the **Apple menu**. These preferences apply to all workspaces on your Mac, and they do not transfer with workspaces copied or moved to other Macs.



At launch

You can choose amongst five behaviors for QLab when it's launched:

- **Show the Launcher Window.** Display the Launcher Window, which lets you choose amongst your recent workspaces or create a new workspace based on one of your templates, and which provides links to manage your licenses, view this documentation, or contact technical support.
- **Restore most recent workspaces.** Re-open the workspace or workspaces that were open last time QLab was running.
- **Create a new blank workspace.** This is the same behavior as previous versions of QLab.
- **Create a new workspace from default template.** You can select a workspace template as the default by choosing *Manage Templates* from the **QLab menu**, selecting a template, and clicking **Set as Default**. Once you've done that, choosing this option will create a new workspace based on that template.
- **Do nothing.** This is exactly what it sounds like.

Automatically check for 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.

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 disables the following OS-level features which could potentially cause trouble during a performance:

- **App Nap.** QLab reports itself as "latency critical" so as to prevent App Nap.
- **Sleep.** QLab will prevent your computer from sleeping, even if Energy Saver preferences are set to permit sleep.
- **⌘Tab Task Swtching.** Typically, typing **⌘Tab** allows you to quickly move between applications on your Mac. QLab will block access to this keyboard shortcut, 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**.

Audio device buffer sizes

QLab defaults to a buffer size of 512 samples for each audio device. 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 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 using a USB DMX device, always leave this set to *Automatic*.

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.

Logging level

QLab logs messages to the Mac at four possible levels. Level 0 provides minimal informational logging only, and Level 3 provides the most logging possible. Level 3 logging records enough data that it can interfere with smooth playback. We recommend using log level 0 unless you are actively troubleshooting a problem.

Chapter 2: Tools

The Tools Menu

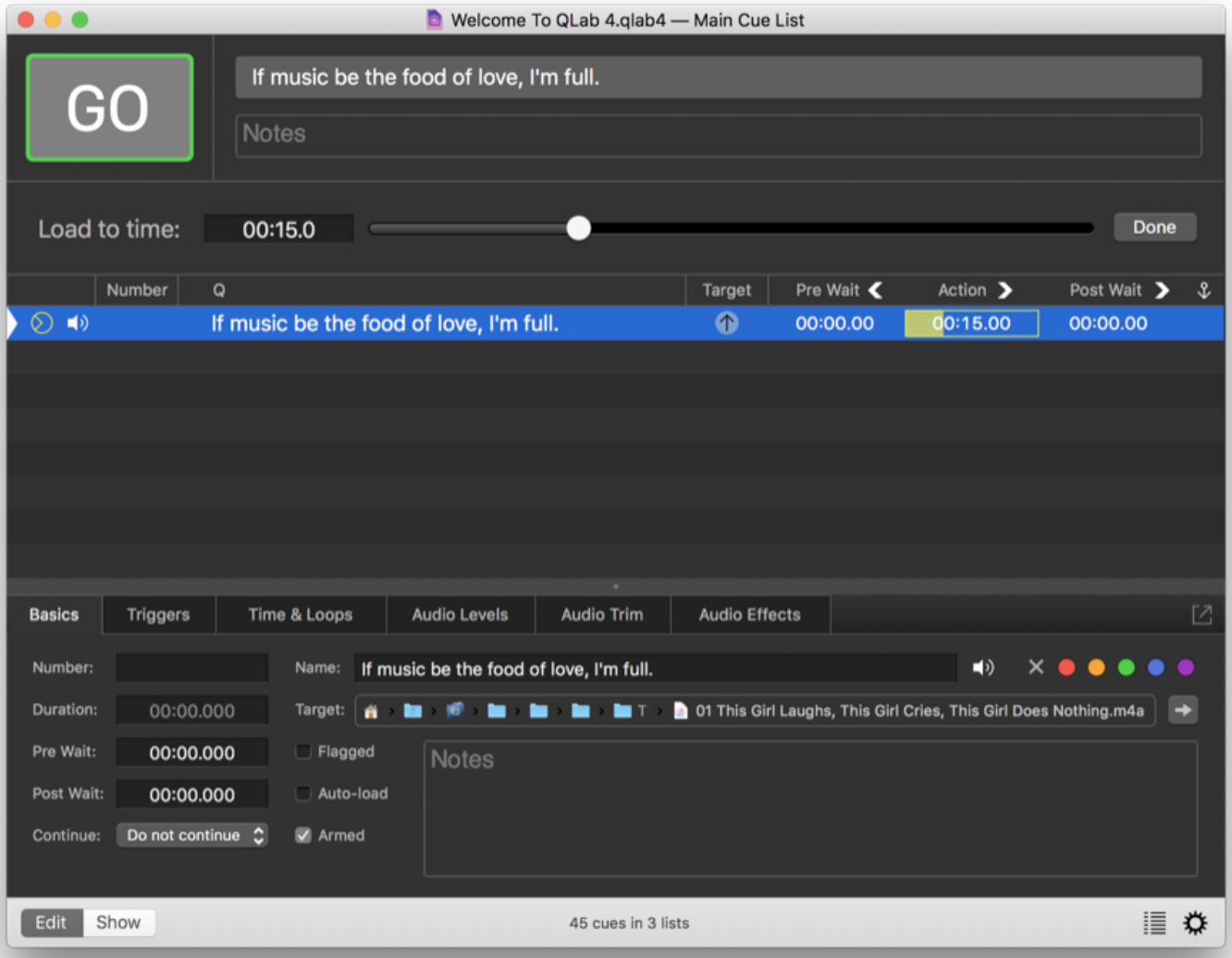
The **Tools** menu contains items relating to workflow, which is to say the process of building cues and cue sequences and operating QLab. The menu is context-sensitive, and can display different tool when different windows in QLab are active. When a cue list or cart window is active the **Tools** menu 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. Then, when the cue is started, 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 feature, 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 along the timeline to load the selected cue or cue sequence to your desired time.

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.

Cue Numbering

Renumber selected cues (⌘R). This will renumber all of the currently selected cues. You will be prompted to enter a starting number and an increment value. QLab will skip any cue numbers belonging to other, non-selected cues in the workspace.

Delete numbers of selected cues (⌘D). This will delete the cue numbers of all selected cues. This will not alter cue targets or anything else at all, just the cue numbers.

Jump

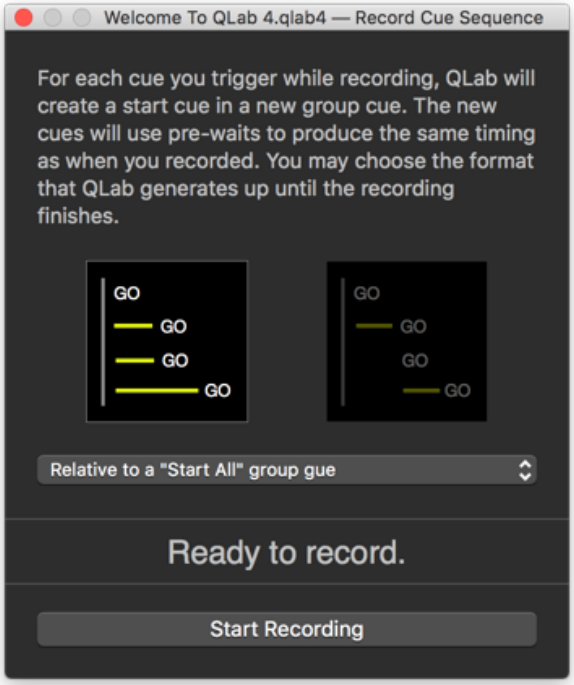
Jump to cue... (⌘J). This lets you quickly move the selection to a specific cue. You'll be prompted to enter a cue number. Note that if the playhead and selection are locked together, the playhead will also move. If they are unlocked, only the selection will move.

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.

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, each triggering the cues that you triggered with pre-waits that exactly match your timing.

To do this, choose *Record cue sequence...* from the **Tools** menu. Then, choose whether you want to create a Group cue set to "start all children simultaneously" with pre-waits on the Start cues, or a Group cue set to "start first child and go to next cue" with pre-waits and auto-continues on the Start cues. 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 trigger the original cues.

Live Fade Preview

When live fade preview is turned on, any adjustments you make in a Fade cue will be immediately reflected, as long as the Fade cue's target 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 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 without disturbing them.

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, 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 Audio and Video Cues

When an Audio or Video cue is selected, the **Tools** menu also shows the following 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, etc. 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 "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.

Tools for Fade Cues

When a Fade cue is selected, the **Tools** menu also shows the following fade-specific items:

- **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 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

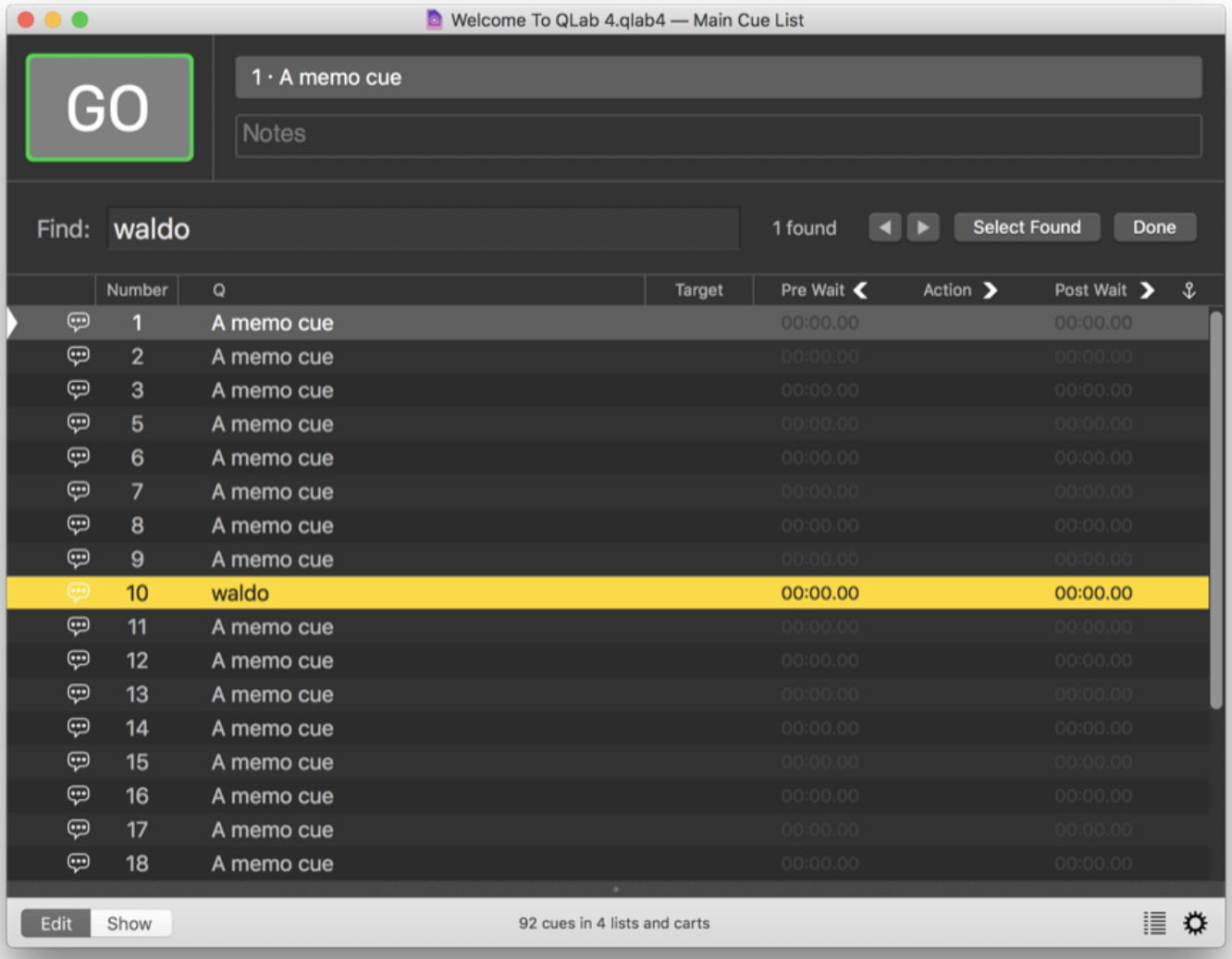
When the Light Dashboard is the frontmost window, the contents of **Tools** menu is re-populated with items pertinent to lighting. You can learn about those items from the page on the Light Dashboard in the Lighting section of this documentation.

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 UDP messages, 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 the found cues at once.

Clicking *Done* will clear the search and close the Find tool.

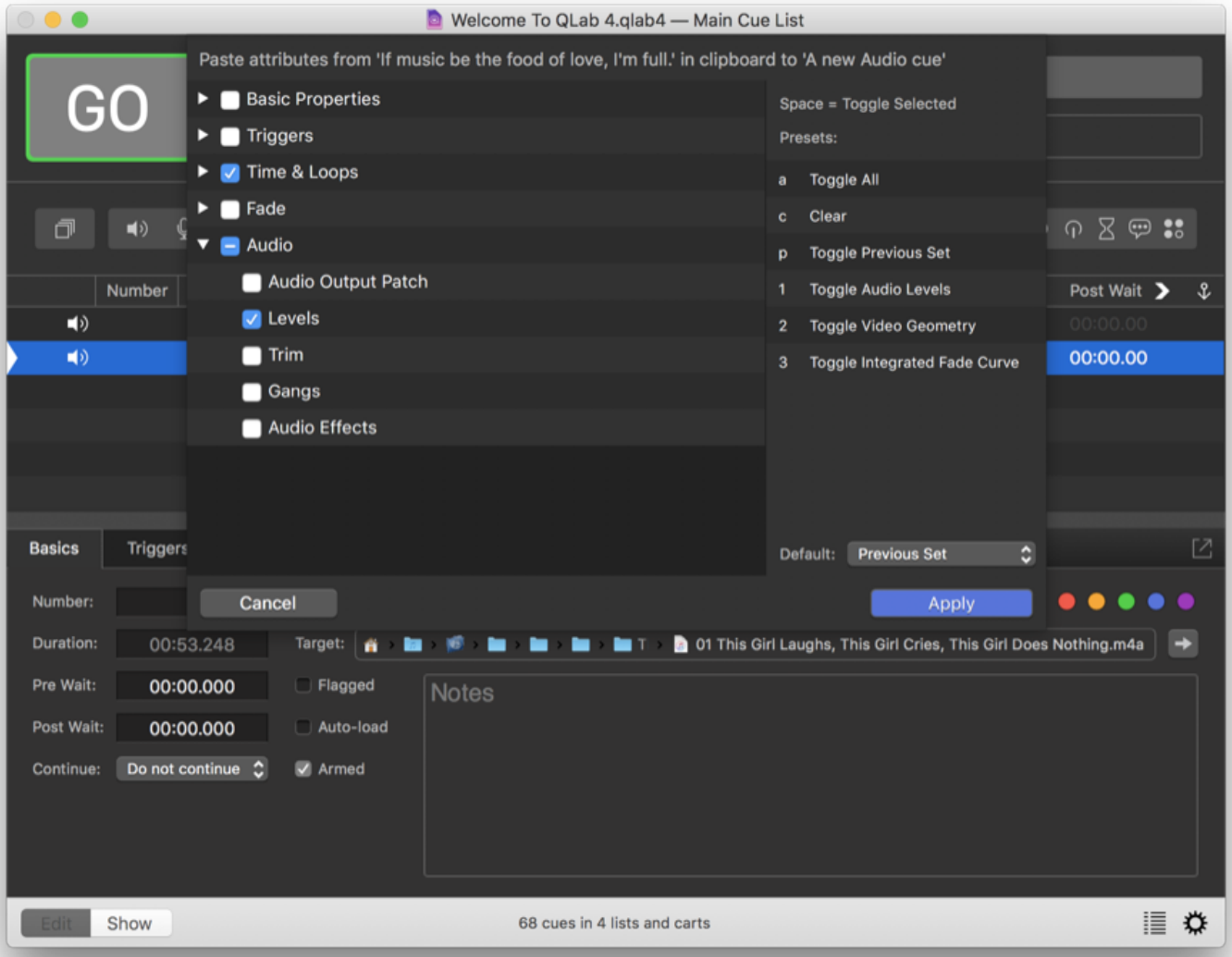
Paste Cue Properties

QLab 4 allows you to paste properties, or parameters, of one cue onto one or more 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. So in this screen shot, for example, Geometry and Text properties are not listed because an Audio cue is currently selected, and you can't paste video properties onto an Audio cue.

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.

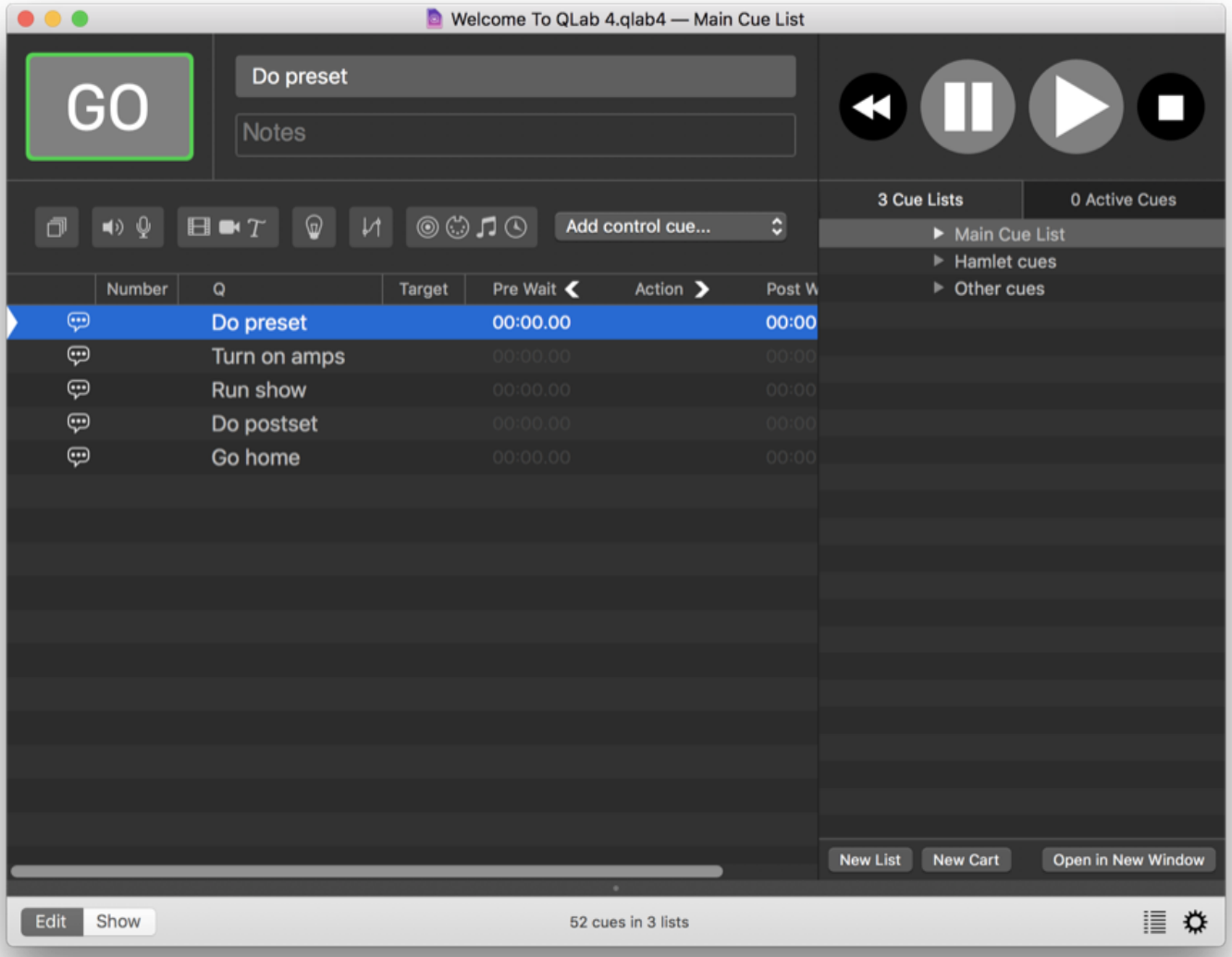
Hitting the enter key or clicking *Apply* will paste the selected properties onto the selected cue or cues. Hitting escape or clicking *Cancel* will close the sheet without making any changes.

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 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 Sidebar

The sidebar, hidden by default, can be shown by choosing *Lists / Carts & Active Cues* from the **View** menu, or by using the keyboard shortcut ⌘**L**. The sidebar shows four transport control buttons at the top, and two lists below them. You can click the headings of these two lists to switch between them, choose *Toggle Between Lists / Carts & Active Cues* from the **View** menu, or use the keyboard shortcut ⇧⌘**L**.



(The inspector, shown in most screen shots in this documentation, is hidden here for clarity.)

The Buttons

- **Reset All.** Stops all playing cues, sets the playhead to the top of the currently active cue list, and resets any temporarily changed properties of cues to their saved values.
- **Pause All.** Pauses all currently playing cues.
- **Resume All.** Resumes all currently paused cues.
- **Panic All.** Fades and stops all currently playing cues, and unloads any loaded cues.

Cue Lists and Carts

The left tab of the sidebar shows cue lists and cue 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 new workspace will contain one cue list named “Main Cue List.” You can change this name as you prefer. You can add cue 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).

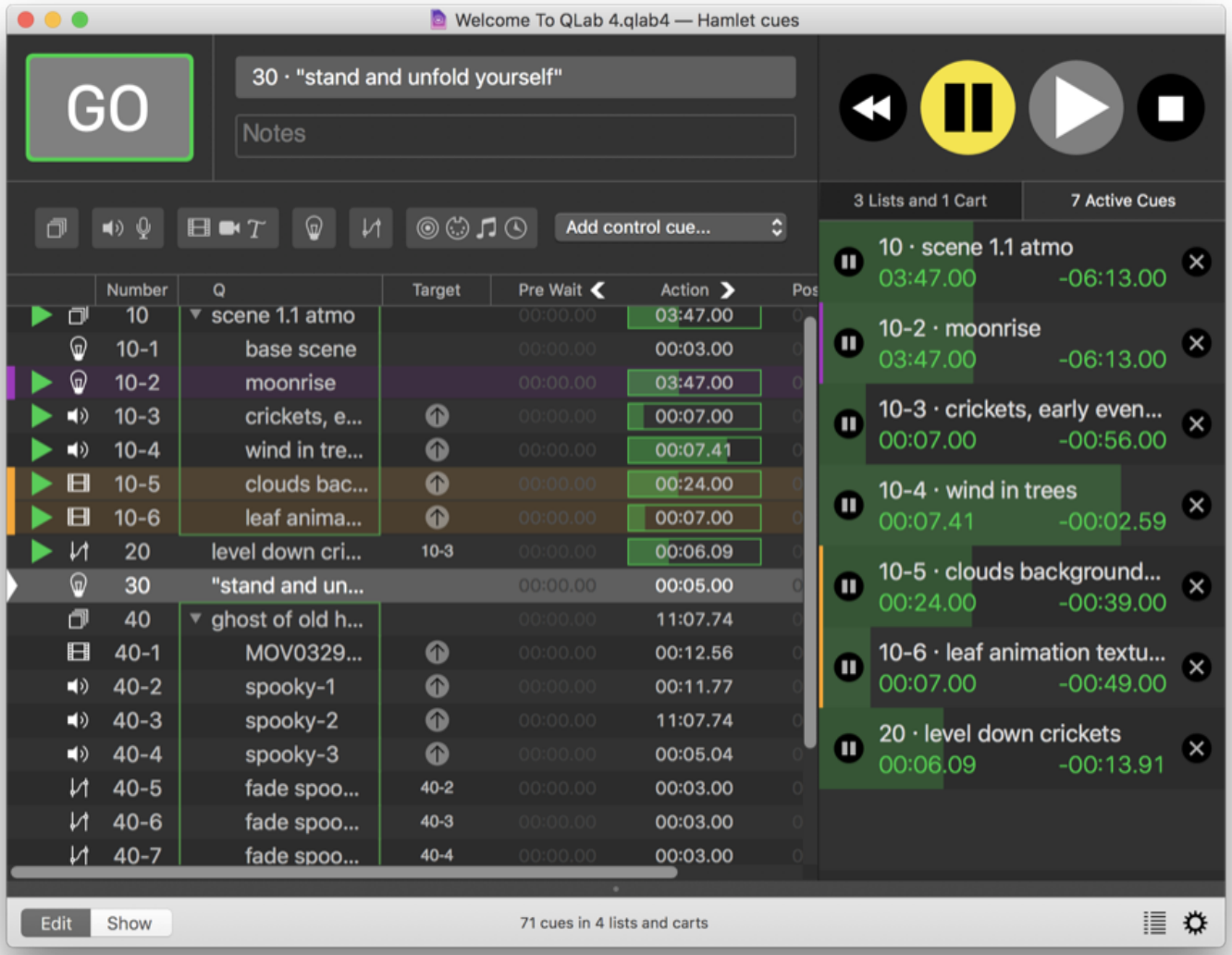
Starting with QLab 4.1, cue lists and carts can be opened in their own windows, allowing you to view multiple lists and carts simultaneously. Secondary windows always appear in show mode, meaning they have a larger GO button, no toolbar, and no inspector. Starting with QLab 4.4, when the workspace is in edit mode you can edit the cue number, cue name, pre-wait, action, post-wait, and continue mode of cues in secondary windows.

You can open a cue list or cart in a new window by using the *Open in New Window* button, or by right-clicking (or control-clicking, or two-finger-clicking) on the name of the cue list or cart.

For more information about cue lists and cue carts, check out the cue list section and the cue carts section of the documentation.

Active Cues

The Active Cues tab shows all cues which are currently playing or paused.



Each active cue is displayed in its own row. The row shows the cue number and name, and 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.

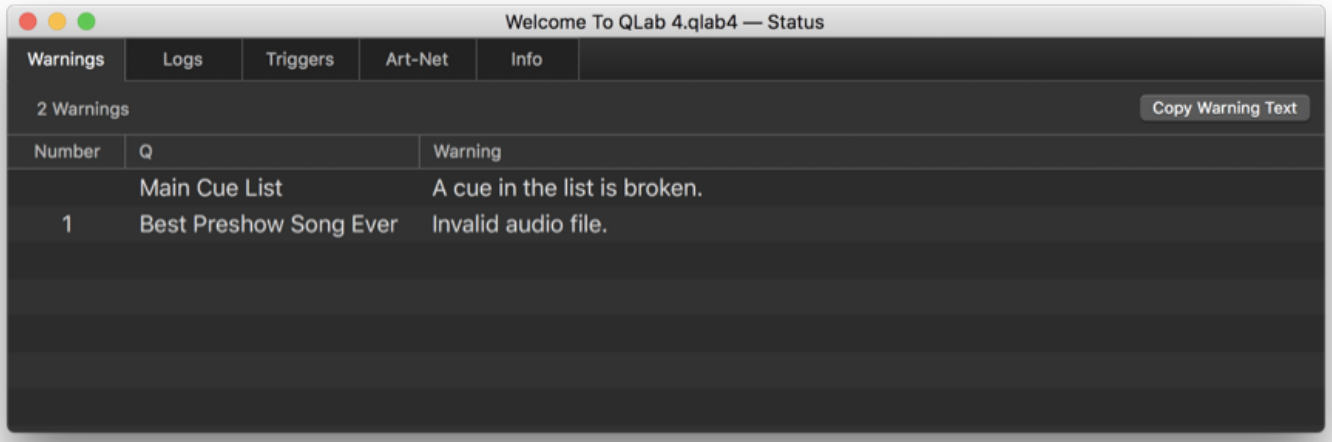
If you hover your mouse over an active cue, the leading edge of the progress bar will show a thick yellow line, which can be dragged left or right to scrub the cue backward or forward.

The Status Window

The Workspace Status window contains five tabs: Warning, Logs, Triggers, Art-Net, and Info, each of which provides information and feedback about your workspace as a whole.

To access the Workspace Status window, select *Workspace Status* from the **Window** menu or use the keyboard shortcut ⇧⌘**W**.

Warnings



The Warnings tab lists all cues which are either broken or flagged. Cues break when they are unable to perform their function for any reason. You'll see each broken cue listed in the Warnings tab, along with a brief description of the problem. Broken cues are also indicated in the cue list by a red ✕ in the leftmost column. Hovering your mouse over that red ✕ shows a tooltip which also shows a brief description of the problem.

Clicking on a cue in the Warnings tab will select that cue in the main workspace window.

You can click *Copy Warning Text* in the upper right corner of the Warnings tab to copy the full text of all the warnings in the tab. This can be helpful if you want to email the warnings to the support team, or save them to a text file for future reference.

Here are some common reasons that a cue might be broken:

- A target has not been assigned to a cue which requires one.
- A Fade cue has no activated parameters to fade.
- A Group cue contains one or more broken cues within it.
- An Audio cue has not been assigned to an audio patch.
- There is a problem with the audio patch to which an audio cue has been assigned.
- A Video cue has not been assigned to a surface.
- There is a problem with the surface to which a Video cue has been assigned.
- A MIDI, MIDI file, or Timecode cue has not been patched to a destination.
- There is a problem with the destination to which a MIDI, MIDI file, or Timecode cue has been assigned.
- The cue requires a license which is not installed.

If your workspace contains any broken cues, a warning icon (⚠) appears on the right side of the workspace window footer. Clicking the icon will open the Workspace Status window with the Warnings tab selected.

A broken cue will behave like a disarmed cue when it comes to being played as part of a cue sequence. Its pre-wait, post-wait, and follow status will be respected wherever possible.

Flagged cues, also listed in the Warnings tab, aren't necessarily broken. They're just listed for the purpose of having ready access to all flagged cues.

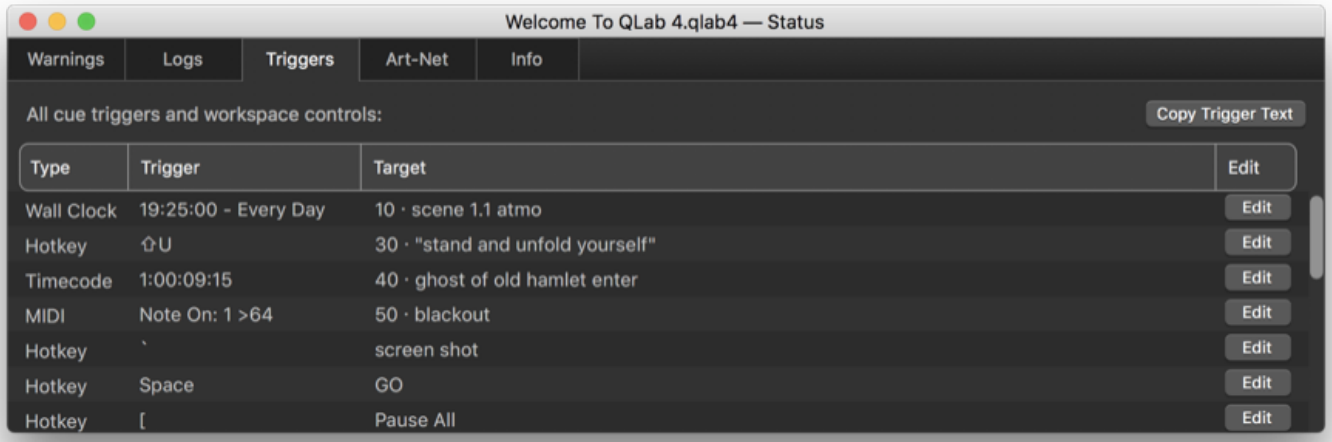
When a broken cue is fixed, its red ✕ will disappear. When a workspace contains no broken cues and no flagged cues, the warning icon (⚠) in the workspace footer will disappear as well.

Logs

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 computer's Console logs.

- **Cue triggers.** QLab will report any time a cue, cue list, or cue cart is triggered.
- **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 replies.** QLab will report any replies that it receives after sending OSC messages to other workspaces, devices, or programs.

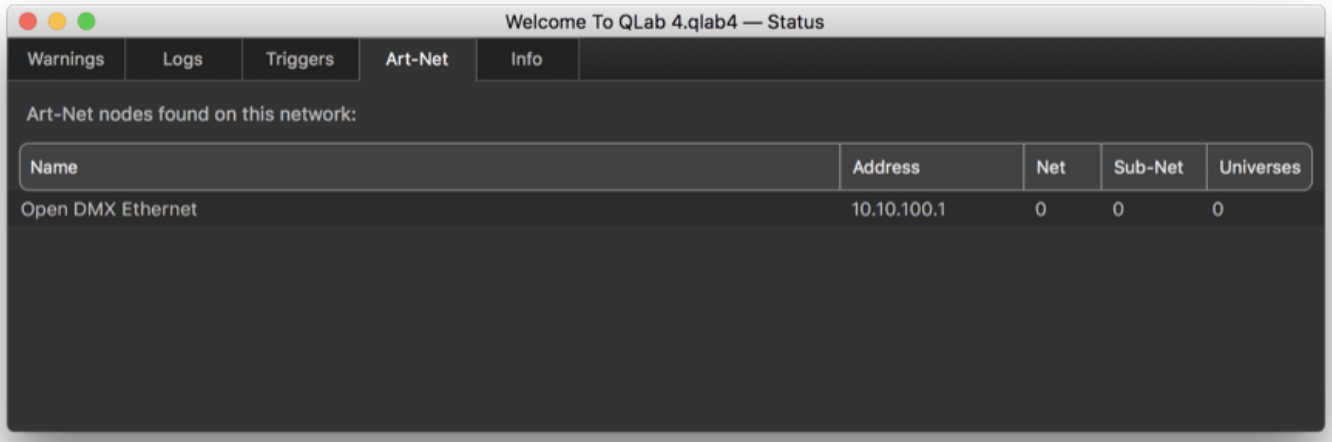
Triggers



The Triggers tab shows a table of all triggers and hotkeys in your workspace. This includes the keyboard shortcuts in the Key Map, as well as all types of Cue Triggers: hotkeys, MIDI triggers, Wall Clock triggers, and Timecode triggers.

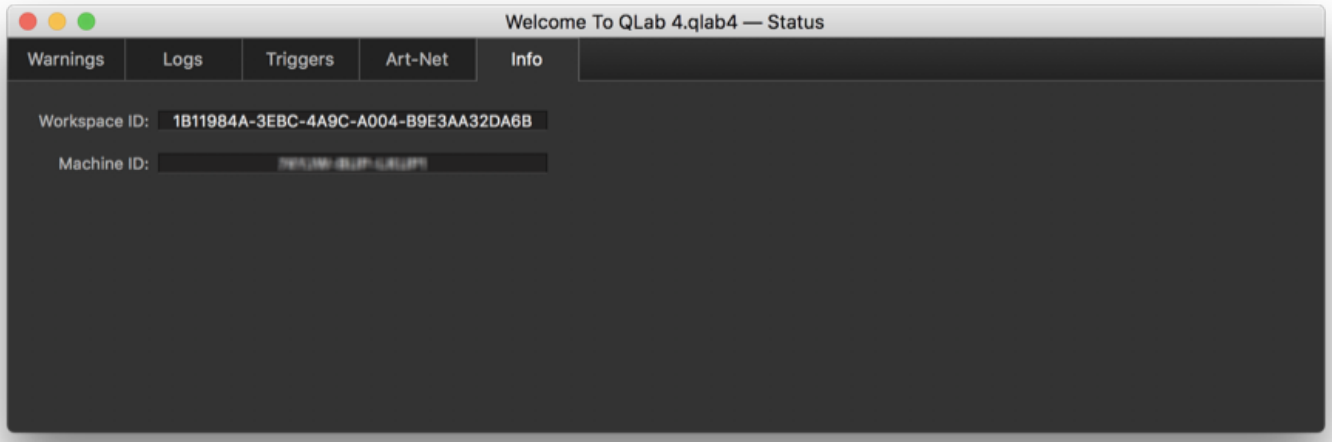
If you click the **Edit** button in each row, QLab will navigate to the appropriate window or inspector tab to allow you to edit that trigger or shortcut.

Art-Net



If your workspace contains at least one lighting instrument, QLab will scan the 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.

Info



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.

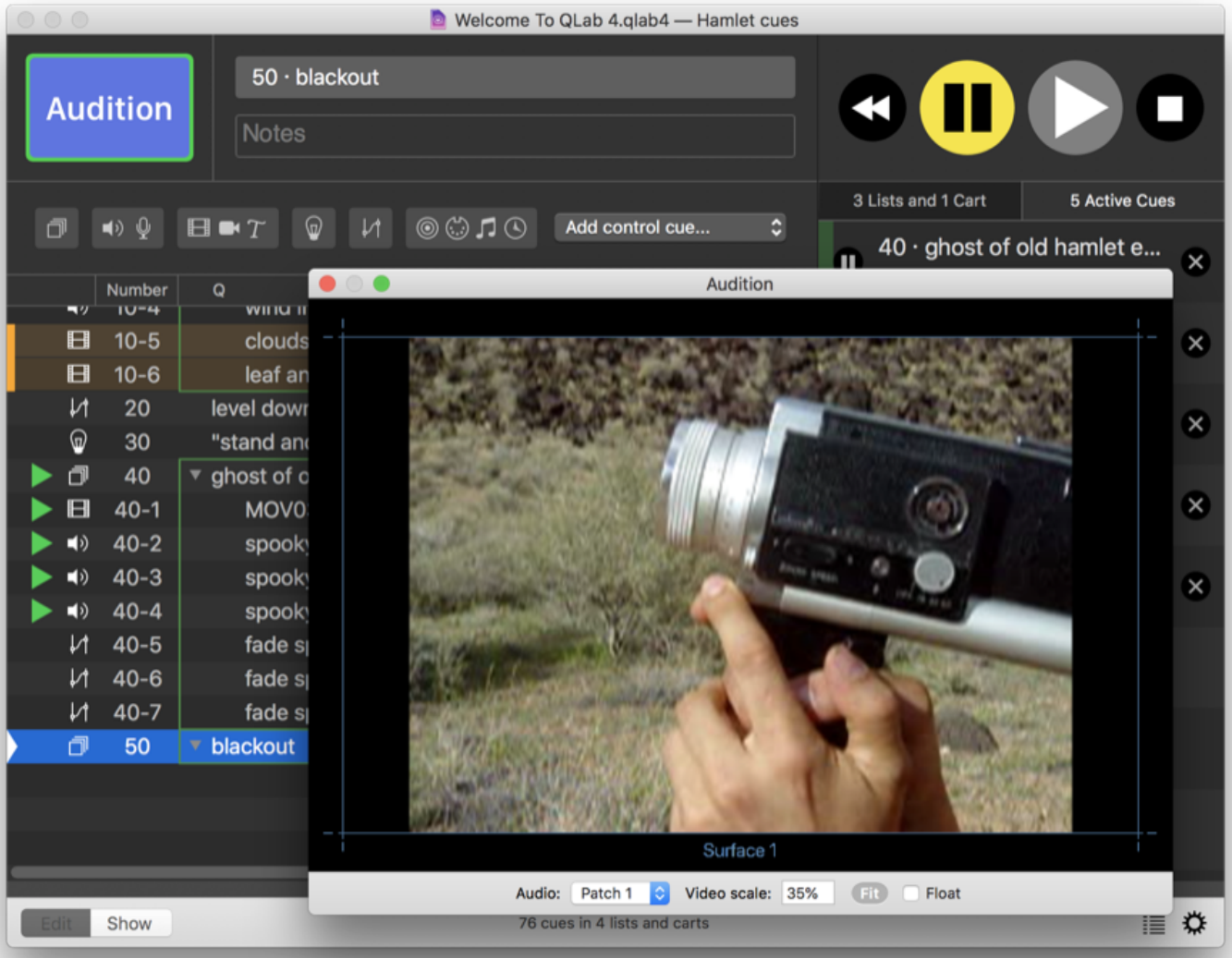
Machine ID. This number is unique to each Mac, and is the identifier that the license system uses to associate license seats with Macs. You may need to grab this number if you're communicating with support@figure53.com about license questions.

The Audition Window

The Audition Window allows you to listen to or watch cues privately without running them through your entire sound or video system. It also allows you to work with your cues when your playback system is not connected, or not turned on. You can think of it as a surrogate destination for all outgoing audio and video.

To access the Audition Window, select *Audition Window* from the **Window** menu or use the keyboard shortcut ⇧⌘**A**.

When the Audition Window is open, the GO button is relabeled "Audition" to indicate that triggering new cues will send them to the Audition Window rather than playing them normally.



You can optionally set the Audition Window to float on top of all other QLab windows by checking the box marked *Float*.

Audio

The Audition Window has its own audio patch selector; any cues started when the Audition Window is open will route their audio to that patch instead of the patch selected in the cue.

This is a convenient way to simulate a headphone or pre-fader listen (PFL) bus: if you're using patch 1 for your audio interface, then you can assign patch 2 to your headphone jack or other local monitoring output, and hear the audio out of that alternate device by opening the Audition Window.

Video

While the Audition Window is open, Video cues appear in the window instead of on their regularly assigned surface. If Video cues played to multiple surfaces are played at the same time, they are shown side by side in the Audition Window.

Any cues already playing when the Audition Window is opened will continue on their normal outputs. Only newly started cues are sent to the Audition Window and its audio patch.

You can scale video played in the Audition Window by any desired percentage using the scale control in the footer of the Audition Window, or you can click *Fit* to have QLab set the scale according to the current size of the window. It's best to click *Fit* while a Video cue is playing, otherwise QLab doesn't know what to measure against.

Lighting

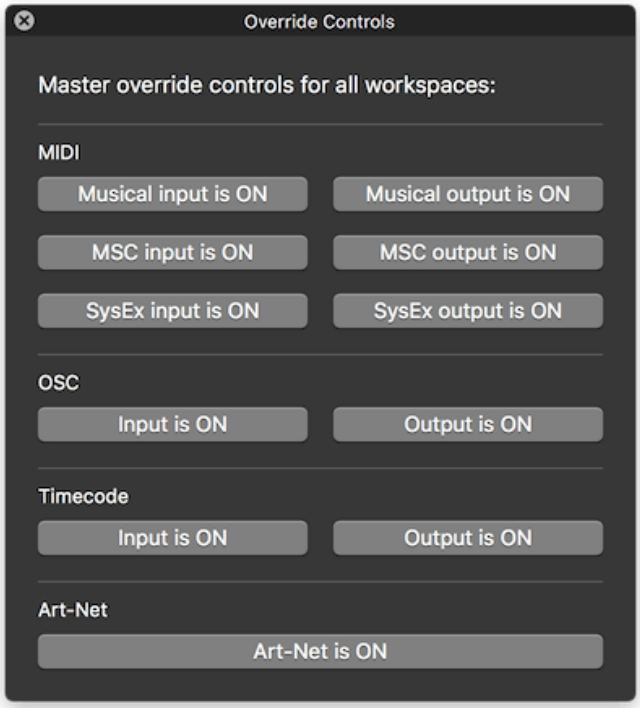
While the Audition Window is open, QLab will not transmit changes to Art-Net or DMX outputs. You can run cues and make changes in the Light Dashboard, but the actual lights in your system will remain unchanged.

Closing the Audition Window

To exit the Audition Window, simply close it or type ⇧⌘**A** again, and your workspace will return to normal. Any Audio, Mic, Video, Camera, and Titles cues which were started while the Audition Window was open will stop, and any cues which were started before the Audition Window was opened will remain unchanged. For lighting, QLab will resume sending DMX data, and the lights in your system will change to reflect the current state shown in the Light Dashboard.

Override Controls

The Override Controls window can be opened by choosing *Override Controls* from the **Windows menu** or by using the keyboard shortcut ⇧⌘**O**. This window provides a way to temporarily suspend input and output of various types of messages to and from QLab.



You can independently override input and output of each of the following types of messages:

- “Musical” MIDI voice messages
- MIDI Show Control (MSC)
- MIDI SysEx
- OSC
- Timecode
- Art-Net (and USB DMX)

Because QLab does not respond to incoming Art-Net messages, there is no Art-Net 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 which sends an overridden message will show a red circle icon in the status column (⃠) to indicate that the cue will not send its message when it is triggered.

Override controls are global, which means that they apply to all open workspaces.

Chapter 3: Audio

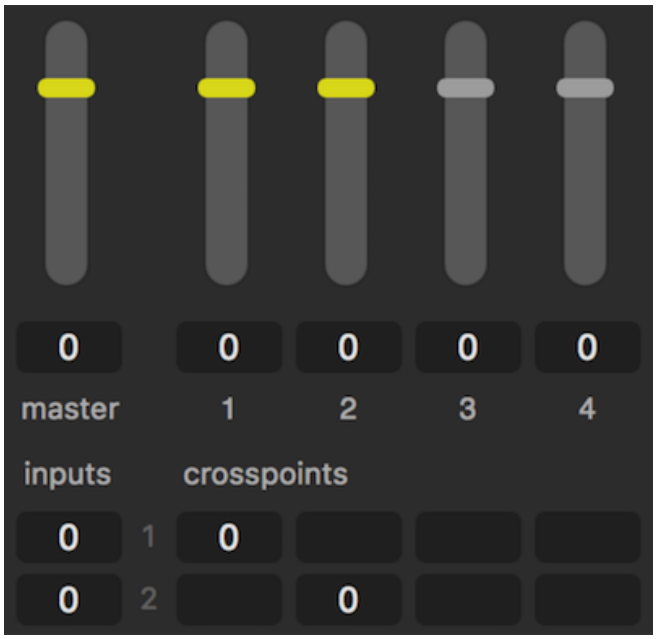
Introduction to Audio

Audio in QLab can originate from an audio file, the audio track within a video file, or a live input on an audio device connected to your Mac. From there, audio in QLab is routed using two matrix mixers. The first mixer is unique to each cue, and can be found in the Audio Levels tab of the inspector for Audio and Mic cues, or in the Audio Levels tab of the inspector for Video cues. The second mixer is unique to each audio device patched in your workspace, and it serves to route the output of the cues' individual mixers to the actual physical audio device. In its simplest form, therefore, audio in QLab traces the following path:

1. Audio file or device input,
2. Cue matrix mixer,
3. Device matrix mixer,
4. Audio hardware output,
5. Cable plugged into an audio console, amplifier, or powered speaker.

What Is A Matrix Mixer

A matrix mixer is made up of inputs and outputs, just like any mixer, that can be thought of as a grid of rows and columns. The rows represent inputs to the mixer, and the columns represent outputs from the mixer. The point at which each row intersects with each column is referred to as a crosspoint. If you imagine that each crosspoint is actually a volume knob which allows you to set the level of the input as it flows into the output, like an auxiliary send knob, then what you have is a matrix mixer. What makes it a matrix mixer is that every input can route into every output with just such a volume knob. There are no "main" outputs, no fixed-send busses, and no limitations on which inputs can go to which outputs.



In QLab, every cue that can deal with audio (that's the Audio cue, the Mic cue, the Video cue, and the Fade cue) has its own matrix mixer in which the rows represent either channels in the audio file, channels of audio embedded in a video file, or live inputs on an audio device. The columns represent cue outputs, which are like busses on a mixer.

The left-most cell of each row, labeled **inputs**, is the level control for that input.

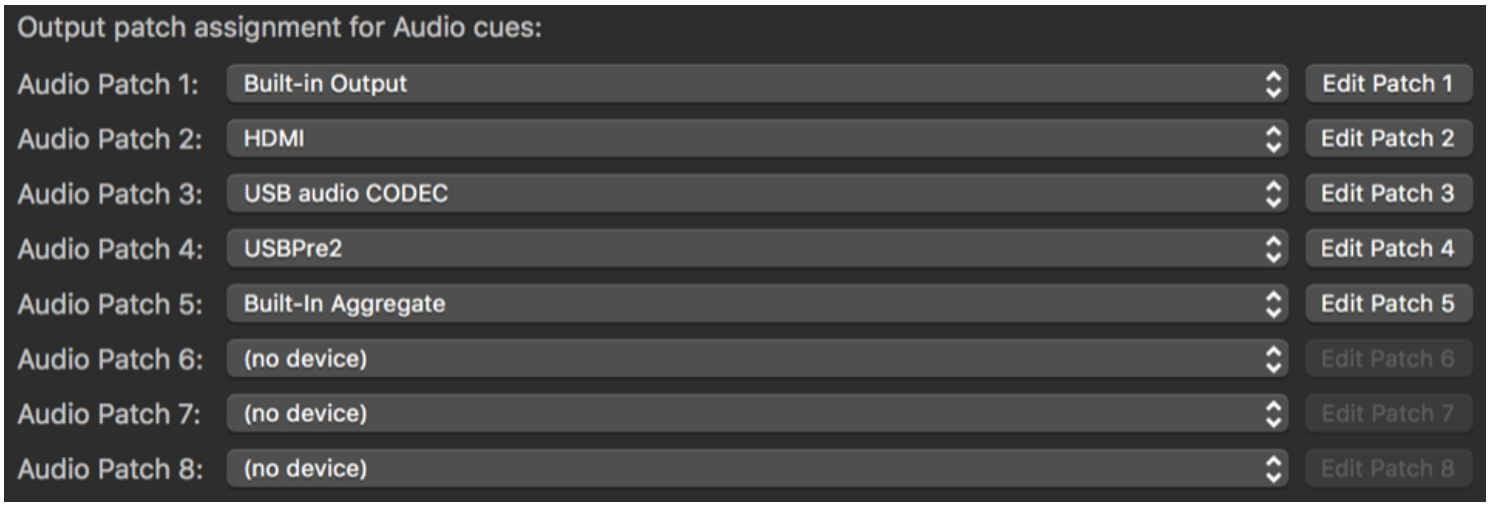
The top-most cell of each column, which is accompanied by a vertical slider control, is the level control for that cue output.

The top-left cell of the matrix, labeled **master** and accompanied by a vertical slider control, is the master output level of the entire cue.

The grid of cells within the matrix, labeled **crosspoints**, are the crosspoint level controls for each input/output connection.

Cue Outputs and Device Outputs

Cue outputs do not connect directly to the headphone jack or plugs on your audio interface. Instead, they use one of eight output patches, which are eight more individual matrix mixers that bridge the connection from cue output to actual physical outputs.



Each Audio, Mic, or Video cue can be assigned to one Audio Patch, and each Audio Patch contains a matrix mixer in which incoming audio from the cues' cue outputs are the rows, and actual outputs of the assigned audio device are the columns. By default, these are routed on a one-to-one basis; cue output 1 routes to device output 1, cue output 2 to device output 2, and so on. If you have a Pro Audio or Pro Bundle license, you can adjust the routing to suit your needs by clicking the *Edit Patch* buttons.

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. 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 what.

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.

Audio Cues

Audio cues allow you to play sound files with precise control over timing, levels, and routing. Audio cues must have a target, which is a sound file on your computer, and an output patch, which is a sound output destination such as your computers' speakers, headphone jack, or an audio interface.

When an Audio cue is triggered, it begins playing its target file. All Audio cues within a sequence play in sample-accurate sync as long as they're assigned to the same output patch.

Audio cues may target any file type supported by Core Audio, Apple's audio framework, but we recommend the following types:

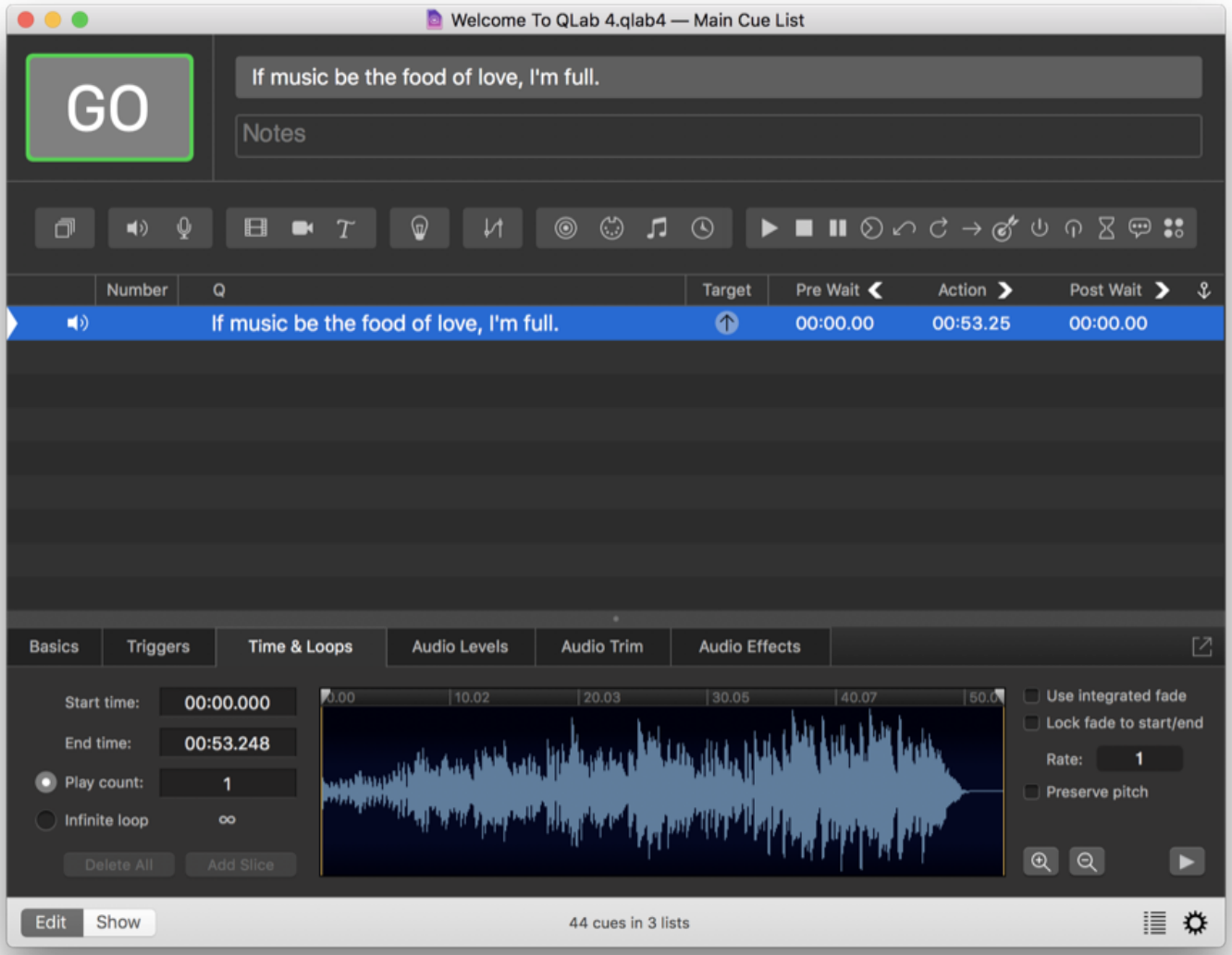
- AIFF
- WAV
- CAF
- AAC
- MP4
- M4A
- MP3 (Because MP3s have inherent timing problems, we do not recommend using them.)

When an Audio cue is selected, six tabs will appear in the Inspector:

- Basics
- Triggers
- Time & Loops
- Audio Levels
- Audio Trim
- Audio Effects

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Time & Loops



Start time and end time

You can set the start and end times in three ways:

1. You can type values into the start time and end time text fields.
2. You can drag the start and end time markers, which are downward-pointing grey triangles at either end of the waveform view.
3. You can click in waveform view, or preview the cue and then pause it, and then use the keyboard shortcut ⇧**I** ("I for in") to set the start time, or ⇧**O** ("O for out") to set the end time.

Play count and infinite loops

Play count is the number of times that the sound file will be played when the cue is run. The default is 1, meaning that the sound file will play once through and then stop. You can change the play count to any whole number to loop the sound file that number of times. Alternately, you can select *Infinite loop*, right below *Play count*, to loop the sound indefinitely. Starting with QLab 4.1, these options are available whether or not the cue contains slices.

Slices and the waveform view

The center of the Time & Loops tab shows a waveform view of the target audio file. You can zoom in and out using the + and - buttons to the right of the waveform, or by scrolling vertically.

The *Play count* and *Infinite loop* options can loop the entire sound file, but QLab can also loop specific sections of the file. To do 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. 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 `0, inf`, or in fact any text that is not a whole number.

You can click the handle at the top of the 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. Note that slice markers cannot be closer together than .05 seconds.

You can also click and drag within the waveform to highlight and select a section of the track, which will cause the *Add Slice* button to place slice markers on both sides of the selection.

If you set an Audio cue to target an AIFF or WAV file with preexisting 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 will not be included in bounced files, making QLab's automatic importing of makers somewhat less valuable. 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!

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. To learn more about this, visit the section on Devamp cues in this documentation.

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 marked *Use integrated fade* to the right of the waveform view. A yellow line will appear along the top of the waveform view, and you can click and drag along the line to create fade points. QLab creates a smooth fade between points, but you can make sharper changes by adding points close together.

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 check the box marked *Lock fade to start/end* to automatically stretch the integrated fade curve to fit within the start and end time of the cue. If you leave the box unchecked, the integrated fade curve will remain locked to the natural start and end time of the sound file, regardless of the start and end times of the cue.

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, 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 on octave

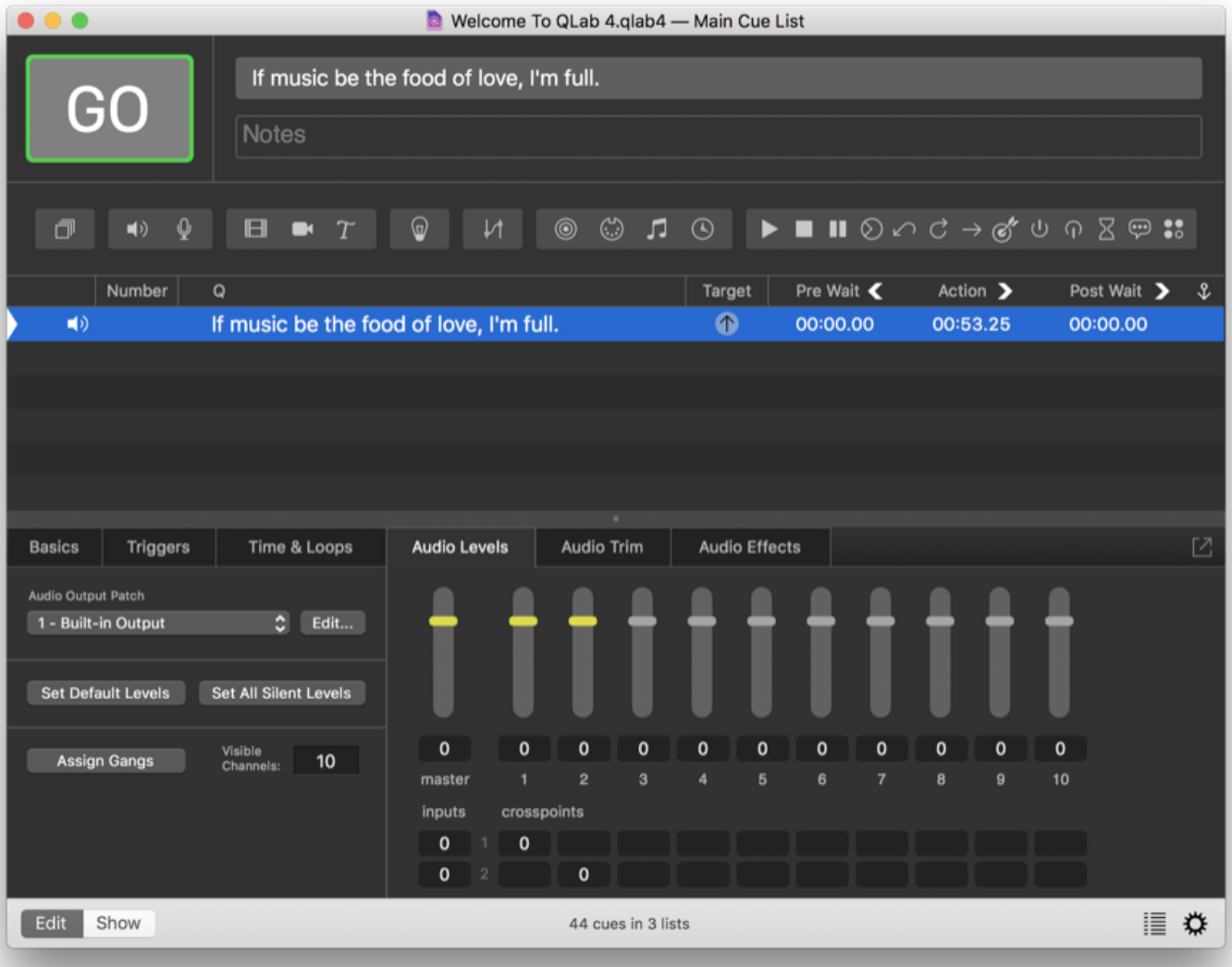
The minimum rate is `0.03`, and the maximum rate is `33`.

It is worth noting that checking the *Preserve pitch* box requires slightly more processor power than leaving the box unchecked.

Preview

In the lower right corner of the *Time & Loops* tab is the *Preview* button which is labeled with a right-pointing triangle. Clicking this 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 playing, this button changes to a pause button.

Audio Levels



Audio Output Patch

Choose your output patch from the drop-down menu in the top-left corner of the *Audio Levels* tab. The *Edit* button directly to its right is a shortcut to the Audio Patch Editor for the selected patch. Generally, you won't need to touch this much after you set it up for your show. You can learn more about the Audio Patch Editor in its section of this documentation.

Level shortcuts

Default Levels. This button sets all levels in the current Audio cue to match the Audio cue template, which can be found and adjusted to suit your needs in the Cue Templates section of Workspace Settings.

Set all silent. This button does exactly what it seems: it sets all levels for the current Audio cue to `-inf`.

Gangs

When you click *Assign Gangs*, 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.

Visible channels. This control allows you to set the number of output channels currently visible in the mixer. Outputs which are not displayed are not disabled, they're simply hidden from view. This is only relevant if you have a Pro Audio or Pro Bundle license, since the free version of QLab only permits a maximum of two outputs. The maximum number of outputs available with a Pro Audio or Pro Bundle license is 64.

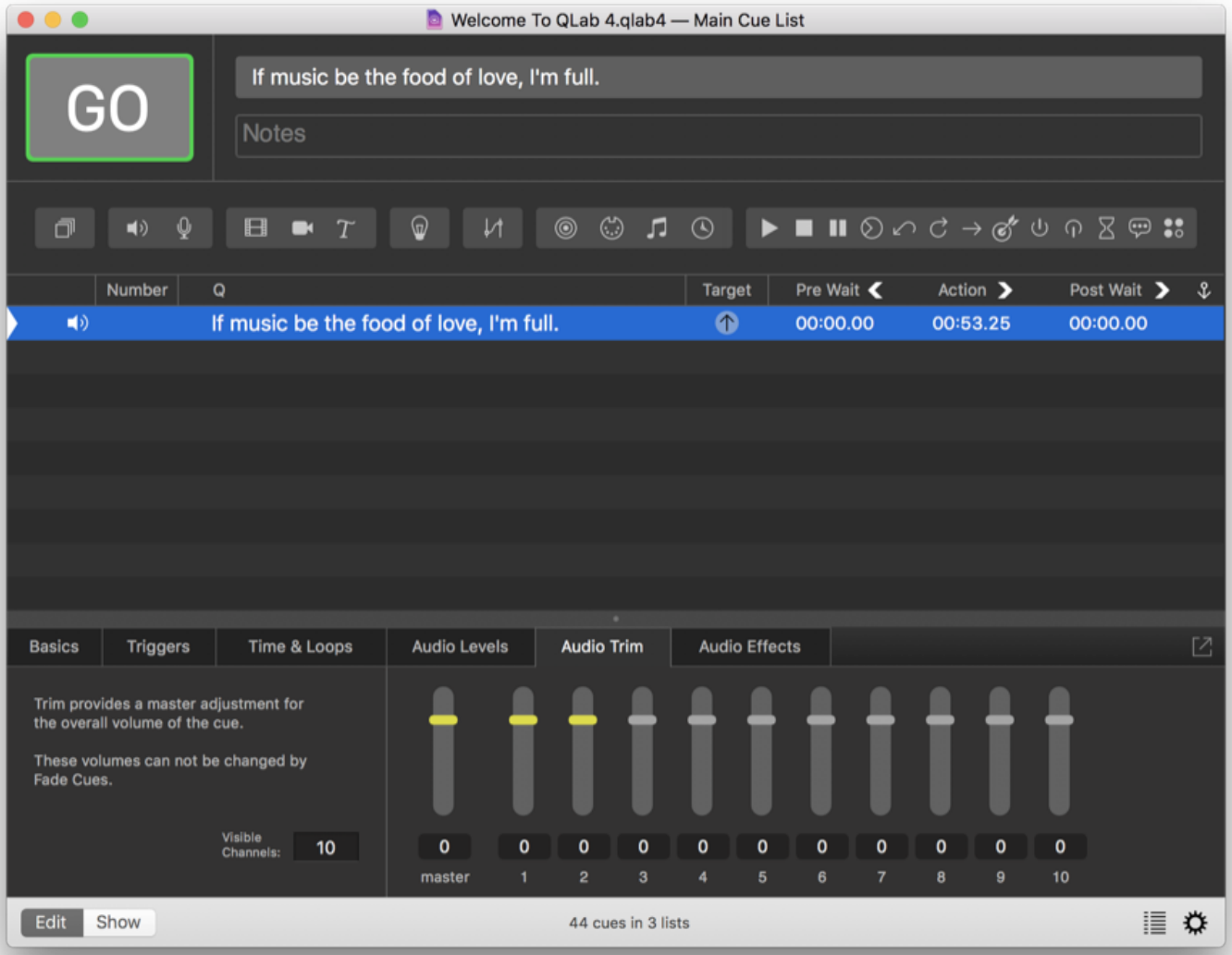
The Matrix Mixer

The matrix mixer is a grid made of rows and columns, like a spreadsheet. The master output level for the cue is the on the top left, the cue outputs are the column headers, and channels in the audio file are the rows. Typically, there will be two rows besides the header: one for the left channel and one for the right, but QLab supports audio files of up to 24 channels with a Pro Audio or Pro Bundle license.

Crosspoints are the level controls for routing a given input (row) to a given output (column).

Levels can be typed in to 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 in any level above `0` manually, limited by the maximum level set in audio settings. Since the majority of levels set are below `0 db`, QLab will interpret a number entered without a + or - sign as a negative number.

Audio Trim



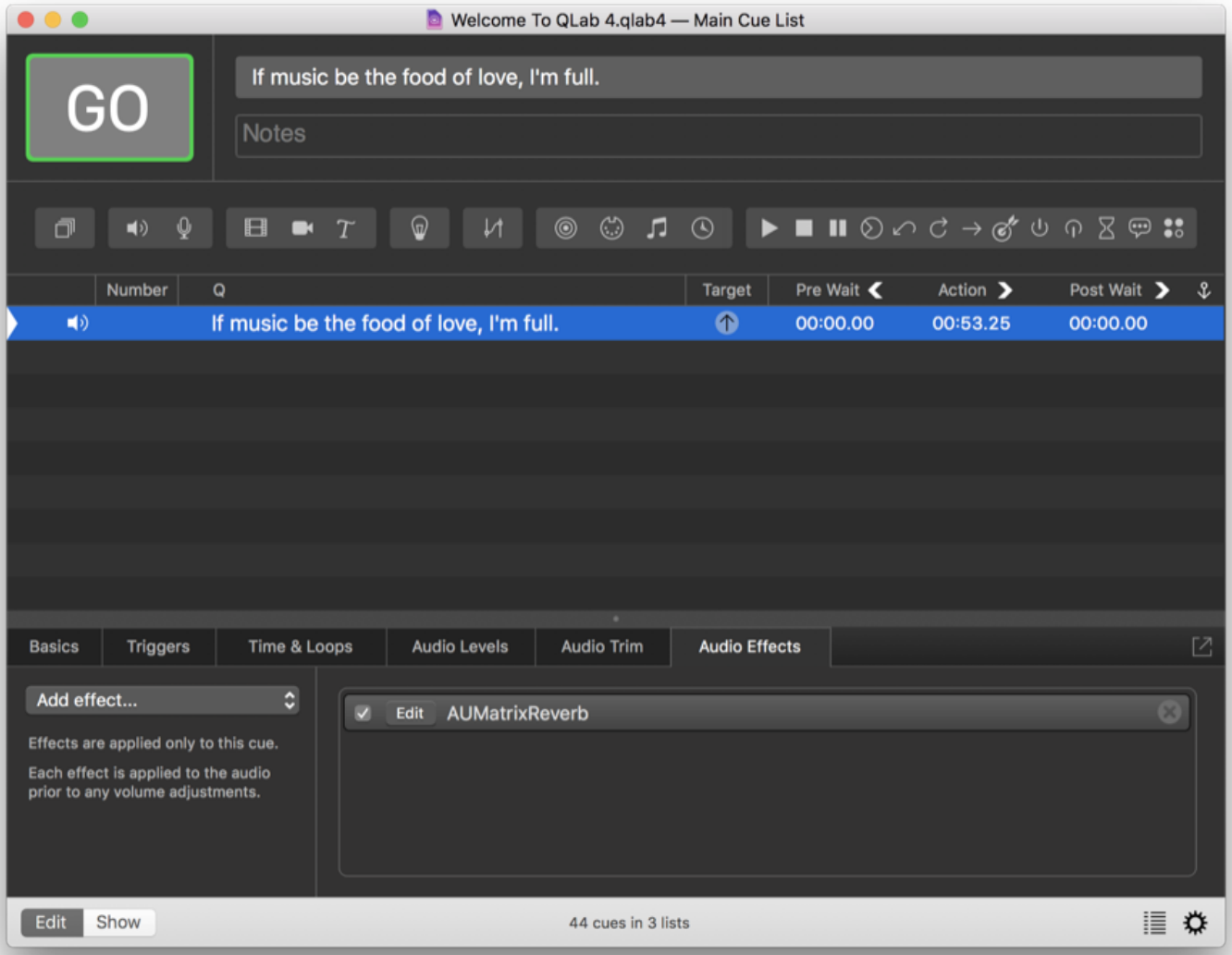
The trim tab is similar to trim on an audio mixer: it provides a master adjustment for the overall volume of the cue. These volumes can not be affected by Fade Cues.

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 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.

Audio Effects

With a Pro Audio or Pro Bundle 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 64-bit, 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 on your audio file needs to match the number of channels supported by the AudioUnit. If there is a mismatch in channel count, some AudioUnits will pass audio without rendering, and others won't pass any audio at all. 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 apply an effect to an Audio cue, select it from the drop-down menu labeled *Add 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 *Edit* 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.

Please note that meters in AudioUnits used in Audio cues do not render at this time.

You can bypass effects by unchecking the box to the left side of the effect, and you can remove effects by clicking the *X* to the right side. If the AudioUnit editor window is open, you can also turn the effect on or off by checking or unchecking *Enabled* in the top right corner of the effect window.

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 Patch Editor section of this documentation.

Broken Cues

Audio cues can become broken for the following reasons:

Invalid Audio File

Either the file is missing or damaged, or it's not one of the supported audio file types.

No audio device. Select one in the "Audio Levels" tab.

You may also need to visit the Audio section of Workspace Settings and connect an audio device to the desired patch.

One or more audio effects in this cue are missing.

This typically happens when a workspace that uses audio effects is moved from one computer to another, and the new computer does not have the necessary AudioUnit installed. Either choose a different audio effect, or install the necessary AudioUnit.

One or more audio effects on the cue outputs are missing.

This typically happens when a workspace that uses audio effects is moved from one computer to another, and the new computer does not have the necessary AudioUnit installed. Either choose a different audio effect, or install the necessary AudioUnit.

One or more audio effects on the device outputs are missing.

This typically happens when a workspace that uses audio effects is moved from one computer to another, and the new computer does not have the necessary AudioUnit installed. Either choose a different audio effect, or install the necessary AudioUnit.

A pro license is required to use audio effects.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the audio effects from this cue.

Timecode triggers require a Pro license.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the timecode trigger from this cue.

Mic Cues

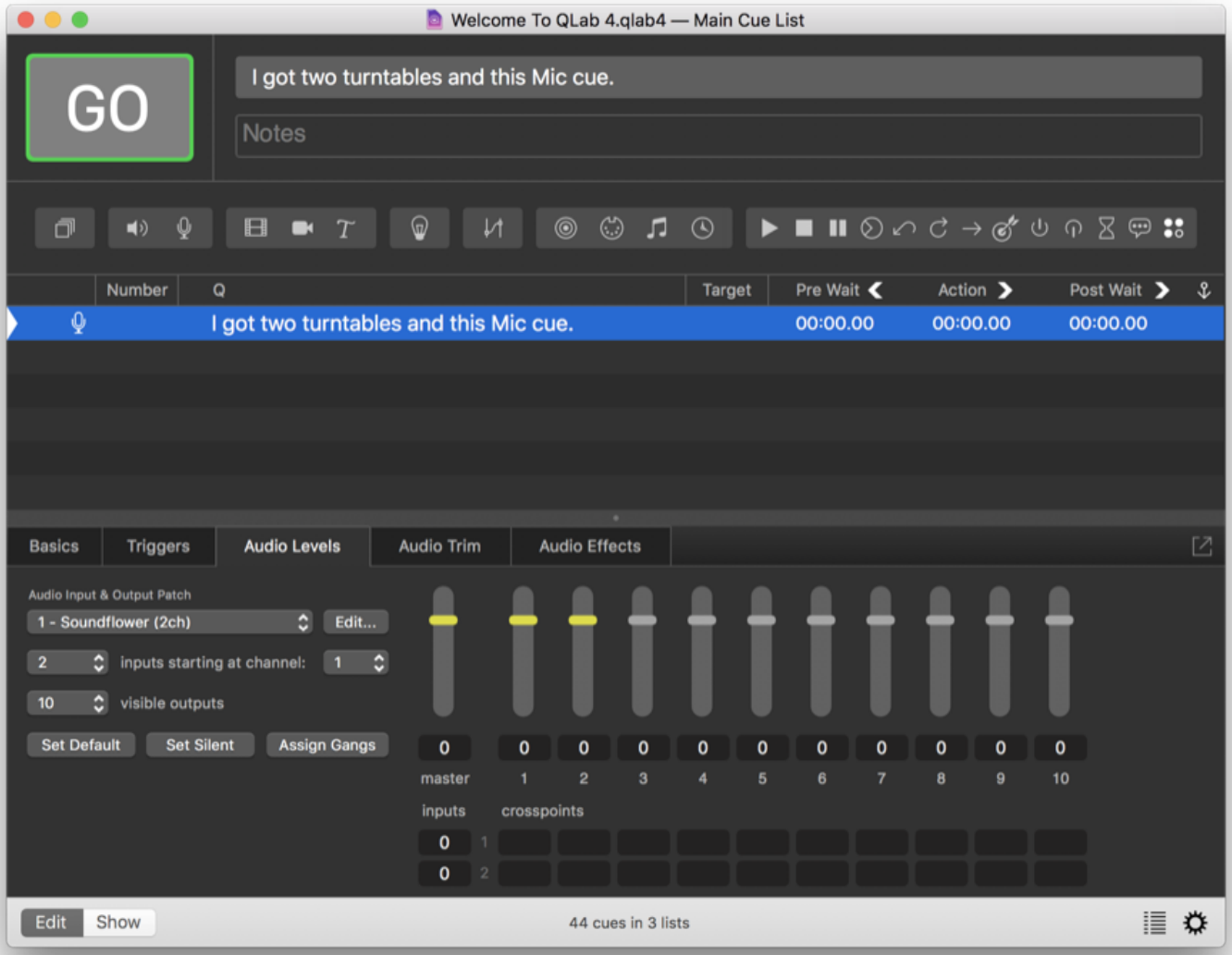
Mic cues are similar to Audio cues, except instead of targeting a track on your computer, Mic cues take in live audio from a microphone or any live source of audio plugged into your audio interface.

When a Mic cue is selected, four tabs will appear in the Inspector:

- Basics
- Audio Levels
- Trim
- Audio Effects

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the the page on the Inspector in the General section of this documentation.

Audio Levels



Audio input & output patch. A Mic cue must use the same audio device for both its input and its output. This can be the same device used for Audio cues in the same workspace, but it must be set up separately and will appear as a separate device in QLab. If you need to use one device for input and a separate device for output, you must set up an aggregate audio device (using an Apple-supplied application called Audio MIDI Setup, found in Applications/Utilities. The process is described in this Apple support document).

The *Edit* button is a shortcut to open the patch editor for the selected input & output patch. Typically, this should be set up once and then left alone for the majority of a show, but during early rehearsals it can be convenient to have quick access here.

Setting Inputs

The pair of drop-down menus that looks like *X inputs starting at channel: Y* lets you select which input channels, and how many, you want to use for the Mic cue. Mic cues can use anywhere between 1 and 24 inputs at a time, and the total number of possible input channels depends on the audio interface that you're using.

The first drop-down menu lets you choose the number of inputs you want to use. Letting you choose the number of inputs lets you keep your cues looking clear and simple, but it also lets you choose a specific number of inputs to allow compatibility with some Audio Effects which require a specific number of inputs. For example, AUMatrixReverb only works on cues with two input channels.

The second drop-down menu lets you choose which input channel on the audio interface will be mapped to the first input in the Mic cue. If you set the cue to use more than one channel, the channel chosen in the first drop down menu will be represented by the first row in the matrix mixer, then the following

channel will be the second row, and so on.

Visible Outputs. This control allows you to set the number of output channels currently visible in the matrix mixer. Outputs which are not displayed are not disabled, they're simply hidden from view.

Default Levels. This button sets all levels in the current Mic cue to match the Mic cue template, which can be found and adjusted to suit your needs in the Cue Templates section of Workspace Settings.

Set all silent. This button does exactly what it seems: it sets all levels for the current Audio cue to `-inf`.

Assign Gangs. Assigning gangs for Mic cues is exactly the same as for Audio cues; please refer to Audio Cues.

Matrix Mixer. The matrix mixer in Mic cues works the same as in Audio Cues.

Audio Trim

Audio Trim for Mic cues operates the same as Trim for Audio Cues.

Audio Effects

Effects for Mic cues operate the same as Effects for Audio Cues.

Broken Cues

Mic cues can become broken for the following reasons:

No audio device. Select one in the "Audio Levels" tab.

You may also need to visit the Audio section of Workspace Settings and connect an audio device to the desired patch.

One or more audio effects in this cue are missing.

This typically happens when a workspace that uses audio effects is moved from one computer to another, and the new computer does not have the necessary AudioUnit installed. Either choose a different audio effect, or install the necessary AudioUnit.

One or more audio effects on the cue outputs are missing.

This typically happens when a workspace that uses audio effects is moved from one computer to another, and the new computer does not have the necessary AudioUnit installed. Either choose a different audio effect, or install the necessary AudioUnit.

One or more audio effects on the device outputs are missing.

This typically happens when a workspace that uses audio effects is moved from one computer to another, and the new computer does not have the necessary AudioUnit installed. Either choose a different audio effect, or install the necessary AudioUnit.

A pro license is required to use audio effects.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the audio effects from this cue.

Timecode triggers require a Pro license.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the timecode trigger from this cue.

Fading Audio

A Fade cue can be used to adjust the volume levels and audio effect parameters of a targeted Audio, Video, or Mic cue. 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” 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 target and a duration, and must adjust at least one level or parameter.

To learn how to set a target for a Fade cue, please refer to the section on targeting other cues in the **Getting Started** section of this documentation.

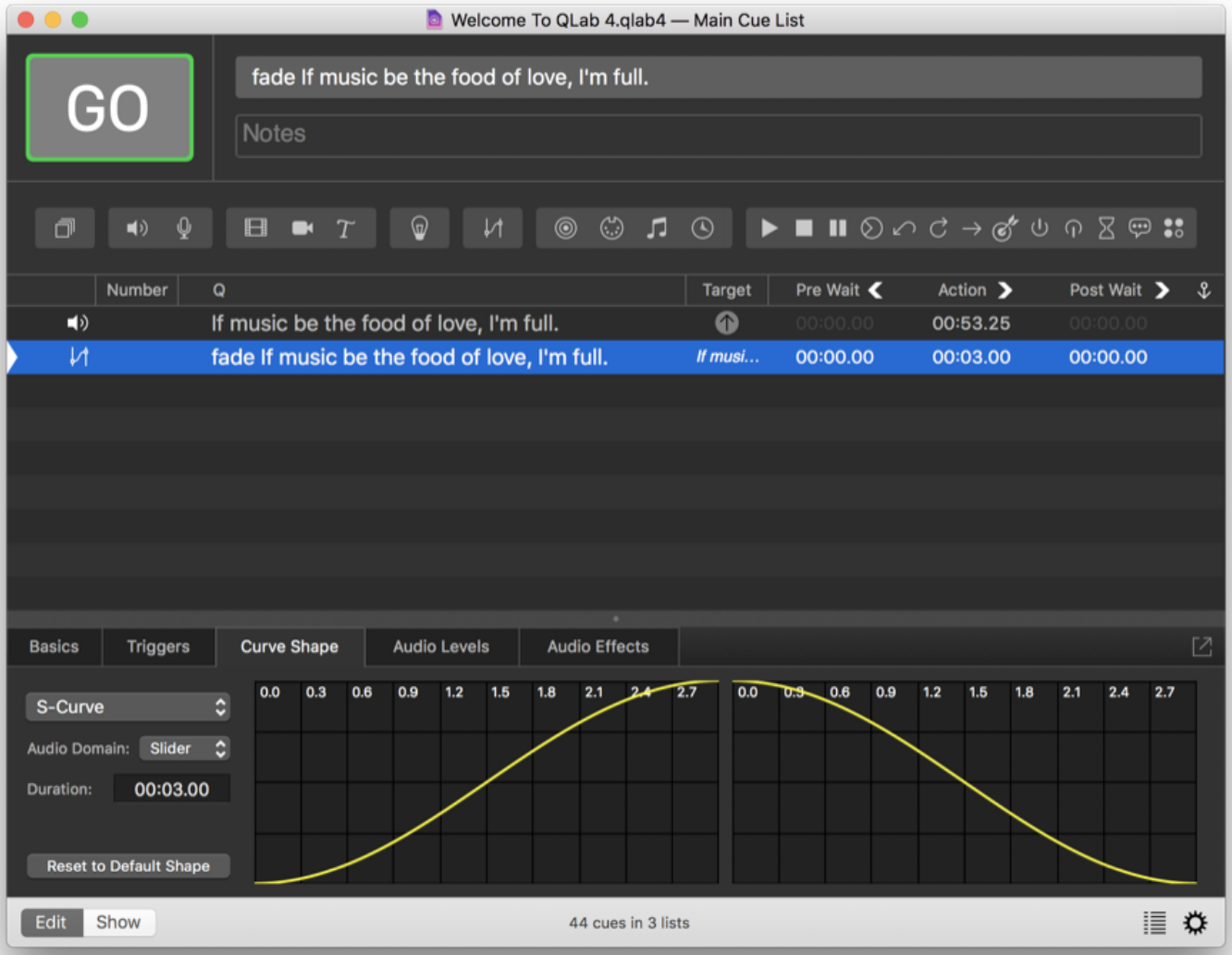
When a Fade cue which targets an Audio or Mic cue is selected, five tabs appear in the Inspector:

- Basics
- Triggers
- Curve Shape
- Levels
- Audio Effects

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Curve Shape

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 are increasing, and the curve on the right is for levels which are decreasing. The curve shape that appears by default is set according to the Fade cue’s template, but you can choose another fade shape from the drop-down 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 b dragged to change the shape of the curve. 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.
- **Parametric Curve.** If you choose this option, a text field labeled *Intensity* appears below the drop-down menu. This allows you to use a mathematically precise parametric fade shape.
- **Linear Curve.** This option provides a straight, linear fade curve.

The *Audio Domain* drop-down menu lets you choose the scale that QLab uses to fade audio levels. This drop-down is only relevant to the Audio Levels tab.

- **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.

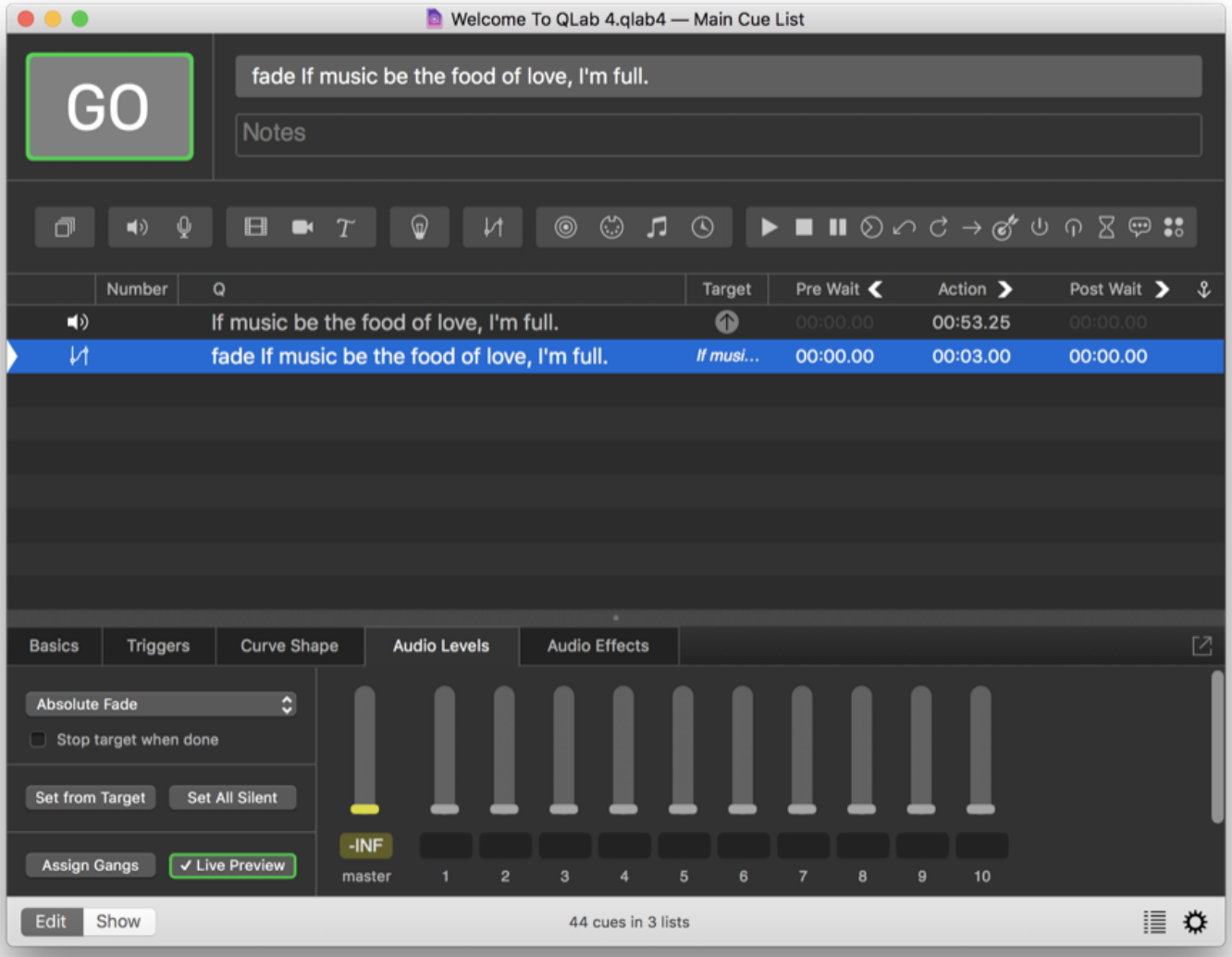
Equal Power and Equal Gain fades

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.**

Audio Levels

The Audio 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.



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.

Stop Target When Done. Check or uncheck this box depending on whether you would like the target cue to continue playing after the Fade cue is complete, or stop once the Fade is complete.

Set from Target. Clicking this button, or using the keyboard shortcut ⇧⌘**T**, 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.

Set All Silent. Clicking this button will return the fade to a pristine state.

Assign Gangs. This button behaves the same way as it does in an Audio cue.

Live Preview. This button provides quick access and a visual reference to the live fade preview setting.

The right side of the Levels tab is a matrix mixer which shows the same number of input rows as the Fade's target cue. 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 Effects

In addition to adjusting audio levels, Fade cues can be used to adjust any audio effects on their target cue. When you apply an effect to an Audio cue, any Fade cues that target that Audio cue will automatically recognize the effects you've applied, and they will be listed in the Audio Effects 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 control 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 Effects from Target* to copy the state of the audio effect 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 Audio 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 Audio 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 Audio 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. To learn more about effects on outputs, refer to the section of the documentation on the Audio Patch Editor.

Fading Rate

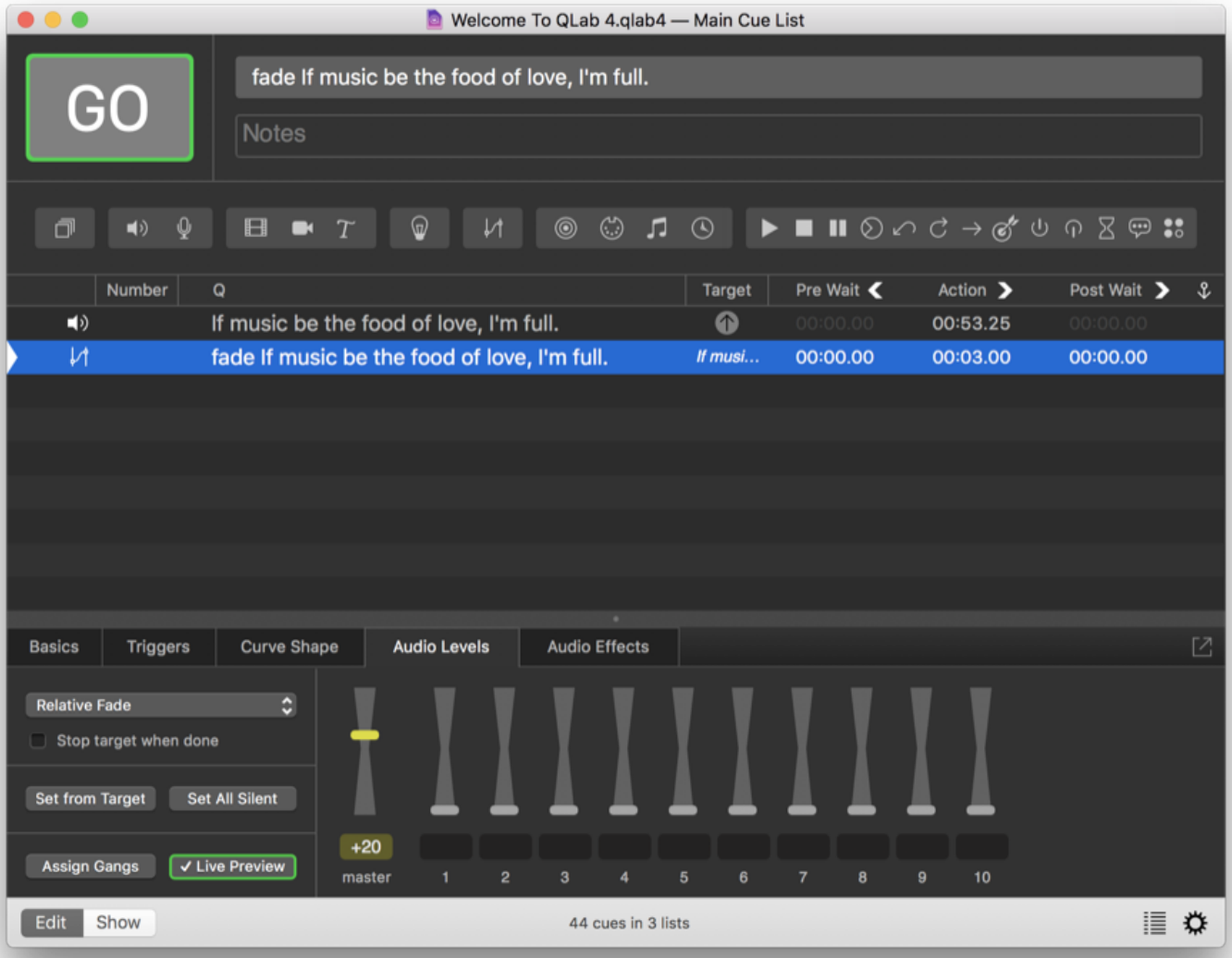
The Audio Effects tab of the inspector also lets you fade the playback rate of the target cue. You can check the box labeled *Fade rate to*, and set a target rate. The maximum playback rate in QLab is `33`, and the minimum is `0.03`.

Please note that to fade the rate of a Video cue, that Video cue's target media must have an audio track.

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 master level of a target Audio cue to `-12`, then the master level of that Audio cue will end up at `-12` after running the Fade no matter what the level was when the Audio cue started off.

However, using the drop-down menu at the top left corner of the Levels tab, you can set a Fade cue to be a *relative* fade. Relative fades add or subtract a given amount to the active parameters, so the starting point of those parameters very much matters.



In this screen shot, the Fade cue raises the master level of the target cue by 20 db. So if the target cue were at `-40`, running this Fade cue would bring the level to `-20`. If the target cue were at `-10`, running this Fade cue would bring the 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 level of an Audio cue from a very soft level, say `-50`, up to `-10`. You'd set the Audio cue to `-50`, then create a relative Fade cue with a level change of `+40`. If you accidentally ran that Fade cue twice, QLab would attempt to fade the Audio cue to `+30`, which is rather loud!

To help protect you against this, QLab has a maximum level, which is set in the Audio section of Workspace Settings. By default, this is `+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 loud.

Using a mixture of absolute and relative fades on the same target cue can have unpredictable results due to the different ways that they operate. If you do use a mix of fade types, be sure to test your cues and cue sequences thoroughly.

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 that targets a mono track, and you want to pan it from one speaker to another. QLab will recognize the track has one channel, and one row will appear in the Audio Levels tab of the inspector for that Audio cue. In the Audio cue, set the level for the first speaker (cue output 1, the first column of the matrix) to `0`, and set the level for the second speaker (cue output 2, the second column of the matrix) to `-INF`. (Note that QLab treats a blank field in this matrix 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 play the Audio cue, you will hear it exclusively from output 1. When you play the Fade cue, it will fade down in output 1 and fade up in output 2 over the same duration.

Remember too that one Fade cue can affect as many controls as you wish. If you have a track with eight channels instead of just one, and you have ten 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 an eight-channel Audio cue fading amongst ten 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 the target of the selected Fade cue 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 Cues

Fade cues can become broken for the following reasons:

No target cue.

Assign a target to the cue.

No fade parameters have been enabled; pick at least one thing to fade.

You can enable or disable an audio control by clicking in it; active controls will be highlighted in yellow. You can enable or disable a video control by checking or unchecking the box next to it.

One or more audio effects in this cue are missing or broken.

This typically happens when a workspace that uses audio effects is moved from one computer to another, and the new computer does not have the necessary AudioUnit installed. Either choose a different audio effect, or install the necessary AudioUnit.

A pro video license is required to fade video geometry.

You'll need to either install a Pro Video or Pro Bundle license, or adjust the Fade cue to not fade video geometry parameters.

A pro license is required to fade the playback rate.

You'll need to either install a Pro Audio, Pro Video, or Pro Bundle license, or adjust the Fade cue to not fade rate.

A pro audio license is required to fade audio effects.

You'll need to either install a Pro Audio, Pro Video, or Pro Bundle license, or adjust the Fade cue to not fade audio effects.

Timecode triggers require a Pro license.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the timecode trigger from this cue.

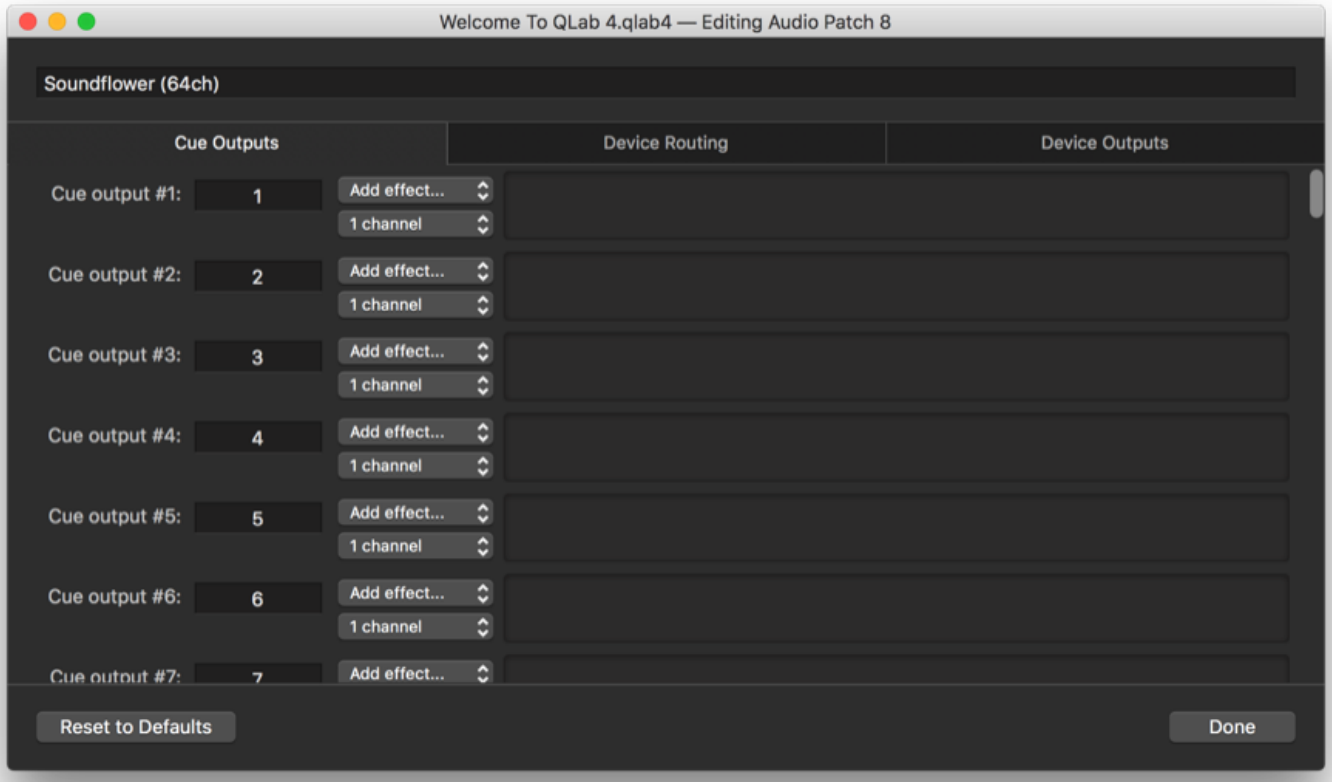
Audio Patch Editor

The Audio Patch Editor, which is available with a Pro Audio or Pro Bundle license, allows you to customize the routing of Cue Outputs to Device Outputs, and add audio effects to each. In this way, QLab can handle many of the tasks usually performed by a sound console or system processor.

To get to the Audio Patch Editor, go to Workspace Settings and choose *Audio* from the list on the left, then click the *Edit* button next to one of the eight audio patches.

The Audio Patch editor has three tabs, Cue Outputs, Device Routing, and Device Outputs. Above the tabs is a text field which lets you enter a custom name for the device patch. In the lower left corner of the window is a button labeled *Reset to Defaults* which will restore all controls and settings in the current audio patch to their defaults. In the lower right corner is a button labeled *Done* which commits the changes you've made, and closes the window.

Cue Outputs



Each row in this tab represents one Cue Output, which are the columns in the matrix mixer in the Audio Levels tab of the inspector for Audio, Mic, Video, and Fade cues.

You can rename each cue output as needed 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 Audio Levels tab of the inspector. There's not a lot of space there, so it's best to use fairly brief names.

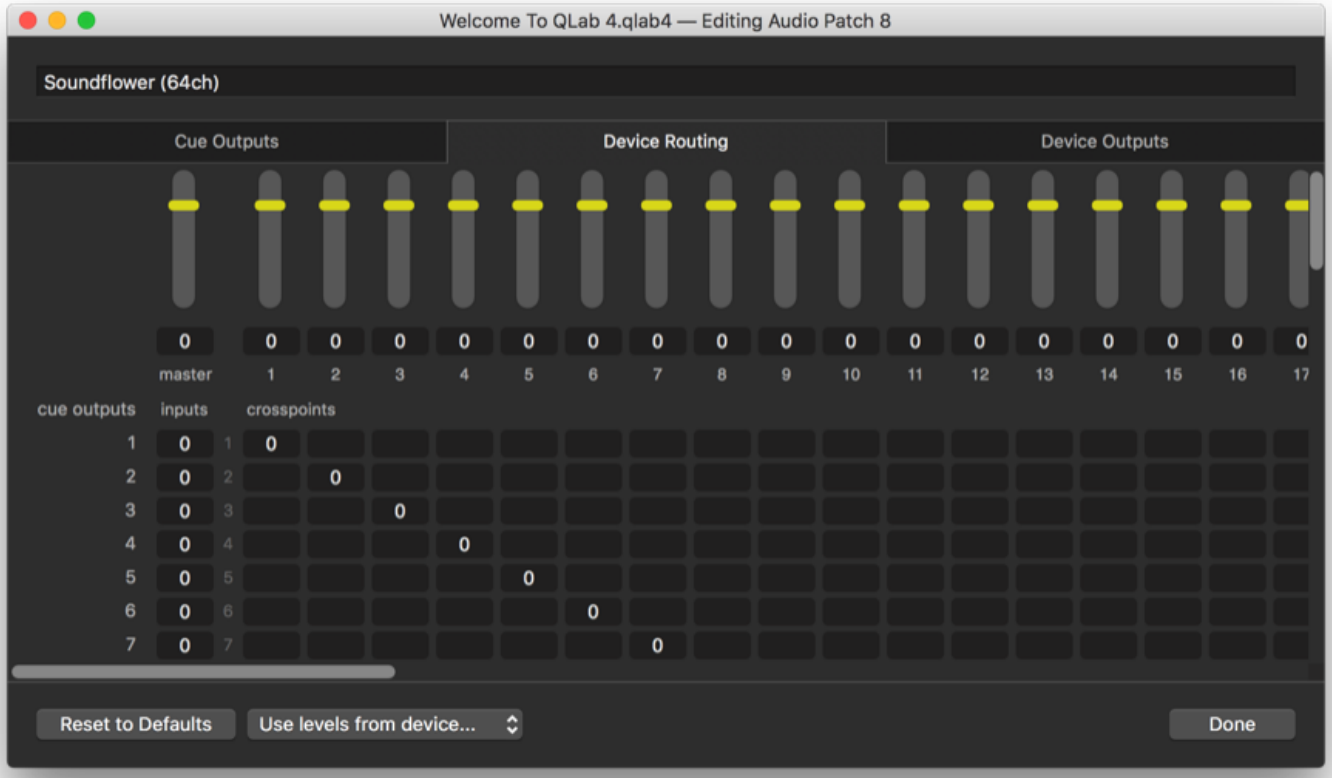
Next are two drop-down menus, which say *Add effect...* and *1 channel*. The first drop-down menu allows you to choose an available AudioUnit effect to insert on the Cue Output. All AudioUnits installed on the Mac which appear to be compatible with QLab will be listed under the drop-down.

Some AudioUnits, notably Apple's built in AUMatrixReverb, only work when they're supplied with two channels of input, so for this reason the *1 channel* drop-down can be used to gang a Cue Output with the following Cue Output. This ganging-together is only relevant to AudioUnits on Cue Outputs, and has no

effect anywhere else in QLab.

AudioUnits inserted on Cue Outputs will appear in the area to the right of the drop-down 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.

Device Routing

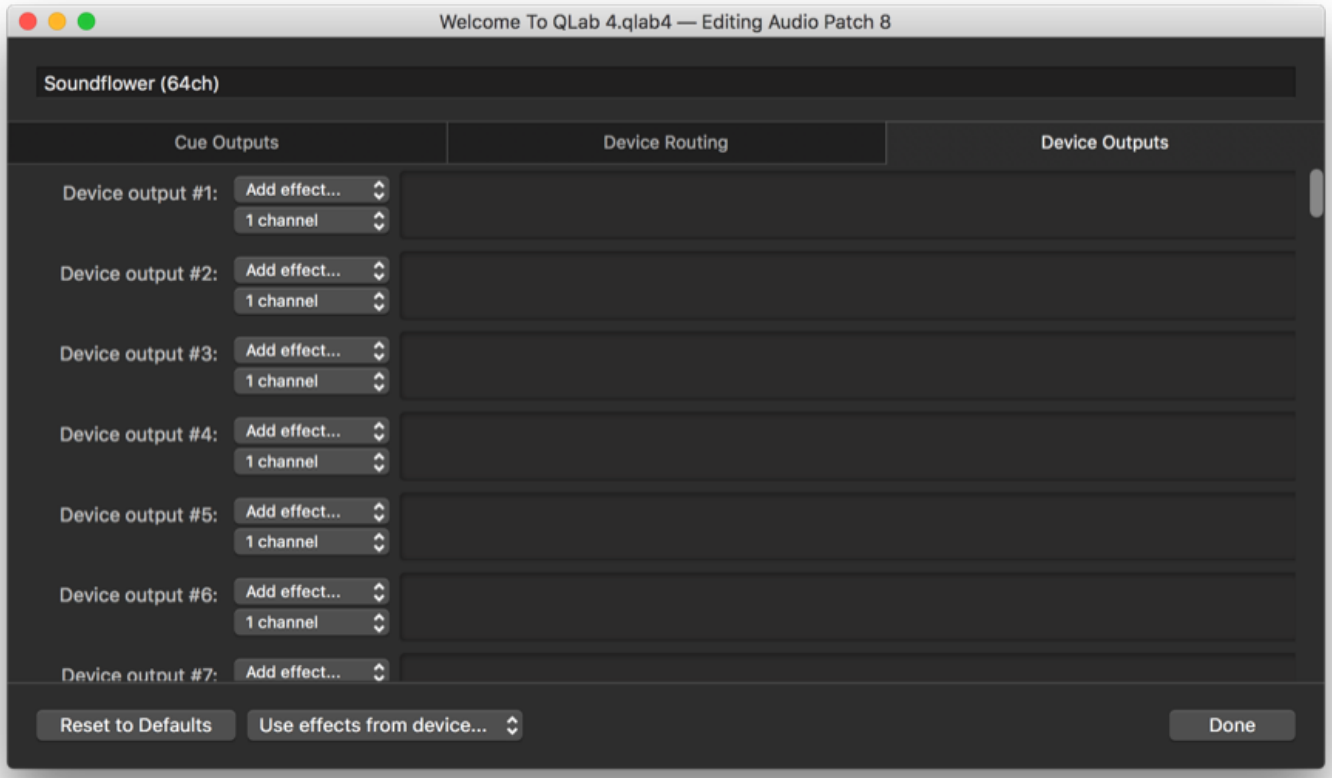


The matrix mixer in this tab allows you to do the actual routing of Cue Outputs to Device Outputs. Rows in the mixer represent Cue Outputs, and columns represent Device Outputs.

The overall volume of all audio output from your workspace can be adjusted using the master level control in the top left corner of this matrix.

Any Cue Output can be routed to any Device Output, or to more than one. 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.

Device Outputs



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.

Chapter 4: Video

Introduction to Video

Video in QLab is designed to be flexible and adaptable. The video workflow is intended to allow you to set up your workspace, configure your hardware, and then basically forget about it and just focus on your projection design.

A note on style

> Following the nomenclature of professional theater in the US, "projection" is taken to mean any form of video or film as used in a live performance. No matter 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, a Video cue is the type of cue that deals with projection. "Screen" means any physical device that displays the contents of Video cues. The words "screen" and "display" are more or less interchangeable.

QLab uses a concept called *Surfaces* to output video. In QLab, a surface is a sort of virtual video output which has one or more actual video screens assigned to it. By creating QLab surfaces which represent the physical surfaces on which you're displaying video (using projectors, monitors, LED walls, or anything else), QLab allows you to focus on the content of your design rather than the mechanical details of your projection system.

The idea is that each surface in a QLab workspace corresponds conceptually to a physical projection space on the stage. For example, you may be projecting onto two walls and a door, all at different angles and all covered by one projector, in which case you could create three surfaces defined in your workspace ("Stage Left Wall", "Stage Right Wall", "Door"). Or you may have four projectors edge blended on a scrim, which you wish to use as one big projection area. In that case you could create just one surface ("Scrim"). In all cases, once the surfaces are set up, you can stop thinking about projectors and simply assign your Video cues to their intended surface.

Sources Of Video

Video cues in QLab play back pre-recorded video or still image files on your computer. Camera cues play live video from webcams, Blackmagic Design video capture interfaces, and other programs on your Mac via Syphon. Text cues render styled text as still images. Once these cues are running, the imagery that they produce is all treated the same way by QLab. Each cue is assigned to a surface, and then all the cues playing simultaneously to a each surface are composited and sent to the actual physical displays.

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" or "1080p", and these are all terms for video standards that have specific resolutions, aspect ratios, frame rates, and other attributes. There are so many standards that it's nearly impossible to keep track of them.

Fortunately, QLab is completely agnostic when it comes to these matters. Video cues can play any compatible media file onto any surface, without the need to preemptively match resolution or frame rate.

When you create or edit a surface, you can set its width and height to suit your exact needs. If the surface you create is larger than the size of an image you project onto it, then the surrounding area will be filled with black pixels. If the surface is smaller than the image, the image will simply extend off the "canvas" of the surface. In either case, you can scale an individual Video cue 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. Video cues can play files at any frame rate, even multiple Video cues at the same time at different frame rates, and the Mac will automatically, invisibly, seamlessly handle the necessary computation to make everything look correct.

Outputs

QLab can output video to any device that's directly, physically connected to your Mac and that appears in the *Displays* section of *System Preferences* will be available as a screen in QLab. Additionally, Blackmagic Design devices which support outputs can be used.

While it's possible that USB-connected displays will also work with QLab, we do not recommend nor do we support using them. They have too many unpredictable variables to be fully reliable. Likewise, AirPlay is not supported in QLab due to its variable latency.

Take Your Time

As we've said in other parts of this documentation, 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.

Video Cues

Video cues allow you to play videos and still images with precise control over timing, sizing and placement on screen, and levels and routing of embedded audio. Video cues require a target, which must be a video or image file on your computer, and a surface, which is a video output destination made up of one or more screens attached to your Mac (such as a monitor, LED wall, or projector).

For moving images, QLab can play files in any format supported by AVFoundation. We strongly recommend the following formats, listed in order of preference, for videos without transparency:

- ProRes 422 Proxy
- ProRes 422 LT
- PhotoJPG
- Hap*

For videos with transparency, also referred to as alpha channel support, we recommend:

- Hap Alpha*
- ProRes 4444

* Hap and Hap Alpha are terrific codecs created by the fine folks at VIDVOX. At the moment, the programming frameworks available to incorporate Hap support into QLab do not afford the very best performance that Hap offers. We hope to revise QLab's Hap support in the near future.

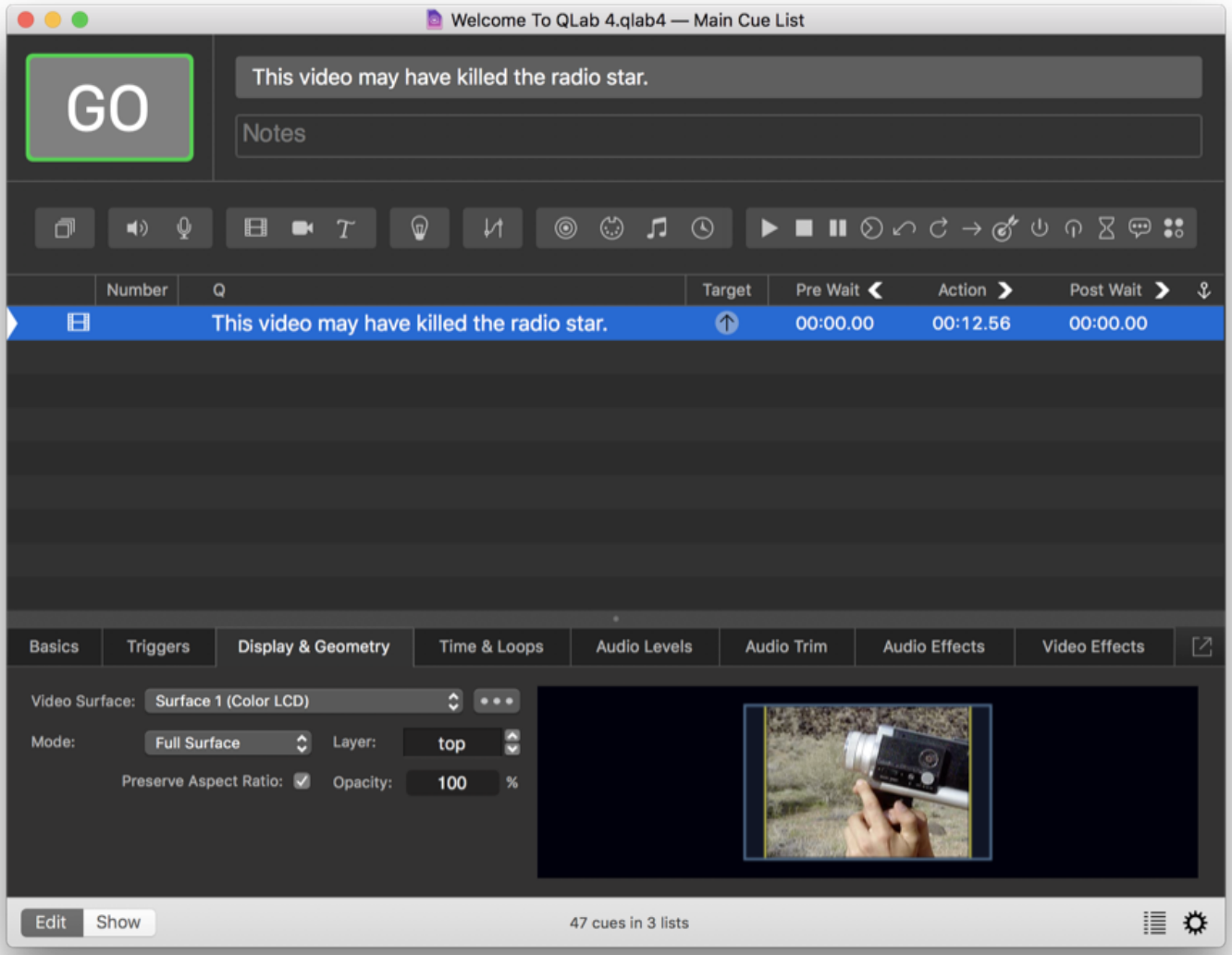
Video cues can also play still images in all common formats. We recommend PNG and JPG. We do not recommend using PSD or PDF formats.

When a Video cue is selected, eight tabs will appear in the Inspector:

- Basics
- Triggers
- Displays & Geometry
- Time & Loops
- Audio Levels
- Audio Trim
- Audio Effects
- Video Effects

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Display & Geometry



Video Surface. This drop-down menu will show a list of the surfaces defined in the workspace. Every Video cue must be assigned to exactly one surface. If you do not have a Pro Video or Pro Bundle license installed, Video cues will only play to the *default* video surface, which is the surface chosen in the Video cue's cue template.

💬 The ellipsis button is a shortcut to open the Surface Editor for the selected surface. You can learn more about surfaces from the page on the Surface Editor.

Mode. Select either *Full Surface* or *Custom Geometry* from the drop-down menu. Note that custom geometry is only available with a Pro Video or Pro Bundle license installed.

Layer. QLab handles layers differently from most media servers. Rather than thinking of layers as a container into which individual cues get placed, QLab treats layers as the order in which video cues stack up on top of each other. Therefore, multiple cues can be assigned to the same layer at the same time.

QLab has 1001 layers: `top` is closest to the viewer, then layer `999`, then 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 triggered. So, for example, if you have three cues assigned to layer `10`, whichever cue is triggered last will appear on top of the other two, but any cue assigned to layer `11` will appear on top of all three cues on layer `10`, regardless of the order in which those cues are triggered.

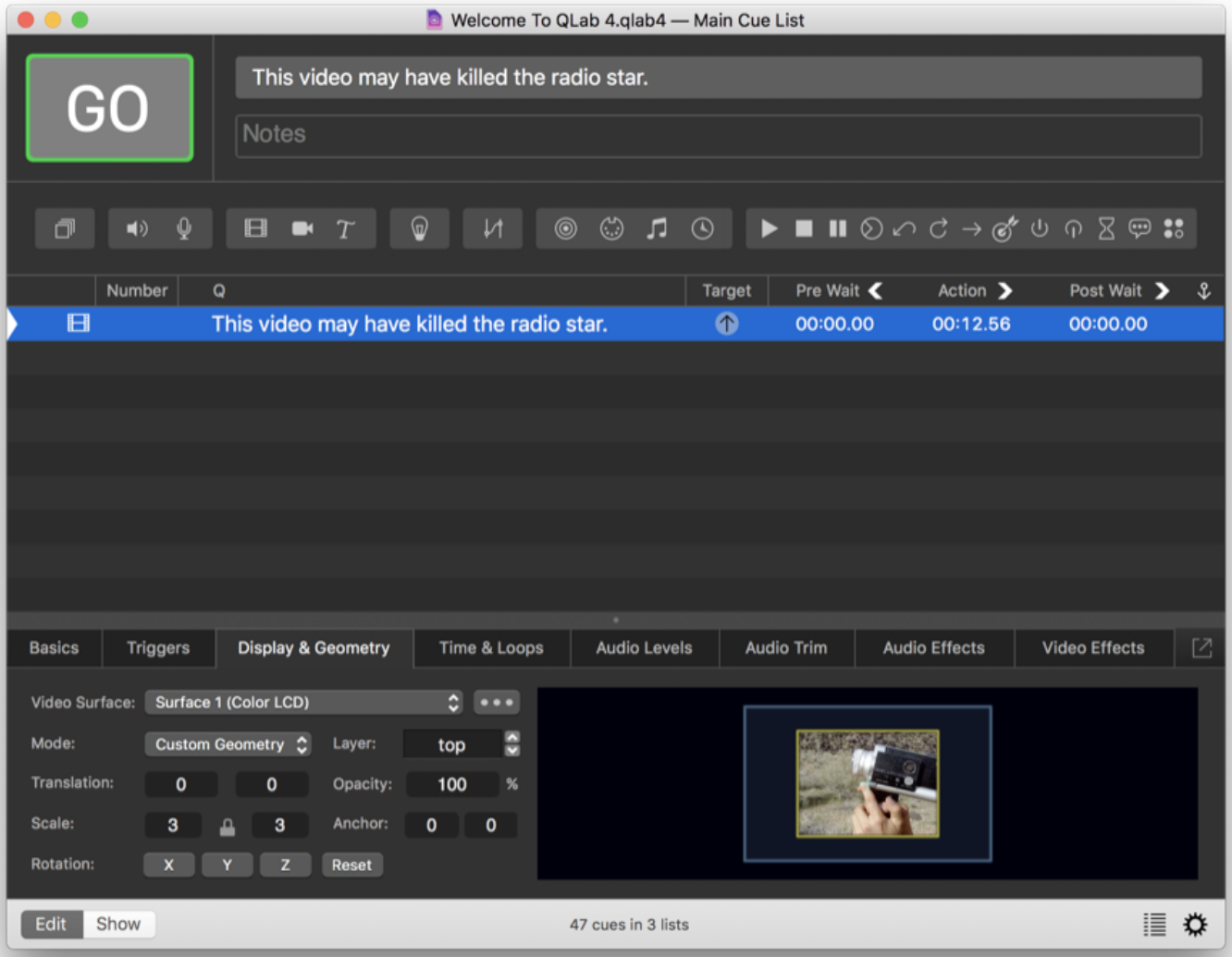
Opacity. Enter values from 0% to 100% to adjust the opacity of the cue.

Full Surface

Preserve Aspect Ratio. This checkbox is only available when *Mode* is set to *Full Surface*. When the box is checked, the video will be played as large as possible without altering its shape, and QLab will pad any remaining space on the surface with black. If the box is unchecked, the video will be stretched horizontally or vertically to fill every pixel of the surface.

Custom Geometry

When *Mode* is set to *Custom Geometry*, more controls become available:



Translation. You can type in values to move the video along the X and Y axes, or click and drag the video within the preview thumbnail to the right of the controls to move it. Translation is measured in pixels relative to the center of the surface, so `0,0` is centered, negative values are down and left, and positive values are up and right.

Scale. You can type in values to enlarge or shrink the video, or scroll up and down on the preview thumbnail to the right of th controls. 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 open it and adjust the height and width independently.

Anchor. The anchor point 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 image. You can type in values to move the anchor point, or click and drag the light blue cross in the preview thumbnail to the right of the controls.

Rotation. You can click and drag on the *X*, *Y*, or *Z* buttons to manipulate rotation around the desired axis, or click and type values into the text field that appears. You can start over by clicking the *Reset* button, which will only reset rotation and not any other attribute of the cue.

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 Video 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.

Time & Loops

Time & Loops for Video cues operate the same as Time & Loops for Audio cues.

Audio Levels, Audio Trim, and Audio Effects

These three tabs allow you to control the audio associated with your Video cue. They operate the same as their corresponding tabs for Audio cues.

QLab can access all audio channels within the first audio track of your video. This can be confusing, since the words "channel" and "track" are often used interchangeably.

In the world of video files, one video can contain multiple *tracks*, and each track can contain up to sixteen *channels*. When a video is prepared for DVD production, the first track might represent the 5.1 surround mix, and thus contain six channels. The second track could be an alternate stereo mixdown, and thus contain two channels.

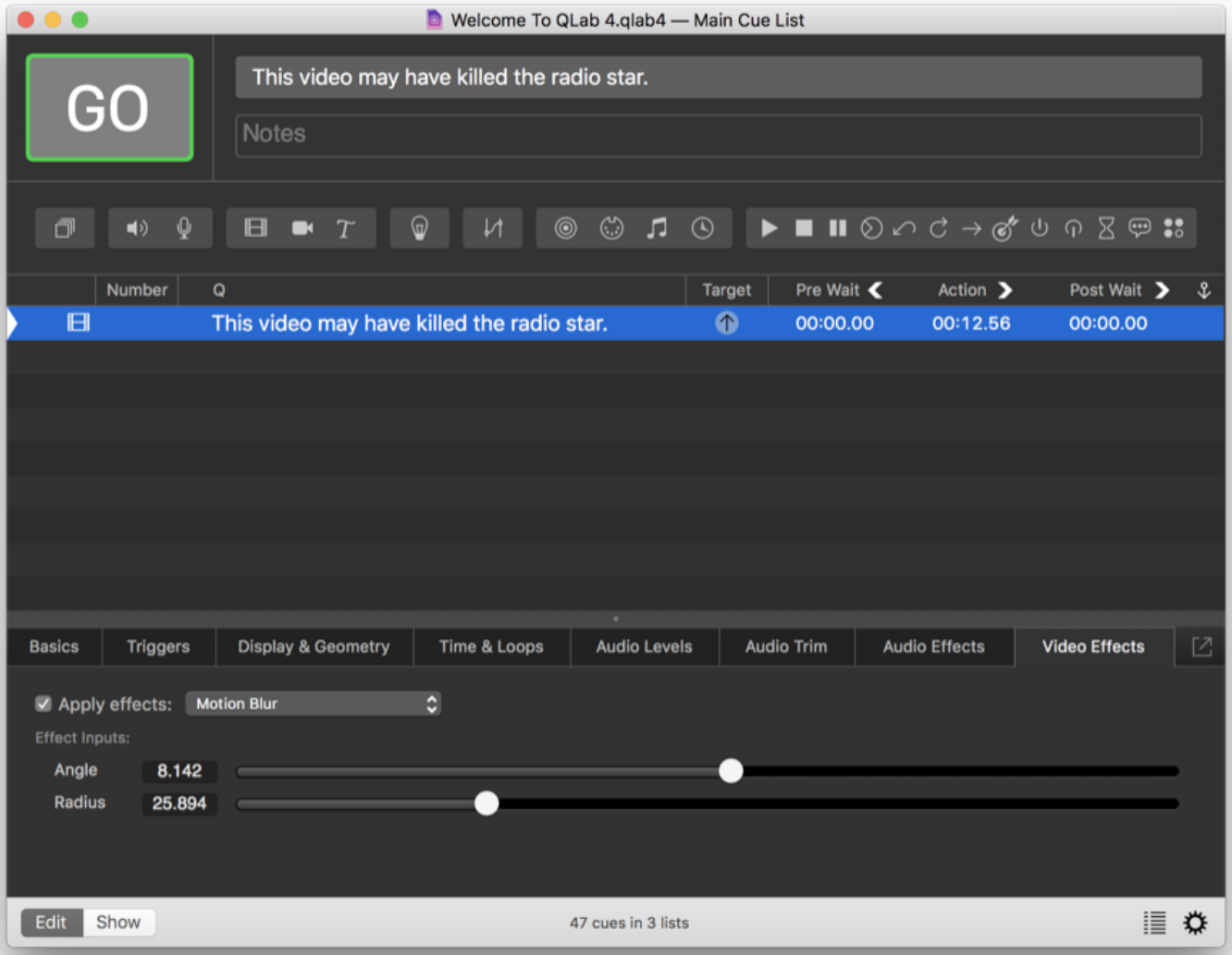
Since QLab can only access the first track of a video's audio content, it's important to keep these differences in mind when preparing your video files.

Video Effects

QLab gives you the ability to apply a wide range of live video effects to your Video cue. Please note that video effects can be extremely processor-intensive, and complicated effects have been known to bog down even very powerful Macs.

Only one video effect may be applied to a cue at a time.

Controls for each parameter of the effect will appear in the inspector; adjust each parameter individually using its appropriate control.



QLab comes with a variety of video effects pre-installed. You can also create your own effects using Quartz Composer. For a composition to be compatible with QLab, it must be an effect (which is one of several types of compositions that Quartz Composer can create) and it must publish an input named `_protocolInput_Image` which is where QLab will send video to the composition, and `_protocolOutput_Image` which is where QLab will receive the effected video from the composition.

To use your own effect, choose *Custom Composition* from the effects drop-down, and then either drag your effect into the area to the right of the drop-down, or double click on that area and choose your effect in the dialog box that opens.

Broken Cues

Video cues can become broken for the following reasons:

Invalid video file.

Either the file is missing or damaged, or it's not one of the supported video file types.

No valid video surface assigned.

Assign a surface in the *Display & Geometry* tab of the inspector. You may also need to visit the Video section of Settings and create or adjust a surface to use with this cue.

There is a problem with this cue's video surface.

Visit the Video section of Settings and create or adjust a surface to use with this cue.

A license is required to send video to anything other than the default surface.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to the default surface for this workspace. The default surface is the surface chosen in the Video cue's cue template.

A license is required to use custom geometry.

You'll need to either install a Pro Video or Pro Bundle license, or set this cue's mode to *Full Screen* in the *Display & Geometry* tab of the inspector.

A pro license is required to send video to a multi-screen surface.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use multiple screens.

A pro license is required to send video to a partial screen.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use partial screens.

A pro license is required to send video to a Blackmagic output device.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use a Blackmagic output device.

A pro license is required to send video to Syphon output.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use Syphon output.

A pro license is required to use video effects.

You'll need to either install a Pro Video or Pro Bundle license, or remove the video effect from this cue.

Timecode triggers require a Pro license.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the timecode trigger from this cue.

Camera Cues

Camera cues, which require a Pro Video or Pro Bundle license, bring live video into QLab. Camera cues will recognize input from:

- Any IIDC-compliant camera or video input device (IIDC devices are often referred to as DV devices.)
- Any UVC-compliant camera or video input device.
- Any Blackmagic DeckLink device (including DeckLink, UltraStudio, and Intensity devices.)
- Syphon inputs (learn more at http://syphon.v002.info.)

It can often be frustratingly difficult to determine if a given camera or device is IIDC-compliant or UVC-compliant. One valuable clue is FaceTime compatibility: if the camera or device works with FaceTime, it will almost definitely work with QLab.

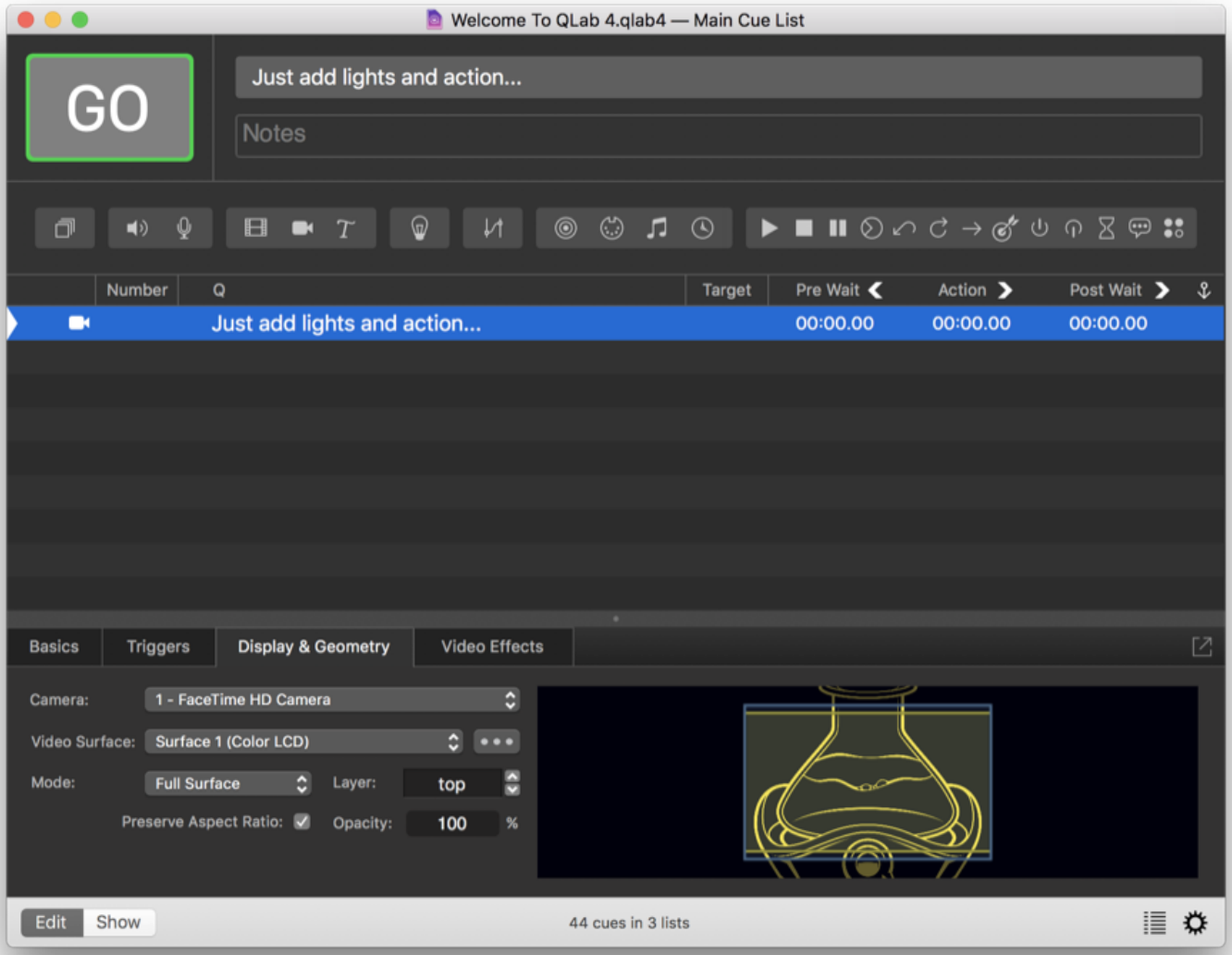
When a Camera cue is selected, four tabs will appear in the Inspector:

- Basics
- Triggers
- Display & Geometry
- Video Effects

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Display & Geometry

The *Camera* drop-down menu lets you choose from among eight camera patches, which are configured in the Video section of Workspace Settings.



Once the camera input is chosen, Camera cues operate exactly the same way as Video cues. Note that Camera cues do not support audio.

Video Effects

Video effects for Camera cues operate the same as Video Effects for Video Cues.

Broken Cues

Camera cues can become broken for the following reasons:

No valid video surface assigned.

Assign a surface in the *Display & Geometry* tab of the inspector. You may also need to visit the Video section of Workspace Settings and create or adjust a surface to use with this cue.

There is a problem with this cue's video surface.

Visit the Video section of Settings and create or adjust a surface to use with this cue.

A license is required to send video to anything other than the default surface.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to the default surface for this workspace.

A license is required to use custom geometry.

You'll need to either install a Pro Video or Pro Bundle license, or set this cue's mode to *Full Screen* in the *Display & Geometry* tab of the inspector.

A pro license is required to send video to a multi-screen surface.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use multiple screens.

A pro license is required to send video to a partial screen.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use partial screens.

A pro license is required to send video to a Blackmagic output device.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use a Blackmagic output device.

A pro license is required to send video to Syphon output.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use Syphon output.

A pro license is required to use video effects.

You'll need to either install a Pro Video or Pro Bundle license, or remove the video effect from this cue.

No valid camera source is assigned.

Assign a valid camera source in the *Display & Geometry* tab of the inspector. You may also need to visit the Video section of Workspace Settings and connect a video input device to the desired patch.

A pro license is required to reactivate this saved cue.

You'll need to install a Pro Video or Pro Bundle license to use this cue.

Timecode triggers require a Pro license.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the timecode trigger from this cue.

Text Cues

Text cues, which require a Pro Video or Pro Bundle license, allow you to display styled text in QLab, using any of the fonts installed on your Mac.

When a Text cue is selected, five tabs will appear in the Inspector:

- Basics
- Triggers
- Display & Geometry
- Text
- Video Effects

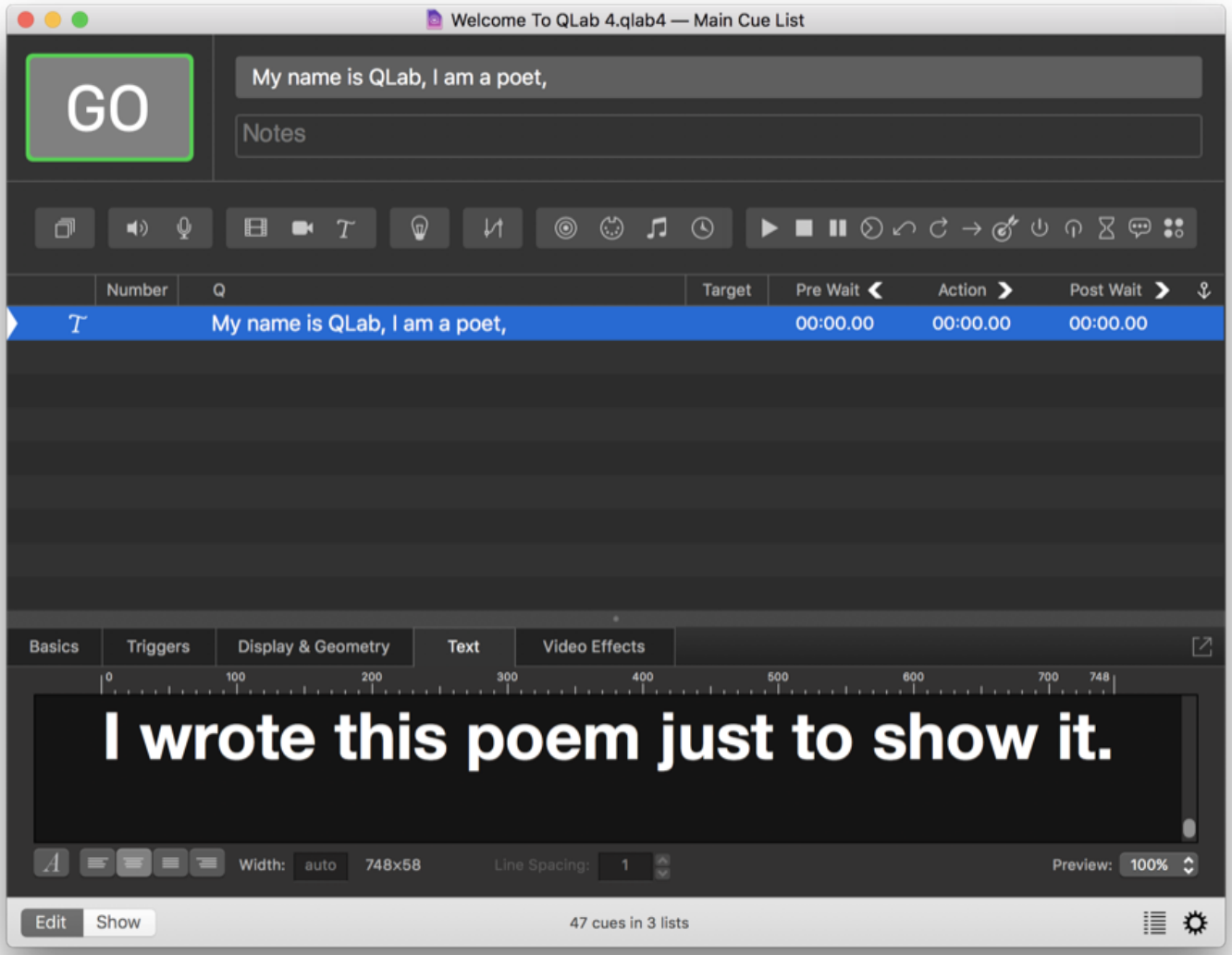
The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Display & Geometry

Display & Geometry for Text cues operates the same as Display & Geometry for Video Cues.

Text

The Text tab lets you input and style the text which will be displayed by the cue.



The *A* button displays the Fonts panel which lets you adjust the font, size, color, and style of the selected text.

Next to that are four buttons which let you choose amongst left-justified, centered, full-justified, and right-justified text.

When a Text cue is triggered, 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 manually increase the width of the (invisible) background, for example to allow necessary space for video effects to properly display. You can manually increase the height of the image by adding carriage returns above and below your text.

The *Line Spacing* field allows you to set the line spacing of the cue text. Line spacing is given in terms of line height, which is dependent on font and point size. Line spacing of `1` is natural spacing, `2` is double spacing, etc.

The *Preview* drop-down 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.

Above the text field is a ruler which shows the actual width of the cue in pixels.

Video Effects

Video effects for Text cues operate the same as Video Effects for Video Cues.

Broken Cues

Text cues can become broken for the following reasons:

No valid video surface assigned.

Assign a surface in the *Display & Geometry* tab of the inspector. You may also need to visit the Video section of Workspace Settings and create or adjust a surface to use with this cue.

There is a problem with this cue's video surface.

Visit the Video section of Workspace Settings and create or adjust a surface to use with this cue.

A license is required to send video to anything other than the default surface.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to the default surface for this workspace.

A license is required to use custom geometry.

You'll need to either install a Pro Video or Pro Bundle license, or set this cue's mode to *Full Screen* in the *Display & Geometry* tab of the inspector.

A pro license is required to send video to a multi-screen surface.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use multiple screens.

A pro license is required to send video to a partial screen.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use partial screens.

A pro license is required to send video to a Blackmagic output device.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use a Blackmagic output device.

A pro license is required to send video to Syphon output.

You'll need to either install a Pro Video or Pro Bundle license, or reassign this cue to a surface which doesn't use Syphon output.

A pro license is required to use video effects.

You'll need to either install a Pro Video or Pro Bundle license, or remove the video effect from this cue.

A video license is required to reactivate this saved cue.

You'll need to install a Pro Video or Pro Bundle license to use this cue.

Timecode triggers require a Pro license.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the timecode trigger from this cue.

Fading Video

A Fade cue can be used to adjust the opacity, translation, scale, rotation, video effect parameters, volume levels, and audio effect parameters of a targeted Video, Camera, or Text cue. Fade cues can also target Audio cues and Mic 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” 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 target and a duration, and must adjust at least one level or parameter.

To learn how to set a target for a Fade cue, please refer to the section on targeting other cues in the **Getting Started** section of this documentation.

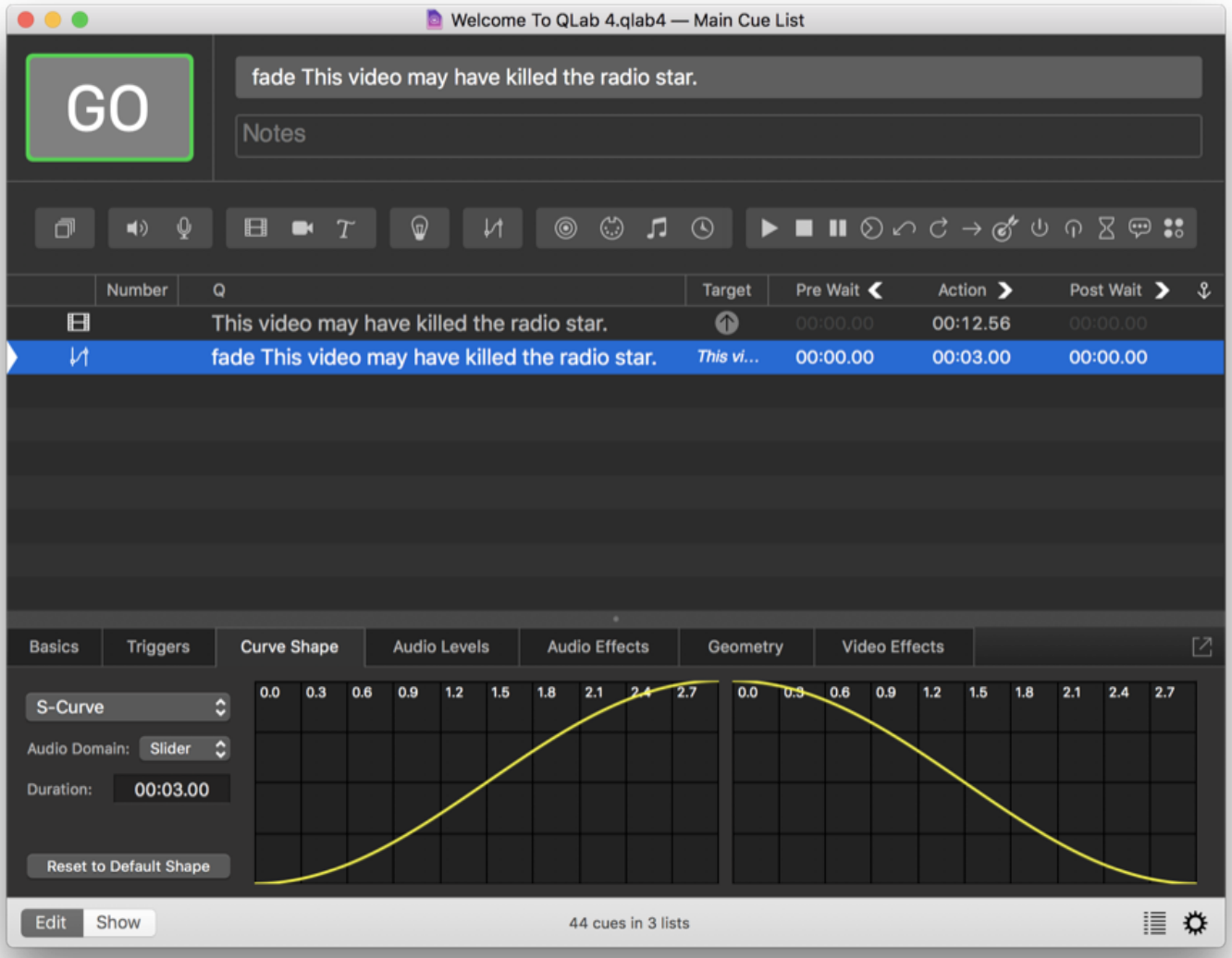
When a Fade cue which targets a Video, Camera, or Text cue is selected, seven tabs appear in the Inspector:

- Basics
- Triggers
- Curve Shape
- Levels
- Audio Effects
- Geometry
- Video Effects

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Curve Shape

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 are increasing, and the curve on the right is for levels which are decreasing. The curve shape that appears by default is set according to the Fade cue’s template, but you can choose another fade shape from the drop-down 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 b dragged to change the shape of the curve. 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.
- **Parametric Curve.** If you choose this option, a text field labeled *Intensity* appears below the drop-down menu. This allows you to use a mathematically precise parametric fade shape.
- **Linear Curve.** This option provides a straight, linear fade curve.

The *Audio Domain* drop-down menu lets you choose the scale that QLab uses to fade audio levels. This drop-down is only relevant to the Audio Levels tab.

- **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.**

Check or uncheck the *Stop target when done* box under the Duration text field depending on whether you would like the target cue to continue playing after the Fade cue is complete, or stop once the Fade is complete.

Audio Levels

The Audio 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.

For Camera cues and Text cues, this tab has no effect.

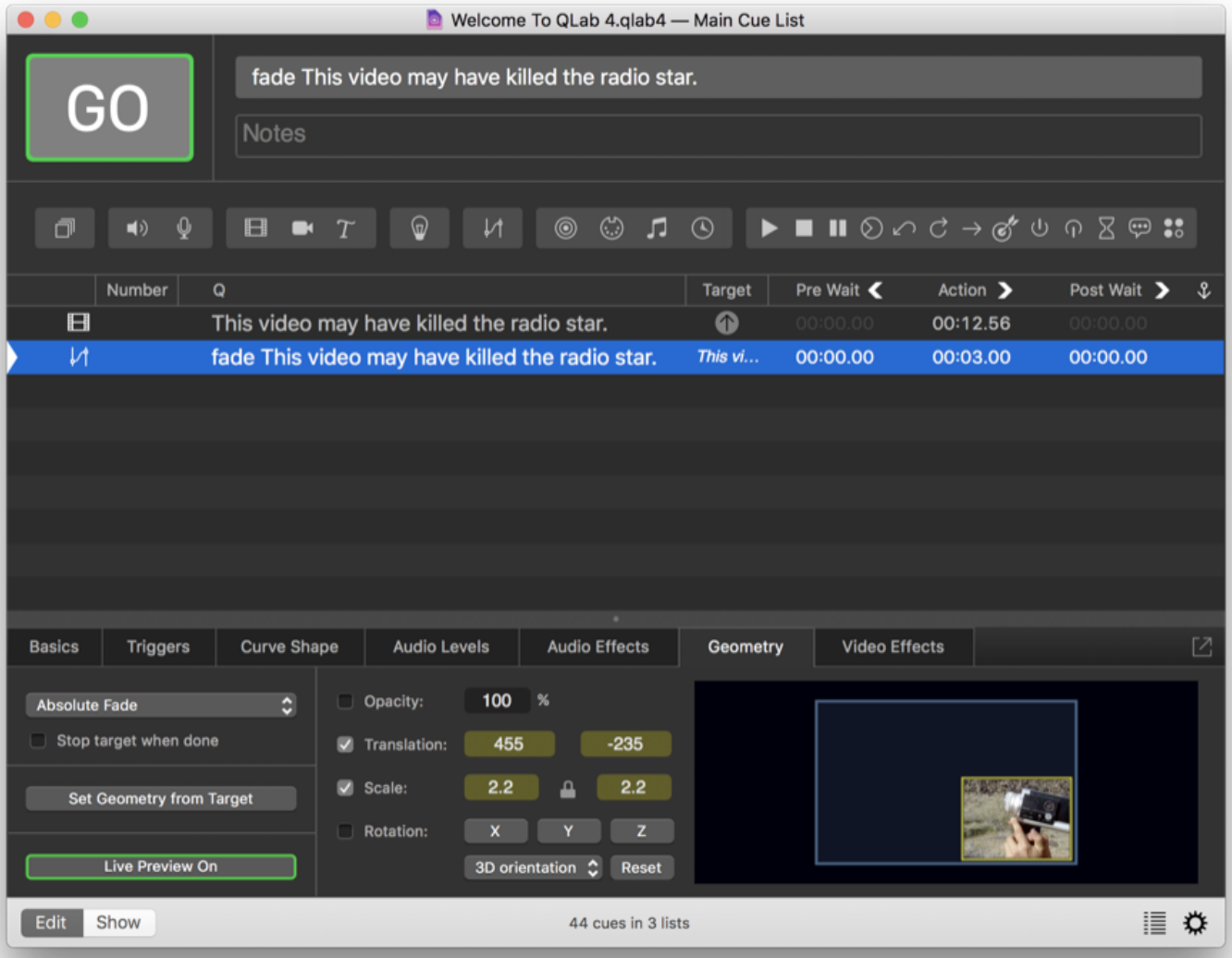
To learn more about the Audio Levels tab in Fade cues, see the page on Fading Audio in this documentation.

Audio Effects

To learn about the Audio Effects tab in Fade cues, see the page on Fading Audio in this documentation.

Geometry

The Geometry tab allows you to specify which parameters of the Video cue you wish to fade, and what their final value will be.



By default, the checkbox next to each parameter will be unchecked, meaning that the Fade cue will not adjust that parameter. To fade a parameter with the Fade cue, just check the box next to that parameter.

Set Geometry from Target. Clicking this button, or using the keyboard shortcut ⇧⌘**T**, 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.

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 to the desired spot in the preview box.

Scale. You can scale the target cue in both the X and Y axes together, or individually by clicking the padlock icon.

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 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 drop-down menu to choose an axis, and then enter the number of degrees you wish to rotate.

Reset. Clicking the *Reset* button will zero out the rotation values in the Fade cue on all three axes, which means the Fade cue will rotate its target Video cue to the default "front facing" rotation.

Video Effects

When you apply an effect to an Video cue, any Fade cues that target that Video cue will recognize the effect you've applied. QLab will list all of the effect's parameters automatically under the Video Effects tab in the Fade cue.

By default, the checkbox next to each effect parameter will be unchecked, meaning that the Fade cue will not adjust that parameter. To fade a parameter with the Fade cue, just check the box next to that parameter.

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 that Video cue will end up at 2 after running the Fade no matter what the scale was when the Video cue started off. If instead you set the Fade to be *relative,* then a Fade with a scale of 2 will double the current scale of the target Video cue.

There are a few peculiar things that you need to keep in mind when using relative fades with Video cues.

1. **Relative rotation is weird.** There's really no better way to say it; using multiple relative fades with rotations adds up to some strange and seemingly unpredictable results. Quaternion math and relative fade math combine in this manner, and the best thing to do is just take your time and test things thoroughly.
2. **Relative and absolute fades don't play well together.** Try to avoid using both relative and absolute fades on the same target Video cue within a single cue sequence. Nothing bad happens, but because of the different way that relative and absolute fades behave, using both at once can add up to unpredictable results, or can cause a cue to "stick" and not respond to one type of fade or the other.
3. **The magic number for a relative fade-in of opacity is 10,000.** To fade in a Video cue (or a Group full of Video cues) using a relative opacity fade, start the Video cue or cues at 1% opacity, and set the Fade cue to 10,000% opacity. 1% * 10000% = 100%

Broken Cues

Fade cues can become broken for the following reasons:

No target cue.

Assign a target to the cue.

No fade parameters have been enabled; pick at least one thing to fade.

You can enable or disable an audio control by clicking in it; active controls will be highlighted in yellow. You can enable or disable a video control by checking or unchecking the box next to it.

One or more audio effects in this cue are missing or broken.

This typically happens when a workspace that uses audio effects is moved from one computer to another, and the new computer does not have the necessary AudioUnit installed. Either choose a different audio effect, or install the necessary AudioUnit.

A pro video license is required to fade video geometry.

You'll need to either install a Pro Video or Pro Bundle license, or adjust the Fade cue to not fade video geometry parameters.

A pro license is required to fade the playback rate.

You'll need to either install a Pro Audio, Pro Video, or Pro Bundle license, or adjust the Fade cue to not fade rate.

A pro audio license is required to fade audio effects.

You'll need to either install a Pro Audio, Pro Video, or Pro Bundle license, or adjust the Fade cue to not fade audio effects.

Timecode triggers require a Pro license.

You'll need to either install a Pro Audio or Pro Bundle license, or remove the timecode trigger from this cue.

Video Surface Editor

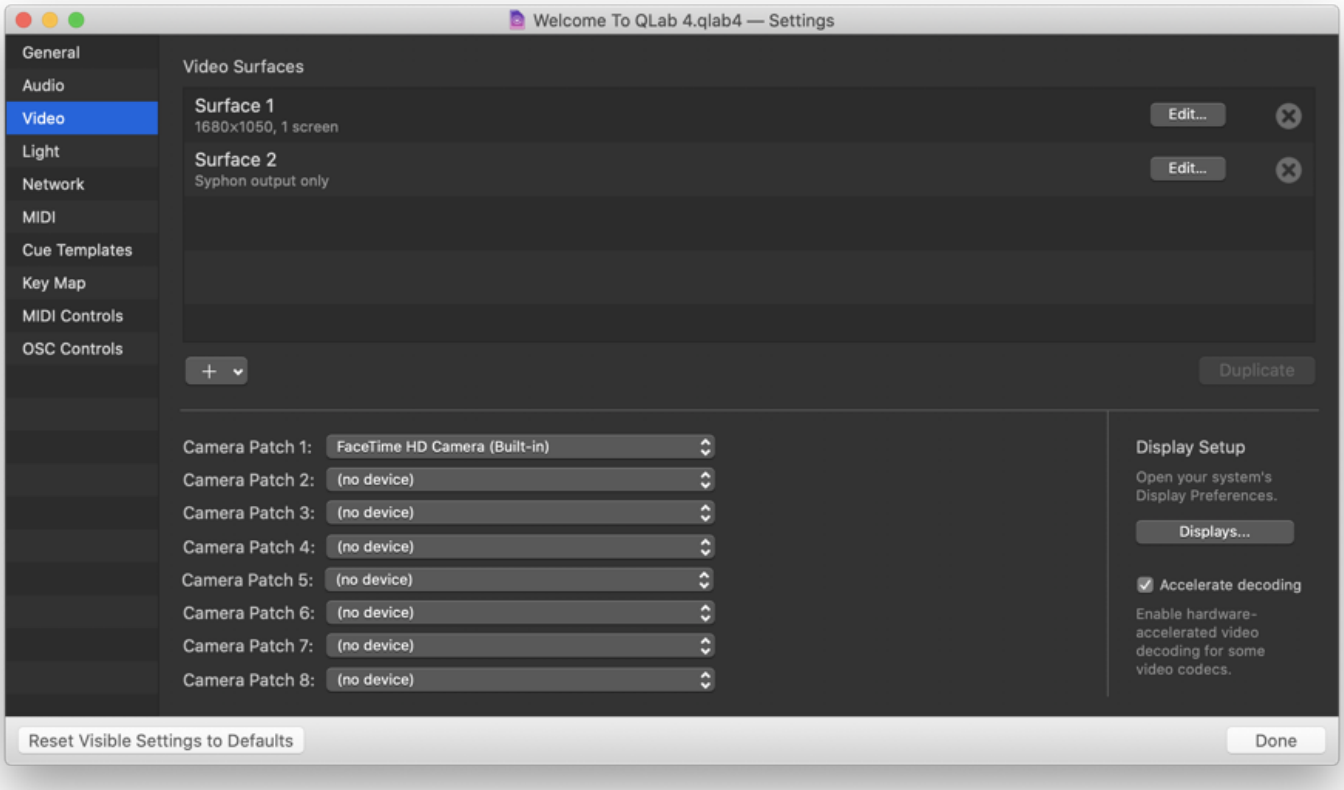
Surfaces are designed to accommodate an extremely wide range of workflows and setups, from single screens, to video walls, to multi-projector blends on curved surfaces. As such, working with surfaces can be fairly straightforward or it can be very involved. The Video Surface Editor is where you set up your workspace to output video to the world.

To get to the Video Surface Editor, go to Workspace Settings and choose *Video* from the list on the left, then click the *Edit* button next to one of the video surfaces.

Getting started with surfaces

When you create a new workspace, QLab will automatically add a surface for each attached display, with that display assigned to the surface. That way, if you don't need anything fancy you can start programming cues as quickly as possible. You can edit these surfaces to suit your needs, or you can delete them and create your own surfaces.

You can see the list of surfaces in your workspace by visiting Workspace Settings and choosing Video from the list on the left.



This screen shot was taken on a 2017 15" Retina MacBook Pro with a 25060x1440 monitor connected to the Thunderbolt port. When this workspace was created, QLab automatically created Surface 1 with the MacBook Pro's internal display assigned to it, and Surface 2 with the external monitor assigned to it.

To create a new surface, click the + popup button below the surface list in the Video page of workspace settings. There are three workflows for creating a new surface:

1. Select *New Empty Surface* to create a totally blank surface, to which you can add screens manually.
2. Select *New With Display* and choose a display to create a surface with that display assigned to the surface. The surface will be created at the same size as the display.
3. Select *New Multi-Screen Surface* to have QLab walk you through creating a surface with a number of regularly arranged screens, such as a multi-projector blended surface or an LCD video wall.

We'll go into more detail about each of these options later on this page.

Once a surface is set up, you can make any number of copies of it by selecting the surface you want to copy and clicking the *Duplicate* button.

To edit a surface, click on the *Edit* button to the right of that surface's name.

To delete a surface, click on the ⊗ button all the way to the right of that surface's name.

Surfaces that are broken will appear with a yellow exclamation point (⚠), and cues routed to broken surfaces will also appear as broken. A surface is defined as broken if it has no valid displays attached. If at least one assigned display is available, the surface will not report as broken, although it will alert you to any disconnected displays in the surface list. Surfaces may also appear as broken if they use features not supported by any of your currently installed licenses.

The Video Surface Editor

Clicking the *Edit* button next to a surface brings up the Surface Editor in a separate window. We're not going to lie to you: this is definitely the most complex-looking part of QLab, and there is a lot going on in this window. Give yourself plenty of time to learn about it, and remember that you can always write to support@figure53.com at any time if you have any questions, large or small, even if you haven't bought a QLab license.

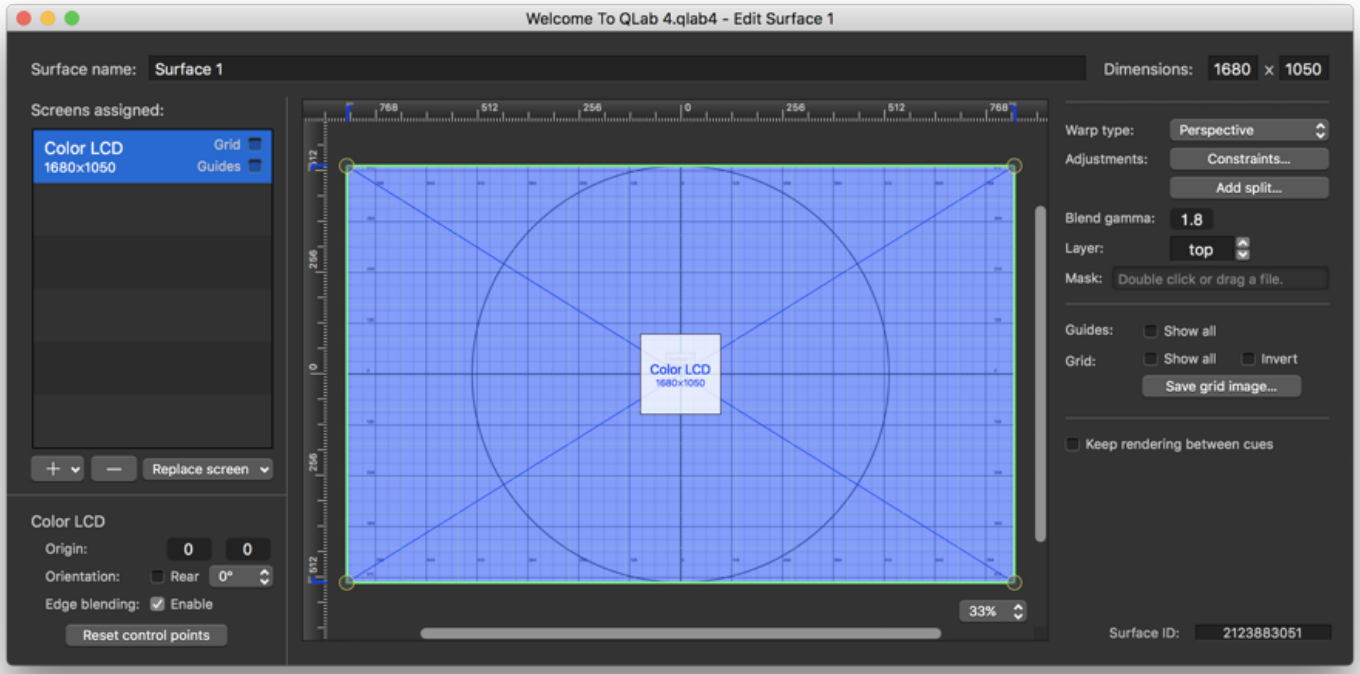
Many features visible in the Edit window require a Pro Video or Pro Bundle license, and will be disabled if an appropriate license is not installed. The features which are allowed without a pro license will function at all times.

The header area shows the surface's name and pixel dimensions.

The leftmost column shows a list of all displays assigned to the surface, as well as a section for information and controls pertaining to the currently selected display.

The center column is occupied by the surface canvas, where you can arrange screens on the surface and adjust control points for each screen.

The right column contains controls for parameters of the surface itself, as well as the currently selected control point or split (when applicable).



Surface name. The name for a newly created surface defaults to something like “Surface 1” or “Surface 2”, and you can rename the surface here to whatever you like. Surface names must be unique within a workspace, so QLab will not allow you to rename a surface to have the same name as another surface.

Dimensions. Every surface in QLab starts its life as a rectangle measured in pixels. The size of the surface does not need to be anything specific, or follow any formula or ratio. If you are creating a surface that will only use one screen, and you want to use the whole of that screen, then the size of the surface should match the size of the screen. That is to say, if you are using a projector with a resolution of `1024 x 768`, then your surface should have dimensions of `1024 x 768`. There is no exact official limit to the size of a surface, but very large surfaces can require more VRAM than your computer has, which can lead to problems.

Displays

This area is where you add, remove, replace, and adjust the display(s) assigned to a surface. At the top is a list of all currently assigned screens, including the raster dimensions of each, and checkboxes to toggle the grid and guides for each. You can find information about grid and guides below.

To assign a screen to the surface, click on the + button and select the screen you wish to add. If you have a Pro Video or Pro Bundle license installed, you can also choose partial screens or Syphon (explained just below).

To remove the selected screen, click the - button.

Syphon output

Although Syphon appears in the list of screens, it does not show up visually in the canvas view of the surface. Syphon output simply mirrors the full surface (within any constraints), with no warping applied. You can add Syphon to any number of surfaces within a QLab workspace; each will create its own Syphon server, so each surfaces can be made available to other Syphon-enabled applications.

Adding Syphon output to a surface has no impact on processor load or performance, because each surface already uses a private Syphon server internally to send frames to the renderers that output video to each display. Adding Syphon to a surface simply makes that server public, so there is no additional work happening within QLab. There is, of course, a CPU and GPU load associated with whatever application is on the receiving end.

You can learn more about Syphon here.

Partial screens

Display splitters such as the Matrox DualHead2Go and TripleHead2Go and the Datapath x4 show up to OS X as a single, very large display. Using partial screens in QLab allows you to split that large display back up and address each output of the splitter as an independent virtual display. Partial screens can be arranged as 2-wide, 3-wide, 4-wide, or 2×2.

Partial screens require a Pro Video or Pro Bundle license.

Partial screens are designed exclusively for working with hardware splitters, and are not useful for making a surface cover only part of a projector. Instead, to restrict a surface to one part of a projector, use constraints, or simply reduce the surface dimensions and adjust the screen's origin.

Replace Screen

The *Replace screen* drop-down allows you to select a screen from the list, and then remap that screen to a different display. The drop-down lists only displays which are currently connected to your Mac. This is helpful if your hardware has changed, for example if you've prepared a show using an external monitor, then moved to the theater where you're using a projector. It can also help you if OS X mistakenly identified a screen as new, when in fact it's the same display you've been using. When you replace a screen, QLab updates the screen's raster dimensions to match the new display, and any adjustments you've made such as control points are preserved as closely as possible.

Screen Controls

The lower section of this column contains controls for the currently selected screen:

Origin controls determine (in X and Y pixels) the position of the screen's lower left corner relative to the lower left corner of the surface.

Orientation controls are composed of the *Rear* checkbox, which flips the screen's rendering horizontally for rear projection, and a *Rotation* pop-up, which rotates the image in 90° increments.

Edge blending can be disabled on a single projector (for example, if the projector has its own built-in blend controls) by unchecking this checkbox. This control is checked by default, but has no effect if the surface doesn't contain at least two partially overlapping screens.

If the selected screen corresponds to a Blackmagic DeckLink-compatible device, a pop-up button will appear for selecting the resolution and frame rate of the device output.

Edge blending

QLab automatically detects when edge blending should happen and feathers the edges of each screen as appropriate, according to the following rules:

- If two screens overlap by more than 90%, they will be treated as double-stacked, and QLab will send the same image to both screens. For example, if you have two projectors that you want to combine to make a brighter image, overlapping them completely will tell QLab to mirror the output to the two projectors.
- If two screens overlap by more than 0% but less than 90%, QLab will automatically feather the overlapping edges, to create a seamless blend between projectors, unless you have disabled blending for the surface or its screens.
- If two screens do not overlap at all, the displays will render as separate parts of the surface with no blending.

The *Reset control points* button returns all the perspective/warping control points of the current screen to their default locations.

The Editor Canvas

The canvas shows a visual overview of the surface layout. With a Pro Video or Pro Bundle license installed, you can drag screens to arrange them on the surface, drag splits to adjust them, and drag control points to adjust perspective correction and warping.

The surface is displayed as a white rectangle with a black focus grid. The name and dimensions of the surface are displayed in the center of the surface. Screens assigned to the surface appear as translucent colored rectangles, also labeled with their name and size. If your surface is `1024 x 768` and you assign a screen that is `1024 x 768`, then the screen will cover the entire surface. If the surface is larger, you'll see that part of the surface is covered by the screen, and part is not.

The canvas is bordered on the top and left by rulers that show surface measurements in pixels. These measurements are centered relative to the middle of the full surface, before any constraints.

The drop-down menu in the lower right corner of the canvas allows you to scale the view in the Surface Editor. It has no effect on QLab's output.

Surface Controls

The controls in this column represent parameters of the surface itself.

Warp type

Warping controls require a Pro Video or Pro Bundle license.

Each screen assigned to a surface comes with a set of control points that you can use to warp the final rendering in various ways. This allows you to project a flat image onto an object that is oddly shaped or at an angle to the projector, and have it appear perfectly aligned to the audience.

Each screen starts off with control points in each corner. You can add splits to a surface in order to add more control points.

Control points are associated with individual screens, so warping adjustments made to one projector will not affect the image displayed by another projector on the same surface.

Note that it is best to arrange all the displays on a surface, and split the surface as desired, before adjusting control points. We haven't discussed splits yet, but that's coming up real soon.

QLab offers three types of warping. Each surface can use any of the three types:

- **Perspective** (the default mode) is useful for simple vertical and horizontal keystone correction, also called corner pinning) when you're projecting onto a flat surface that the projector is not perfectly aligned with. Any adjustments made are perspective-correct, meaning that once the correction is complete, an image in one part of the surface will appear exactly the same size if moved to a different part of the surface.
- **Linear** is more useful for warping onto a complex shape; this is often called mesh warping. Linear warping guarantees a continuous image across a split, with the tradeoff that it is not perspective-correct, and may distort images if used with extreme corrections.
- **Bézier** is a more complex mode, used to project onto curved surfaces. This mode gives you more control points, for full control of a 2D cubic Bézier surface. Like Linear mode, it is continuous across splits and useful for mesh warping.
- **Hint:** Bézier warps take longer to set up because of the additional control points; however, if you need to make a quick perspective tweak to a Bézier warp, you can do so by holding down the **Command** key while dragging one of the corner points. While **Command** is held down, QLab will interpolate the other points to approximate a perspective-correct corner pinning adjustment. This is not entirely precise, but it can be useful for minor tweaks.

Perspective correction is the most common use of QLab's warp controls. In the simplest case, a single projector directed at a flat screen may need corner pinning in order to compensate for skewing or keystoning caused by the angle of incidence of its beam.

To make corrections such as these, first check the *Grid* checkbox for the display being corrected, to display an adjustment grid on the projector. Then, in the canvas of the Surface Editor, drag the yellow control points at the corners until the grid displayed by the projector appears aligned. Fine-tune these adjustments, if desired, by holding down the **Shift** key while dragging control points, or by adjusting the selected control point numerically using the *Selected control point* fields in the right column of the Surface Editor. When finished, uncheck the *Grid* checkbox to dismiss the grid.

In many cases, multiple projectors are edge-blended onto a single screen. Since each beam may hit the screen at a different angle, each projector has its own set of control points. It may be helpful to work with one projector at a time, adjusting control points with only a single grid shown, then to fine-tune the control points with all grids up and ensure that they overlap continuously.

Whenever possible, QLab attempts to adjust control points automatically to match other changes you make to a surface, such as moving screens or changing the layout of splits. However, once control point adjustments have been made, it is generally not possible to interpolate correct new locations for the control points when the surface setup changes. Rather than wildly guessing in these cases (since manual adjustment will be necessary anyway,) QLab leaves the control points in place for you to adjust. This can cause the image to appear distorted until you fix the control points, because they no longer match closely with the area they represent.

You may notice that the first adjustment you make to a screen's control point causes that screen's outline in the canvas to appear thicker. This indicates that an adjustment has been made, and the control points are fixed in place relative to the screen until you adjust them yourself. If you click on the *Reset control points* button, the thinner outline will return, and the control points will begin automatically adjusting to match the changes you make.

Usually, saving control point adjustments for the end of the process is the easiest way to ensure that moving screens and splits doesn't distort the image, and that the control points all start from a reasonable starting place. If that order of operations is not possible, you can still adjust control points manually, or use the *Reset control points* button to start fresh.

Constraints

Constraints are a means of taking an existing surface and bringing its sides in to reduce its effective dimensions, while preserving all the warps and splits previously defined on the surface. The result, for the purposes of cue geometry, is simply a smaller surface.

To constrain a surface, click on the *Constraints...* button, and use the number fields in the popover to bring in each edge.

Constraints are most useful in conjunction with duplicating surfaces. For example, if you have a paneled wall defined as one surface, and want to define the individual panels as separate surfaces, you can first set up the full wall, along with any control point adjustments, then duplicate that surface and pull the duplicate's constraints in to isolate a single panel. This saves time compared to redoing all the corner pin adjustments to match the full wall's geometry perfectly.

Splits

Adding splits divides a surface up into separate regions, called "patches", each of which has its own set of control points. This allows you to map your surface onto complex shapes, like the corner of a set piece or the façade of a building.

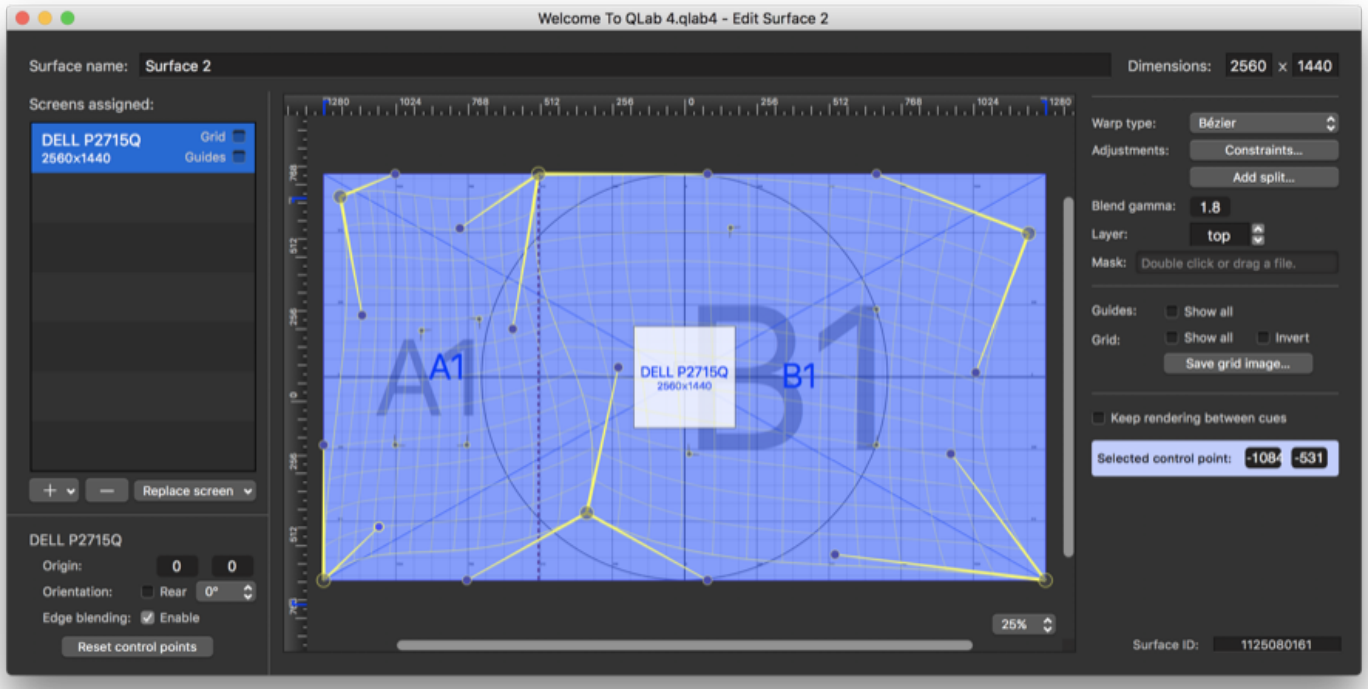
To add a split, click on the *Add split...* button. In the popover that appears, choose whether to split the surface horizontally (resulting in a left half and a right half) or vertically (into a top half and a bottom half), and provide the location for the split. You can set the location by clicking in the number field and dragging up and down. A thin dotted line will appear in the surface canvas as a preview, and will appear thicker once you click on *Add split* to commit it.

To move an existing split, click and drag it in the surface canvas. You can also adjust its positioning numerically by clicking to select it in the canvas, then using the *Selected patch* field.

To delete a split, click to select it in the canvas, then click the *X* button next to the *Selected patch* field.

Once a surface is split into patches, each patch is labeled with an identifier such as **A1**, **A2**, **B1**, etc. These labels show up in two places: The surface itself is displayed with large gray labels, and each display on the surface is shown with smaller, colored labels for patch identifiers. This allows you to determine which part of the surface corresponds to which set of control points, even if the control points are dragged to a different part of the editor than the surface patch they correspond to.

It is important to note that each set of control points on a screen corresponds only to the part of the surface patch they intersect, and not necessarily to the corners of the whole patch. For example, if a `1920x1080` surface is split evenly, at `x = 0`, into patches **A1** and **B1**, and the first projector ends just 10 pixels to the right of that split, then the control points labeled **B1** for that projector will control only that 10-pixel-wide strip, and not the whole 960-pixel width of patch **B1**.



If you move a screen until it no longer intersects with a patch at all, the number of control points associated with that screen will change. This can have unexpected results if you have control point adjustments in place, as it may change which patch a set of control points corresponds to. Again, saving control point adjustments for the end of the process will avoid this confusion.

Blend gamma

This adjusts the curve of the edge blending calculated by the surface. Adjust this if you see a dark or bright band running down the “seam” between projectors.

Surface Layer

The layer control for surfaces behaves just like the layer control for Video cues, but it’s a separate control. So if surface “Hamlet” is set to layer 1, and surface “Ophelia” is set to layer 2, then *all* cues assigned to Ophelia will render on top of all the cues assigned to Hamlet.

Surface Mask

Surface masks are a way to completely or partially block certain pixels of a surface from displaying. They may be a familiar concept from graphics editing software, and they work similarly in QLab. Masks have several uses, including:

- Altering the shape of otherwise rectangular surfaces.
- Feathering the edges of a surface for a vignetting effect.
- Defining a region where another surface shows through from underneath.

To add a mask to a surface, double-click on the *Mask* well and select a file, or drag a file from Finder onto the well.

A mask should be a greyscale image that matches the dimensions of the surface, but QLab will convert it to greyscale and re-size it to fix the surface if necessary.

Black pixels in the mask are regarded as opaque, white pixels are regarded as transparent, and grey pixels are partially transparent; the darker the grey, the less transparent they are.

Any standard image format can be used as a surface mask. 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.

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.

Guides

The *Guides* checkbox toggles a set of lines and targets for projector alignment. With the basic layout of screens on a surface in place, you can use the guides to “rough in” the projectors to cover the desired physical area on stage.

Red guides show you the edges and center of each projector, and a green dotted line shows where the overlap with each adjacent projector should end. Guides show the bounds of the full projector raster, and are not affected by control points.

Grid

Use the *Grid* checkbox to send a black-and-white grid image to each projector while you adjust control points. The grid’s dimensions match those of the surface, and only the portion corresponding to the area covered by each projector is output. In other words, it is essentially equivalent to running a video cue with a still image of a grid in full-screen mode, except that it can be sent to individual screens in isolation. When all corner points are in place and grids for all screens are enabled, you should see a single, perspective-correct, continuous grid for the whole surface.

By default, the grid is rendered black-on-white; use the *Invert* checkbox to switch it to white-on-black.

The *Save grid image...* button creates an image file that can serve as a useful reference when constructing or fine-tuning masks in a graphics program.

Keep rendering between cues

Whenever a Video, Camera, or Text cue is running, QLab renders a black background on the whole surface that the active cue is assigned to. If no cues are playing, QLab does not show a black background, which allows the desktop to show through. If this checkbox is checked, however, QLab will keep rendering the black background until QLab quits or the workspace receives the command to panic. This can ensure a smoother transition between Video cues, particularly on Macs whose GPU performance is possibly not quite as terrific as one might wish.

Selected point or split controls

Below the *Grid* controls are fields for numerically adjusting the selected control point or split. When none is selected, these controls are hidden.

Surface ID

When a surface is created, QLab gives it a unique surface ID number. This number can be useful when using OSC or AppleScript to interact with surfaces. The surface ID is displayed in the lower right corner of the Surface Editor for easy reference and copying.

Chapter 5: Lighting

Introduction 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 4 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 and dimmers 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.

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
- 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 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 are the type of connector used for ethernet. 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 which lights it will be controlling, and what DMX addresses those lights are using. You do this for each QLab workspace in the Light section of Workspace Settings. There, you create **instruments** in the workspace and map them to the Art-Net/DMX addresses of real lights or dimmers in the physical world. You can also define **light groups** which behave as shorthand for collections 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 every 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.

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 copy and edit these definitions to suit your needs, and you can create new light definitions 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 master library, as well as a library 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, 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 `0`. Consider a show with six conventional lights in it:

Cue	1	2	3	4	5	6

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 documentation.

Light Cues

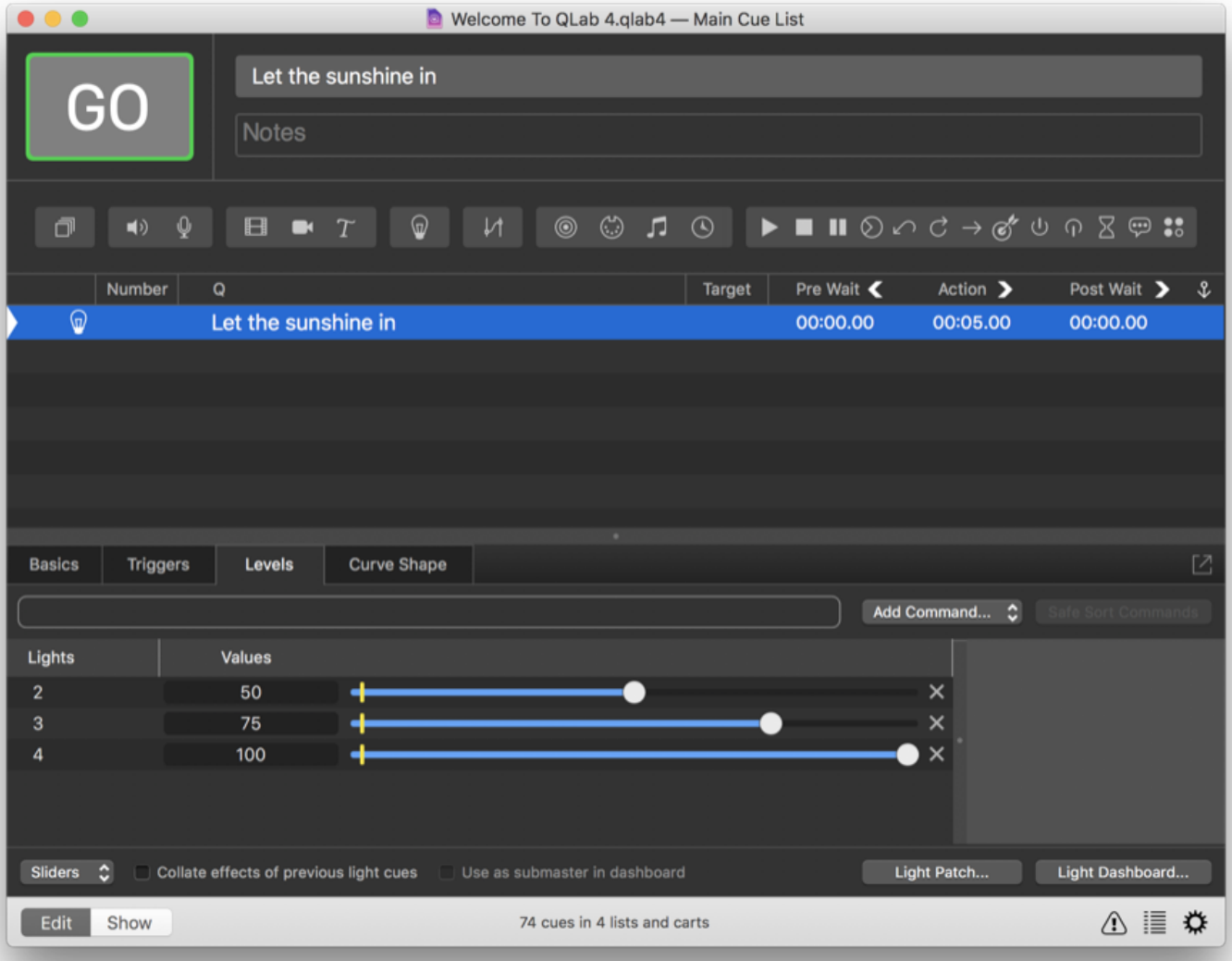
When a Light cue is selected, four tabs will appear in the Inspector:

- Basics
- Triggers
- Levels
- Curve Shape

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Levels

The Levels tab shows the lighting commands contained by this cue.



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.

Add Command...

The Add Command drop-down menu contains an ordered list of all instruments and light groups contained in the workspace's light patch. Selecting the instrument will add a command for that instrument 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.

Safe Sort Commands

Commands in a Light cue are displayed and interpreted in the order in which they were added to the cue. Clicking *Safe Sort Commands* will sort the commands alphabetically so long as the sorting does not alter the results of the commands. If moving a particular command by sorting would change the result, that command is left un-sorted.

The Lighting Commands list

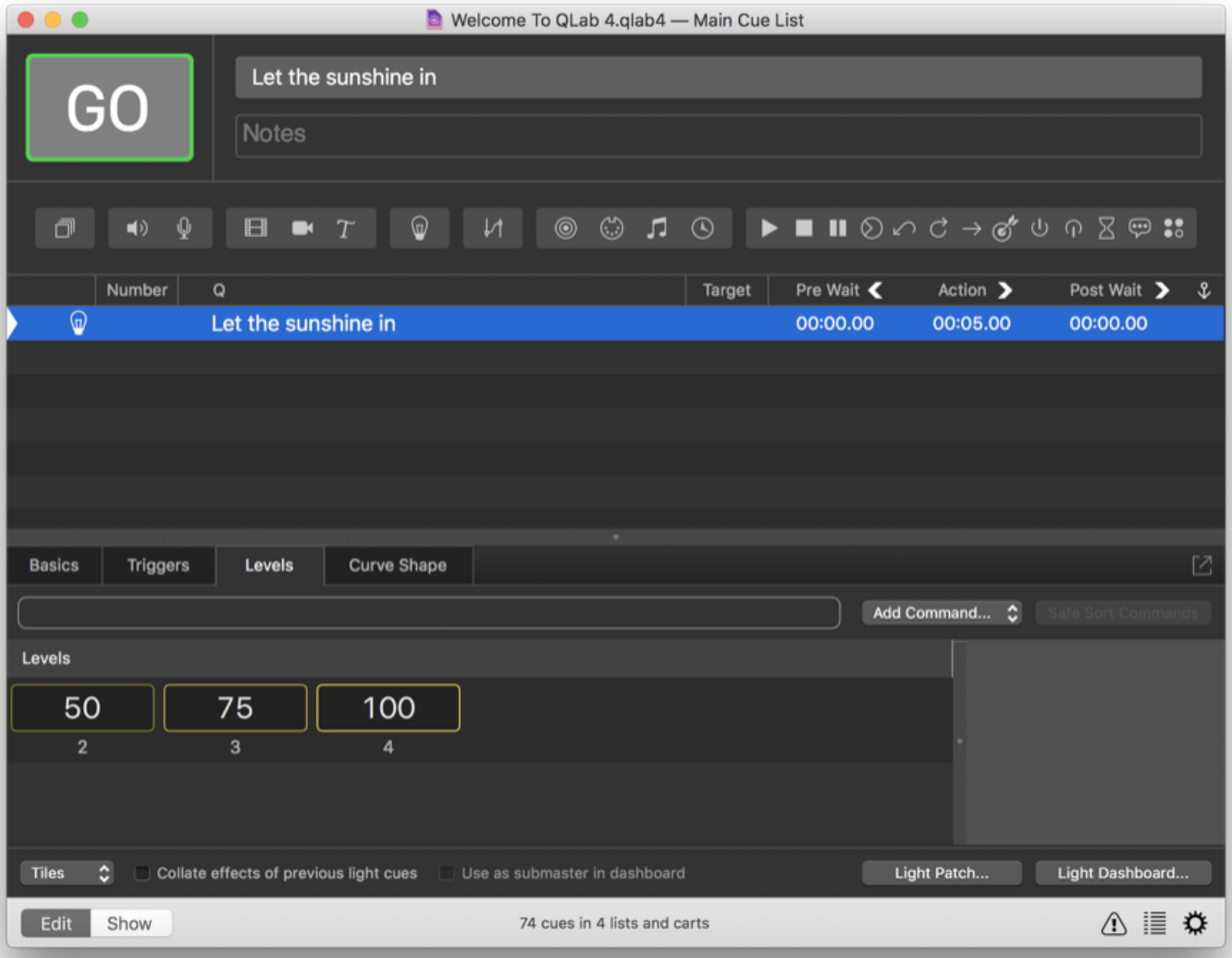
The commands contained by the Light cue are listed here in the order in which they were added to the cue, with the most recent addition at the bottom. You can view the cue's lighting commands as sliders, tiles, or text.

In slider view, a yellow mark indicates the current value of each parameter. For example, in the screen shot above, instruments 2, 3, and 4 are all currently at `0`.

Commands can be dragged up and down in the list to re-order them. You can type in new values in the text fields, or drag the slider handles to change levels. Changes made in a Light cue will not be immediately reflected on stage; you need to run the Light 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.

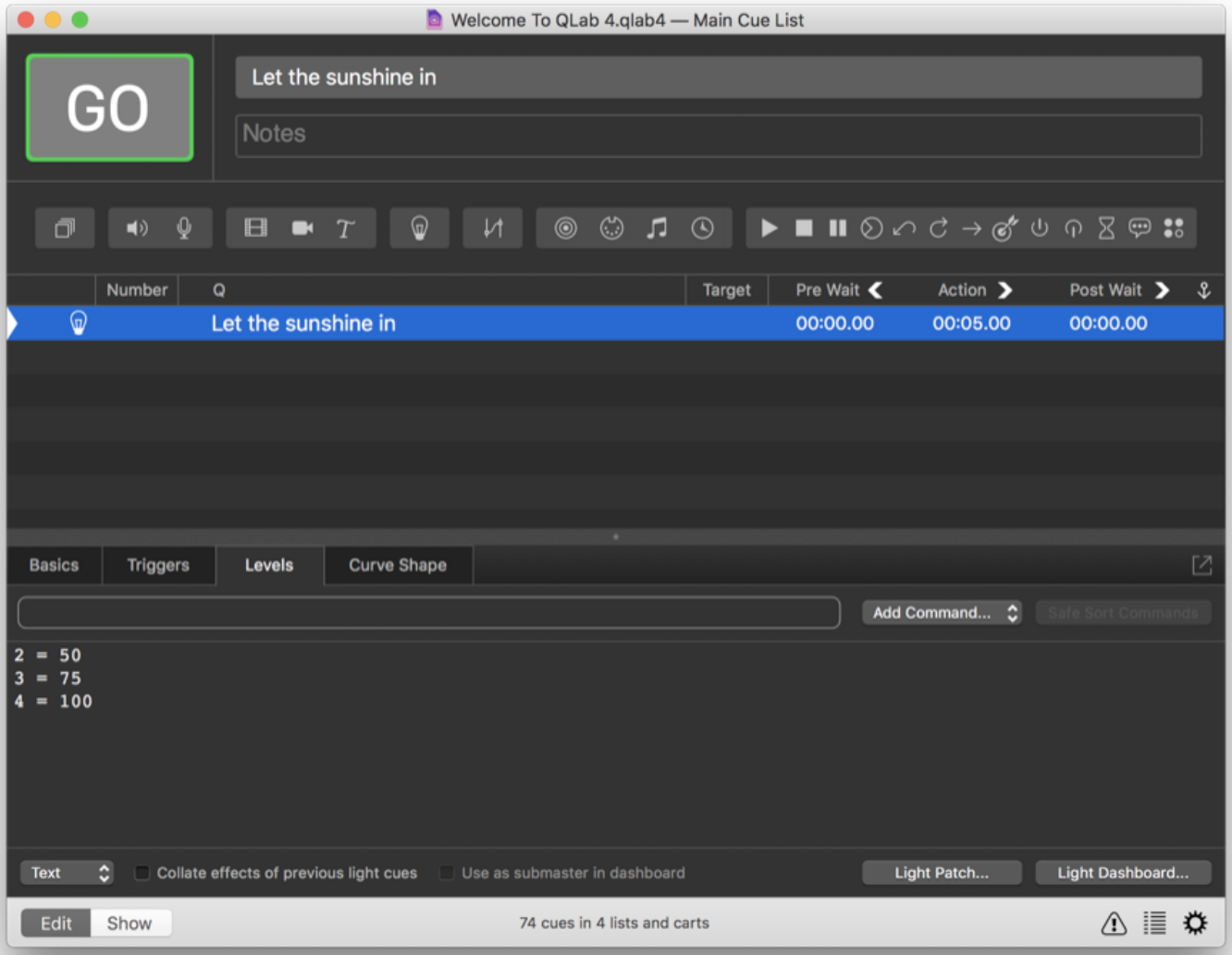
To edit "live", make your changes in the Light Dashboard.

You can click on the ✕ on the right side of each command to remove it from the cue.



In tile view, a yellow outline reflects the level for each command; brighter for higher values.

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 gives you direct access to edit light commands textually. You can delete a command from the cue in text view by selecting and delete the text of that command.

Sliders, Tiles, Text

This drop-down 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.

Collate effects of previous light cues when running this cue

When this box is checked, QLab will behave as though all prior Light cues in the same list have been run when running this Light cue. With this box checked, triggering the Light cue will result in the exact same total look on stage, no matter which cues have or have not been run. With this box checked, QLab's behavior most closely resembles the behavior of a traditional lighting console.

Use as submaster in dashboard

When this box is checked, the cue becomes available to use as a submaster in the Light Dashboard. You can learn more about submasters from the Submasters section of the Light Dashboard page of this manual.

Light Patch...

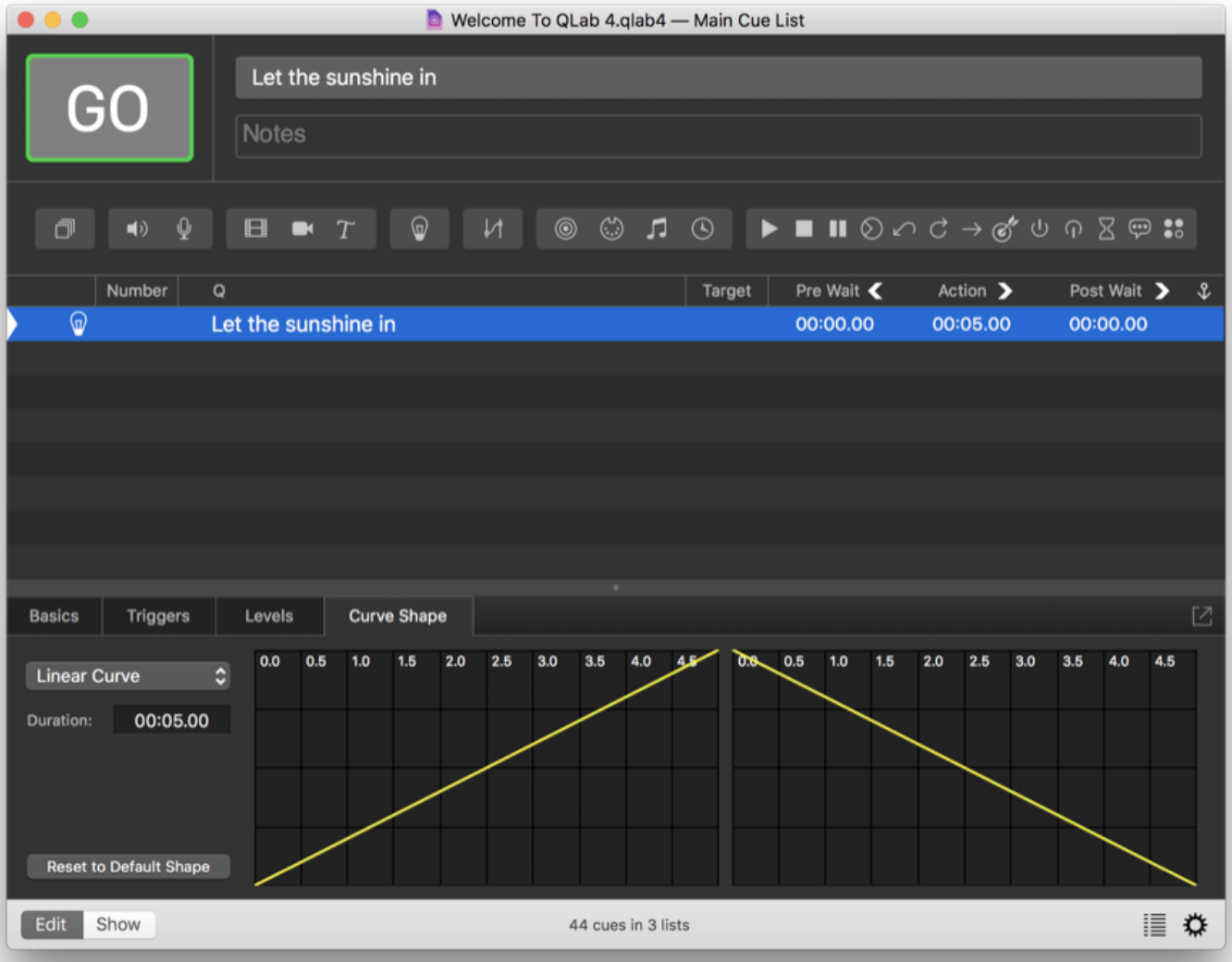
This button opens the Light Patch window.

Dashboard...

This button opens the Light Dashboard window.

Curve Shape

The curve shape determines the manner in which the parameter or parameters are adjusted over the course of the fade. QLab defaults to a linear curve, but any curve can be drawn in the Curve Shape tab by selecting *Custom Curve* from the drop-down menu in the top left corner of the tab.

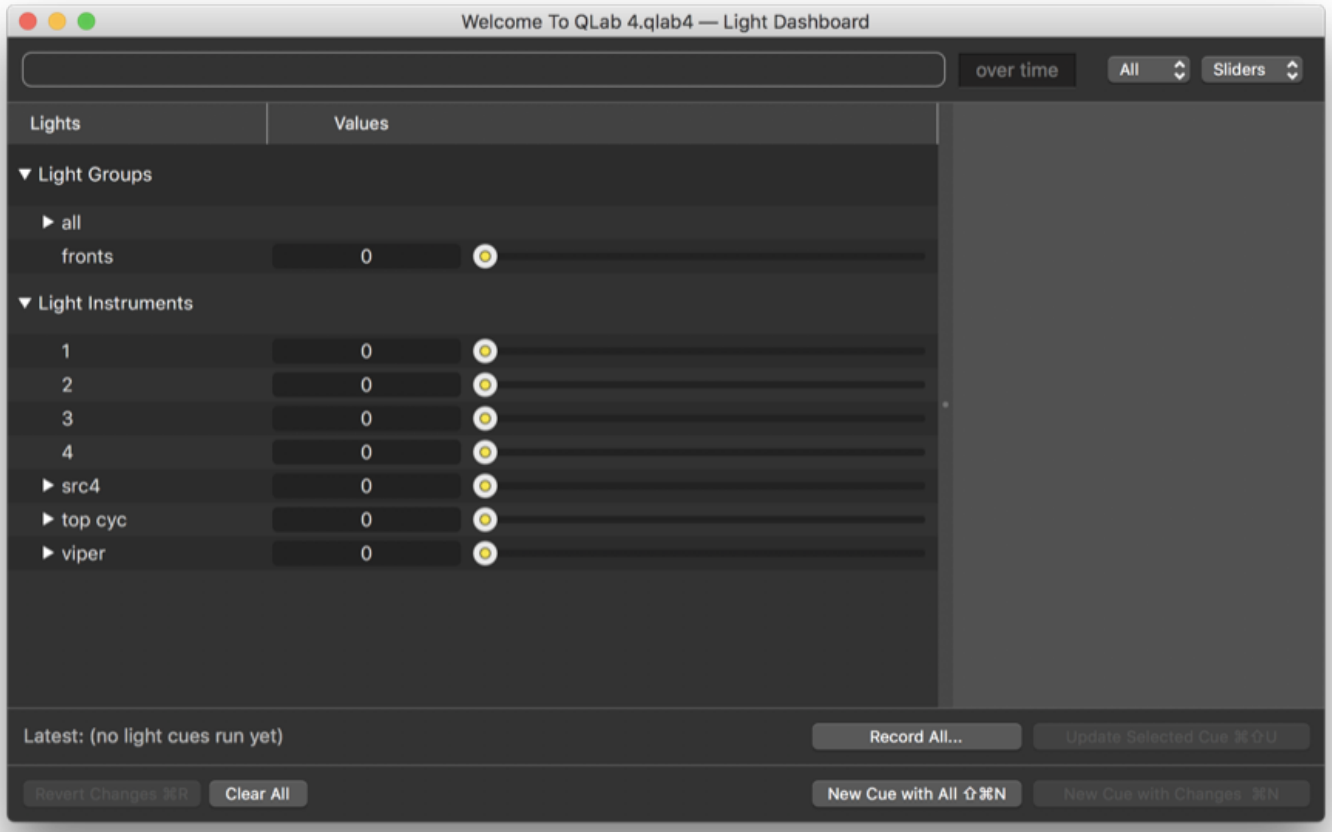


The curve is drawn in yellow, overlaid on a grey grid. The curve on the left is used for levels which are increasing, and the curve on the right is used for levels which are decreasing. If you've chosen *Custom Curve*, you can click anywhere along the yellow line and a yellow dot, called a control point, will appear. Drag it to change the shape of the curve. Click on the curve again to create more control points. The active point will be filled-in yellow circle, and others are yellow outlines. To delete one, 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.

Light Dashboard

The **Light Dashboard** 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 live. Similarly, when you run a Light cue, you'll see the changes made by that cue reflected in the Light Dashboard as it happens.

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;
- Starting with QLab 4.4, by clicking and dragging on parameters' text fields.

Starting with QLab 4.2, 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 level has been modified in the Dashboard, that parameter's control turns yellow.

The Light Command Line

This textual command line allows you to manipulate the Dashboard quickly and powerfully. You can learn more about how to use it in the Lighting Command Language section of this documentation.

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 drop-down menu is set to **All**, the Dashboard will display every instrument and light group defined in the workspace's patch.

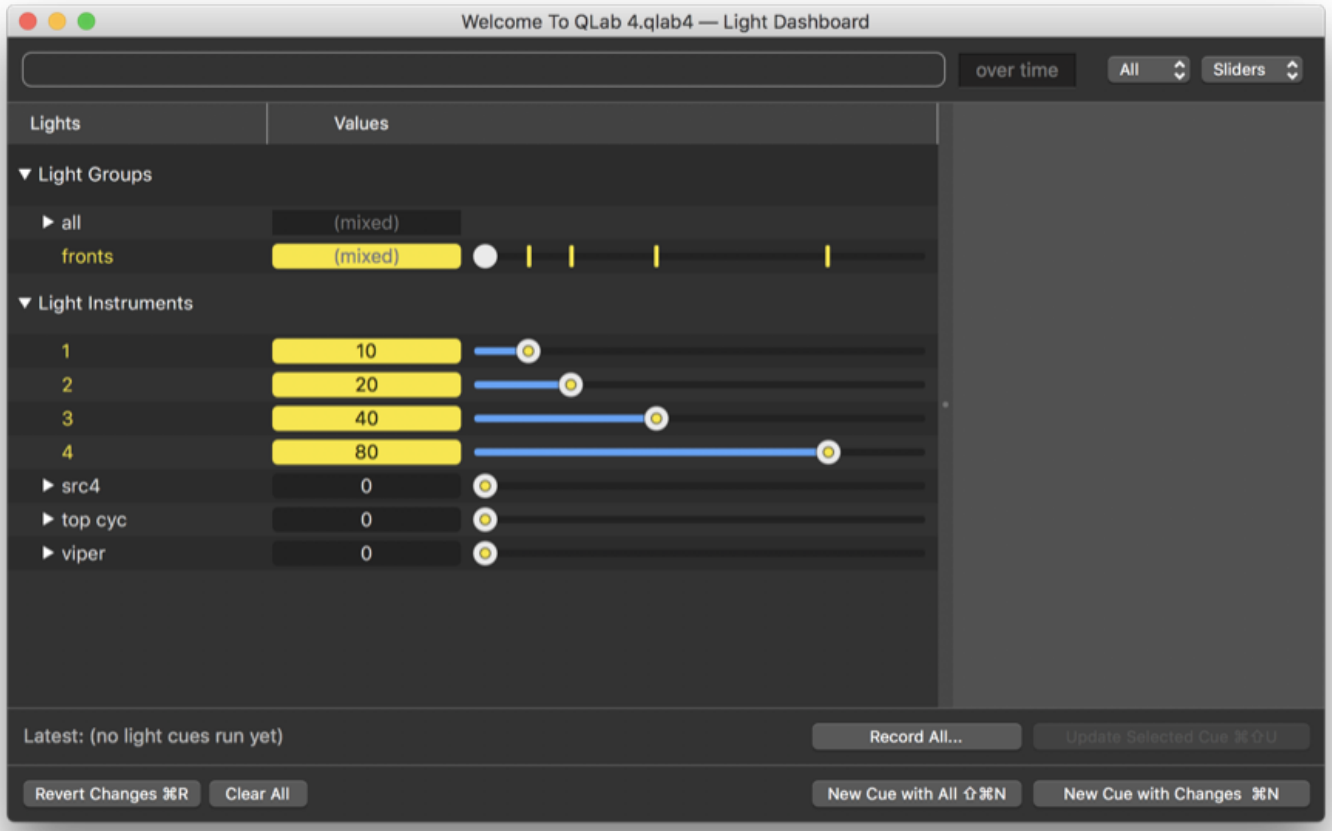
When the drop-down is set to **Used**, the Dashboard will only display instruments and light groups which are currently recorded into cues plus any instruments or light groups which have been manually adjusted (i.e. any control that is drawn in yellow.)

Sliders or Tiles

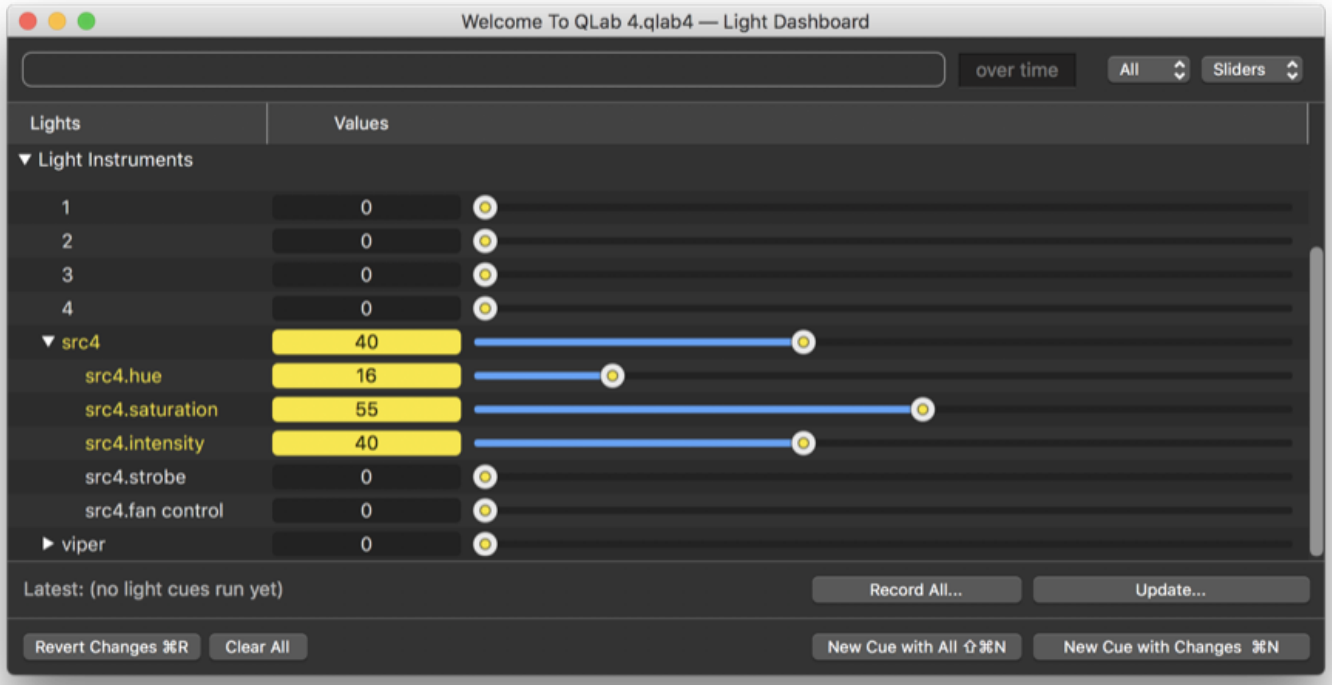
The Dashboard offers two ways to look at your lighting:

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.

Any row containing a value that has been manually adjusted since a Light cue was run will display a yellow background. Rows which represent groups show yellow marks to indicate different levels of individual instruments within the group.

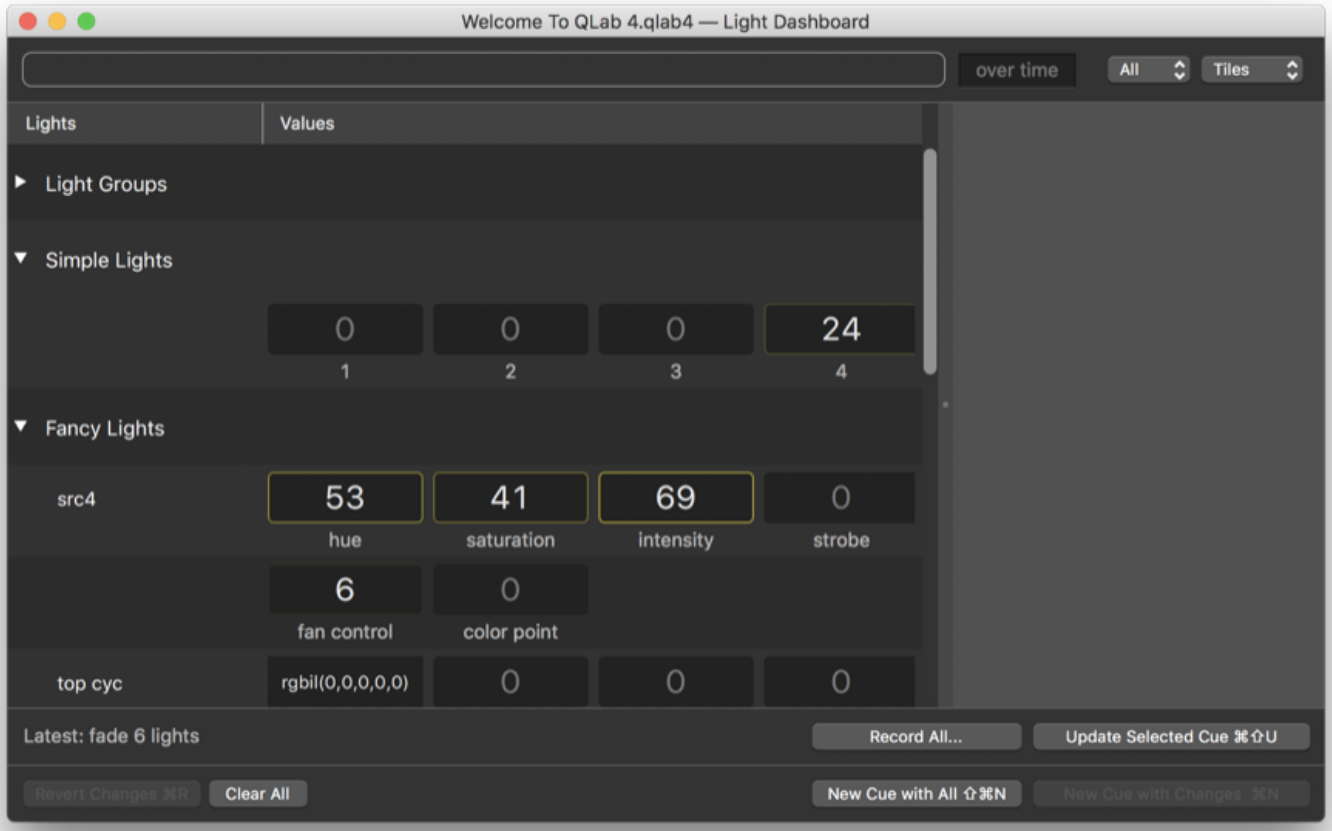


Instruments and light groups that have more than one parameter will have a disclosure triangle on the left edge of the Light Dashboard. When the triangle is pointing to the right, only the default parameter of the instrument or light group (typically the intensity parameter) will be shown. If you click the triangle, the slider will “unfold” to show all parameters.



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, “simple” lights, which are all single-parameter instruments, and “fancy” lights, which are all multi-parameter instruments. The brightness of the outline of each level display reflects the current value of that parameter.

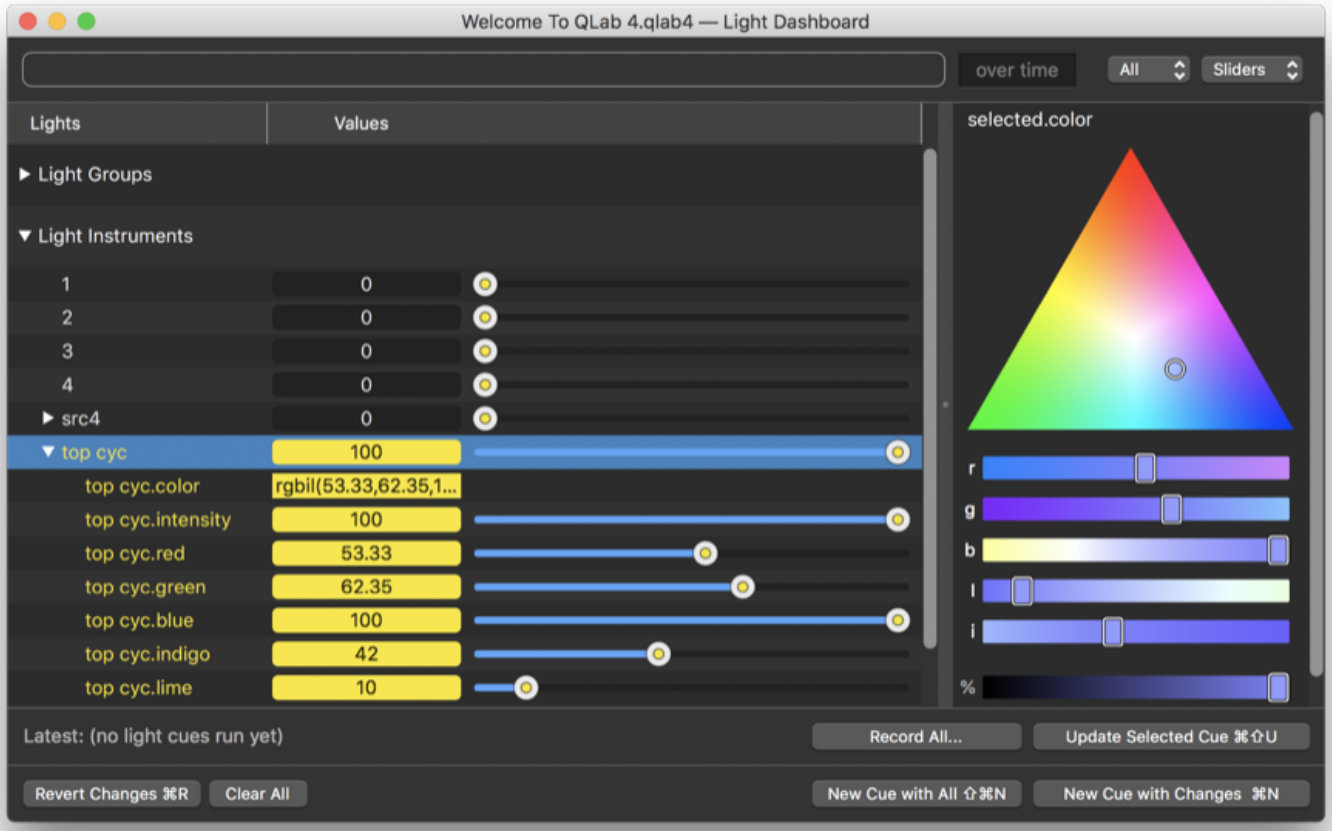


Any tile containing a value that has been manually adjusted since a Light cue was run will display a yellow background.

Virtual Controls

Starting with QLab 4.5, whenever an instrument is selected that has an additive color, subtractive color, or pan/tilt virtual parameter, controls for those virtual parameters are displayed in the sidebar of the Light Dashboard.

The Additive Color Picker



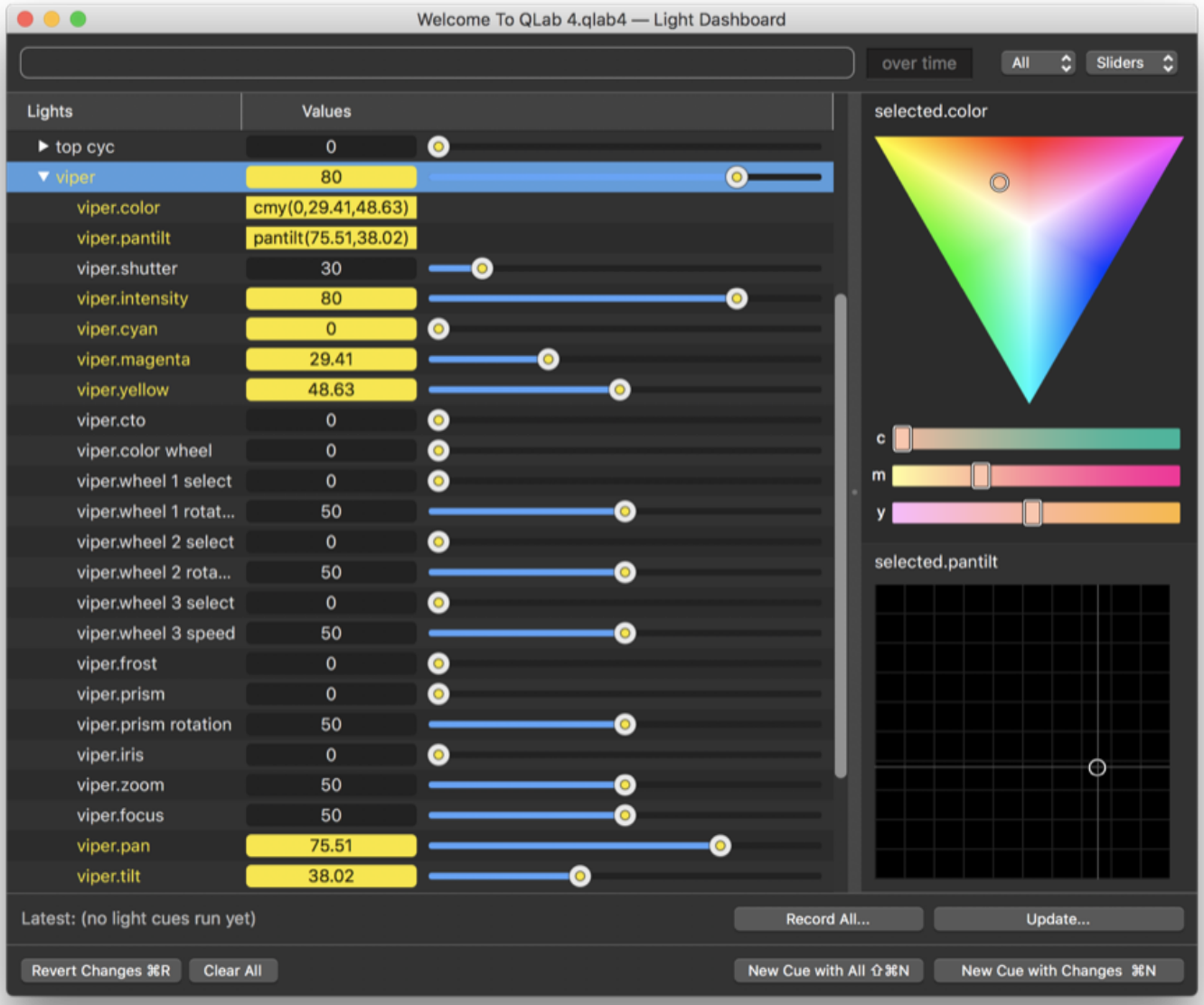
The additive color picker has at least red, green, and blue components, but can also include other components such as amber, white, lime, indigo, or others as specified by the light definition.

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 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 view all controls in the sidebar.

Latest Light Cue

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 triggered, 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.

To revert only individual changes, rather than everything you’ve changed, the Dashboard supports undo (and redo).

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 only 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, and it displays a pop-up menu with three options, each of which may only 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.

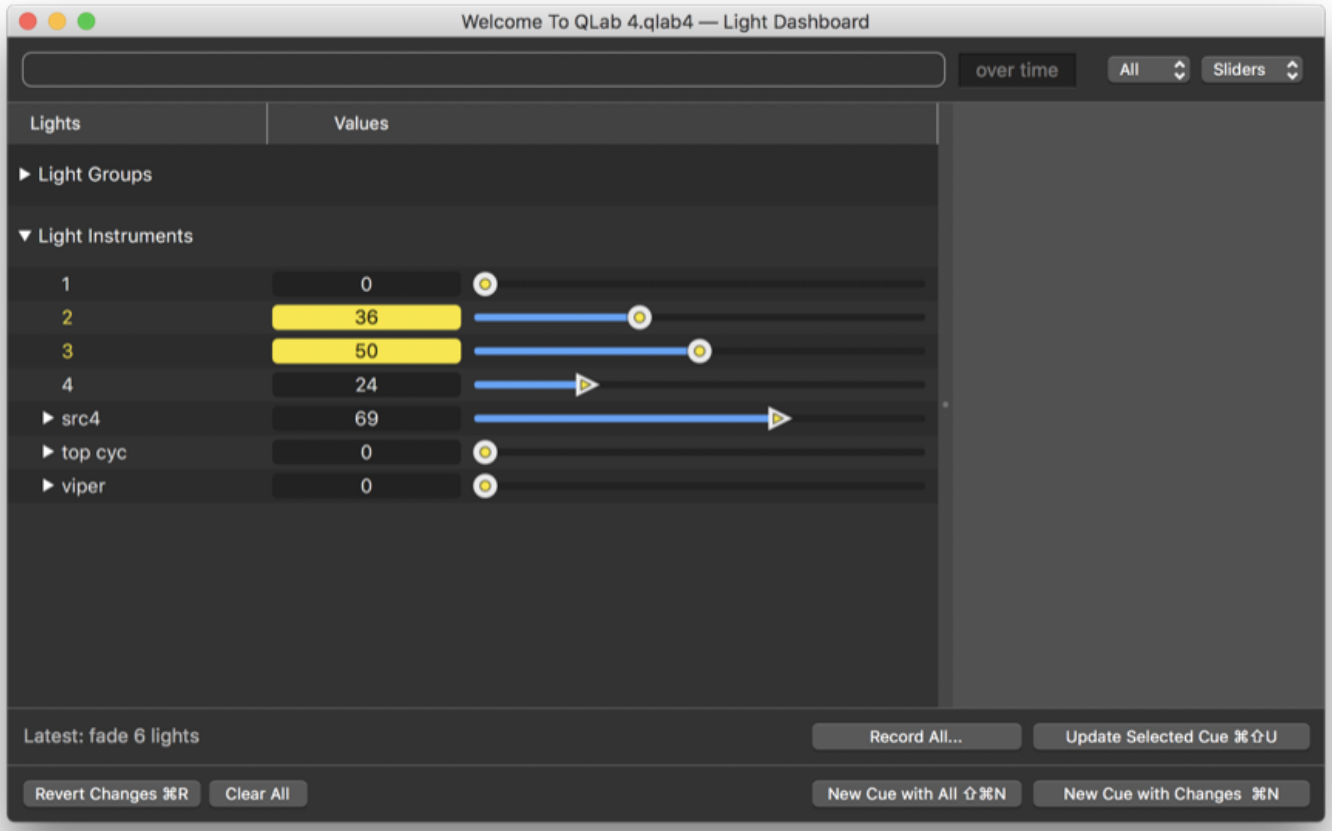
Originating Cue (⌥⌘U). Copy all modified light levels into the cue or cues which *originated* their current levels. This concept is most similar to “record track” and is explained in more detail below.

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
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 instruments 3, 5, and 6 to new levels, let's say all to 75, the button would read **Update 3 Originating Cues** because those three instruments' levels originated in three 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	15	15	15			
3		80				
4				25	**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:

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

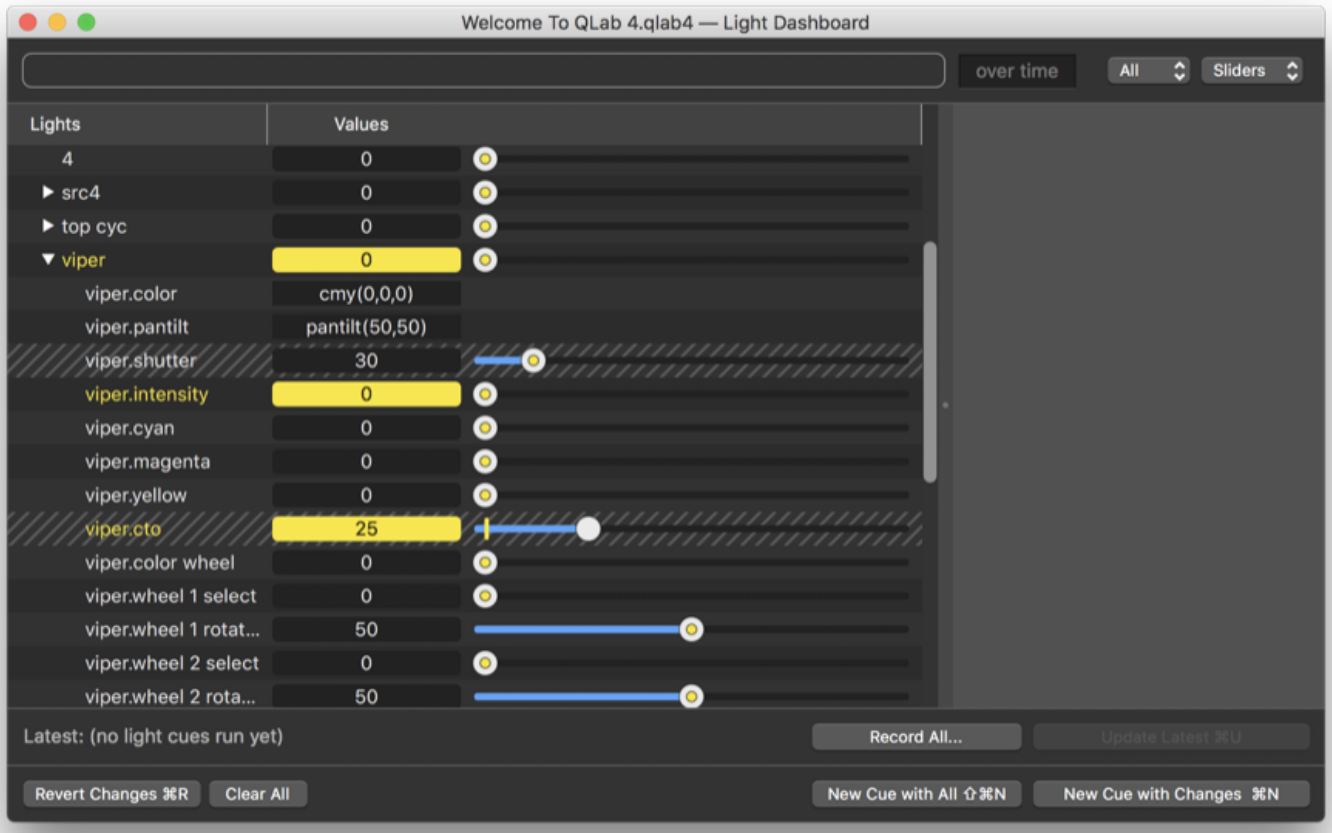
Parking an instrument freezes its current state until it is unparked. While parked, neither cues nor the Light Dashboard can alter the actual live state of the parameter, and the parked value cannot be recorded into cues. This can be helpful for things like keeping house lights at a glow during focus, 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.



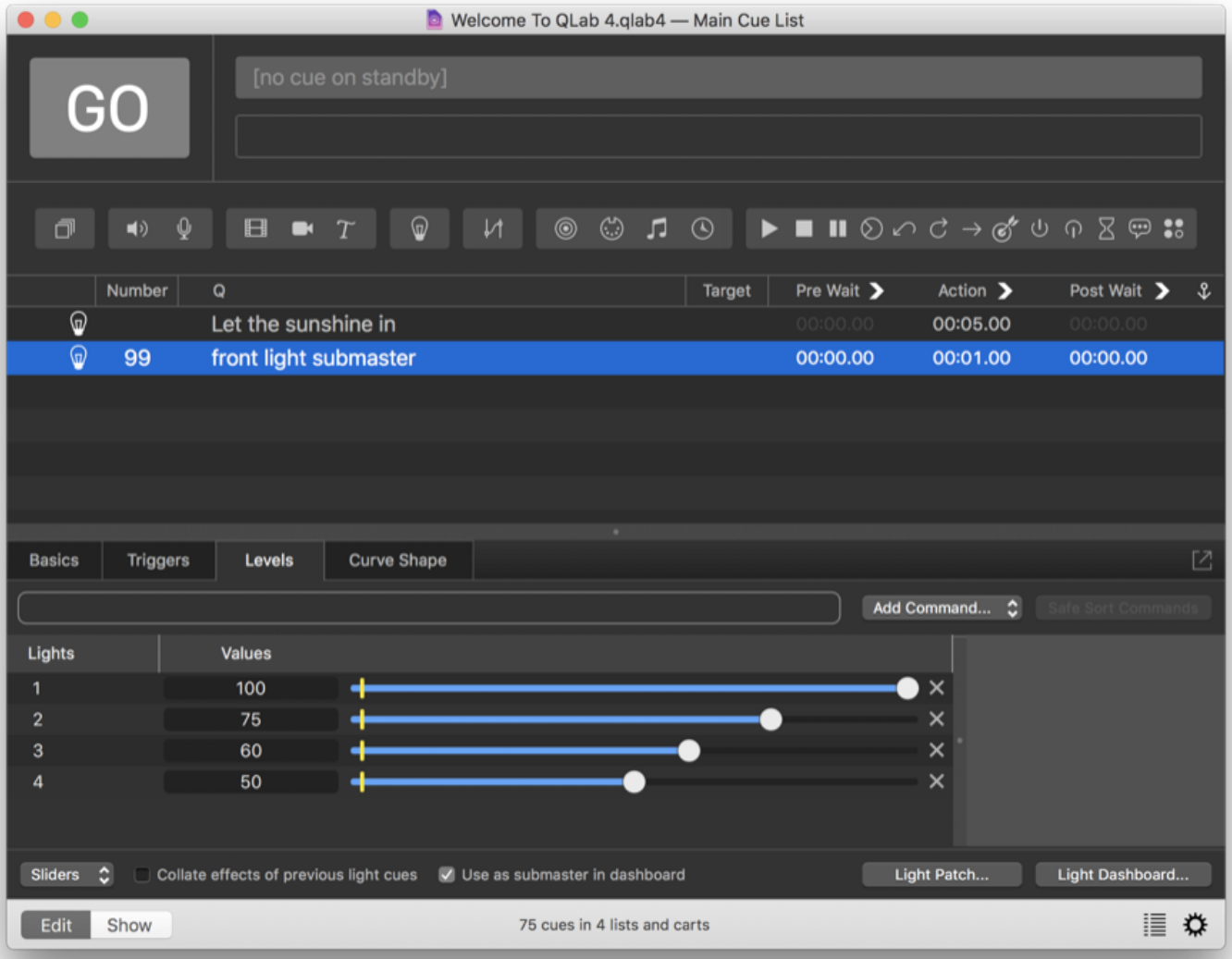
In this screen shot, two parameters of the Viper have been parked: the `shutter` parameter is parked at `30`, and the `cto` parameter is parked at `0`. The `cto` parameter has subsequently been set to `25`, and the light mark shows that the current live value of the `cto` parameter remains at `0`. All 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.

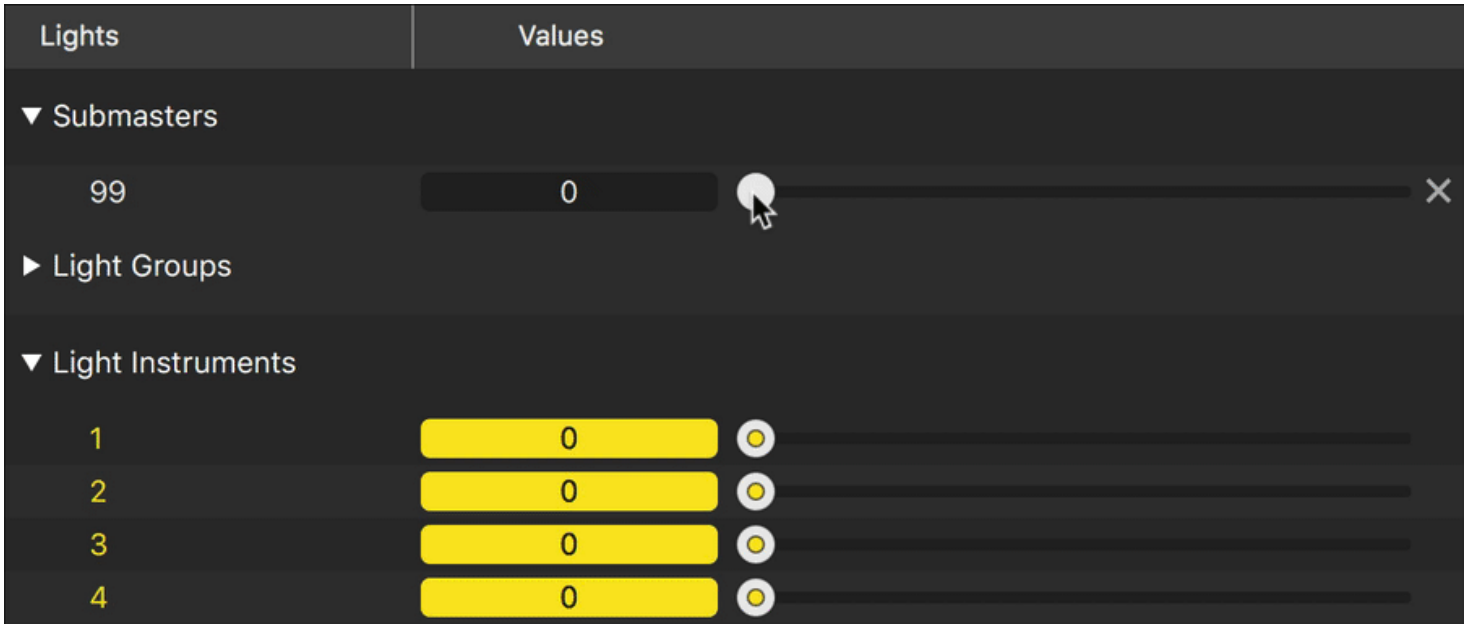
Submasters

Submasters can be thought of as light groups which have proportional control over the parameters which they include. Starting with QLab 4.5, a submaster is defined by creating a Light cue, giving that cue a number, and checking the box labeled *Use as submaster in dashboard* in the Levels tab of the inspector.



Again, only Light cues with cue numbers can be used as submasters.

When that checkbox is checked, a slider appears in the Light Dashboard labeled with the cue number of the cue. Dragging the slider, or typing or dragging the level field, proportionally controls all the parameters recorded into the submaster.



Any parameters can be included in submasters, so it's important to be careful and specific when building Light cues that will be used as submasters. 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

Submasters interact with cues according to a rule called *highest takes precedence*, often referred to simply as *HTP*. When both a cue and a submaster are simultaneously setting levels for a parameter, the parameter is set to whichever level is higher. For example, if a submaster is holding a light at 20% and a cue runs which brings all lights out, the one light in the submaster will remain at 20%. If another cue runs which brings all lights to full, the light in the submaster will go up to full as well, since that's the higher level.

Submasters 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.

Lighting Command Language

QLab 4 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, 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 `left`, `center`, and `right`. If a cue consists of the following commands:

```
backlight = 75
center = pass
```

then instruments `left` and `right` would be set to `75`, and `center` 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 groups.

Pull From Cue

Starting with QLab 4.1, you can instruct a cue to "pull" a value from another cue when it's 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, enter `cue` and the number of the cue you want to pull as the level for a light command. For example:

```
10 = cue A
```

This command sets instrument 10 to whatever values it has 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

Starting with QLab 4.5, you can scale pulled values using the asterisk (*) character:

```
10 = cue A * .5
```

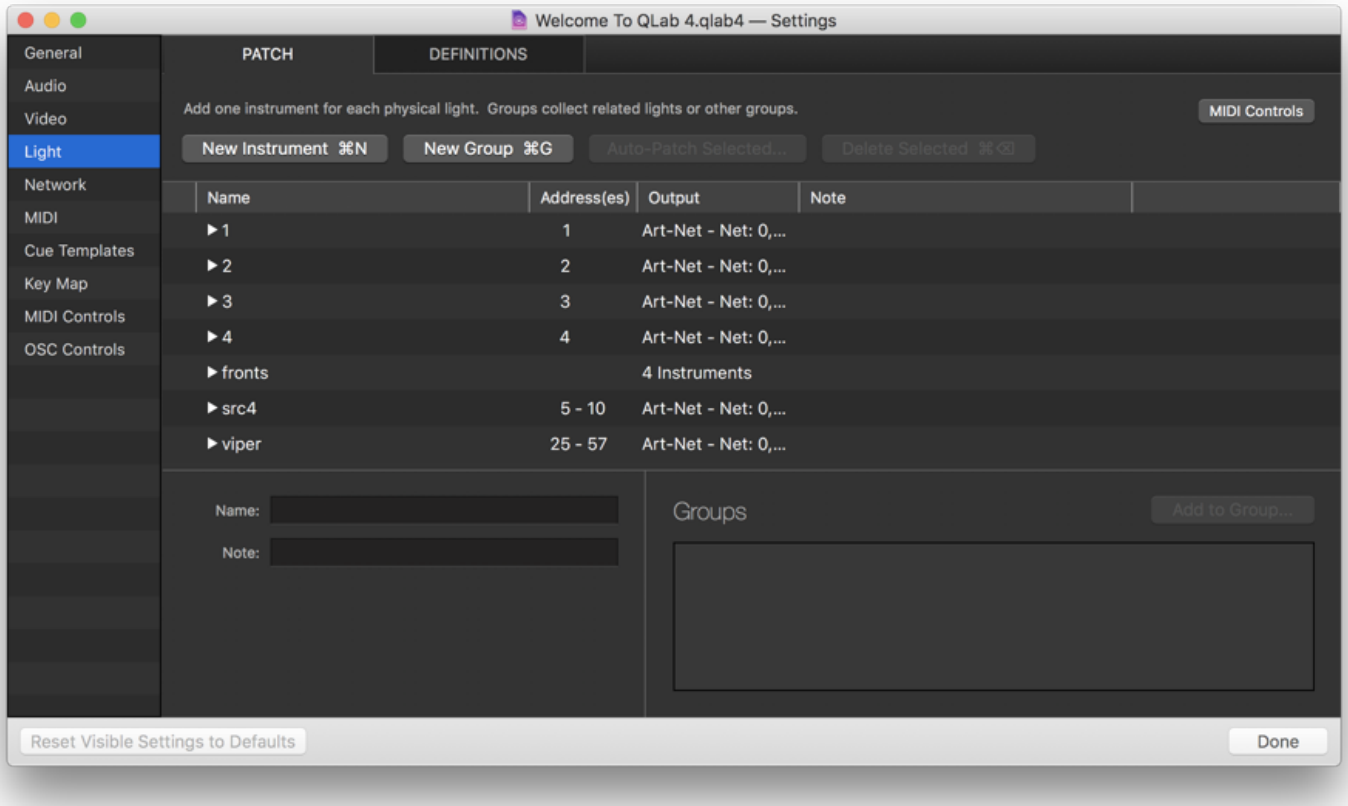
This command sets instrument 10 to half of whatever values it has in cue A. This can be an easy way to mimic a look from an earlier cue, but make it a bit dimmer or a bit brighter.

Lighting Patch Editor

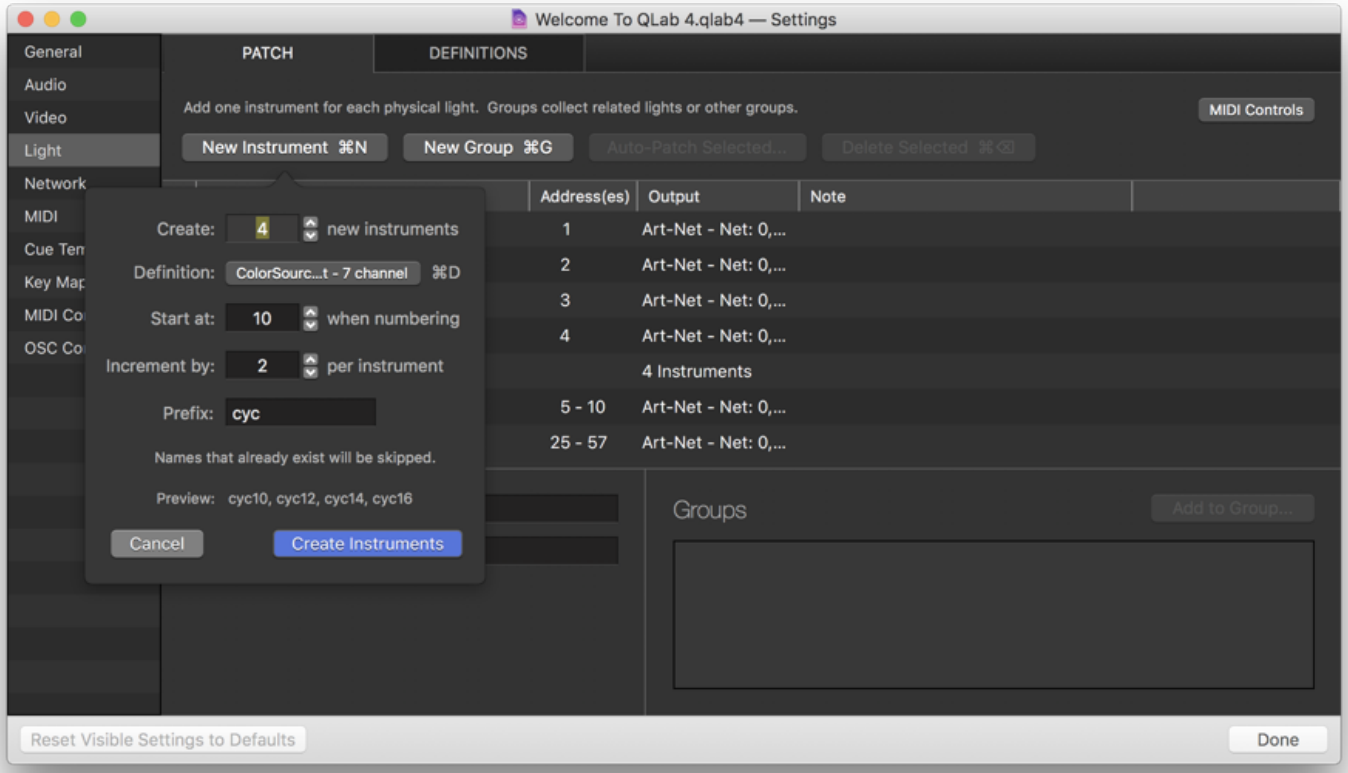
The Lighting Patch Editor, found in Workspace Settings, is divided into two tabs: Patch and Definitions.

The Patch Tab

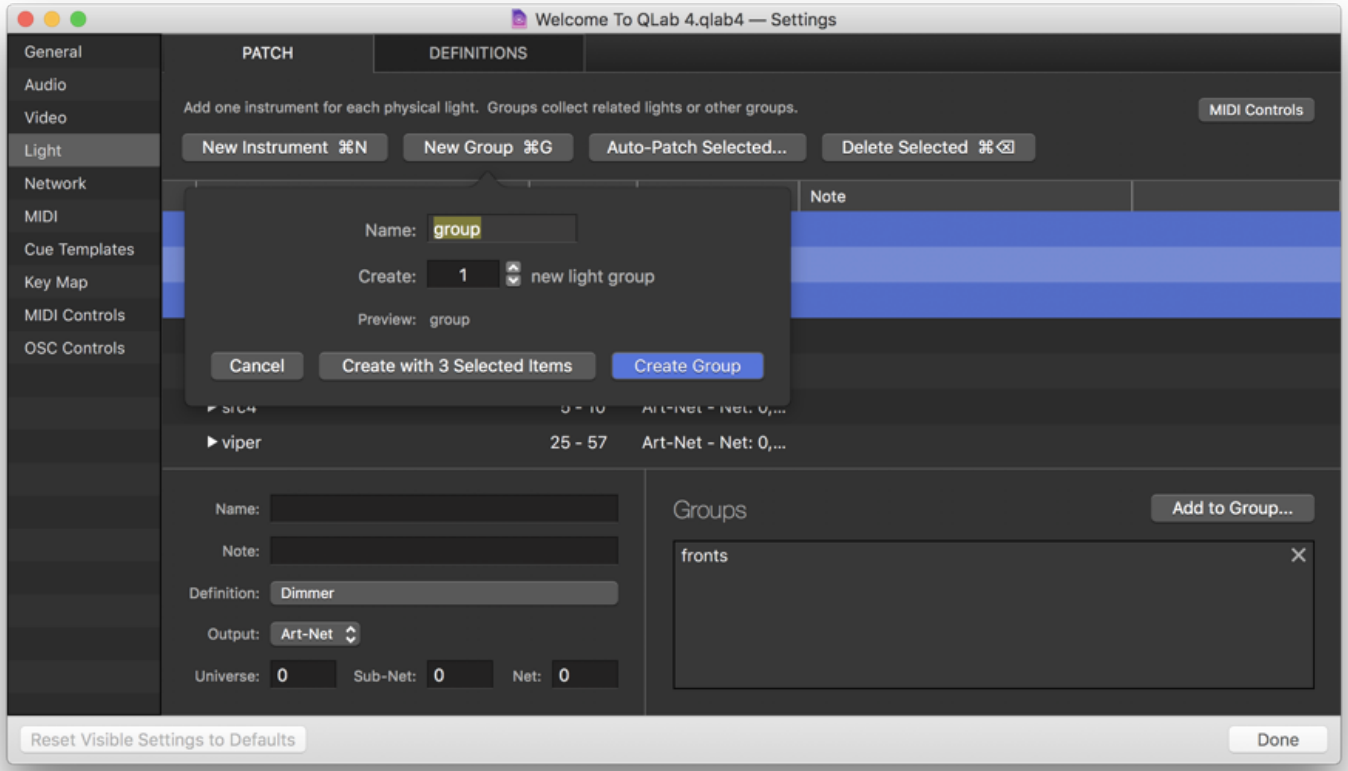
The Patch tab allows you to view and edit the instruments and light groups in the workspace.



New Instrument ⌘N. Click here to 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.



New Group ⌘G. Click here to 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.



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 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.

Delete Selected ⌘⌫. Delete the selected instrument(s) and/or light groups(s) from the workspace.

MIDI Controls. Click here to jump to the **Light Dashboard** section of the MIDI Controls section of Workspace Settings.

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.

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 to edit the address. 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 highlighted in red, and those instruments will not work until the conflict is solved. Without an applicable license installed, any instruments using more than the freely allowed 16 addresses will also be highlighted in red.

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 system 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 several 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. Starting with QLab 4.1, you can use both Art-net and 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 USB, QLab will display a **Device** drop-down 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.

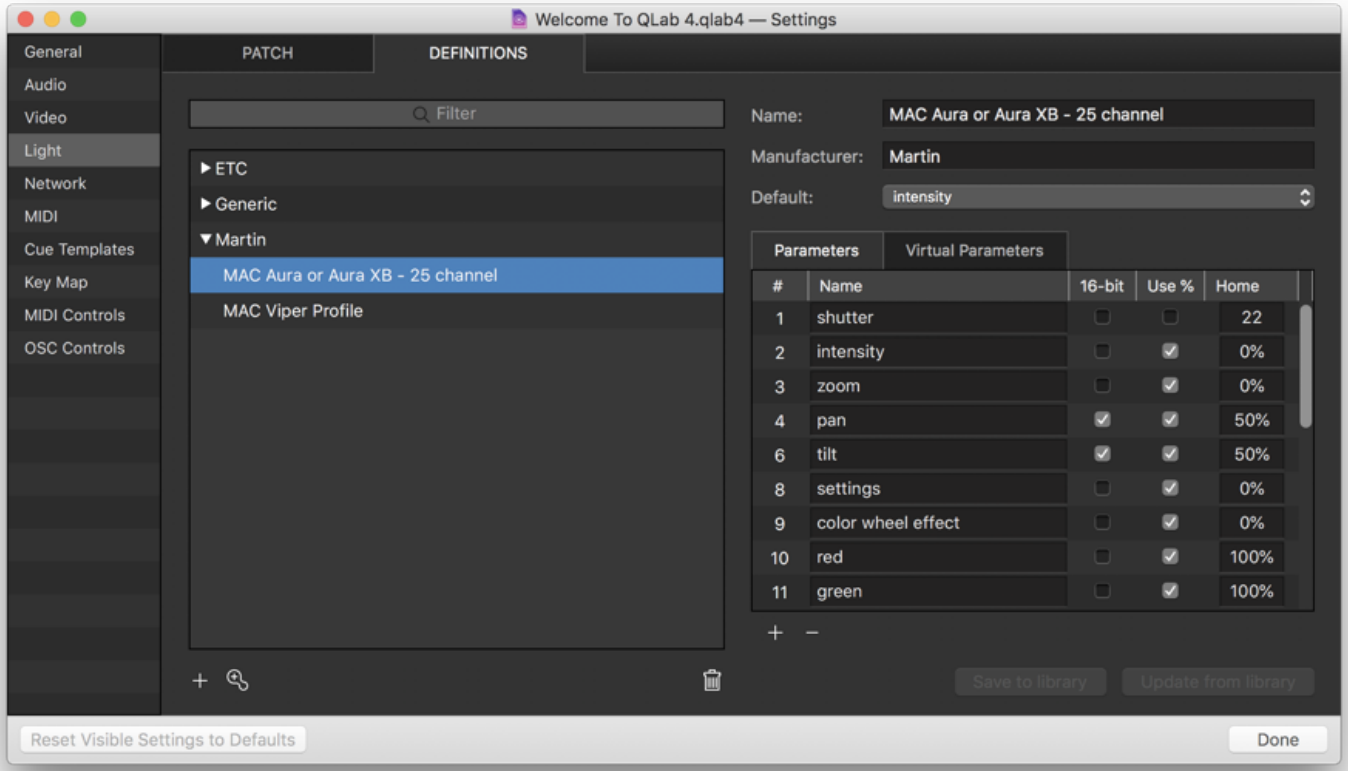
It's important to remember that all Art-net nodes on a network share one set of Universes, Sub-Nets, and Nets, whereas all USB DMX devices have their own, unique sets of Universes. If you have two single-universe Art-net nodes both set to universe 1, sub-net 2, net 3, then the address space is shared 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.

Add to Group... Clicking this button will display a pop-up of all light groups in the workspace, and allow 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 Definitions Tab

The 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 chosen in the 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.



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.

There are three buttons beneath the list:

- + Create a new instrument definition in the workspace.
- Duplicate the selected definition within the workspace.
- Delete the selected definition from the workspace. QLab will not permit you to delete a definition that is currently being used.

Editing Instrument Definitions

Instrument definitions must include a unique *name*, a *manufacturer*, and at least one *parameter*. Starting with QLab 4.5, lights can also have *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 + button below the Parameters table. Click − to delete the last parameter from the current instrument definition.

The Parameters table has 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 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.

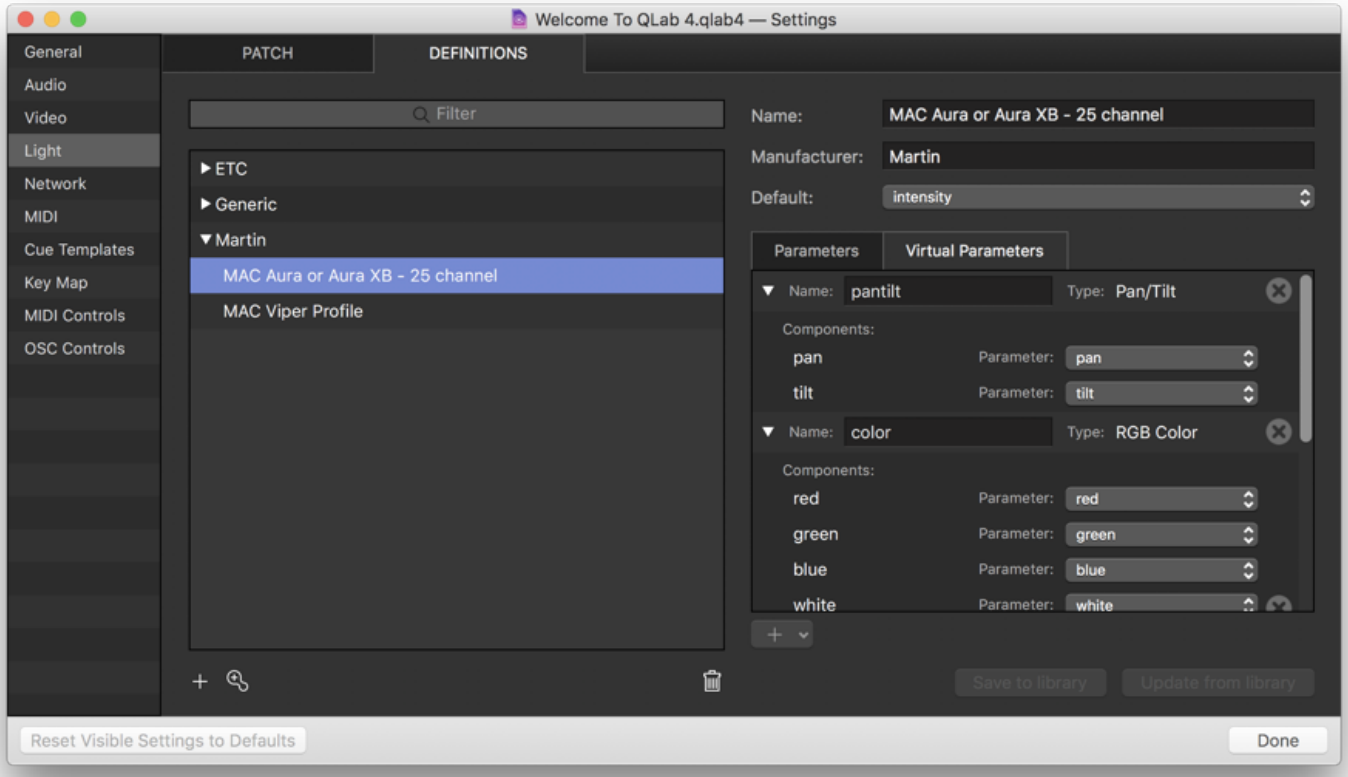
Below the table are two buttons:

Save to library. If the definition does not exist in QLab's global Light Library, you can click here to add it. Saving a definition to the global Light Library allows that definition to be used in any workspace.

Update from library. If the definition in the workspace has the same name as a definition in the global Light Library, but different parameters, you can click here to update the workspace's definition to match the global one.

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 + pop-up 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 in order to use an additive color picker control in the Light Dashboard. 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 channels on lights which use subtractive color mixing (or simulated subtractive color mixing) in order to use a subtractive color picker control in the Light Dashboard.
- **Pan/Tilt.** This virtual parameter is used to combine `pan` and `tilt` parameters in order to use a pan/tilt control in the Light Dashboard.
- **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 a master 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 a master intensity control for the whole fixture. `One-to-Many` virtual parameters can include both regular parameters and other virtual parameters.

The Patch Menu

When the Lighting Patch Editor is active, the Tools menu becomes the **Patch** menu, containing tools for working with the lighting 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.

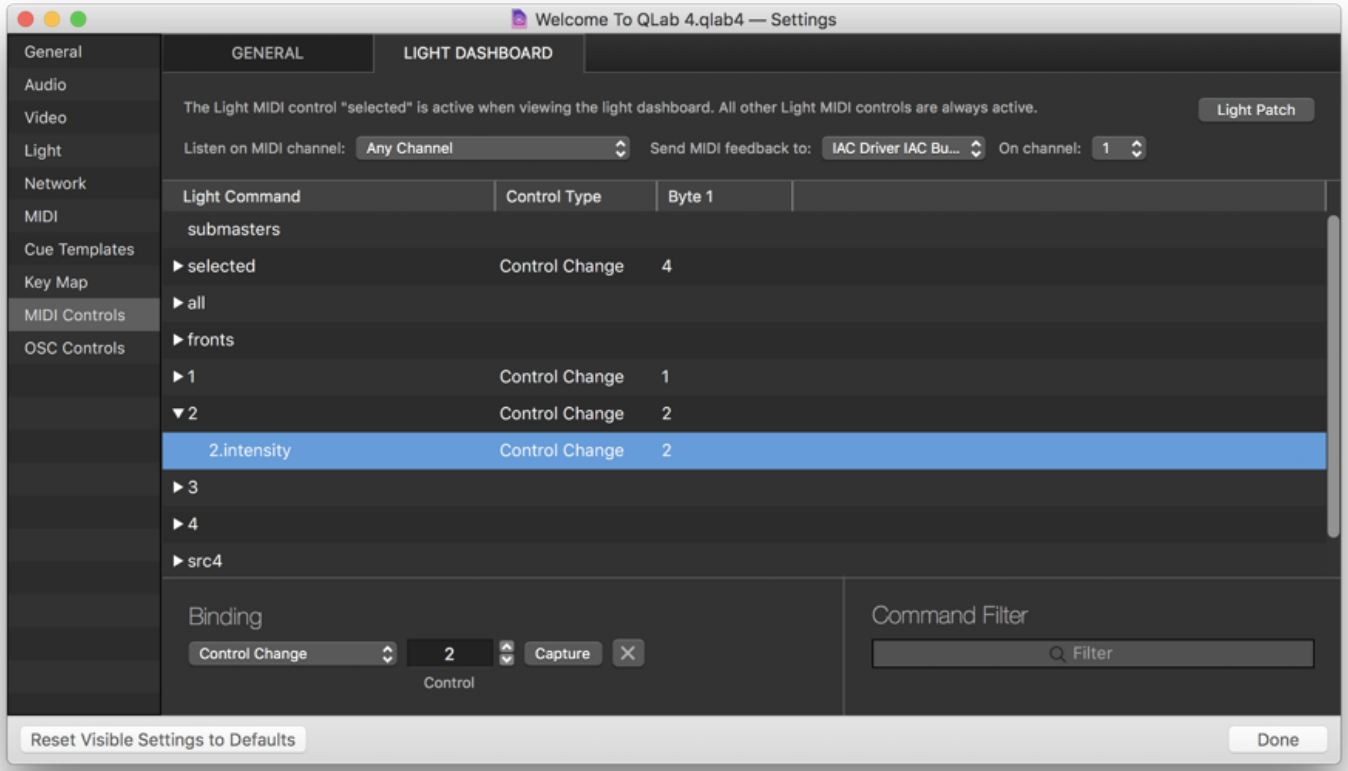
Unpatch Selected. Remove the DMX address assignment for the instruments currently selected in the patch list.

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.

MIDI Controls

Starting with QLab 4.3, the MIDI Controls section of Workspace Settings allows you to configure, or *bind*, MIDI messages to control and respond to instruments and light groups. This makes it possible to use any MIDI device as a physical controller for the Dashboard.



Listen on MIDI channel. This lets you restrict the MIDI channel that QLab will use for incoming MIDI control of the Dashboard. It can be set to follow the channel set in the **General** tab, or set to "any", or to any MIDI channel. This setting has no effect on any other MIDI behavior of QLab.

Send MIDI feedback to. If you choose a MIDI output here, QLab will send MIDI feedback for any instrument or group which has a MIDI control assigned to it. This is needed for motorized MIDI fader controllers or touchscreen controllers to follow along with QLab. If you are using a non-motorized physical controller, this setting has essentially no effect. Otherwise, it's generally best to set this to send MIDI feedback to the MIDI device you're using with the Dashboard.

On channel. This is the MIDI channel that QLab will use to send MIDI feedback. It can be the same or different as the listening channel, although most MIDI setups will generally use the same channel.

Light Patch. Click here to jump to the Light Patch section of the Light section of Workspace Settings.

The MIDI Bindings Table

All instruments and light groups in the workspace are listed here, sorted alphabetically with light groups listed first, along with a special "selected" light group which is discussed below. You can click on the disclosure triangles to view the parameters of each instrument or group.

Selecting an instrument, group, or parameter in the list allows you to bind a MIDI command to that item. You can edit the binding manually by choosing a message type from the drop-down menu and typing in an associated value, or you can click the **Capture** button and QLab will listen for and capture the next incoming MIDI message.

Click the **X** button to remove the assigned binding.

When an instrument, group, or parameter has a binding assigned, the type and first byte of the MIDI message will be displayed in the MIDI Bindings table.

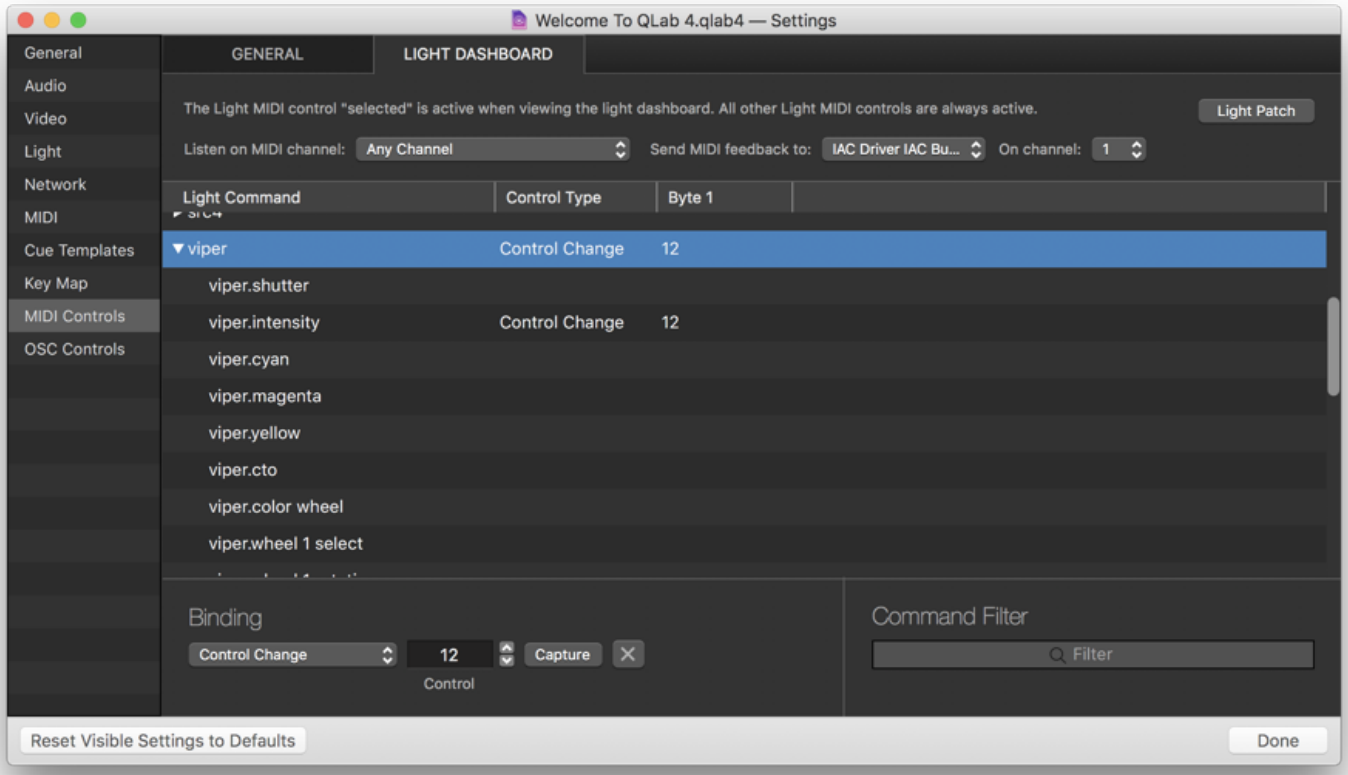
Selected

The special "selected" group which appears here shows all parameters of all instruments in the workspace, much like the "all" group. When a MIDI message is assigned to the "selected" group, that MIDI message is used to control any instruments and/or groups which are currently selected in the Dashboard.

This can be a very convenient way to use a relatively small MIDI controller to control any light in your workspace, or to use a controller as a way to adjust a set of parameters (like color, or pan and tilt) for any given instrument.

Multi-parameter Instruments and Light Groups

If a MIDI message is bound to the top-level of a multi-parameter instrument or light group, the message will be passed to the default parameter of that instrument or light group. To underscore this, the same MIDI message is listed again in italic text next to the default parameter.



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 Lighting 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 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. If you would like us to add a specific fixture, or a whole range of fixtures, to QLab's library, please contact us at support@figure53.com and tell us about it. We'll be happy to create the instrument definitions and 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

Altman

- AP-150 RGBW LED Par
- Spectra Series fixtures (all models)
- PHX Series fixtures (all models)

American DJ

- 15 Hex Bar IP
- 12P Hex
- 18P Hex
- 3 Sixty 2R
- 5P Hex
- 64B LED Pro
- 7PZ IP
- Asteroid 1200
- Chameleon QBar Pro
- COB Cannon Wash
- COB Cannon Wash DW
- Dotz Par
- Dotz Par 100
- Encore Burst RGBW IP
- Encore Burst UV IP
- Entour Ice
- Entour Venue
- Entourage
- Event Bar Pro
- Flat Par QA5XS
- Flat Par TRI7XS
- Flat Par TRI18XS
- Flat Par TW5
- Flat Par TW12
- Focus Spot One
- Focus Spot Two
- Focus Spot Three Z
- Focus Spot 4Z
- Fog Fury Jett
- Fog Fury Jett Pro
- 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 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
- Starship
- Stinger Spot
- Sweeper Beam Quad LED
- UB 6H
- UB 9H
- UB 12H
- 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
- Vizi Beam 5RX
- Vizi Beam RXONE
- Vizi BSW 300
- Vizi CMY300
- Vizi Hex Wash7
- Vizi Hybrid 16RX

- Vizi Q Wash7
- Vizi Roller Beam 2R
- Vizi Wash Z19
- XS 600

Cameo

- Auro Bar 100
- Auro Beam 150
- Auro Beam 200 DC
- Auro Matrix 500
- Auro Spot 100
- Auro Spot 200
- Auro Spot 300
- Auro Spot 400
- Azor B1
- CL 200
- Evos S3
- F2 D
- F2 FC
- F2 T
- Flat 1 RGB 10 IR
- Flat 1 TRI 3W IR
- Flat 1 TW
- Flat Moon
- Flat PAR RGBW IR
- Flat Pro 12
- Flat Pro 12 IP
- Flat Pro 18
- Flat Pro 18 IP
- Flat Pro 7
- Flat Pro 7 IP
- Flat Pro 7Spot
- Flat Pro 7XS
- Flat Pro Flood 600 IP65
- Flat Pro Flood IP65 TRI
- Flat Star
- Flat Storm
- Flat UV
- Hydrabeam 1000 RGBW
- 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
- Nanospot 120
- Nanospot 300
- Opus S5
- Opus SP5
- Opus SP5 FC
- 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
- Q-Spot 15 RGBW
- Q-Spot 15W
- Q-Spot 40 CW
- Q-Spot 40 RGBW
- Q-Spot 40 WW
- 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 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 W300
- Zenit W600
- Zenit W600-D
- Zenit Z120
- Zenit Z120 G2

Chauvet

- 4Bar Tri
- COLORado 1 Quad
- COLORado 1 Quad Zoom
- COLORado 1 Solo
- COLORado 1 Tri IP
- COLORado 1 Tri Tour
- COLORado 2 Quad Zoom
- COLORado 2 Solo
- COLORado 3 Solo
- COLORado Batten 72
- COLORado Batten 72X
- COLORado M Solo
- COLORado Panel Q40
- COLORado Solo Batten
- COLORado Solo Batten 4
- COLORband PiX
- COLORdash Accent Quad
- COLORdash Batten-Quad 12
- COLORdash Batten-Quad 6
- COLORdash Par Hex 12
- COLORdash Par Hex 7
- COLORdash Par Q12 IP
- COLORdash Par Quad 7
- COLORdash S-Par 1
- Cloud 9
- EZpar T6 USB
- FXpar 3
- FXpar 9
- Legend 230SR
- Maverick MK Pyxis
- Maverick MK1 Hybrid
- Maverick MK1 Spot
- Maverick MK2 Profile
- Maverick MK2 Spot
- Maverick MK2 Wash
- Maverick MK3 Profile
- Maverick MK3 Spot
- Maverick MK3 Wash
- Maverick Storm 1 Wash
- Ovation B-1965FC
- Ovation B-565FC
- Ovation CYC 1 FC
- 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-55WW
- Ovation P-56FC
- Ovation P-56UV
- Ovation P-56VW
- Ovation P-56WW
- Rogue R1 Beam
- Rogue R1 BeamWash
- Rogue R1 Spot
- Rogue R1 Wash
- Rogue R1X Spot
- Rogue R1X Wash
- Rogue R2 Beam
- Rogue R2 Spot
- Rogue R2 Wash
- Rogue R2X Spot
- Rogue R2X Wash
- Rogue R3 Spot
- Rogue R3 Wash
- Rogue R3X Wash
- Rogue RH1 Hybrid
- Strike P38

- Strike Saber

Chroma-Q

- Color Force 12
- Color Force 48
- Color Force 72
- Color One 100
- Color One 100X

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
- Alpha Beam 700
- Alpha Beam 1500
- Alpha Profile 700
- Alpha Profile 800 ST
- Alpha Profile 1500
- Alpha Spot 300
- Alpha Spot HPE 1500
- Alpha Spot HPE 300
- Alpha Spot HPE 700
- Alpha Spot QWO 800
- Alpha Wash 300
- Alpha Wash 700
- Alpha Wash 1500
- Axcor Beam 300
- Axcor Spot 300
- Axcor Wash 300
- Axcor Profile 900
- Hepikos
- K-EYE K10 HCR
- K-EYE K20 HCR
- K-EYE S10 HCR
- K-EYE S20 HCR
- Mythos
- Mythos 2
- Scenius Profile
- Scenius Spot
- Scenius Unico
- Sharpy
- Sharpy Wash
- Sharpy Wash 330 PC

- Spheriscan
- Supersharpy
- Supersharpy 2

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 Wash CW-WW
- LED Wash RGBW
- LEDWash XL
- Orion
- Poseidon
- Quadcolor Mini PAR
- Serius
- Spectrum P1
- Spectrum P2
- Tricolor Mini PAR
- Xena
- Yara

Elation

- 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
- 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
- Paladin
- 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
- 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 Spot
- 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 Four LED Series 1 (Lustr+ and Studio HD)
- Source Four LED Series 2 (Lustr, Daylight HD, and Tungsten HD)

German Light Products (GLP)

- Force 120
- GT-1
- Highlander
- Impression 120RZ
- Impression 240XL
- Impression 90 RGB
- Impression 90 RGB Static
- Impression 90 WhiteAmber
- Impression E350
- Impression FR1
- 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
- JDC1
- Volkslicht RGB
- Volkslicht RGB Spot
- Volkslicht Spot
- X4 Atom PSU-6
- X4 Atom PSU-12

High End Systems

- Quad
- SolaFrame 750
- SolaFrame 1000
- SolaFrame 1500
- SolaFrame 2000
- SolaFrame 3000
- SolaFrame Theatre
- SolaHyBeam 1000
- SolaHyBeam 2000
- SolaSpot 1000
- SolaSpot 2000
- SolaSpot 3000
- SolaSpot Pro CMY
- SolaWash 1000
- SolaWash 2000
- SolaWash 3000
- TurboRay

K9

- Bulldog
- Bulldog Pro
- Labrador
- Pup

Martin

- Atomic 3000
- Atomic 3000 LED
- ELP CL
- ELP WW
- ERA 300 Profile
- ERA 400 Performance CLD
- ERA 400 Performance WRM
- MAC 250
- MAC 250+
- MAC 250 Krypton
- MAC 250 Entour
- MAC 250 Wash
- 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 XB
- MAC Axiom Hybrid
- MAC Encore Performance CLD
- MAC Encore Performance WRM
- MAC Encore Wash CLD
- MAC Encore Wash WRM
- MAC Quantum Profile
- MAC Quantum Wash
- MAC Viper AirFX
- MAC Viper AirFX Quadray
- MAC Viper Performance
- MAC Viper Profile
- MAC Viper Wash
- MAC Viper Wash DX
- 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

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

Panasonic

- PT-RZ570

Philips Selecon

- PLCYC1 mkII
- PLFRESNEL1 mkII
- PLPROFILE1 mkII
- PLPROFILE4 mkII

Robe

- BMFL Blade
- BMFL Followspot
- BMFL Followspot LT
- BMFL Spot
- BMFL Wash
- BMFL Wash XF
- BMFL WashBeam
- BMFL WashBeam EV
- CycFX 4
- CycFX 8
- Cyclone
- DigitalSpot 3500 DT
- DL4F Wash
- DL4S Profile
- DL4X Spot
- DL7F Wash
- DL7S Profile
- Dominator 1200 XT
- Esprite
- iPointe
- LEDBeam 100
- LEDBeam 100 DayLight
- LEDBeam 100 SmartWhite
- LEDBeam 150
- LEDWash 300
- LEDWash 300 SmartWhite
- LEDWash 300+
- LEDWash 600
- LEDWash 600+
- LEDWash 800
- LEDWash 800 DayLight
- LEDWash 800 SmartWhite
- LEDBeam 1000
- LEDWash 1200
- MegaPointe
- MiniMe
- MiniMe DV
- MiniPointe
- MMX Blade
- MMX Spot
- MMX WashBeam
- 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
- RoboSpot Motion Camera
- SilverScan
- Spiider
- Spikie
- SuperSpikie
- Strobe IP
- T1 Fresnel
- T1 PC
- T1 Profile
- T1 Profle FS
- Viva
- Viva CMY

Rosco

- I-Cue
- RevoPRO

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

- SL Bar 620
- SL Bar 640
- SL Bar 660
- SL Bar 720ZT
- SL eStrip 10 RGBW
- SL Hydrus 350
- SL LEDSpot 300
- SL Nitro 510C
- SL PAR 155 Zoom RGBW
- SL Punchlite 220
- SL Strip 10 IP

Showtec

- Sunstrip Active

Vari*Lite

- VL10 BeamWash
- VL440 Spot
- VL500 Wash
- VL550 Wash
- VL770 Spot
- VL800 BeamLine
- VL800 EVENTPAR RGBA
- VL800 EVENTPAR WW
- VL800 EVENTWASH
- VL800 PROPAR
- VL880 Spot
- VL1000 (TI, AI, TS, AS)
- VL1100 (TI, AI, TS, AS)
- VL1100 LED
- VL1100 LED HP
- 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
- VL4000 BeamWash
- VL4000 Spot
- VL6000 Beam
- VL6500 Wash
- VLX Wash
- VLX3 Wash
- VLZ Profile
- VLZ Spot
- VLZ Wash

Wybron

- Coloram
- CXI
- Forerunner

Yorkville

- LP-LED/x Bar series

Chapter 6: Control

QLab Remote

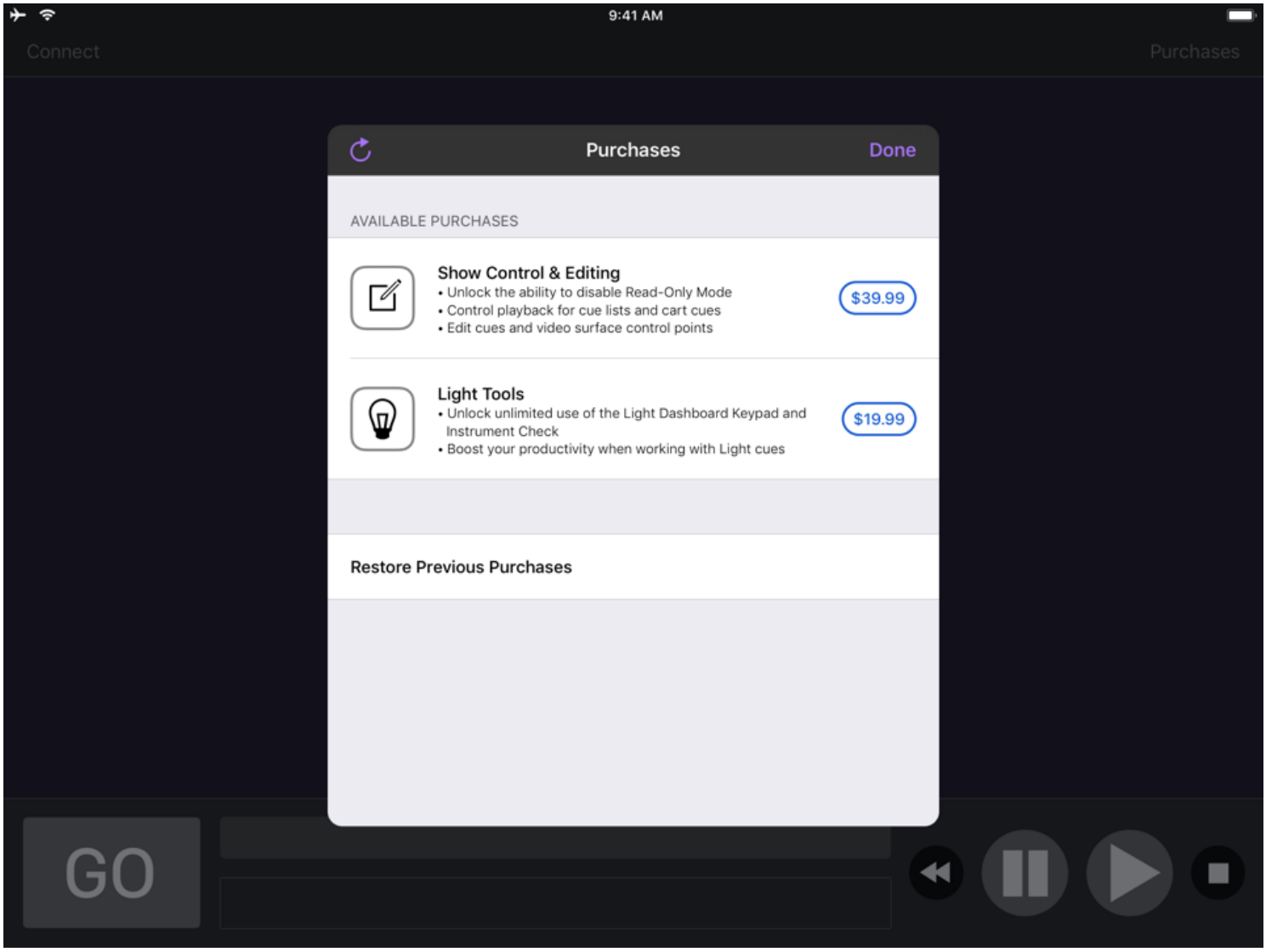
QLab Remote is an iOS app, available on the App Store, that can connect to QLab to remotely view, edit, and run cues. You can run QLab Remote on any iOS device running iOS 9.0 or later.

QLab Remote adapts to different screen sizes and orientations as best as it can, and it supports Split View. Some features will only be available when screen space permits, which means some features are only available on iPads.

Features

Beginning with version 4.2, released March 2018, 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 QLab Remote prior to version 4.2, 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 in QLab 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 surface 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 and QLab 4. 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.

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.

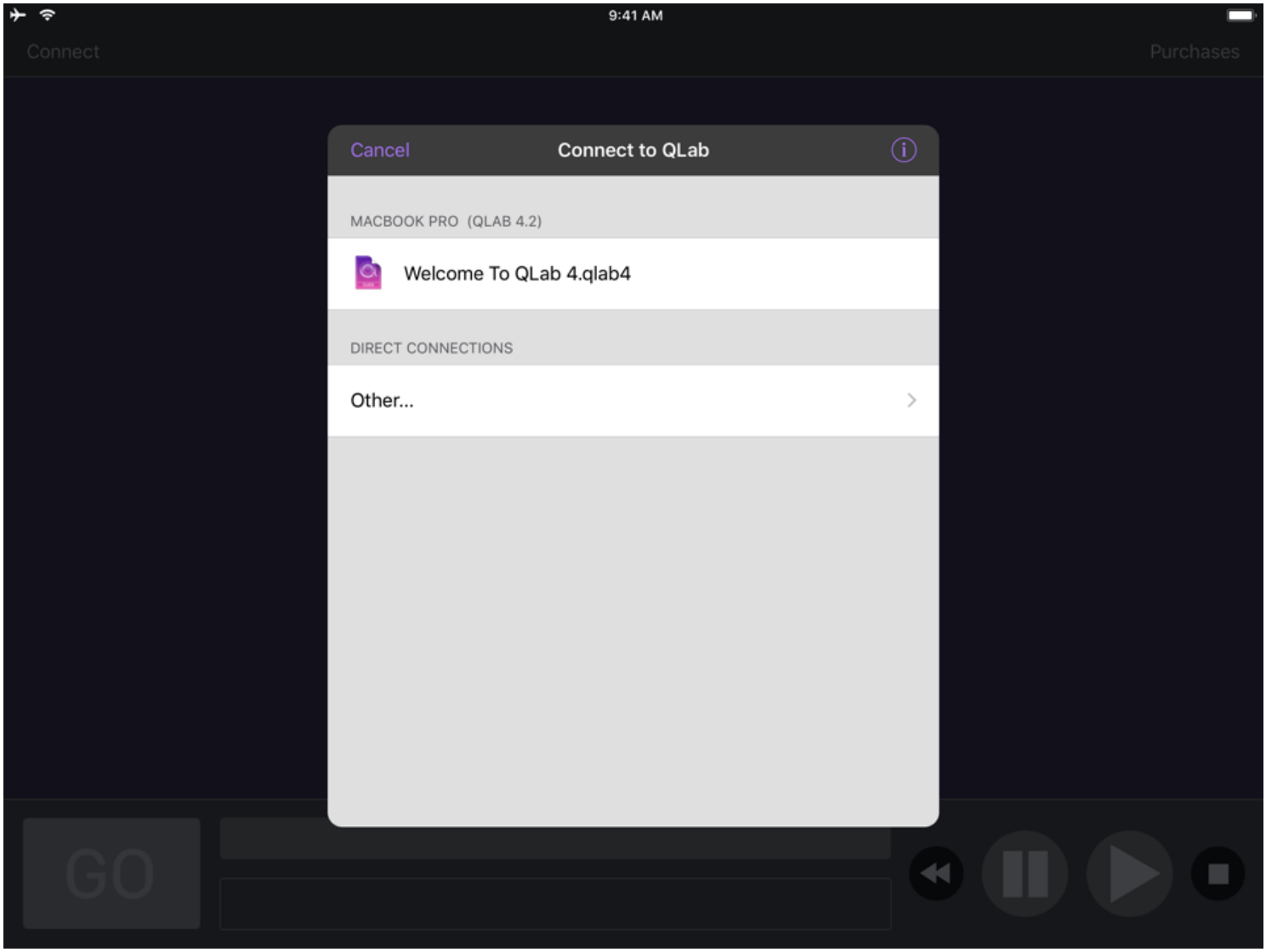
Getting Started

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.

Alternately, if you're using QLab 4.1 or later, you can connect your iOS device to the Mac running QLab using a Lightning to USB cable, or a 30-pin to USB cable for older iOS devices.

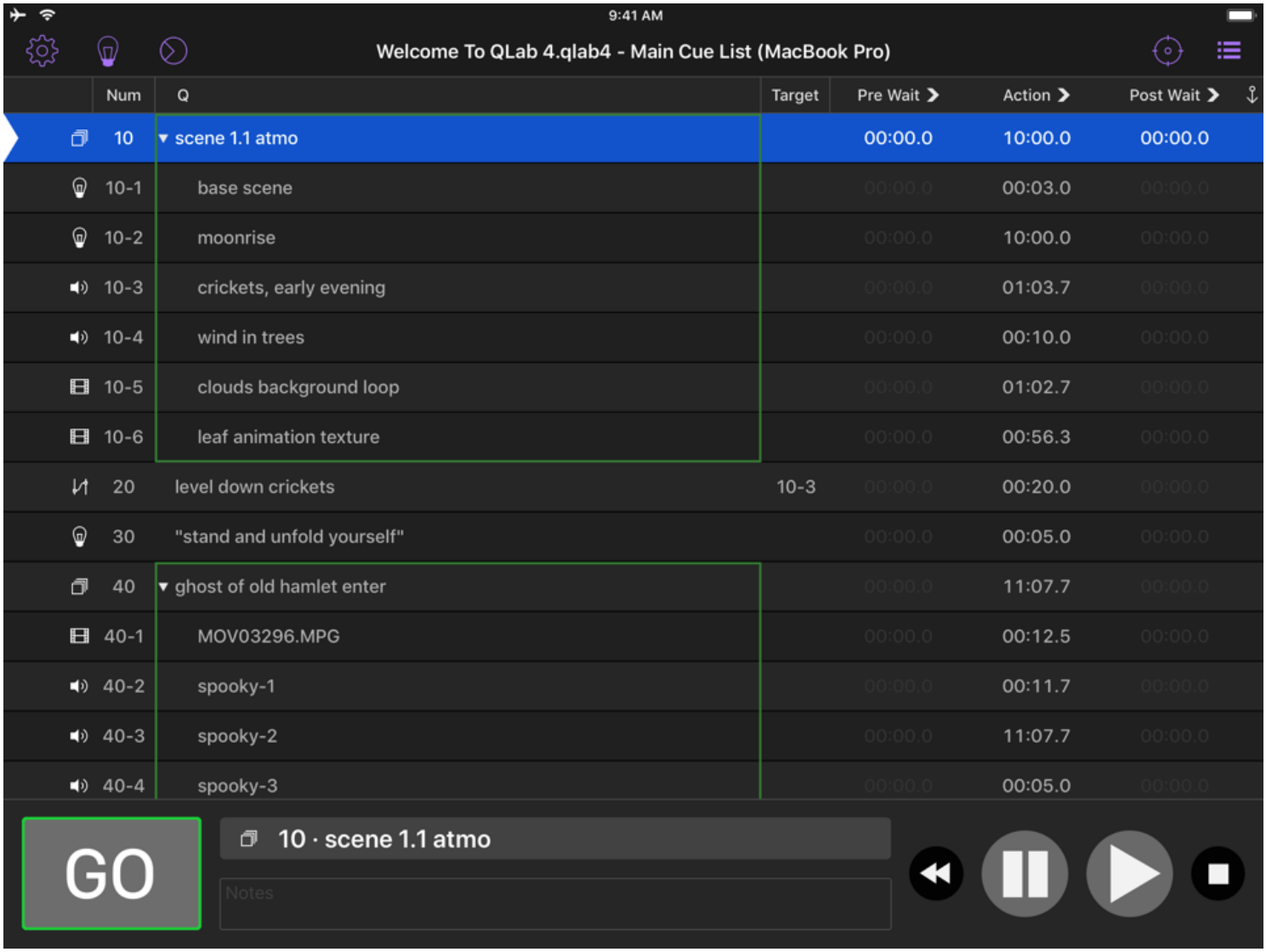
Whether you're connecting via a network or via USB tethering, your QLab workspace must also be set to use OSC controls, which can be set in the OSC Controls section of Workspace Settings. OSC controls are turned on by default in all QLab workspaces.

With your devices connected physically, and your workspace open on your Mac, launching QLab Remote will give you this screen:



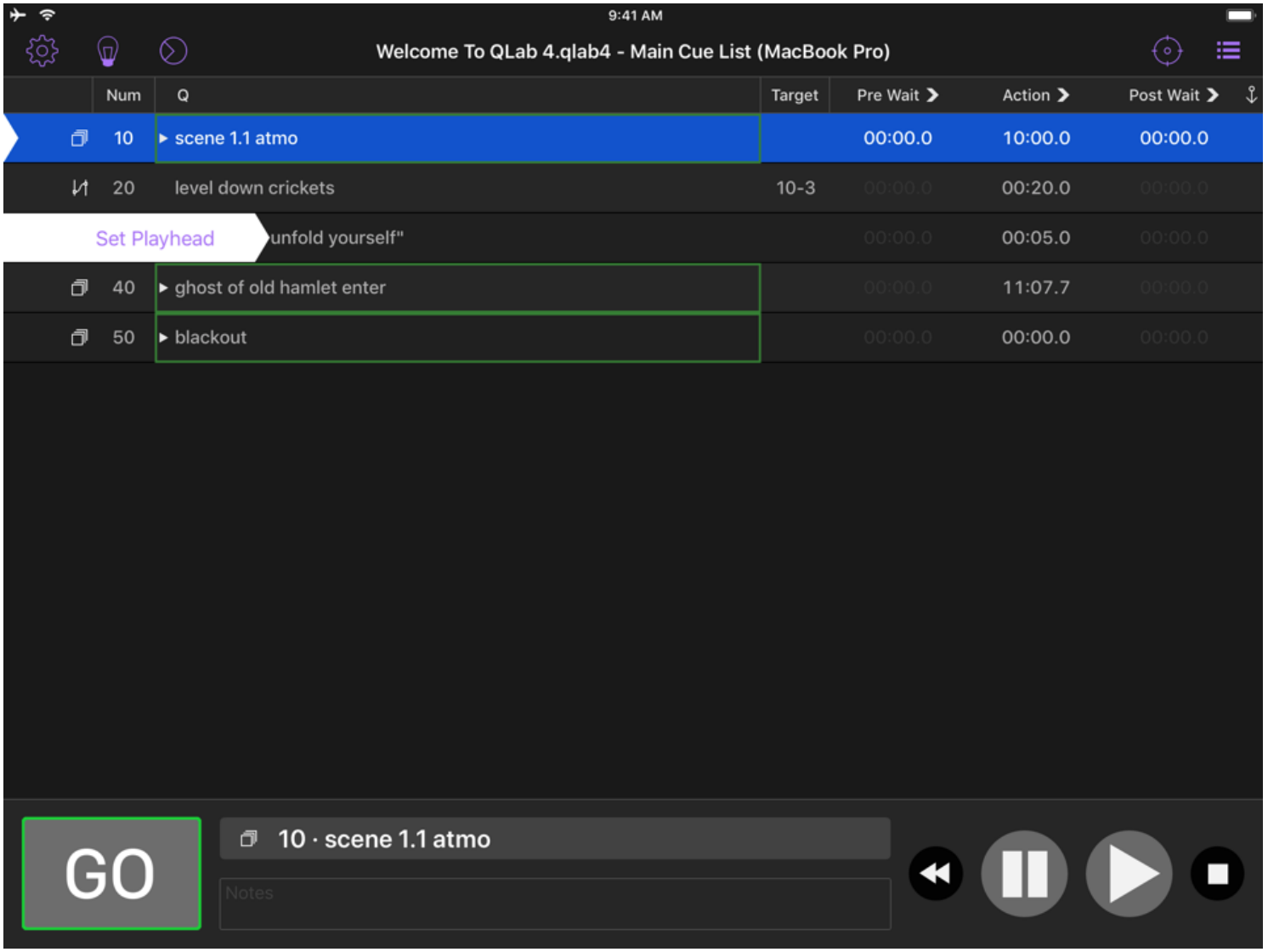
QLab Remote will automatically "see" any workspaces open on your Mac via Bonjour, and you can simply tap to connect. If you have a unusual network setup, you can tap "Other" to directly enter the IP address of the Mac running QLab.

A Tour Of QLab Remote

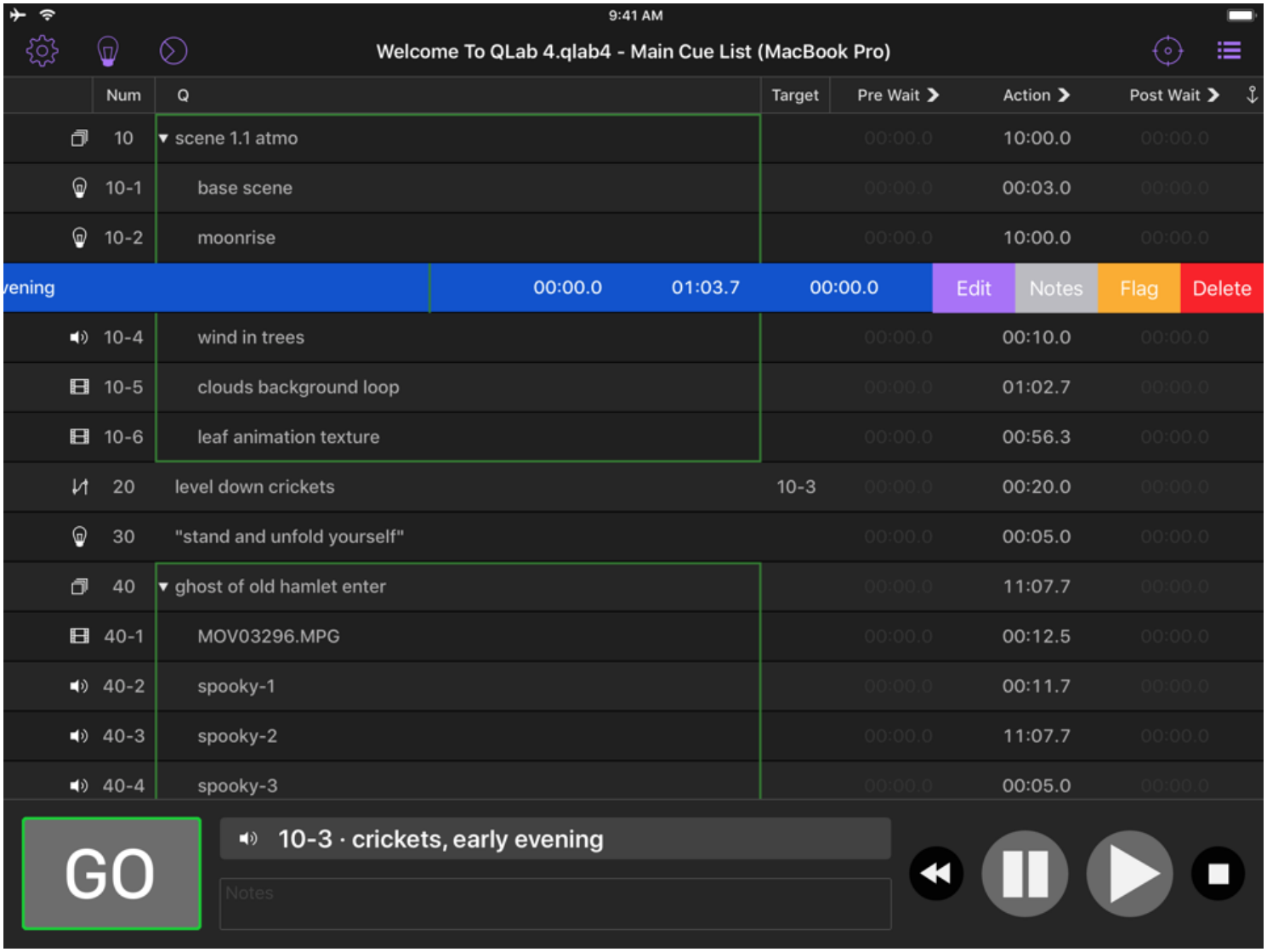


The bottom of the main display shows the familiar GO button, standby indicator, notes field, and transport buttons (reset, pause all, resume all, stop all) from QLab.

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:

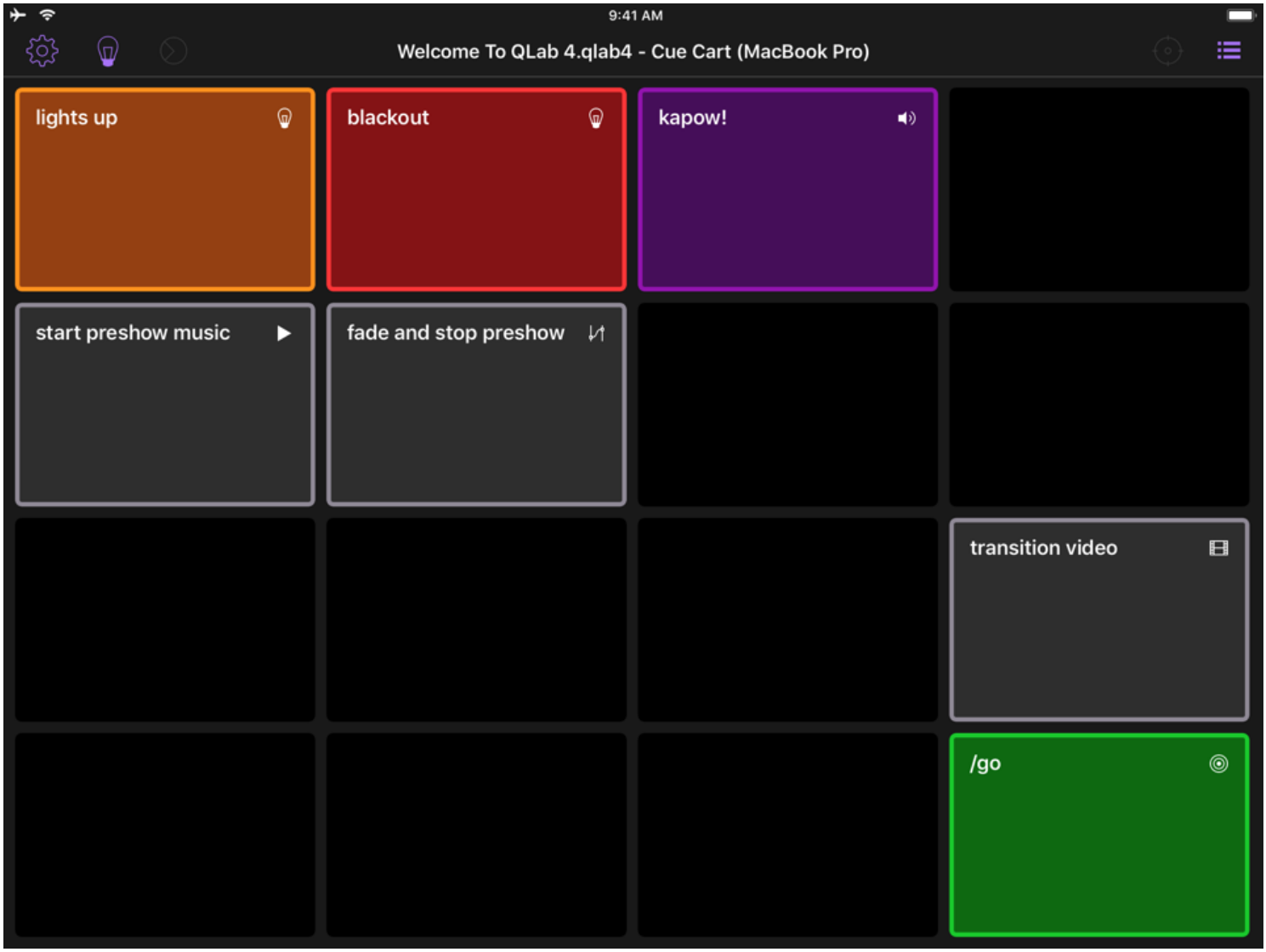


You can also double-tap on a cue to edit that cue.

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:

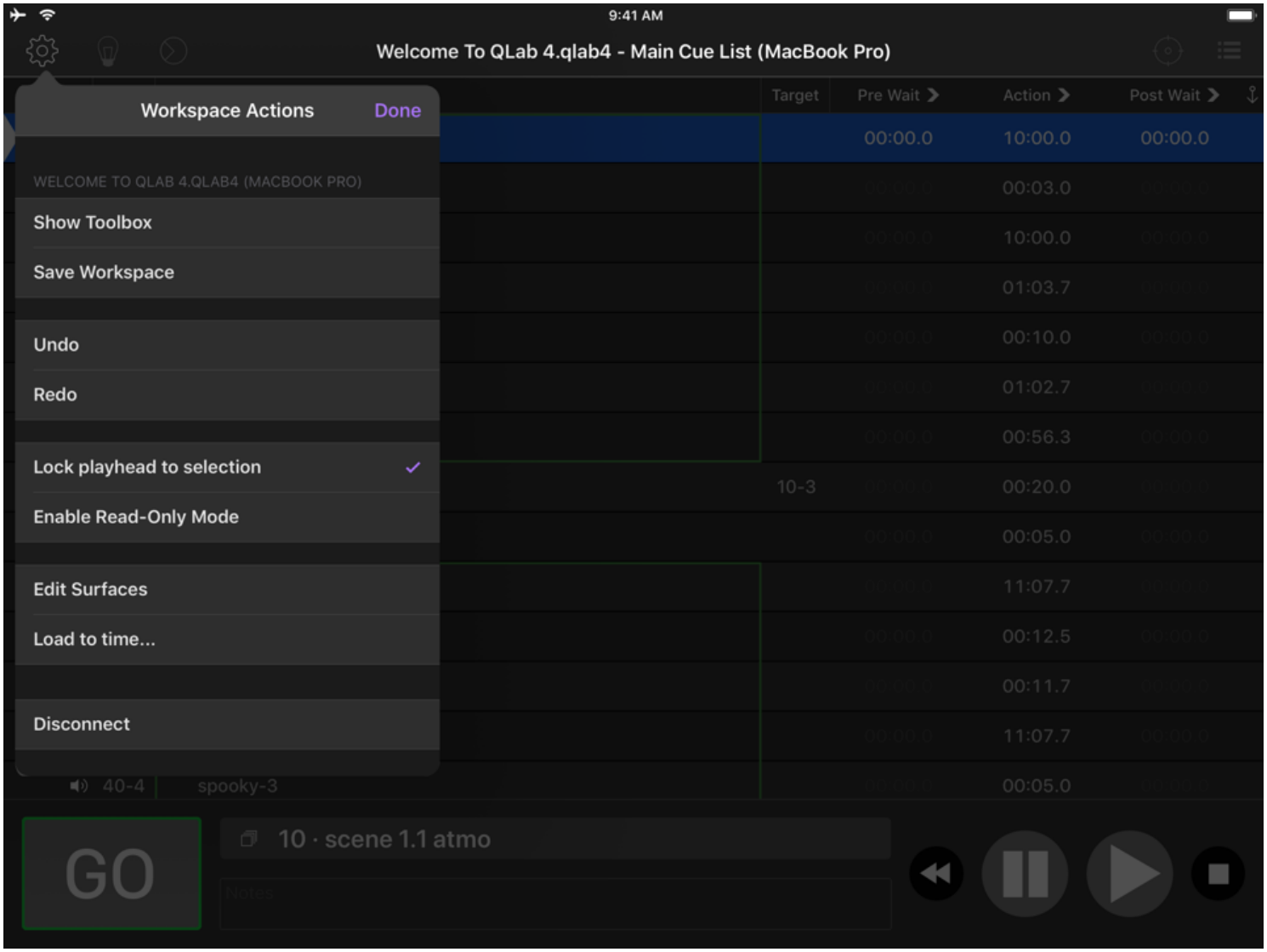
- **Tap** on a cue to trigger it.
- **Two-finger double-tap** on a cue to edit it.
- **Two-finger long-press-and-hold** on a cue to drag it to a new cell in the grid.
- **Pinch** to zoom in or out on the cart view.

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

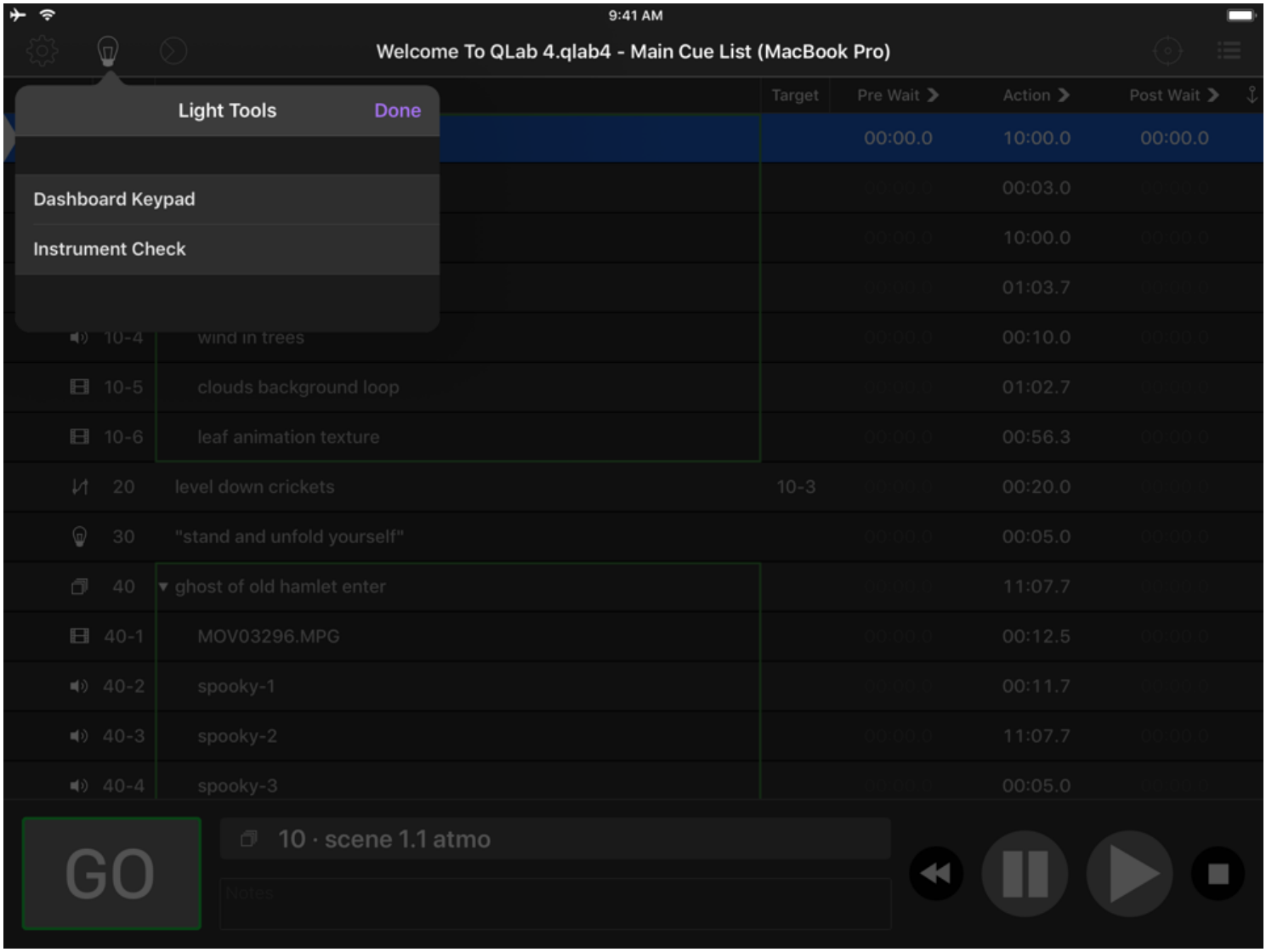
Tapping on the Workspace Actions menu (⚙) reveals nine actions:



- **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 setting from QLab Remote. You can learn more about the setting from the General section of the page on Workspace Settings in this documentation.
- **Enable/disable Read-Only Mode.** Switching off Read-only mode requires the *Show Control & Editing* in-app purchase, as discussed above.
- **Edit Surfaces.** If your workspaces has any video surfaces, and you have a Pro Video or Pro Bundle license installed, you can make basic edits to surface 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

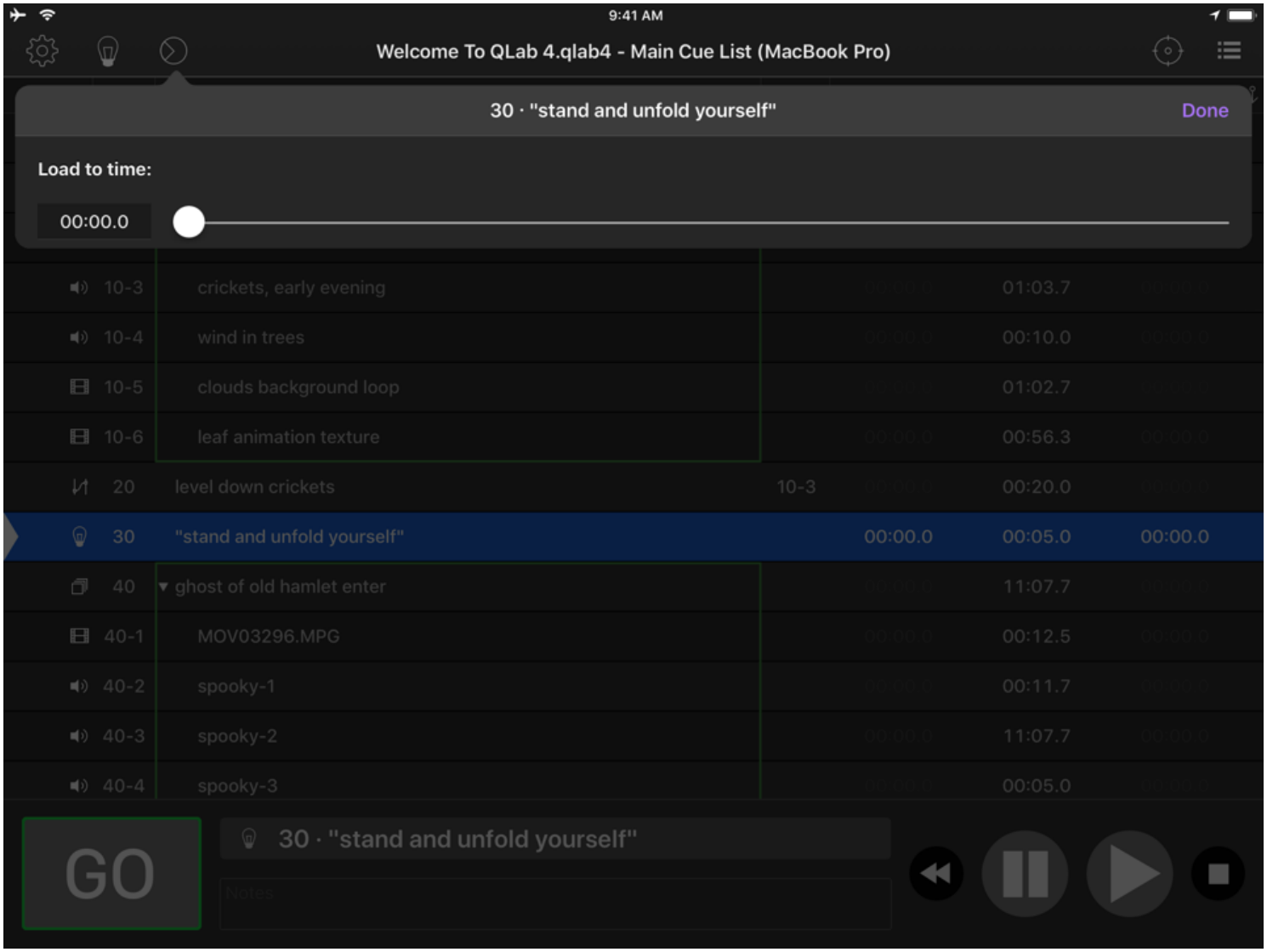
Tapping on the Workspace Actions menu ( ) reveals two actions:



- **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.

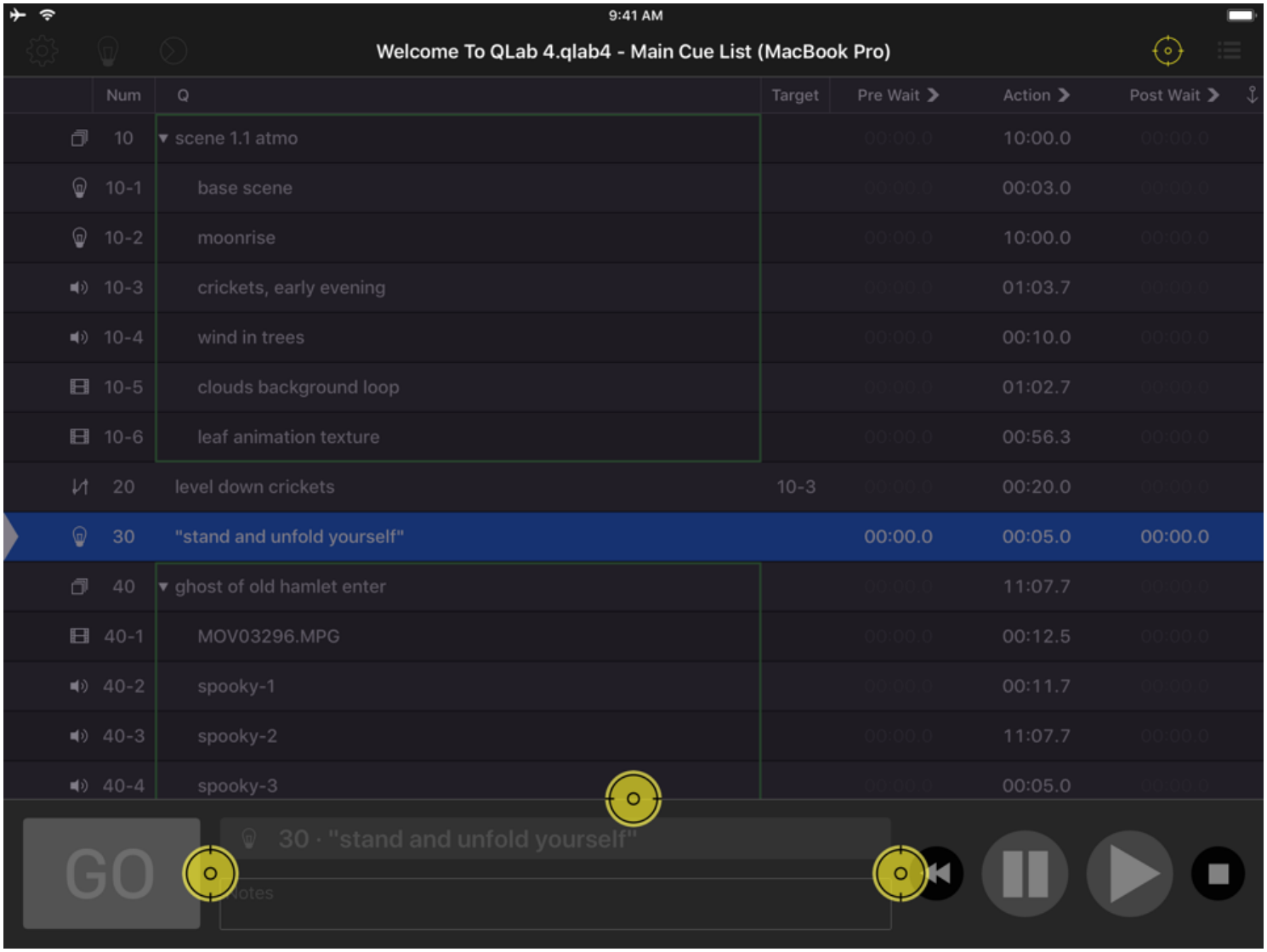
Load To Time

The Load To Time (⏲) tool works just like the Load To Time tool in QLab.



Resize Controls

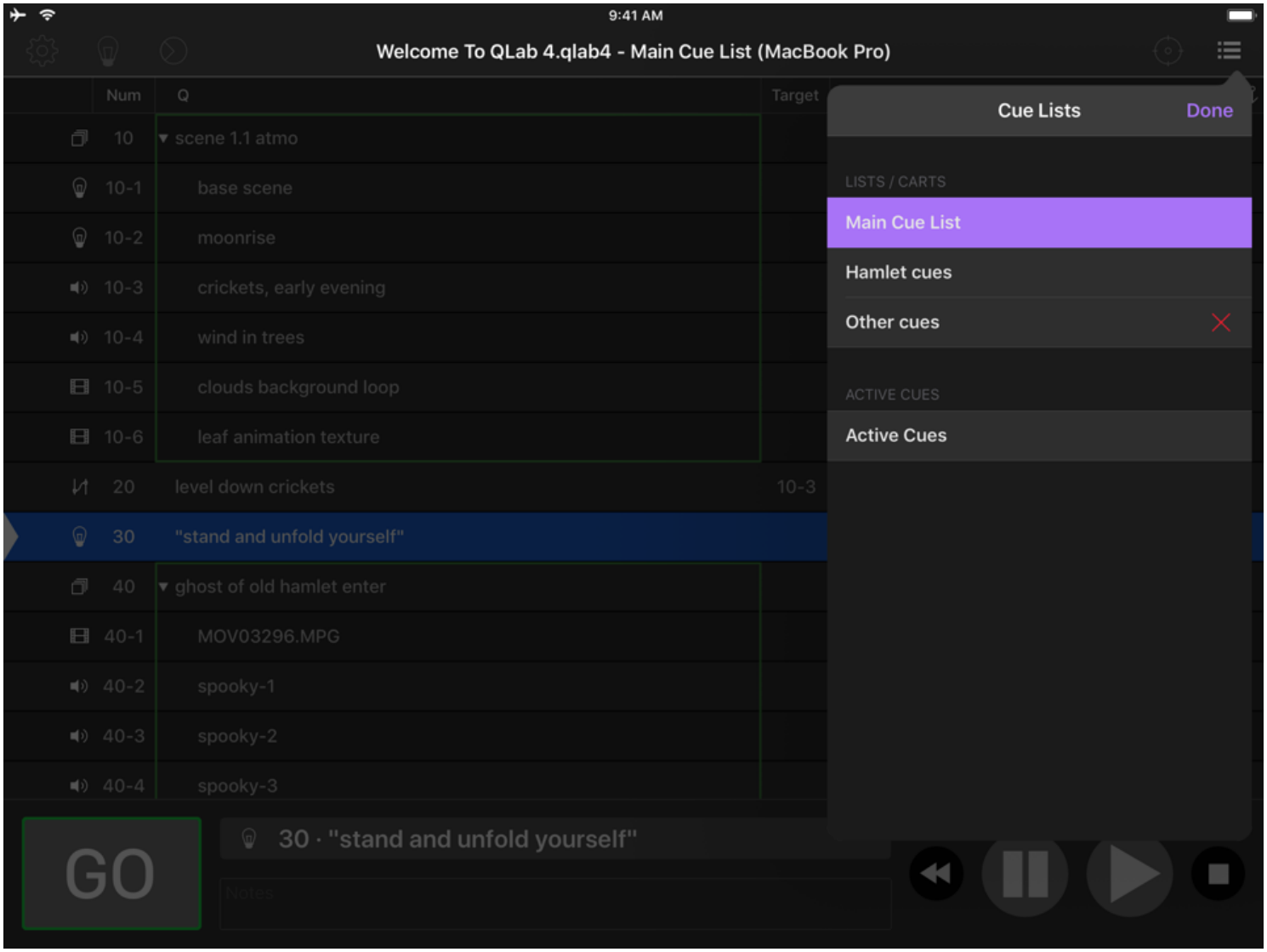
The Resize Controls (⌖) button allows you to resize the footer, the GO button, and the transport controls.



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

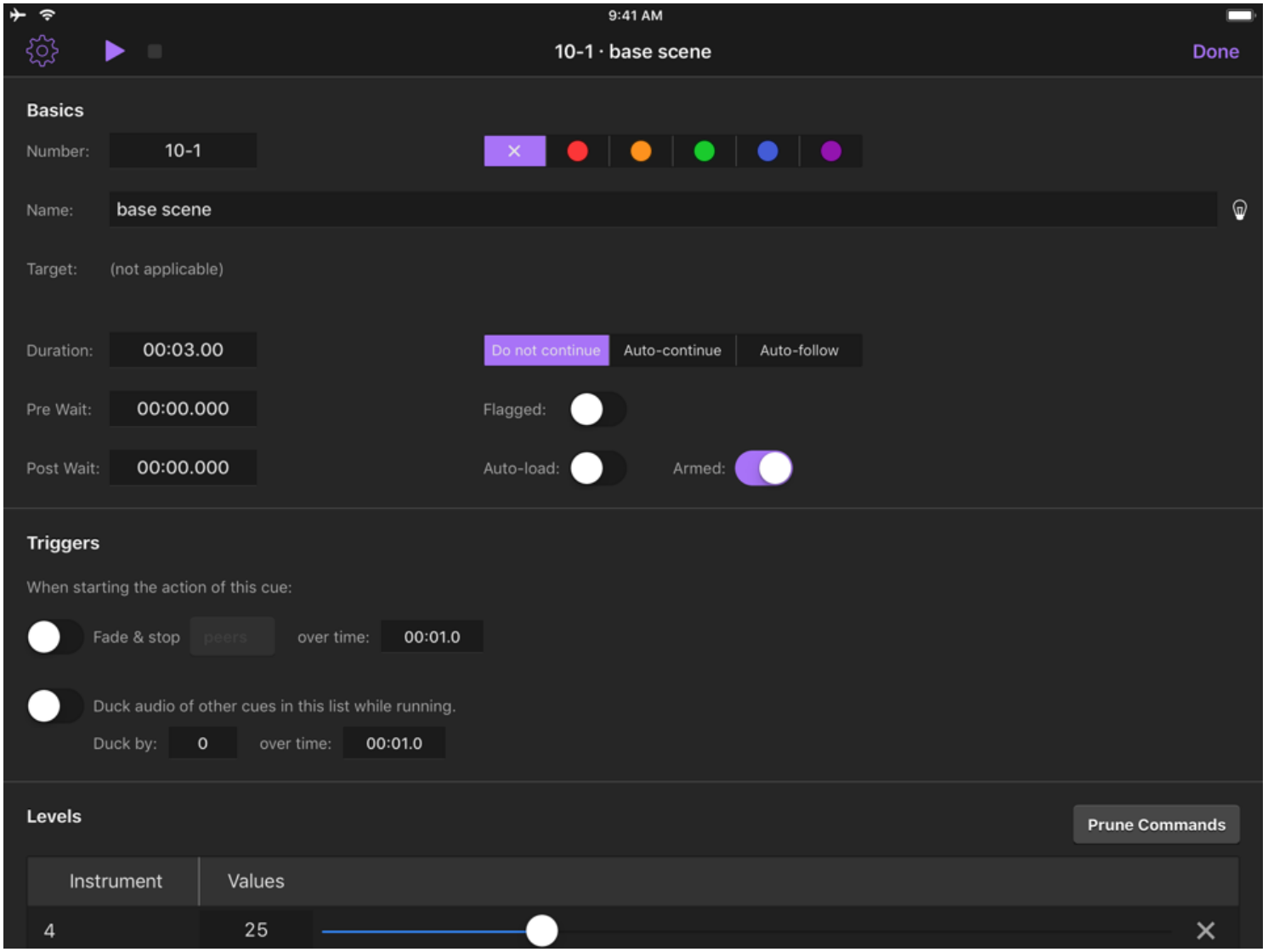
The Cue Lists () menu allows you to switch between the lists and carts in your workspace.



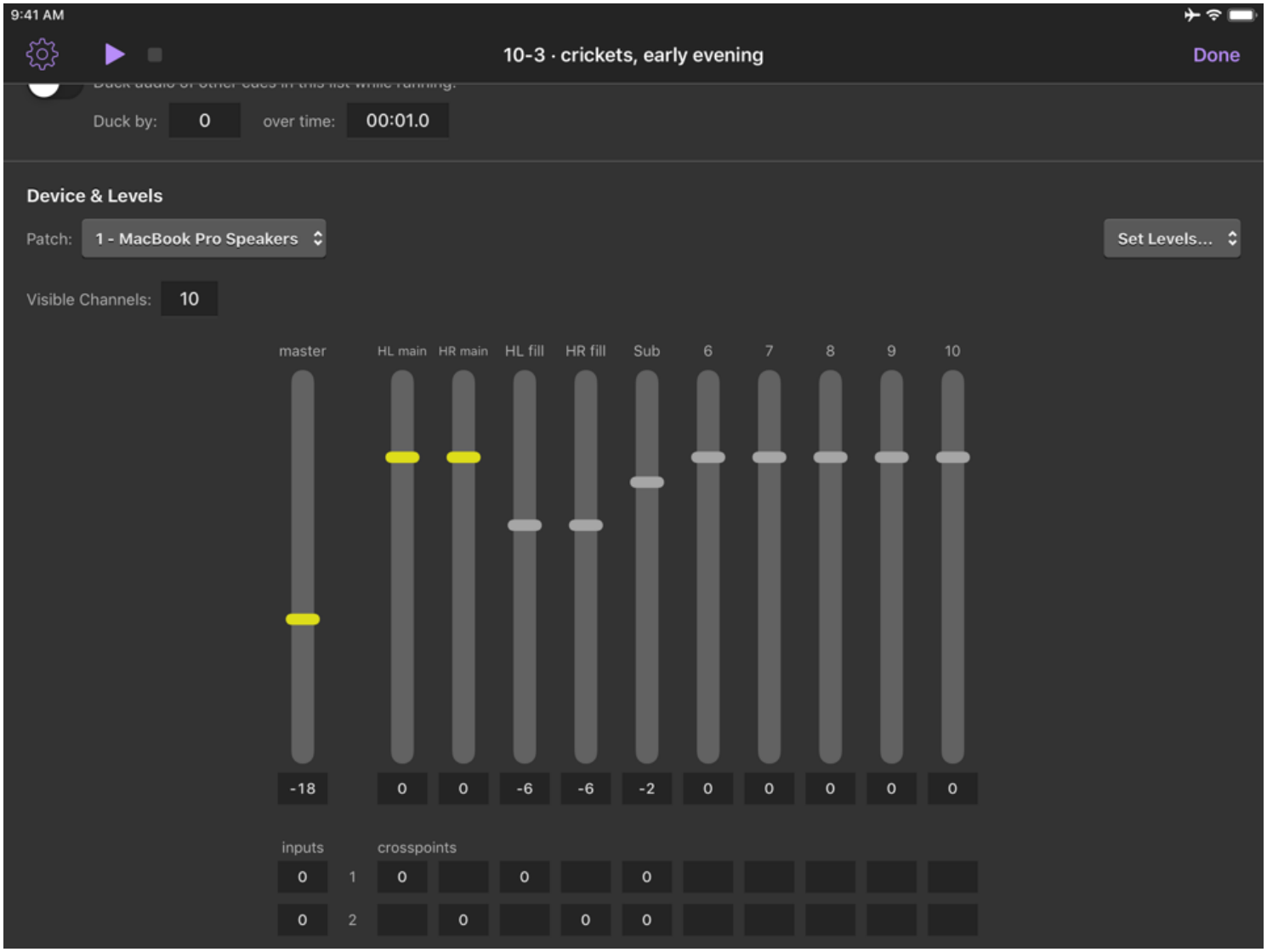
The last item in the Cue Lists menu is Active Cues, which is similar to the Active Cues display in QLab's sidebar.

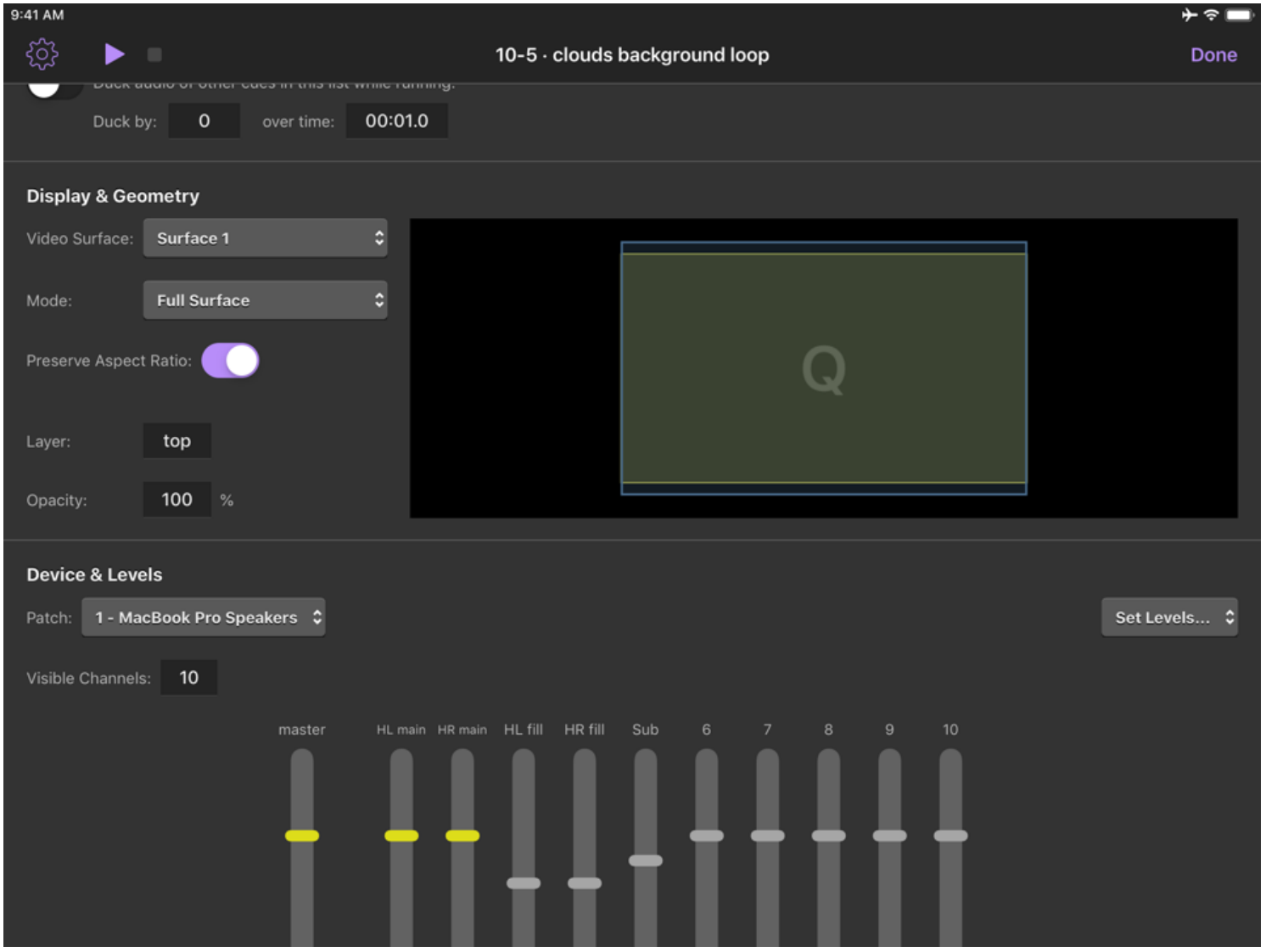
Editing Cues

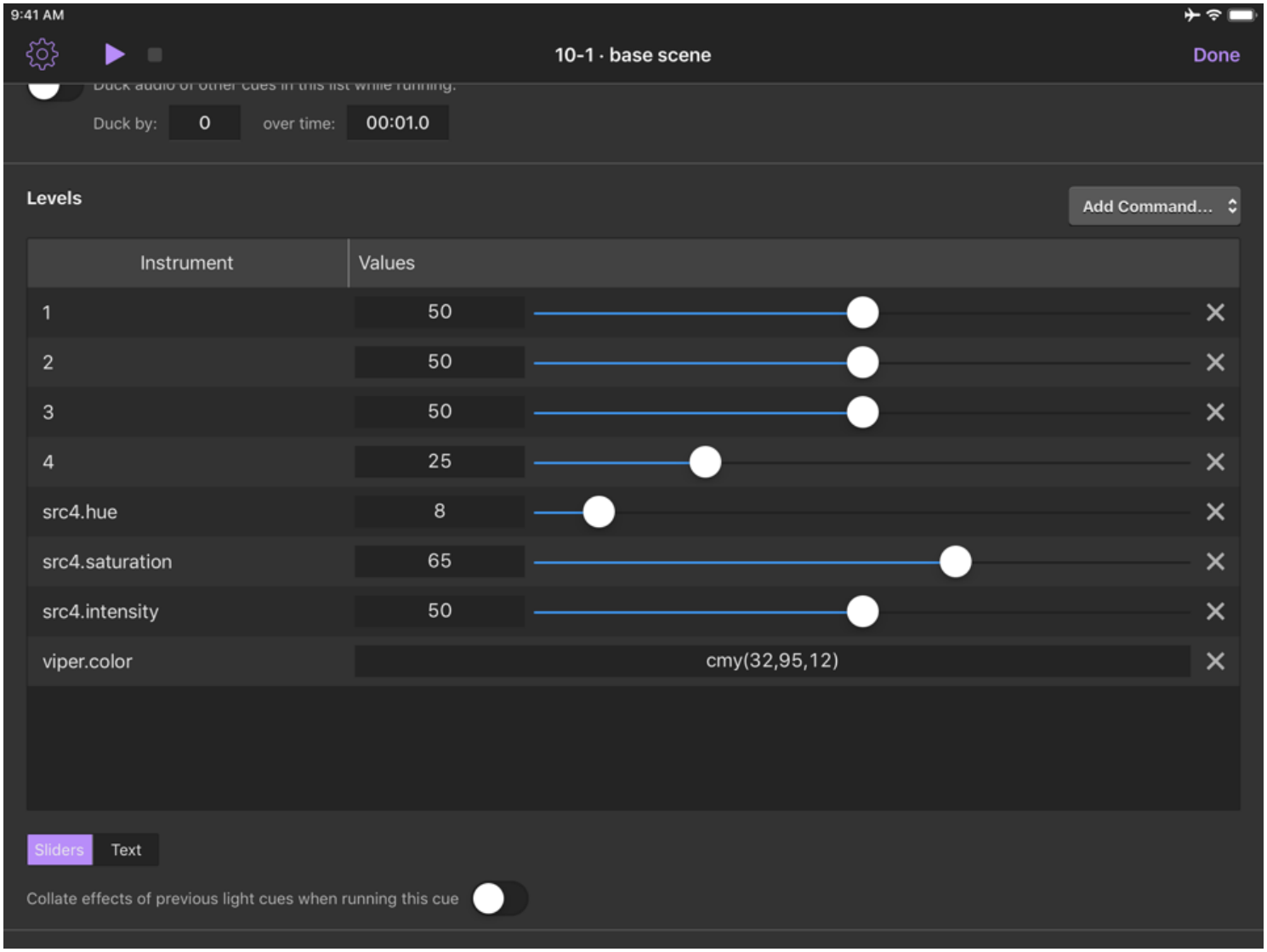
QLab Remote can edit some attributes of all cues:



You can also make some adjustments to Audio cues, Video cues, and Light cues:

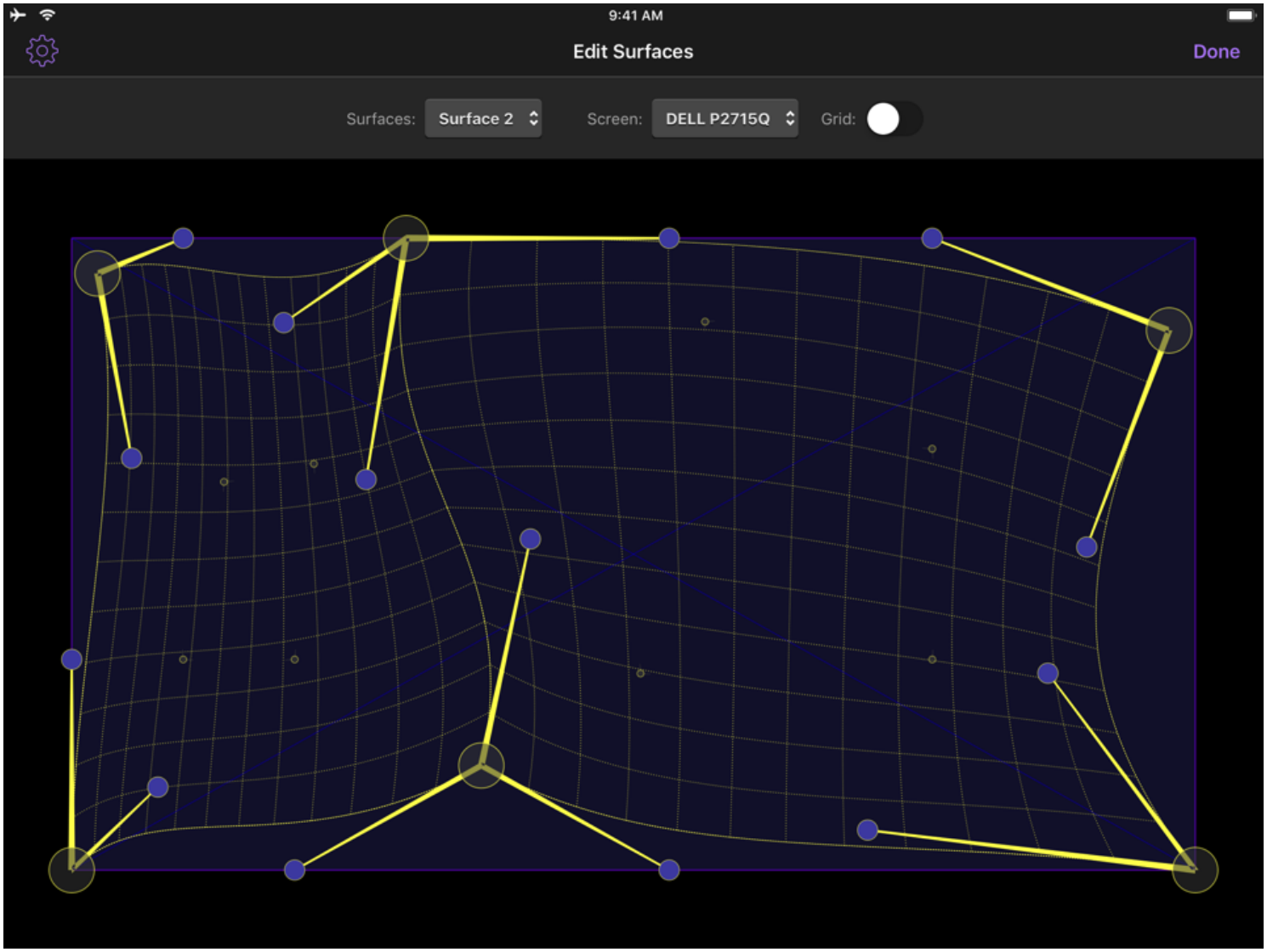

9:41 AM
10-3 · crickets, early evening
Done
Duck by: 0 over time: 00:01.0
Device & Levels
Patch: 1 - MacBook Pro Speakers
Set Levels...
Visible Channels: 10
master
HL main
HR main
HL fill
HR fill
Sub
6
7
8
9
10
-18
0
0
-6
-6
-2
0
0
0
0
0
inputs
crosspoints
0
1
0
0
0
0
2
0
0
0


9:41 AM
10-5 · clouds background loop
Done
Duck by: 0
over time: 00:01.0
Display & Geometry
Video Surface: Surface 1
Mode: Full Surface
Preserve Aspect Ratio:
Layer: top
Opacity: 100 %
Q
Device & Levels
Patch: 1 - MacBook Pro Speakers
Set Levels...
Visible Channels: 10
master
HL main
HR main
HL fill
HR fill
Sub
6
7
8
9
10



Editing Video surfaces

If you have Pro Video or Pro Bundle license installed in QLab, QLab Remote can make basic edits to the geometry of surfaces in your workspace.



You will need to set up the surface 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

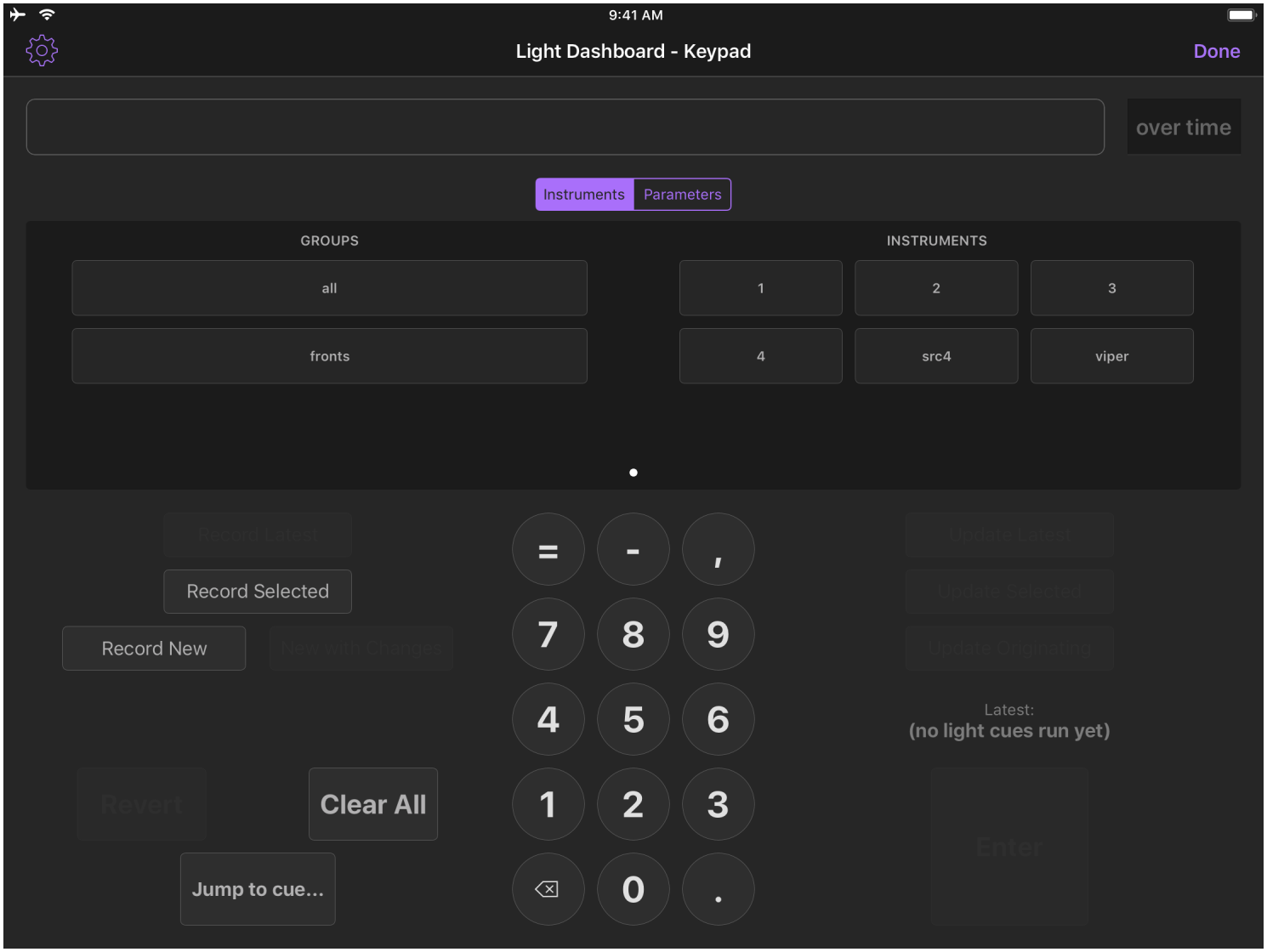
You can tap on the command line and enter text using the regular iOS keyboard (or a physical keyboard connected to your iOS device.) This command line behaves exactly like the the command line in the Light Dashboard. Any text you enter here will be sent to the Dashboard when you press enter.

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

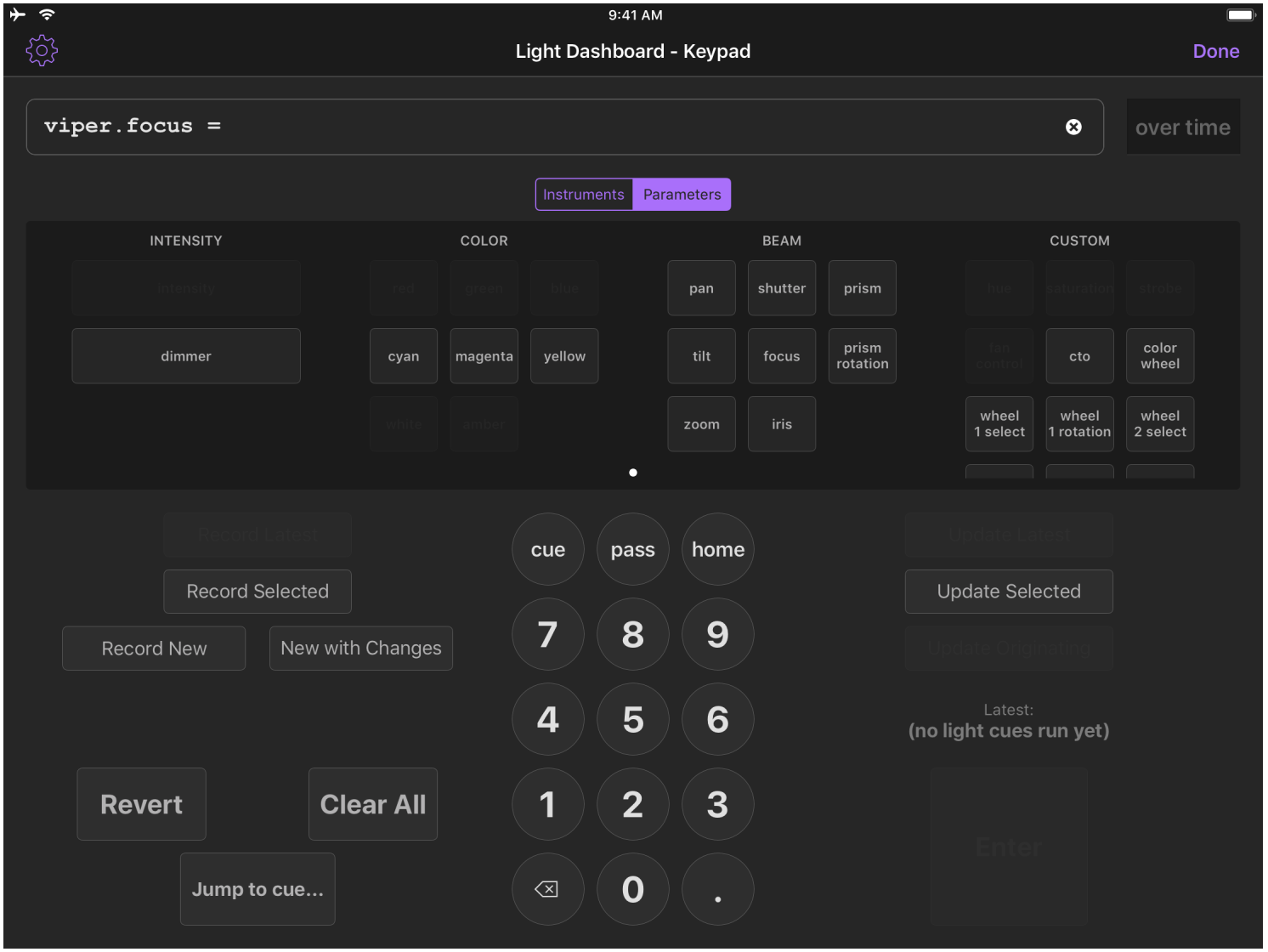
This tabbed area allows you to swiftly select an instrument or light group, then select the parameter you wish to edit. When the Light Keypad first opens, you'll be presented with the Instruments tab:



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.

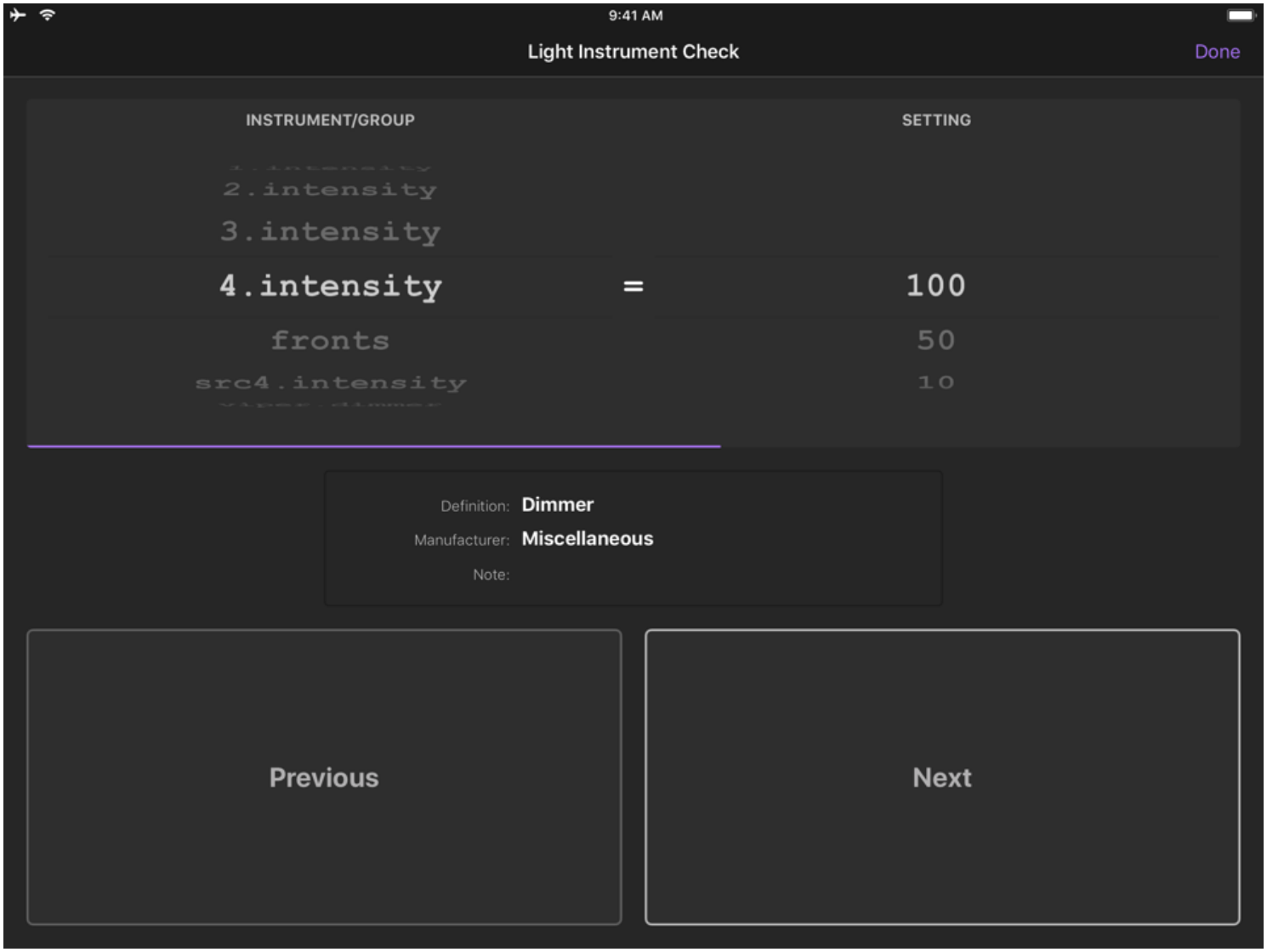
Each button is only enabled when the corresponding action is available.

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.

Requirements

The following features in QLab Remote are available when connected to a specific minimum version of QLab.

QLab 4.0 or newer:

- 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

QLab 4.1 or newer:

- GO in QLab Remote is covered by workspace double-go protection
- QLab Remote respects Show/Edit mode

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 4.3 or newer:

- Browsing folder aliases in the File Target document picker of the cue inspector

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 4.6 or newer:

- Options for Second Triggers in the Triggers inspector panel
- Display audio output names in the Device & Levels inspector panel

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 already in place, or easy to implement if not.

Understanding OSC

All software or devices which support OSC have their own dictionary of commands. You can use programs like Max/MSP, Medialon Manager, and TouchOSC, or hardware like ETC's EOS family to send messages that exist in QLab 4'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 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 commands* 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.

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.

/cue/playhead - the command will target the cue that's currently standing by.

/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.

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`.

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, whereas 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: 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 need 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.

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.

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.

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.

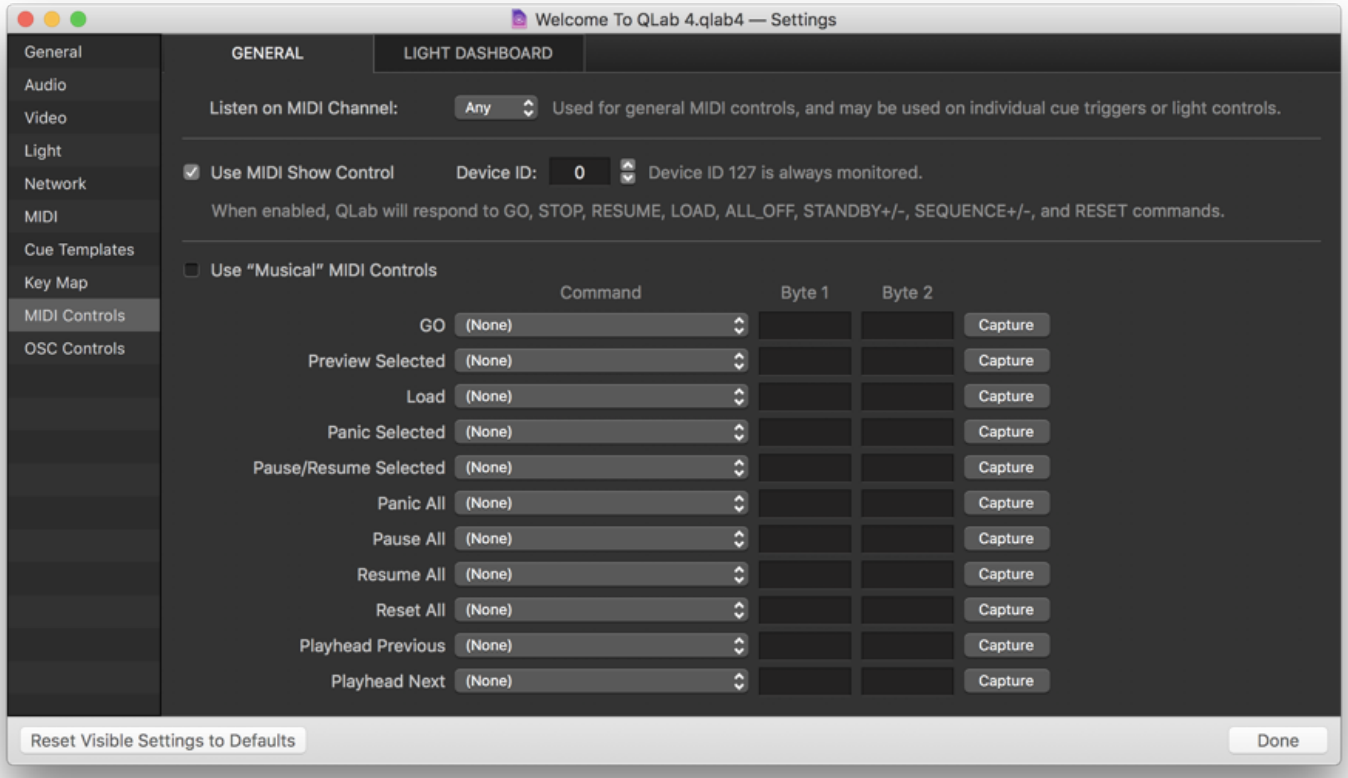
Using MIDI

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.

To configure QLab to respond to MIDI Voice Messages, open the Workspace Settings window by clicking the gear icon in the lower right corner of the workspace, or by using the keyboard shortcut ⌘**,** (that's command-comma) and then select *MIDI Controls* from the list on the left.



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 trigger GO, and you're using a velocity-sensitive keyboard, you may wish to enter `>0` for byte 2, so that no matter the velocity of the keypress, the GO is triggered.

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 trigger individual cues in your workspace. You can learn more about that in the Triggers Tab section of the page on the Inspector.

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 triggers 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 triggered. The only way to ensure that certain cues trigger over MSC and others don't is to use matching cue numbers for the trigger cues, and make sure that no other cues in QLab 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 triggers to the lighting console, since QLab only sends triggers 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.

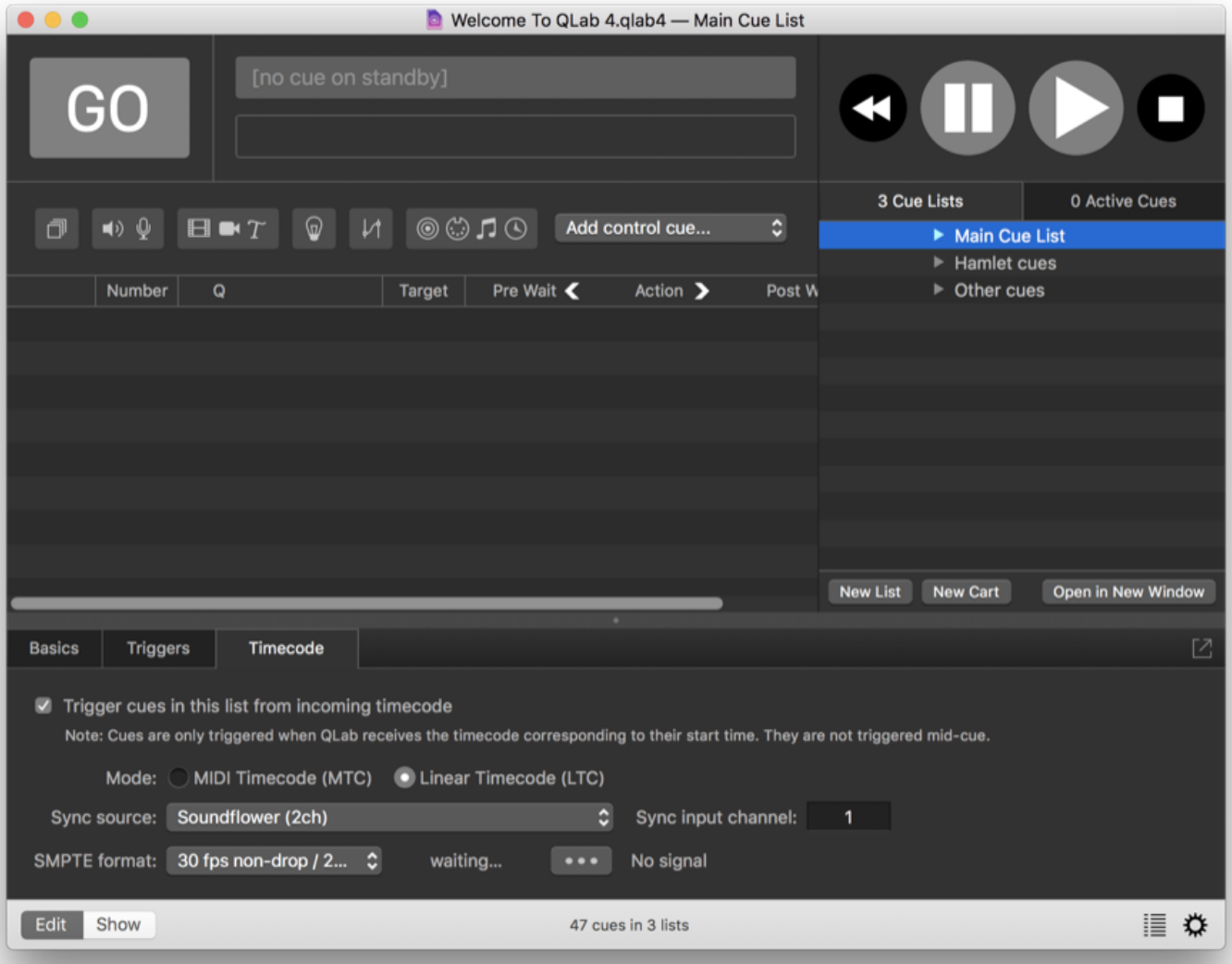
Using Timecode

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 (occasionally) theaters as a mechanism for linking lighting, audio, video, and automation equipment.

QLab can be triggered by either LTC (linear or longitudinal timecode) or MTC (MIDI timecode). It's important to understand that QLab does not sync to, or "chase", timecode, it only *triggers* off of timecode.

Controlling Individual Cues

To trigger cues off of timecode, you first need to enable incoming timecode for the cue list that contains those cues. To do this, select the cue list in the sidebar, and navigate to the Timecode tab in the inspector.



Check the box labeled *Trigger cues in this list from incoming timecode*, and then choose the *mode*, *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.

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 drop-down to *Timecode* and enter the time in the format `reel:minute:second:frame`, or set the drop-down 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 never changes. 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.

The main reason why you might want to use timecode is that lots and lots of legacy equipment supports it.

The Perils of Timecode

Timecode is a bit at odds with the design premise of QLab, which is that you don't necessarily know the amount of time that will elapse between cues. After all, that's the whole point of having cues in the first place! Using timecode locks a series of events in place. Obviously, that's sometimes quite useful.

We recommend using timecode when 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.

Network Cues

Network cues allow QLab to send messages to other software or devices using the network connections of your Mac. They can also be used to send control messages to QLab itself, and beginning with QLab 4.2, they can also be used to control the d&b Soundscape.

At present, 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 successor to MIDI.
- **QLab messages**, which use a subset of OSC to send QLab-specific commands.
- **UDP messages**, which send plain text as UDP packets.

When a Network cue is selected, three tabs will appear in the Inspector:

- Basics
- Triggers
- Settings

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Settings

The controls in the Settings tab can vary widely.

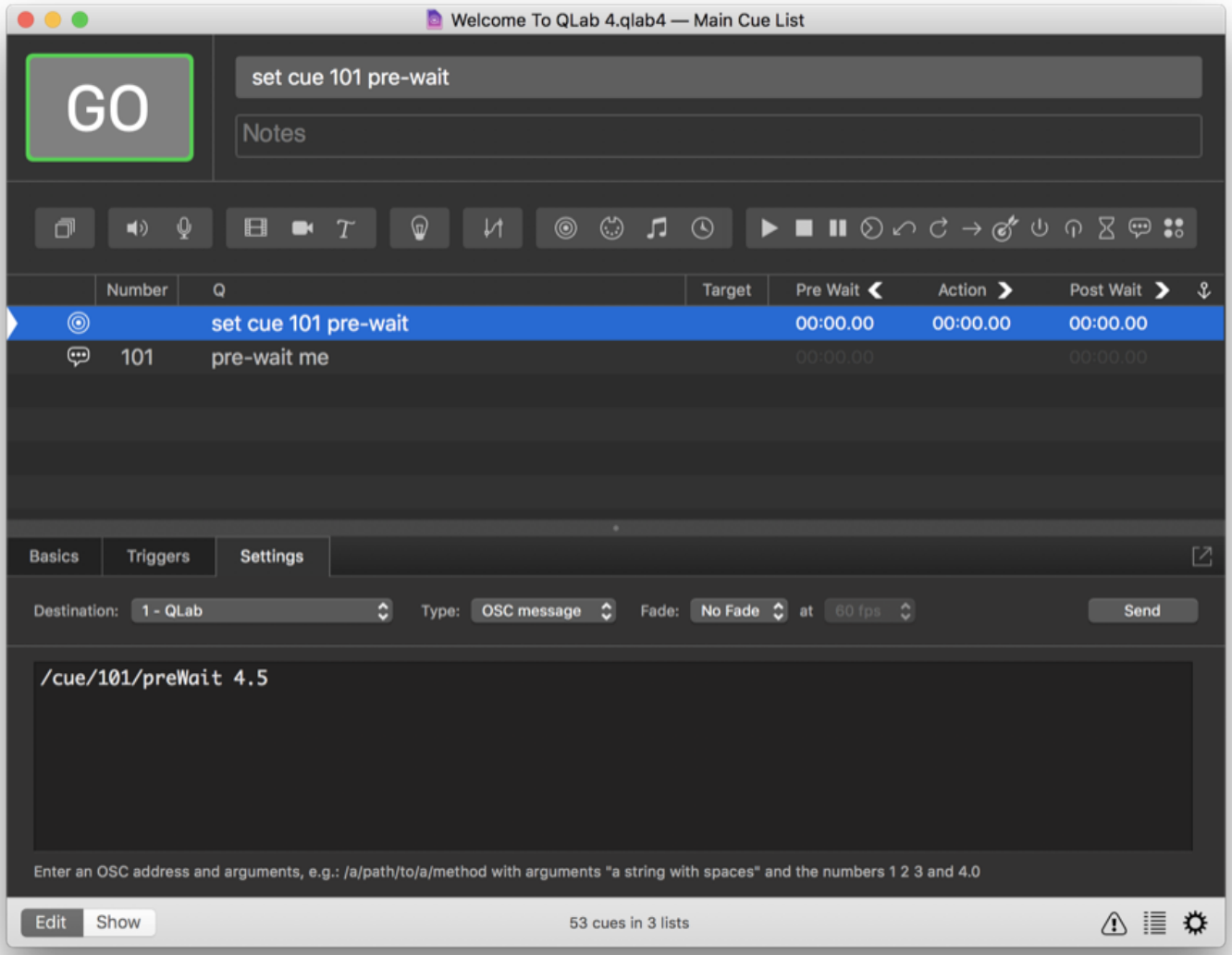
Destination. This drop-down menu shows the sixteen network patches, or destinations, to which QLab can send messages. You can edit this list in Network Settings. If the selected network patch is set to communicate with a Soundscape DS100, the rest of the Settings tab shows DS100-specific controls, described below.

Type. This menu selects which type of message the cue will transmit; an OSC message, a QLab message, or a UDP message.

Send. Click this button to test-send your message.

OSC Messages

If the Network cue is set to send an OSC message, additional menus appear to the right of the Type menu.



The text field allows to enter any single OSC command. For example, to tell a remote copy of QLab to GO, enter the command `/go`. Multiple arguments in a custom OSC message are separated by spaces, like this:

```
/my/groovy/message 2 10 12
```

That sends a message to address `/my/groovy/message` with three integers, `2`, `10`, and `12`, as separate arguments.

QLab supports integers, floats, and strings as outgoing OSC arguments. If you use floats, please remember that QLab is locale-aware. That is to say, if your Mac uses a comma as the decimal separator, your OSC messages should too.

Fade

The Fade drop-down menu lets you set the Network cue to send the message just once, to send it repeatedly over time, or to perform a one-dimensional or two-dimensional fade to send more complex messages over time.

No fade

This option will transmit the OSC message exactly as you entered it, right when the cue is triggered. If you give the Network cue a duration, the message will be sent repeatedly for the duration of the cue. The frequency with which the message repeats is selected using the next drop-down menu. The options range from once per second to 120 times per second.

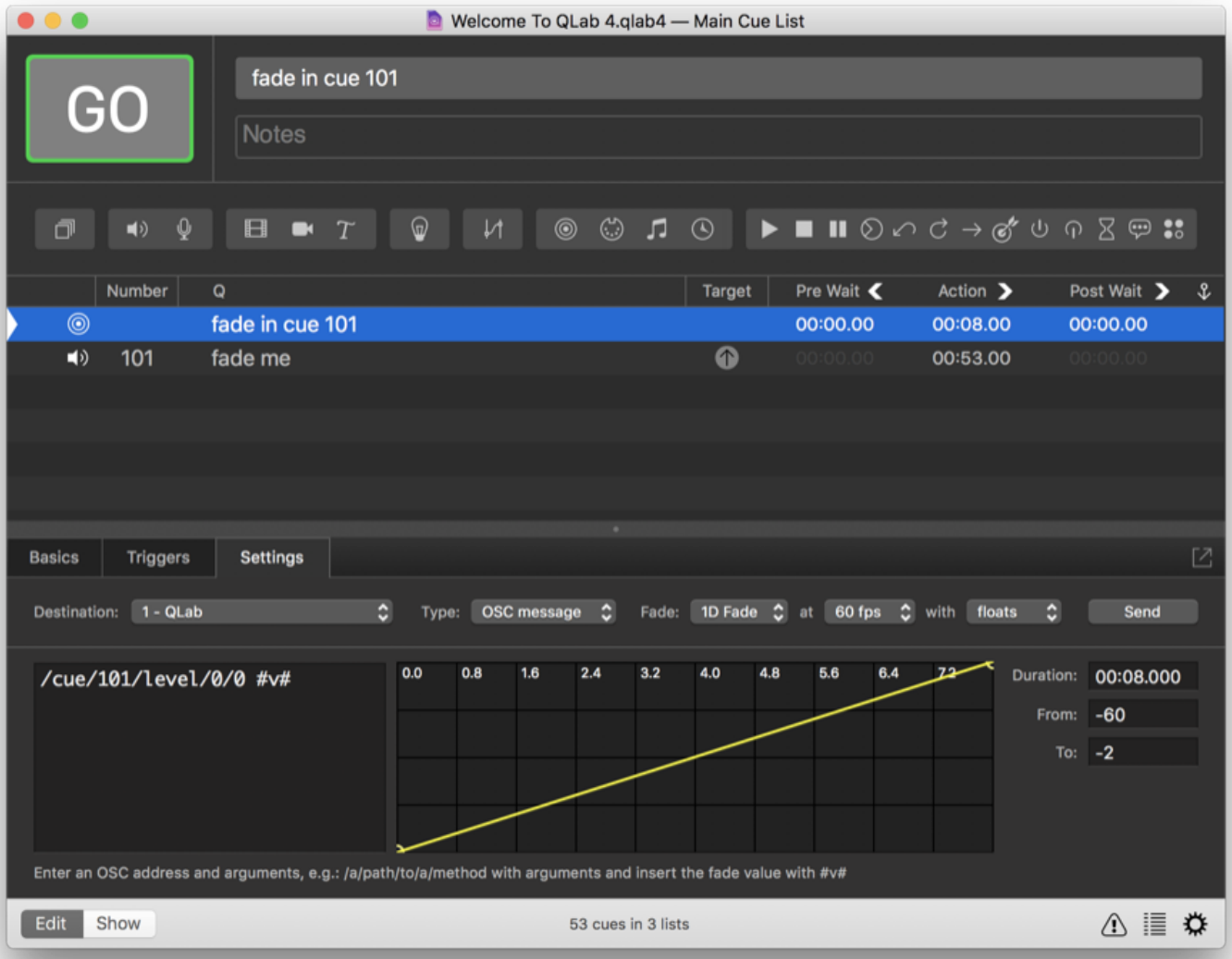
1D fade

This option will allow you to set a starting value and ending value, and have QLab send a series of OSC messages interpolating between them. You need to give the Network cue a duration for this to work properly, and once you do you'll be able to set the frequency with which messages are transmitted, and whether you want QLab to interpolate between the starting and ending values using floating point numbers (numbers with decimal values) or integers (whole numbers.)

Enter your OSC message in the text field, and use the token `#v#` to represent the value which will be faded.

- **Duration** sets the length of the fade, just like a Fade cue. You can also adjust the duration in the Action column of the cue list.
- **From** sets the starting value of `#v#`.
- **To** sets the final value of `#v#`.

You can also adjust the fade curve as needed.



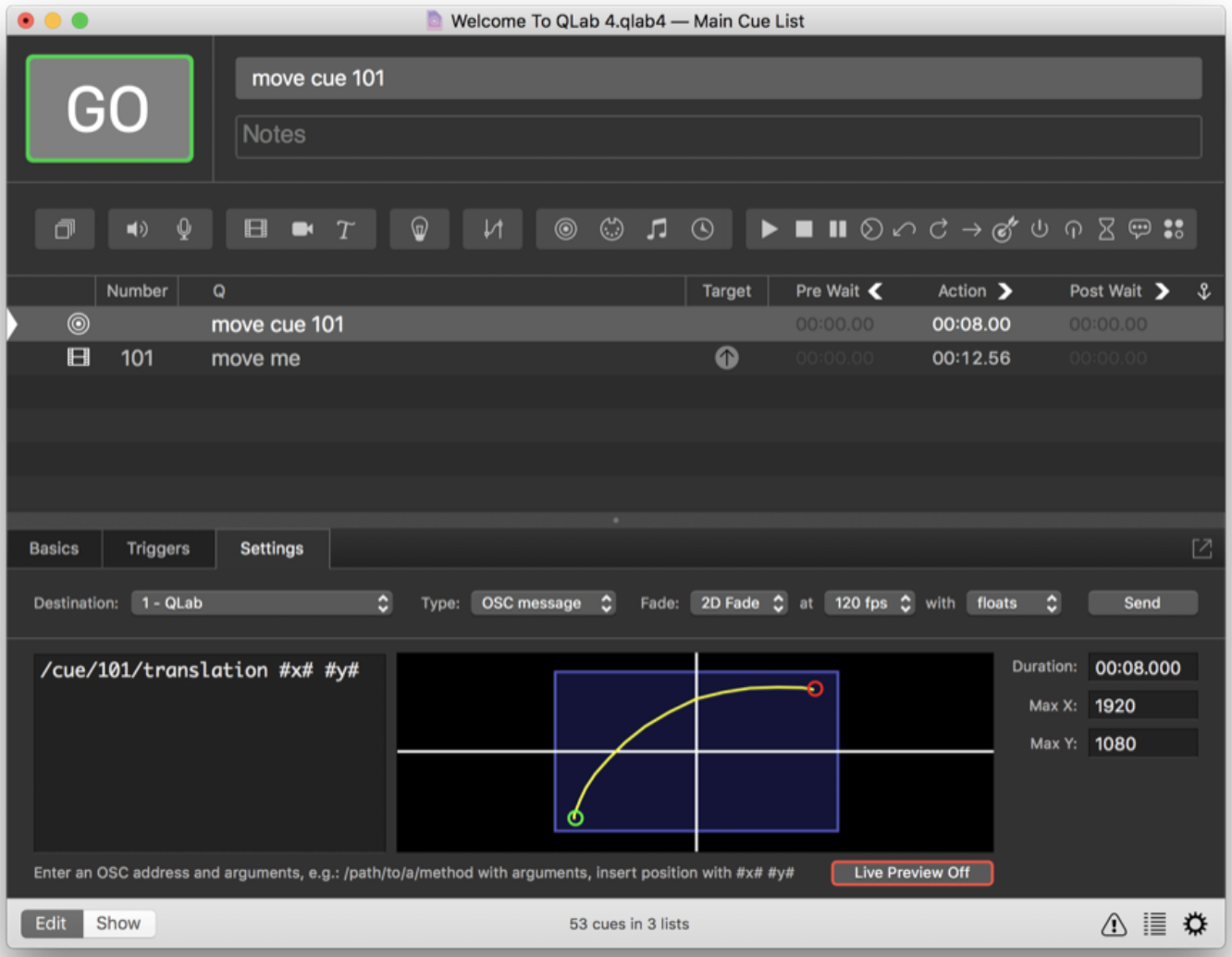
2D fade

If your OSC message has two values you'd like to fade, then *2D Fade* is for you. 2D OSC fades show a blue rectangular area which let you draw a two dimensional path using your mouse or trackpad. To fade a two-value OSC message, enter your OSC message in the text field, and use the tokens `#x#` and `#y#` to represent the two values which will fade.

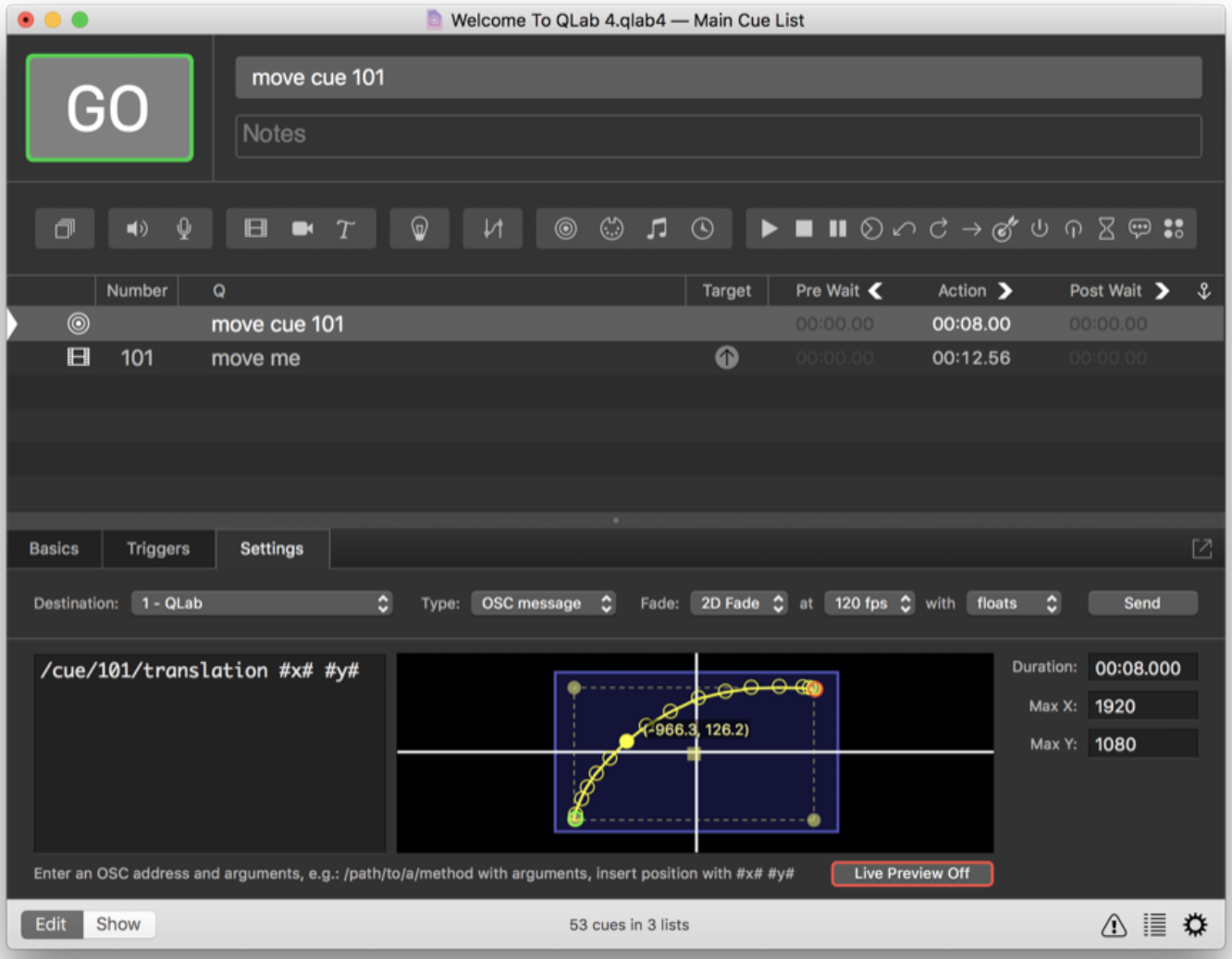
- **Duration** sets the length of the fade, just like a Fade cue.
- **Max X** sets the highest possible value for `#x#`.
- **Max Y** sets the highest possible value for `#y#`.

Then, click and drag to draw a path within the blue rectangular area. A green circle will indicate the starting position of the fade, and a red one will indicate the ending position.

If the **Live Preview** button is set to *On*, QLab will transmit OSC messages as you click and drag so that you can see or hear the results in real time.

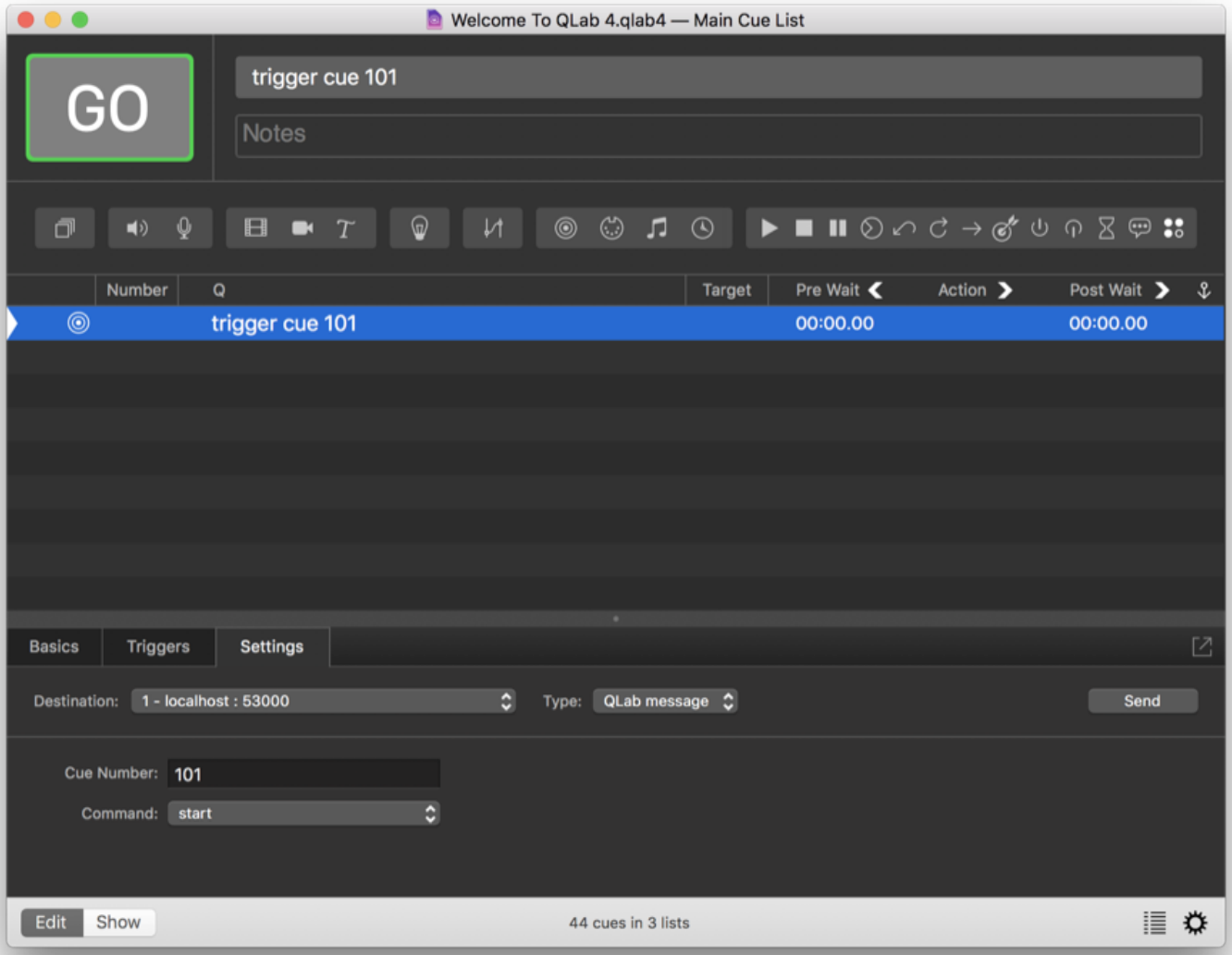


If you hover your mouse over the path, you can click and drag to move, resize, and edit the path. Control points can be moved or deleted, and the whole curve can be moved or scaled. If you make a mistake, just choose *Undo* from the **Edit menu.**



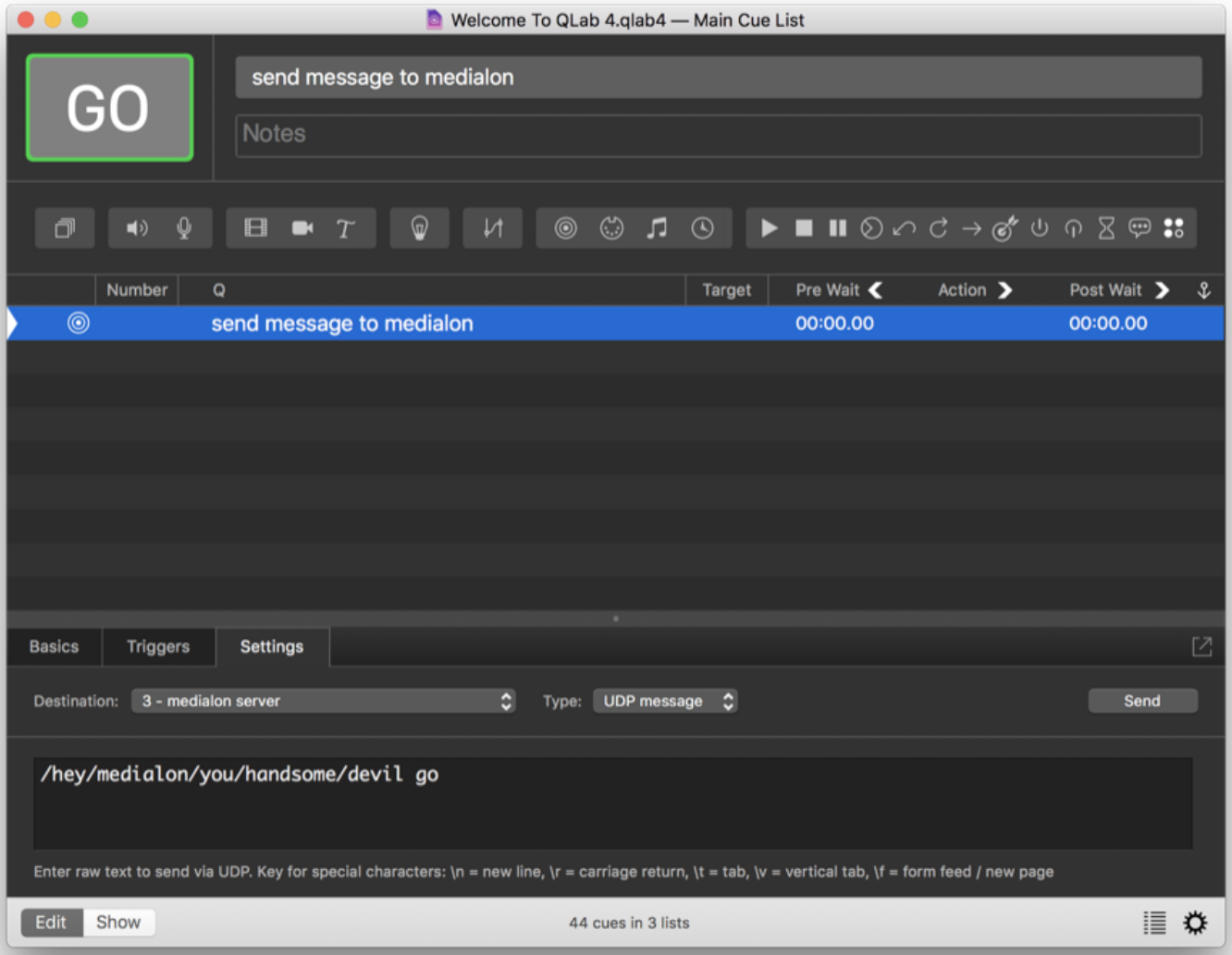
QLab messages

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.



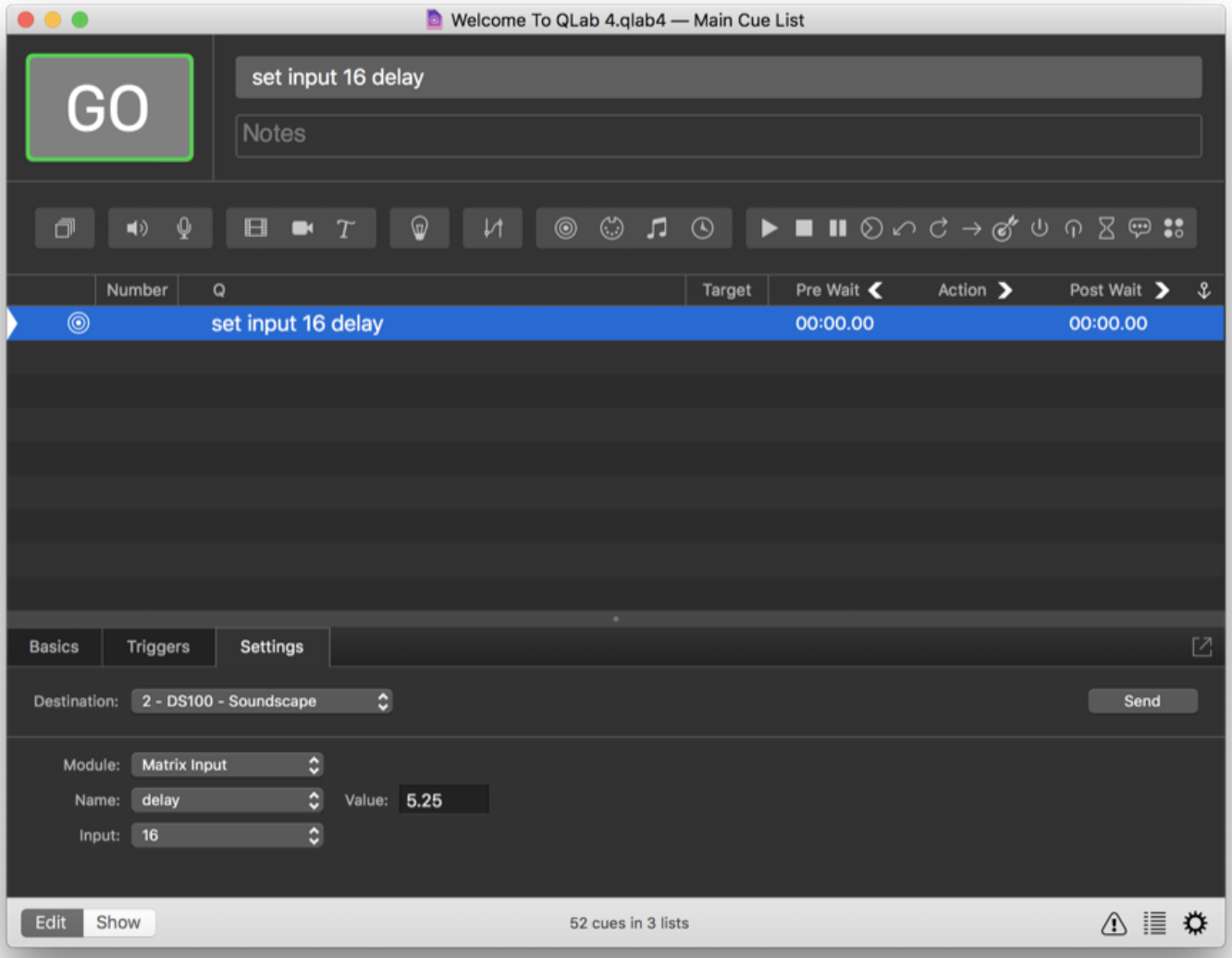
UDP messages

This option allows you to send plaintext strings over UDP. Some software, such as Medialon Manager, prefers to receive commands in this manner.

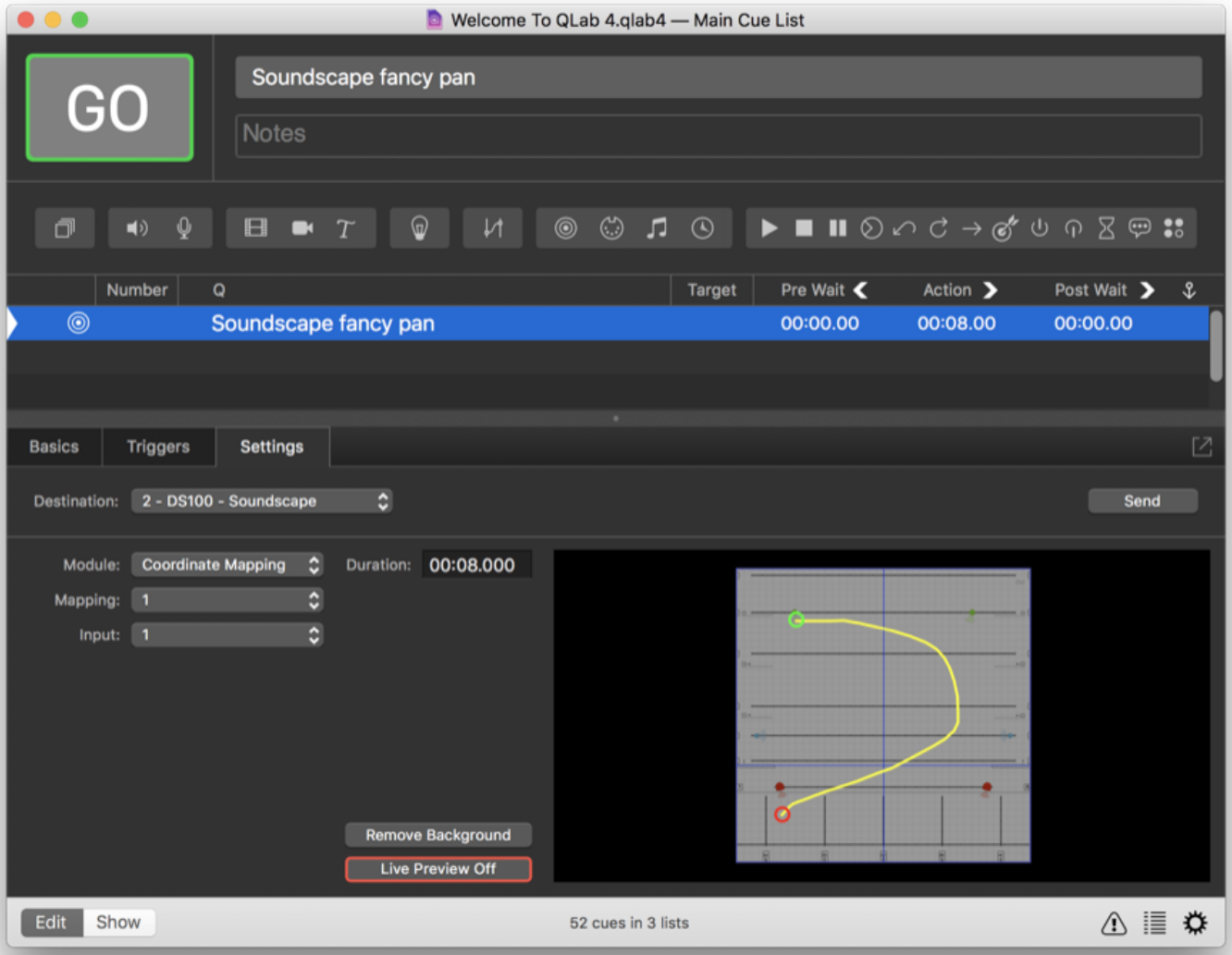


Soundscape

If the Network cue's destination has been configured as a DS100 destination (which can be done in the Network section of Workspace Settings), QLab will display controls specific to controlling a d&b DS100 Soundscape processor. The **Module** drop-down lets you select which control module of the DS100 to adjust, Four modules are supported at this time: Coordinate Mapping, Matrix Input, Matrix Output, and Matrix Node, each with their own associated controls.



The *Coordinate Mapping* module is rather different from the other three, and bears some similarity to the 2D fade discussed above.



One notable difference is that you can drag-and-drop an image into the fade path area and QLab will display it as a background. This can be particularly helpful if you export an image of your coordinate map from d&b's software.

Broken Cues

Network cues can become broken for the following reasons:

Invalid network destination.

Assign a valid network destination in the *Settings* tab of the inspector. You may also need to visit the Network section of Settings and specify a valid destination address and port for the desired patch.

Missing QLab cue number.

Fill in a valid cue number in the *Settings* tab of the inspector. This is only relevant for Network cues set to the *QLab message* type.

Invalid command parameters.

Fill in valid parameters in the *Settings* tab of the inspector. This is only relevant for Network cues set to the *QLab message* type.

No OSC message specified.

Fill in a valid OSC message in the *Settings* tab of the inspector. This is only relevant for Network cues set to the *OSC message* type.

Invalid OSC message.

Fill in a valid OSC message in the *Settings* tab of the inspector. This is only relevant for Network cues set to the *OSC message* type.

No UDP message specified.

Fill in a valid UDP string in the *Settings* tab of the inspector. This is only relevant for Network cues set to the *UDP message* type.

A license is required to reactivate this saved cue.

You'll need to install a license in order to use this cue.

MIDI Cues

MIDI cues allow you to send MIDI voice messages, MIDI Show Control (MSC) messages, or MIDI System Exclusive (SysEx) messages.

When a MIDI cue is selected, three tabs will appear in the Inspector:

- Basics
- Triggers
- Settings

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Settings

MIDI Destination. This drop-down menu shows the eight MIDI patches, or destinations, to which QLab can send MIDI messages. You can edit this list in MIDI Settings.

Message Type. You can choose among MIDI Voice Message, MIDI Show Control Message, and MIDI SysEx Message.

MIDI Voice Message ("Musical MIDI")

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.

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* or *Velocity* underneath the Command drop-down menu, and an ending value, which is the field labeled *Fade to value* or *Fade to velocity* towards the middle of the inspector.

MIDI Show Control Message (MSC)

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.

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 triggering is in the active cue list. Only fill it in if the cue you're triggering 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

Enter your SysEx message here in hexadecimal format. Omit the leading `F0` and `F7`; QLab adds those for you.

Send Message. Click this button to test-send your message.

Broken Cues

MIDI cues can become broken for the following reasons:

No MIDI destination.

Choose a MIDI destination in the *Settings* tab of the inspector. You may also need to visit the MIDI section of Settings and connect a MIDI device to the desired patch.

Illegal characters used in SysEx message.

Edit the SysEx message in the *Settings* tab of the inspector.

Length of SysEx message is invalid.

Edit the SysEx message in the *Settings* tab of the inspector.

A license is required to reactivate this saved cue.

You'll need to install a license in order to use this cue.

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 when synchronizing MIDI-triggered events with a piece of music, for example.

Files supported

The MIDI File cue supports both Type 0 (single-track) and Type 1 (multitrack) standard MIDI files. Type 2 files (an uncommon multiple-sequence format) are not supported.

When a MIDI File cue is selected, three tabs will appear in the Inspector:

- Basics
- Triggers
- Settings

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Settings

MIDI Destination. This drop-down menu shows the eight MIDI patches, or destinations, to which QLab can send MIDI messages. You can edit this list in MIDI File Settings.

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 trigger other cues, or to route to another application running on the same computer.

Playback. 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.

MIDI File cue timings are based on the computer's internal clock, and are not clocked to any audio or video devices.

Broken Cues

MIDI File cues can become broken for the following reasons:

Invalid MIDI destination.

Choose a MIDI destination in the *Settings* tab of the inspector. You may also need to visit the MIDI File section of Settings and connect a MIDI device to the desired patch.

Invalid MIDI File.

Either the file is missing or damaged, or it's not a valid MIDI file.

A license is required to reactivate this saved cue.

You'll need to install a license in order to use this cue.

Timecode Cues

Timecode cues comprise QLab's mechanism for generating outgoing timecode.

Unlike most timecode-enabled applications which have a single, fixed timeline from which timecode can be generated directly, or which can be driven by incoming timecode, QLab allows for multiple independent timelines running simultaneously, since multiple cues can run simultaneously.

When a Timecode cue is selected, three tabs will appear in the Inspector:

- Basics
- Triggers
- Settings

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Settings

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.

LTC (Linear, or Longitudinal, Timecode) is meant to be carried on a line-level audio connection. When LTC is selected, the **Destination** controls allow you to select an audio device and a channel number to output to. 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 have strong high-frequency content which, if accidentally routed out to speakers, can cause serious damage to hearing and to friendships.

MTC (MIDI Timecode) carries most of the same information as LTC, but on a MIDI connection rather than audio. This can be a physical MIDI port, or an IAC bus. No channel number is required, as 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.

Framerate. 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 master clock (via word clock, etc.) as the other machine.

Start time. This control allows you to specify the first frame of timecode that is transmitted when the cue is triggered. Bear in mind that both LTC and MTC can take up to a few frames to transmit enough information to read. If an event needs to be triggered on a specific frame, it is best to start timecode output a few frames ahead, as preroll into that event. This is why the default start time for a Timecode cue is `1:00:00:00` rather than `0:00:00:00`. Because there is no room for preroll before hour 0, the best practice is to treat hour 1 as the beginning of the timeline, with space for preroll beforehand.

Broken Cues

Timecode cues can become broken for the following reasons:

No MIDI destination.

Choose a MIDI destination in the *Settings* tab of the inspector. You may also need to visit the MIDI section of Settings and connect a MIDI device to the desired patch. This is only relevant to Timecode cues set to the *MTC* type.

Invalid LTC audio channel.

Choose an audio device and channel in the *Settings* tab of the inspector. You may also need to visit the Audio section of Settings and connect an audio device to the desired patch. This is only relevant to Timecode cues set to the *LTC* type.

A Pro license is required to reactivate this saved cue.

You'll need to install a Pro Audio, Pro Video, Pro Lighting, or Pro Bundle license in order to use this cue.

Devamp Cues

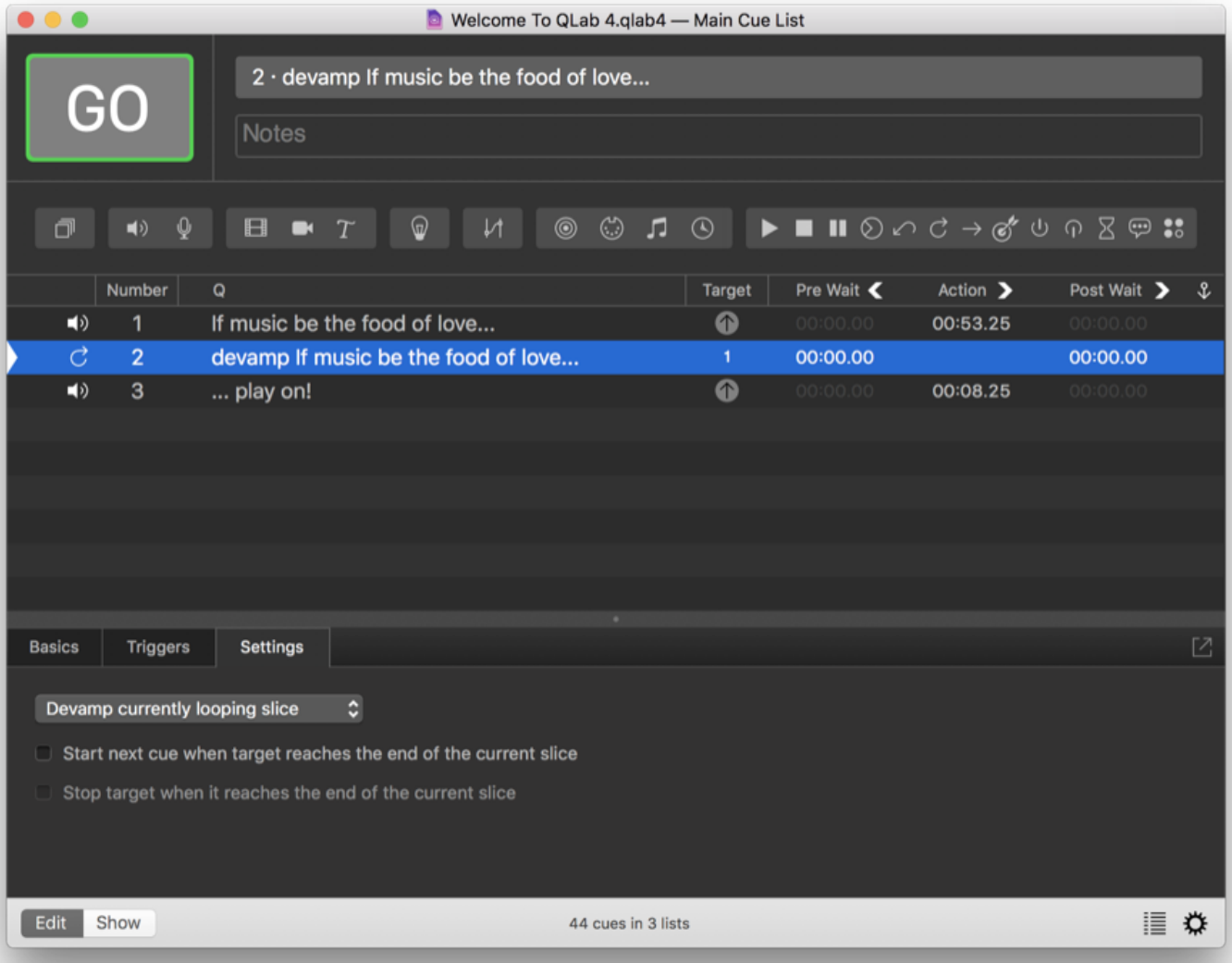
The Devamp cue targets looping Audio and Video cues, and gives you the ability to dynamically exit loops precisely at the end of the loop. You can then have QLab either continue to play through the targeted cue, or trigger a following cue at the precise moment that the loop ends.

Starting with QLab 4.1, Devamp cues can be set to either devamp the currently playing slice of the target cue, or the cue as a whole.

[You can download an example workspace which explores the capabilities of the Devamp cue here.]

There are three ways to use a Devamp cue, illustrated by the following three examples.

Devamp and Continue



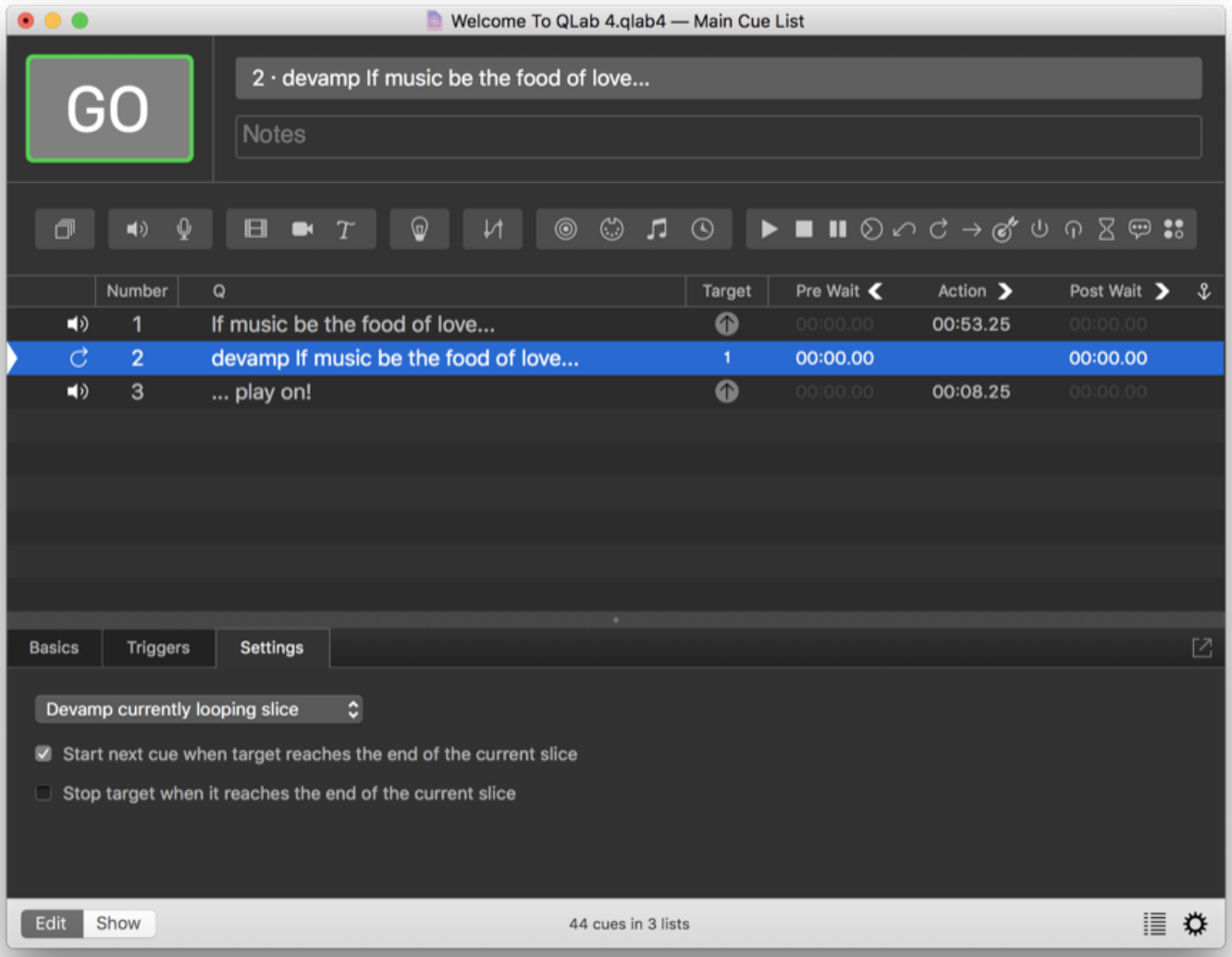
First, create an Audio (or Video) cue with at least one slice towards the beginning of the cue, and set that slice to loop infinitely by double clicking the green slice count in the bottom of the waveform view, and typing `0` or `inf`. For the purposes of this example, it's a good idea for the slice to be longer than a few seconds.

Next, create a Devamp cue and target the Audio or Video cue with the looping slice. Look in the Settings tab of the inspector, and you'll see two checkboxes. Leave them unchecked.

Now, run the Audio or Video cue, and notice when it reaches the looping slice. It will keep repeating that slice indefinitely... until you run the Devamp cue. Run the Devamp, and then when playback reaches the end of the slice, it will no longer loop, and instead proceed onwards.

If you have multiple looping slices in the Audio or Video 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 the Devamp cue is triggered.

Devamp and Start Next



First, create an Audio or Video cue with at least one slice towards the beginning of the cue, and set that slice to loop infinitely by double clicking the green slice count in the bottom of the waveform view, and typing `0` or `inf`. For the purposes of this example, it's a good idea for the slice to be longer than a few seconds.

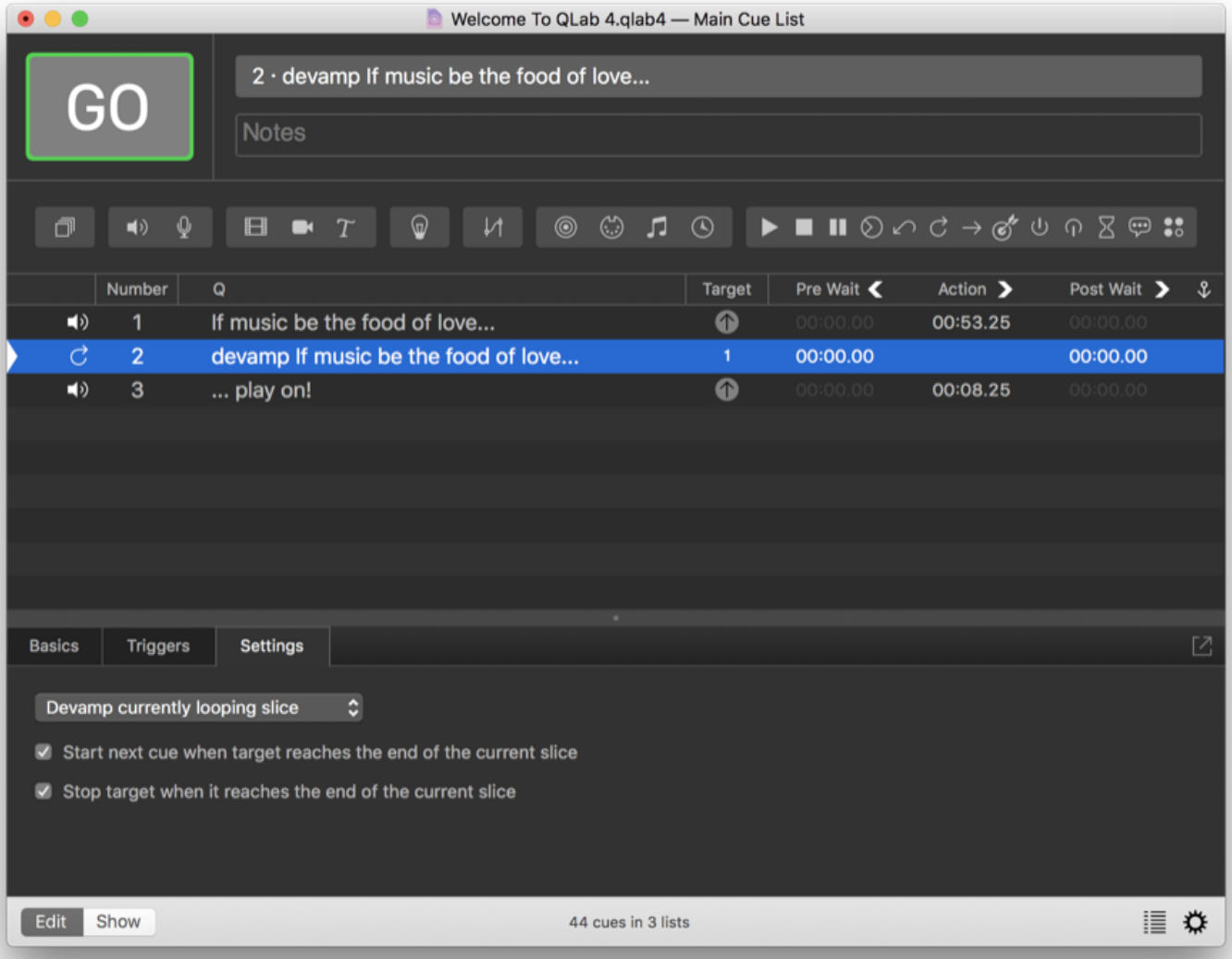
Next, create a Devamp cue and target the Audio or Video cue with the looping slice. Look in the Settings tab of the inspector, and check the box marked *Start next cue when target reaches the end of the current slice*.

Finally, create another cue (of any type) directly after the Devamp cue.

Now, run the Audio or Video cue, and notice when it reaches the looping slice. It will keep repeating that slice indefinitely... until you run the Devamp cue. Run the Devamp, and then when the playback reaches the end of the slice, it will no longer loop, and instead proceed onwards.

At the same moment that the original cue passes the slice marker and continues onwards, the Devamp cue will trigger the following cue to play. This can be very useful for triggering another cue on a musical downbeat, or on an exact frame of a video.

Devamp, Start Next, and Stop Target



First, create an Audio or Video cue with at least one slice towards the beginning of the cue, and set that slice to loop infinitely by double clicking the green slice count in the bottom of the waveform view, and typing `0` or `inf`. For the purposes of this example, it's a good idea for the slice to be longer than a few seconds.

Next, create a Devamp cue and target the Audio or Video cue with the looping slice. Look in the Settings tab of the inspector, and check both the box marked *Start next cue when target reaches the end of the current slice* and the box marked *Stop target when it reaches the end of the current slice.*

Finally, create another cue directly after the Devamp cue.

Now, run the Audio or Video cue, and notice when it reaches the looping slice. It will keep repeating that slice indefinitely... until you run the Devamp cue. Run the Devamp, and then when the playback reaches the end of the slice, it will stop.

At the same moment that the original cue stops, the Devamp cue will trigger the following cue to play. This can be useful for starting a new section of music after a downbeat or for putting a visual button on the end of a looping video.

Thinking In Bars and Beats

The Devamp cue enables QLab to behave as though it's aware of bars and beats by putting slice markers on each beat, and using Devamp cues to trigger 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 Cues

Devamp cues can become broken for the following reasons:

A license is required to reactivate this saved cue.

You'll need to install a license to use this cue.

No target cue.

Assign a valid target cue.

Script Cues

Script cues allow you to execute AppleScript from within QLab.

When a Script cue is selected, three tabs will appear in the Inspector:

- Basics
- Triggers
- Script

The Basics and Triggers tabs are the same for all cue types, and you can learn more about them from the page on the Inspector in the General section of this documentation.

Script

The text field in the Script tab will show the default AppleScript defined in Script Settings.

Text entered here will follow Apple's standard formatting rules for AppleScript.

Compile Script. Click here to compile your script. If your script shows errors, QLab will display an error message next to this button in order to help you find and solve the error.

Run in separate process. By default, 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. If your script, for some reason, needs to execute from within QLab, uncheck this box.

Broken Cues

Script cues can become broken for the following reasons:

Empty script.

Fill in a valid script in the *Script* tab of the inspector.

AppleScript error

Correct the error in the *Script* tab of the inspector.

A license is required to reactivate this saved cue.

You'll need to install a license in order to use this cue.

Other Cues

Start, Stop, and Pause

Start (▶), Stop (■), and Pause (▮▮) cues have no inspector tabs other than the Basics and Triggers tabs. They have only a target, which must be another cue in the workspace. Each of these three cue types has a single function:

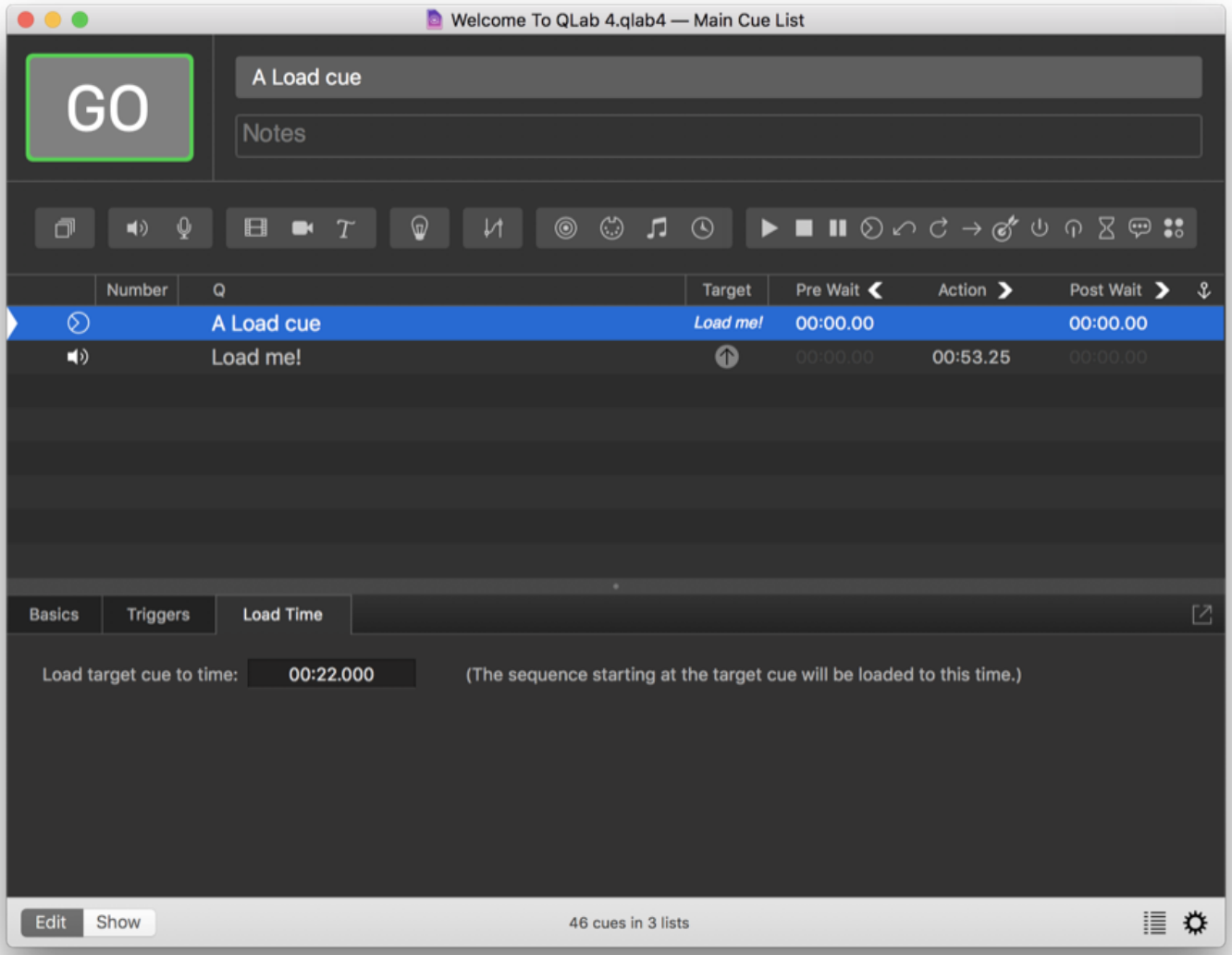
- A Start cue starts its target cue. Note that it does not move the playback position.
- A Stop cue stops its target cue.
- A Pause cue pauses its target cue. Use a Start cue to resume.

Broken Cues

Start, Stop, and Pause cues will only become broken if they have no valid target cue.

Load

The Load cue () loads its target cue. If the target cue has a non-zero action time, then you can load that target cue to a specific time via the Load Time tab in the inspector.

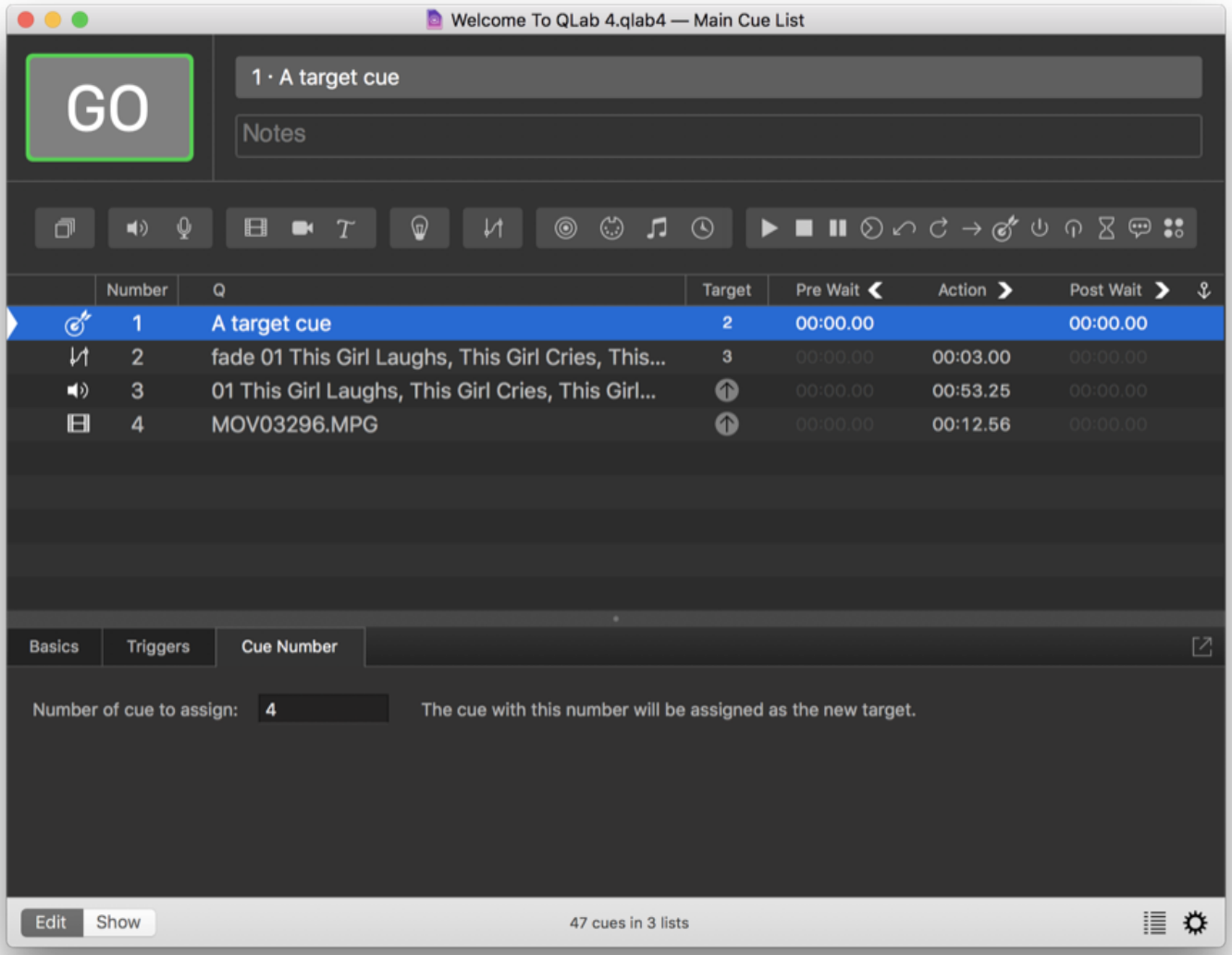


Broken Cues

Load cues will only become broken if they have no valid target cue.

Target

The Target ( ) cue changes the target of another cue in the workspace.



In the above screen shot, Target cue 1 targets Fade cue 2, and sets Fade cue 2's target to Video cue 4.

Broken Cues

Target cues can become broken for the following reasons:

Invalid target cue number.

There is no cue in this workspace with the given number.

Assign a valid cue number in the *Cue Number* tab of the inspector.

No target cue.

Assign a valid target cue.

Reset

The Reset cue (↶) has no inspector tabs other than the Basics and Triggers tabs. It requires a target, and when a Reset cue is triggered, it resets any temporary changes made to its target.

This begs the question: what are temporary changes?

The first and perhaps most obvious answer is that a change of target caused by a Target cue is a temporary change, and using a Reset cue will revert the change made by a Target cue. Additionally, there are a number of OSC methods which make changes to the live state of a cue, which is to say they only have an effect if the cue is currently running. You can use a Reset cue to revert these temporary changes.

Broken Cues

Reset cues will only become broken if they have no valid target cue.

GoTo

The GoTo cue (→) has no inspector tabs other than the Basics and Triggers tabs. It requires a target, and when a GoTo cue is triggered, QLab moves the playback position to that target cue. Note that the target cue is not automatically started; in this respect, the GoTo cue is a complementary cue to the Start cue.

Broken Cues

GoTo cues will only become broken if they have no valid target cue.

Arm and Disarm

Arm ( ) and Disarm ( ) cues have no inspector tabs other than the Basics and Triggers tabs. Their only role is to arm and disarm their target cues, respectively.

Broken Cues

Arm and Disarm cues will only become broken if they have no valid target cue.

Wait

The Wait cue ( ) has no inspector tabs other than the Basics and Triggers tabs, and no settings other than Action. A Wait cue's post-wait time is automatically set to be equal to its duration. You can use a Wait cue in combination with auto-follows or auto-continues as an alternate way to create cue sequences, or as a simple timer for tasks outside of QLab.

Memo

The Memo cue ( ) has no inspector tabs other than the Basics and Triggers tabs, and has no effect when triggered. You can use Memo cues as a place to store notes to your operator (as the name of the Notes cue, or in the cue's notes field), as a visual separator between other cues, or for some similar reason.

Chapter 7: Scripting

OSC Dictionary

QLab has an extensive API (application program 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's OSC implementation.

Transport layer

The QLab OSC API can be used over both UDP and TCP transport layers. QLab listens for incoming OSC on port 53000.

When talking to QLab via UDP, each OSC message corresponds to one UDP datagram. Replies to OSC via UDP are sent on port 53001.

When talking 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 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.

Reply format

All replies from QLab take the form:

```
/reply/{/invoked/osc/method} json_string
```

The reply is sent to the IP address from which the original message was received.

The address of the reply starts with `/reply/` and ends with the address of the invoked method.

`json_string` takes the form:

```
{
    "workspace_id" : string,
    "address": "/invoked/osc/method",
    "status": string,
    "data": value
}
```

`workspace_id` is optional, and only specified if the reply is specifically from the given workspace.

`status` is either `ok` or `error`.

`data` is the JSON-encoded result of invoking the method at `address`

For example, 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}
```

Update format

When a client has requested updates from a specific workspace (i.e. has sent that workspace the `/workspace/{id}/updates 1` command), it will receive push notifications when the client needs to update state for a cue or workspace. The client may receive the following messages at any time:

`/update/workspace/{workspace_id}`

This message is sent if you need to reload the cue lists for the workspace, and also whenever various other aspects of a workspace are updated.

`/update/workspace/{workspace_id}/cue_id/{cue_id}`

This message is sent if you need to reload the state for the given cue. If the cue is a group cue or cue list, you should also reload the children of the cue.

`/update/workspace/{workspace_id}/cueList/{cue_list_id}/playbackPosition {cue_id}`

This message is sent when the playback position of {cue_list_id} changes to {cue_id}. If there is no current playback position, there will be no {cue_id} argument.

`/update/workspace/{workspace_id}/disconnect`

This message is sent if you need to disconnect from the given workspace (because it is going away).

Application methods

QLab will respond to the following general commands:

/alwaysReply {number}

By default, QLab will only send a reply if the method generates a reply to send.

However, if `number` is set to any non-zero number, QLab will send a reply for every OSC message it receives. Messages that would not normally generate a reply will generate one with a JSON string argument that contains:

```
{
    "workspace_id" : string,
    "address": "/invoked/osc/method",
    "status": string
}
```

The `status` string will be either “ok” or “error”.

Note that the `data` field does not exist in this reply.

/auditionWindow {number}

Show or hide the Audition Window. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current visibility of the Audition Window.

/toggleAuditionWindow

Show or hide the Audition Window.

/fontNames

Return an array of the names/PostScript names of all available fonts. For example:

```
[
  "AppleColorEmoji",
  "AppleSDGothicNeo-Bold",
  "AppleSDGothicNeo-ExtraBold",
  "AppleSDGothicNeo-Heavy",
  "AppleSDGothicNeo-Light",
  ...
]
```

/fontFamiliesAndStyles

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"
    ],
  ...
}
```

/liveFadePreview {number}

Enable or disable live fade preview. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current status of live fade preview.

/toggleLiveFadePreview

Enable or disable live fade preview.

/overrides/artNetEnabled {number}

Set the Art-net output override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the Art-net output override.

/overrides/toggleArtNet

Enable or disable Art-net output.

/overrides/midiInputEnabled {number}

Set the MIDI input override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the MIDI input override.

/overrides/toggleMidiInput

Enable or disable MIDI input.

/overrides/midiOutputEnabled {number}

Set the MIDI output override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the MIDI output override.

/overrides/toggleMidiOutput

Enable or disable MIDI output.

/overrides/mscInputEnabled {number}

Set the MSC input override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the MSC input override.

/overrides/toggleMscInput

Enable or disable MSC input.

/overrides/mscOutputEnabled {number}

Set the MSC output override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the MSC output override.

/overrides/toggleMscOutput

Enable or disable MSC output.

/overrides/sysexInputEnabled {number}

Set the MIDI SysEx input override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the MIDI SysEx input override.

/overrides/toggleSysexInput

Enable or disable SysEx input.

/overrides/sysexOutputEnabled {number}

Set the MIDI SysEx output override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the MIDI SysEx output override.

/overrides/toggleSysexOutput

Enable or disable SysEx output.

/overrides/oscInputEnabled {number}

Set the OSC input override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the OSC input override.

/overrides/toggleOscInput

Enable or disable OSC input.

/overrides/oscOutputEnabled {number}

Set the OSC output override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the OSC output override.

/overrides/toggleOscOutput

Enable or disable OSC output.

/overrides/timecodeInputEnabled {number}

Set the timecode input override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the timecode input override.

/overrides/toggleTimecodeInput

Enable or disable timecode input.

/overrides/timecodeOutputEnabled {number}

Set the timecode output override to `true` or `false`. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current state of the timecode output override.

/overrides/toggleTimecodeOutput

Enable or disable timecode output.

/overrideWindow {number}

Show or hide the Override Window. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current visibility of the Override Window.

/toggleOverrideWindow

Show or hide the Override Window.

/replyFormat {format_string}

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. For example, let's say you set QLab's replay format with the following message:

```
/replyFormat "#address# #data#"
```

Then, if you sent `/cue/1/colorName`, you would get the reply: `/cue/1/colorName green`. The `#address#` token resolves to `colorName`, since that was the OSC address you sent, and the `#data#` token resolves to `green`, assuming the color of cue 1 is in fact green.

/timecodeWindow {number}

Show or hide the Timecode Window. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current visibility of the Timecode Window.

/toggleTimecodeWindow

Show or hide the Timecode Window.

/version

Return QLab's version number.

/workingDirectory {path}

Get or set the current working directory. If provided, the `path` string is the working directory you wish to set. 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."

/workspaces

Return an array of workspace dictionaries:

```
[
    {
        "uniqueID": string,
        "displayName": string,
        "hasPasscode": number,
        "version": string
    }
]
```

Workspace methods

Workspace OSC methods use the form `/workspace/{id}/command`, where `{id}` may be either the display name of the workspace, such as `hamlet.qlab4`, or the unique ID of the workspace, which can be found in the Info tab of the 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, diacritical, 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.qlab4/go
```

If you send a workspace method without the `/workspace/{id}` portion of the address, then the method will be sent to the current workspace, which is the front-most, active document. So, if your `hamlet.qlab4` workspace is the front-most, active document, and you send QLab the OSC command `/go`, then `hamlet.qlab4` will GO.

Workspaces respond to the following commands:

/workspace/{id}/basePath

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}

Connect to this workspace with an optional passcode string. If the workspace has a passcode, you MUST supply it before any other commands will be accepted by the workspace or the cues it contains. If the workspace does not have a passcode, the `/workspace/{id}/connect` method is optional.

Returns `ok` if there is no passcode, or the passcode matches.

Returns `badpass` if the passcode does not match.

Returns `error` if the workspace does not exist.

/workspace/{id}/disconnect

Disconnect from this workspace. You should invoke this method when you will no longer be sending messages to this workspace.

If you are communicating to 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. If you are disconnected and the workspace has a password, you will need to reconnect with that password before further commands will be accepted.

If you are communicating to QLab via TCP, QLab will not automatically disconnect your client. The client will remain connected until the client sends the disconnect message or the TCP connection itself is disconnected.

/workspace/{id}/doubleGoWindowRemaining

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}/cueLists
/workspace/{id}/selectedCues
/workspace/{id}/runningCues
/workspace/{id}/runningOrPausedCues

All return an array of cue dictionaries:

```
[
    {
        "uniqueID": string,
        "number": string
        "name": string
        "listName": string
        "type": string
        "colorName": string
        "flagged": number
        "armed": number
    }
]
```

If any of the included cues are Group cues, the dictionary will include an array of cue dictionaries for all children in the group:

```
[
    {
        "uniqueID": string,
        "number": string
        "name": string
        "listName": string
        "type": string
        "colorName": string
        "flagged": number
        "armed": number
        "cues": [ { }, { }, { } ]
    }
]
```

`colorName` may be `none`, `red`, `orange`, `green`, `blue`, or `purple`.

Note: Methods that reply with an array of cue dictionaries may generate large OSC messages. These messages can easily grow larger than the maximum size supported by UDP datagrams. If you need to access these methods you should communicate to QLab over a TCP connection rather than a UDP connection.

Starting with QLab 4.4.3, versions of these commands are available which return less data:

/cueLists/shallow
/selectedCues/shallow
/runningCues/shallow
/runningOrPausedCues/shallow

These methods are identical to the similar methods above, except they do not include any data for the children of Group cues.

/cueLists/uniqueIDs
/selectedCues/uniqueIDs
/runningCues/uniqueIDs
/runningOrPausedCues/uniqueIDs

These methods 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 is included.

/cueLists/uniqueIDs/shallow
/selectedCues/uniqueIDs/shallow
/runningCues/uniqueIDs/shallow
/runningOrPausedCues/uniqueIDs/shallow

These methods are identical to the similar methods above, except they do not include any data for the children of Group cues.

/workspace/{id}/dashboard/clear
/workspace/{id}/dashboard/updateLatestCue
/workspace/{id}/dashboard/updateOriginatingCues
/workspace/{id}/dashboard/updateSelectedCues
/workspace/{id}/dashboard/newCueWithAll
/workspace/{id}/dashboard/newCueWithChanges
/workspace/{id}/dashboard/recordAllToLatest
/workspace/{id}/dashboard/recordAllToSelected
/workspace/{id}/dashboard/revert

These commands are the equivalent of clicking their corresponding buttons in the Light Dashboard.

/workspace/{id}/dashboard/redo

Re-does the last action taken in the Light Dashboard. If nothing has been un-done in the Dashboard, this command has no effect.

/workspace/{id}/dashboard/mode {string}

Set the Light Dashboard's view mode to `string`. Supported modes are `sliders` and `tiles`. If no argument is provided, this method does nothing.

/workspace/{id}/dashboard/nextMode

Toggles between "sliders" and "tiles" view modes in the Light Dashboard.

/workspace/{id}/dashboard/setLight {string} {setting} {time}

Set instrument or light group `string` to level `setting` in the Light 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.

/workspace/{id}/dashboard/undo

Un-does the last action taken in the Light Dashboard. If nothing has been done in the Dashboard, this command has no effect.

/workspace/{id}/delete/{cue_number}
/workspace/{id}/delete_id/{cue_id}
/workspace/{id}/delete/active
/workspace/{id}/delete/selected

Delete the specified cue(s).

/workspace/{id}/fullScreen {number}

`Number` is interpreted as a boolean, and sets whether the workspace is displayed in macOS' full screen mode. If no argument is provided, this returns the current full screen status of the workspace.

/workspace/{id}/toggleFullScreen

Turn full screen mode on or off.

/workspace/{id}/go {cue_number}

Tell the current cue list of the given workspace to GO. `cue_number` is optional; if given, it 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.

In this OSC method, QLab cannot use the same technique it uses in other places to turn numbers into strings when necessary. Therefore, when using `cue_number` in this method, it must always be enclosed in quotation marks.

- ✓: `/go`
- ✓: `/go "53"`
- ✗: `/go 53`

/workspace/{id}/go/{cue_number}

Tell QLab to jump to cue `cue_number` and then GO. `cue_number` must match a cue number in the given workspace.

/workspace/{id}/lightDashboard {number}

Show or hide the Light Dashboard. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no argument is provided, return the current visibility of the Light Dashboard.

/workspace/{id}/toggleLightDashboard

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}

Move the specified cue 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. `new_parent_cue_id` is optional, and must be a string.

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}

Create a new cue. `cue_type` is a string stating which kind 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 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}`.

`{cart_row}` and `{cart_column}` must be integers.

/workspace/{id}/panic

Tell the workspace to panic. A panic is a brief gradual fade out leading into a hard stop. A double panic will trigger an immediate hard stop.

/workspace/{id}/panicInTime {number}

Panic over the specified time, rather than over the panic time defined in the workspace.

/workspace/{id}/pause

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}

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/next
/workspace/{id}/playbackPosition/next

Move the playhead (also called the playback position) of the active cue list to the next cue.

/workspace/{id}/playhead/previous
/workspace/{id}/playbackPosition/previous

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

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

Move the playhead (also called the playback position) of the active cue list to the previous cue sequence.

/workspace/{id}/renumber {startNumber} {incrementNumber}

Renumber the selected cues, starting at `startNumber` and incrementing by `incrementNumber`.

/workspace/{id}/reset

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 method.)

/workspace/{id}/resume

Un-pause all paused cues in the workspace.

/workspace/{id}/save

Tell the given workspace to save itself to disk.

/workspace/{id}/select/{cue_number}
/workspace/{id}/select_id/{id}

Select the specified cue(s).

/workspace/{id}/select/next

Move the selection down one cue.

/workspace/{id}/select/previous

Move the selection up one cue.

/workspace/{id}/selectionIsPlayhead {number}

`Number` is interpreted as a boolean, and sets whether the selection is locked to the playhead. If no argument is provided, return whether the selection is currently locked to the playhead.

/workspace/{id}/toggleSelectionIsPlayhead

Lock or unlock the selection to the playhead.

/workspace/{id}/showMode {number}

`Number` is interpreted as a boolean, and sets whether the workspace is in show mode. If no argument is provided, return whether the workspace is currently in show mode.

/workspace/{id}/toggleEditShowMode

Switch between show mode and edit mode.

/workspace/{id}/stop

Stop playback but allow effects to continue rendering. e.g., playback stops, but reverbs decay naturally.

/workspace/{id}/hardStop

Stop playback and cut all effects immediately.

/workspace/{id}/thump

A simple heartbeat method for this workspace. Returns the data “thump”. (Thump-thump, thump-thump.)

/workspace/{id}/undo
/workspace/{id}/redo

Undo or redo the most recent change of the workspace.

/workspace/{id}/updates {number}

`number` is interpreted as a boolean. If yes, your client wants push notifications of cue changes. If no, your client no longer wants push notifications of cue changes.

Settings methods

Settings methods can be invoked either on a specific workspace (using the `/workspace/{id}/settings/...` address pattern) or "rootless", where the `/settings/...` address pattern is applied to the current front-most workspace.

/settings/audio/maxVolume
/settings/audio/minVolume

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

Read-only; return a JSON dictionary of output names for Audio or Mic output patches. The keys in the dictionary are the patch numbers, and the values are dictionaries of output channel numbers and names. If a given patch does not have customized output names, that patch will not be included in the dictionary.

/settings/general/minGoTime {number}

If `number` is given, set the minimum time required between each GO to `number` seconds. If not, return the minimum time required between each GO.

/settings/video/surfaces

Read-only; return an array of dictionaries describing all video surfaces defined in the workspace.

Each dictionary takes the form:

```
{
    "surfaceID" : number,
    "surfaceName" : string,
    "width" : number,
    "height" : number,
    "warpType" : number,
    "patchSplitsX": array of numbers,
    "patchSplitsY": array of numbers,
    "screenAssignments": array (see below),
}
```

Each item in the `screenAssignments` array is a dictionary in this form:

```
{
    "name" : string,
    "frame" : string representation of a rectangle (i.e. "{{0,0} {1280,800}}"),
    "enableGrid" : number (BOOL),
    "enableGuides" : number (BOOL),
    "controlPoints" : array containing arrays of string representations of points (i.e. "{0,0}"),
}
```

/settings/video/surfaces/{surfaceID}

Read-only; return a dictionary for the specified surface. The dictionaries take the same form as above.

/settings/video/surfaces/{surfaceID}/{screenIndex}/enableGrid {number}

Shows or hides the grid display for the specified screen in the specified surface. `number` is interpreted as a boolean; 0 equals false (don't show grid), any other number equals true (do show grid.) If no argument is provided, return the current display state of the grid. `screenIndex` is a range of whole numbers,

starting with 0, representing the list of screens assigned to a surface. The first screen listed in the Surface Editor is `screenIndex 0`, the second is `screenIndex 1`, and so on.

/settings/video/surfaces/{surfaceID}/{screenIndex}/enableGuides {number}

Shows or hides the guides on the specified screen in the specified surface. `number` is interpreted as a boolean; 0 equals false (don't show guides), any other number equals true (do show grid.) If no argument is provided, return the current display state of the guides.

/settings/video/surfaces/{surfaceID}/{screenIndex}/origin {location}

Set the origin point for the specified screen in the specified surface to `location`. `location` must be a string of the form: `{xPosition,yPosition}`, including the curly braces, where `xPosition` and `yPosition` are both integers.

If no argument is provided, return the current location of the origin for the specified screen in the specified surface.

/settings/video/surfaces/{surfaceID}/{screenIndex}/controlPoint {row_index} {column_index} {location}

Set the specified control point on the specified screen in the specified surface to `location`. `location` must be a string of the form: `{xPosition,yPosition}`, including the curly braces, where `xPosition` and `yPosition` are both integers.

`row_index` and `column_index` refer to the grid of control points in their original, unaltered positions.

If no argument is provided, return the current location of the specified control point on the specified screen in the specified surface.

/settings/video/surfaces/{surfaceID}/{screenIndex}/resetControlPoints

Reset all control points for the specified screen in the specified surface.

Cue methods

Cue methods can be invoked either on a specific workspace (using the `/workspace/{id}/cue/...` address pattern) or "rootless", where the `/cue/...` address pattern is applied to the current, front-most workspace.

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 the commands below, any instance of the address pattern `/cue/{number}` can be replaced by the equivalent unique ID address pattern `/cue_id/{id}`.

Alternately, 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 `{number}` or `{id}`. For example:

`/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.

Increment/Decrement Syntax

Simple number properties of cues can be incremented or decremented with the following syntax:

```
/cue/1/property/+ {delta}
```

```
/cue/1/property/- {delta}
```

For example, the command `/cue/10/preWait/+ 1` would increase the preWait of cue 10 by one second.

/cue/{cue_number}/actionElapsed

Return the elapsed action (in seconds) of the specified cue.

/cue/{cue_number}/percentActionElapsed

Return the elapsed action (as a percentage of the total action) of the specified cue.

/cue/{cue_number}/allowsEditingDuration

Return `true` if the specified cue has an editable duration, such as an Audio, Video, or Fade cue.

/cue/{cue_number}/armed {number}

Get or set the armed state of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no `number` is given, return the armed state of the specified cue.

/cue/{cue_number}/autoLoad {number}

Get or set the auto-load state of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no `number` is given, return the auto-load state of the specified cue.

/cue/{cue_number}/cartPosition

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/{number}/children/shallow

If the specified cue is a list, cart, or Group cue, return the cue numbers of the child cues of that list, cart, or Group. Nested Groups will not be queried, and so only the first "layer" of the cue hierarchy will be returned.

/cue/{number}/children/uniqueIDs

If the specified cue is a list, cart, or Group cue, return the cue IDs of the child cues of that list, cart, or Group, and the cue IDs of any children of nested Groups.

/cue/{number}/children/uniqueIDs/shallow

If the specified cue is a list, cart, or Group cue, return the cue IDs of the child cues of that list, cart, or Group. Nested Groups will not be queried, and so only the first "layer" of the cue hierarchy will be returned.

/cue/{cue_number}/colorName {string}

If `string` is given, set the color of the specified cue to `string`. If not, return the pre-wait of the specified cue. Valid colors are `none`, `red`, `orange`, `green`, `blue`, and `purple`. Certain other colors may also be valid...

/cue/{cue_number}/continueMode {number}

If `number` is given, set the continue mode of the specified cue to `number`. If not, return the continue mode of the specified cue. Valid continue modes are:

```
0 - NO CONTINUE
1 - AUTO CONTINUE
2 - AUTO FOLLOW
```

/cue/{cue_number}/cueTargetId {string}

If `string` is given, and if the specified cue can have cue targets, set the target of the specified cue to `string`. If not, return the cue ID of the target of the specified cue.

/cue/{cue_number}/cueTargetNumber {string}

If `string` is given, and if the specified cue can have cue targets, set the target of the specified cue to `string`. If not, return the cue number of the target of the specified cue.

/cue/{cue_number}/currentCueTarget

Read-only; return the cue ID of the current target of the specified cue.

/cue/{cue_number}/go

If the specified cue is not a cue list, tell QLab to jump to cue `cue_number` and then GO. `cue_number` must match a cue number in the given workspace.

If the specified cue is a cue list, then 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}/tempCueTargetNumber {string}

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. If `string` is not given, and the specified cue has a temporary target, return the cue number of that temporary target.

/cue/{cue_number}/tempCueTargetId

This works exactly the same as `/tempCueTargetNumber`, but uses the cue ID of the target instead of the cue number.

/cue/{cue_number}/currentDuration

Read-only; return the current duration of the specified cue.

/cue/{cue_number}/tempDuration {number}

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. If `number` is not given, and if the specified cue has a temporary duration, return that temporary duration.

/cue/{cue_number}/currentFileTime

Read-only; if the specified cue has a file target, return the current playback time of the target file in seconds. If the cue is not running, that's `0`. If the cue has been playing for ten and a half seconds, and the playback rate of the cue is `1.0`, then the currentFileTime is `10.5`.

/cue/{cue_number}/defaultName

Return the default name of the specified cue.

/cue/{cue_number}/displayName

Return the display name of the specified cue.

/cue/{cue_number}/listName

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}/duckLevel {number}

If `number` is given, set the duck (or boost) level of the specified cue to `number`. If not, return the duck or boost level of the specified cue.

/cue/{cue_number}/duckOthers {number}

Get or set the state of the *Duck audio of other cues* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no `number` is given, return the state of the *Duck audio of other cues* checkbox of the specified cue.

/cue/{cue_number}/duckTime {number}

If `number` is given, set the *Duck audio of other cues* time of the specified cue to `number`. If not, return the *Duck audio of other cues* time of the specified cue.

/cue/{cue_number}/duration {number}

If `number` is given, and the selected cue has an editable duration, set the duration of the specified cue to `number`. If not, return the duration of the specified cue.

/cue/{cue_number}/fadeAndStopOthers {number}

If `number` is given, set the *Fade and stop* mode of the specified cue to `number`. If not, return the *Fade and stop* mode of the specified cue. Mode `0` is equivalent to the *Fade and stop* checkbox being unchecked. Valid modes are:

```
0 - none
1 - peers
2 - list or cart
3 - all
```

/cue/{cue_number}/fadeAndStopOthersTime {number}

If `number` is given, set the *Fade and stop others* time of the specified cue to `number`. If not, return the *Fade and stop others* time of the specified cue.

/cue/{cue_number}/fileTarget {string}

If `string` is given, and if the specified cue can have file targets, set the target of the specified cue to `string`. If not, return the target of the specified cue. You can provide a target using any of three kinds of paths:

- 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 QLab’s `/workingDirectory` method to set or get the current working directory.

/cue/{cue_number}/flagged {number}

Get or set the flagged state of the specified cue. `number` is interpreted as a boolean; `0` equals false, any other number equals true. If no `number` is given, return the flagged state of the specified cue.

/cue/{cue_number}/hasCueTargets

Read-only; return `true` if the specified cue is able to target another cue, such as a Fade or Stop cue.

/cue/{cue_number}/hasFileTargets

Read-only; return `true` if the specified cue is able to target a file, such as an Audio or Video cue.

/cue/{cue_number}/isActionRunning

Read-only; return `true` if the action of the specified cue (not the pre-wait or post-wait) is running.

/cue/{cue_number}/isBroken

Read-only; return `true` if the specified cue is broken.

/cue/{cue_number}/isLoaded

Read-only; return `true` if the specified cue is loaded.

/cue/{cue_number}/isOverridden

Read-only; return `true` if the specified cue's output is currently suppressed by an override control.

/cue/{cue_number}/isPanicking

Read-only; return `true` if the specified cue is panicking.

/cue/{cue_number}/isPaused

Read-only; return `true` if the specified cue is paused.

/cue/{cue_number}/isRunning

Read-only; return `true` if the specified cue is running.

/cue/{cue_number}/isTailingOut

Read-only; return `true` if the specified cue has an AudioUnit which is decaying.

/cue/{cue_number}/load

Load the specified cue.

/cue/{cue_number}/loadAt {number}

If `number` is given, load the specified cue to `number` seconds. If not, this command is equivalent to `load`.

/cue/{cue_number}/loadActionAt {number}

If `number` is given, load the specified cue to `number` seconds. QLab will automatically add the pre-wait of the specified cue to `number` in order to load the cue to the correct time. If no number is given, this command is equivalent to `load`.

/cue/{cue_number}/loadAndSetPlayhead

Move the playhead to the specified cue and load that cue.

/cue/{cue_number}/maxTimeInCueSequence

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}

If `string` is given, set the name of the specified cue to `string`. If not, return the name of the specified cue.

/cue/{cue_number}/notes {string}

If `string` is given, set the notes of the specified cue to `string`. If not, return the notes of the specified cue.

/cue/{cue_number}/number {string}

If `string` is given, set the cue number of the specified cue to `string`. If not, return the cue number of the specified cue.

/cue/{cue_number}/panic

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}

Panic the specified cue, using `number` for the panic duration instead of the panic duration specified in Workspace Settings.

/cue/{cue_number}/pause

Pause the specified cue, allowing any AudioUnit effects on the cue to decay naturally. If the specified cue is not playing, this command has no effect.

/cue/{cue_number}/hardPause

Pause the specified cue without allowing AudioUnit effects to decay naturally. If the specified cue is not playing, this command has no effect.

/cue/{cue_number}/togglePause

Toggle the paused state of the specified cue. That is, if the cue is playing, this command will pause it. If the cue is paused, this command will resume it. If the specified cue is not playing, this command has no effect.

/cue/{cue_number}/parent

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}/preview

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 triggered.

/cue/{cue_number}/preWait {number}

If `number` is given, set the pre-wait of the specified cue to `number`. If not, return the pre-wait of the specified cue.

/cue/{cue_number}/preWaitElapsed

Return the elapsed pre-wait time (in seconds) of the specified cue.

/cue/{cue_number}/percentPreWaitElapsed

Return the elapsed pre-wait time (as a percentage of the total pre-wait time) of the specified cue.

/cue/{cue_number}/postWait {number}

If `number` is given, set the post-wait of the specified cue to `number`. If not, return the post-wait of the specified cue.

/cue/{cue_number}/postWaitElapsed

Return the elapsed post-wait time (in seconds) of the specified cue.

/cue/{cue_number}/percentPostWaitElapsed

Return the elapsed post-wait time (as a percentage of the total post-wait time) of the specified cue.

/cue/{cue_number}/resume

Resume the specified cue. If the specified cue is not paused, this command has no effect.

/cue/{cue_number}/reset

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}/secondTriggerAction {number}

If `number` is given, set the second trigger action of the specified cue to `number`. If not, return the second trigger action of the specified cue. Valid actions are:

```
0 - does nothing
1 - panics
```

2 - stops
3 - hard stops
4 - hard stops & restarts

/cue/{cue_number}/secondTriggerOnRelease {number}

Get or set the state of the *Second trigger on release* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no `number` is given, return the state of the *Second trigger on release* checkbox of the specified cue.

/cue/{cue_number}/soloCueInTime {number}

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 positive whole or decimal number.

/cue/{cue_number}/start

Start the specified cue.

/cue/{cue_number}/startAndAutoloadNext

Start the specified cue and load the following cue or cue sequence if that cue or cue sequence is set to auto-load.

/cue/{cue_number}/stop

Stop the specified cue. If the specified cue is not playing, this command has no effect.

/cue/{cue_number}/hardStop

HardStop the specified cue. If the specified cue is not playing, this command has no effect. HardStopped cues stop immediately; if the cue has an Audio Effect which ordinarily decays after stopping, such as a reverb, the Audio Effect is also stopped immediately.

/cue/{cue_number}/type

Return the type (Audio, Video, Fade, etc.) of the specified cue.

/cue/{cue_number}/uniqueID

Return the uniqueID of the specified cue.

/cue/{cue_number}/valuesForKeys {json_string}

This special method can be used to request a custom collection of state about the given cue. `json_string` must be a JSON-formatted string representing an array of keys you wish to query. For example:

```
/cue/2/valuesForKeys "[\"opacity\",\"surfaceSize\"]"
```

would return the values of opacity and surfaceSize of cue 2, assuming cue 2 is a Video cue.

/cue/{cue_number}/valuesForKeysWithArguments {json_string}

This special method can be used to request a custom collection of state about the given cue. `json_string` must be a JSON-formatted string representing a dictionary of keys and arguments you wish to query. For example:

```
/cue/2/valuesForKeysWithArguments "{\"level\":[0,0]}"
```

would return a dictionary that contains the value of the master 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 methods

Methods specific to Group cues, cue lists, and carts (which are really also Group cues.)

/cue/{cue_number}/cartColumns
/cue/{cue_number}/cartRows

Read-only; if the specified cue is a cart, these messages return the number of rows or columns in the cart.

/cue/{cue_number}/children

Read-only; return a list of the cues contained within the specified Group, list, or cart. Returns the same data as the workspace `/cueLists` method.

/cue/{cue_number}/go

If the specified cue is a cue list, then 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}/mode {number}

If `number` is given, set the mode of the specified Group cue. If not, return the mode of the specified Group cue. `number` must be a whole number from 1 to 4. A cue list will return mode `0`, but mode cannot be set to `0`. A cue cart will return mode `5`, but mode cannot be set to `5`.

/cue/{cue_number}/moveCartCue/{child} {row} {column}

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.

/cue/{cue_number}/playhead {string}
/cue/{cue_number}/playbackPosition {string}

If the specified cue is a cue list, set the playhead (playback position) to cue `string`. If `string` is not specified, return the cue number of the standing-by cue, or “none” if there is no cue standing by.

/cue/{cue_number}/playheadId {string}
/cue/{cue_number}/playbackPositionId {string}

If the specified cue is a cue list, set the playhead (playback position) to the cue ID `string`. If `string` is `none`, unset the playhead. If `string` is not specified, return the cue ID of the standing-by cue, or “none” if there is no cue standing by.

Audio cue methods

Methods specific to Audio cues.

/cue/{cue_number}/doFade {number}

Get or set the state of the integrated fade curve of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the integrated fade curve of the specified cue.

/cue/{cue_number}/doPitchShift {number}

Get or set the state of the pitch shift checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the pitch shift checkbox of the specified cue.

/cue/{cue_number}/endTime {number}

If `number` is given, set the end time of the specified cue to `number` seconds. If not, return the end time of the specified cue. `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}

Get or set a single crosspoint gang.

`inChannel` is an integer from 0 to 24.

`outChannel` is either an integer from 0 to 64, or a string (the cue output name).

`gang` is an optional text string. When present it is the gang to set.

If no `gang` is given, return the gang of the specified crosspoint.

/cue/{cue_number}/infiniteLoop {number}

Get or set the infinite loop state of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the infinite loop state of the specified cue.

/cue/{cue_number}/level {inChannel} {outChannel} {decibel}
/cue/{cue_number}/level/{inChannel}/{outChannel} {decibel}

Get or set a single crosspoint volume level.

`inChannel` is an integer from 0 to 24. 0 is the master column.

`outChannel` is either an integer from 0 to 64, or a string (the cue output name). 0 is the master row.

`decibel` is an optional whole or decimal number. When present it is the decibel value to set.

If `decibel` is sent as a string (e.g. "-inf") QLab will use the minimum decibel value set in workspace settings.

If no `decibel` is given, return the volume level of the specified crosspoint.

/cue/{cue_number}/levels

Read-only; return all the audio levels currently available in the specified cue's inspector. The levels are returned as an array of arrays, like so: `[row0Array, row1Array, row2Array, ...]`

Row 0 of `/levels` is equivalent to the results of the `/sliderLevels` method.

/cue/{cue_number}/liveAverageLevel/{outputChannel} {low} {high}

Read-only; return the RMS level of `outputChannel`. `low` and `high` are optional values to re-scale the output of this method. For example, sending the message `/cue/1/liveAverageLevel/1 0 100` will return a `0` if the level of output 1 is silent, and `100` if the level of output 1 is as loud as is possible.

/cue/{cue_number}/lockFadeToCue {number}

Get or set the state of the *Lock fade to start/end* checkbox of the specified cue. `number` is interpreted as a boolean; `0` equals false, any other number equals true. If no number is given, return the state of the *Lock fade to start/end* checkbox of the specified cue.

/cue/{cue_number}/numChannelsIn

Read-only; return the number of input channels in the specified cue.

/cue/{cue_number}/patch {number}

If `number` is given, set the patch of the specified cue. If not, return the patch of the specified cue. `number` must be a whole number from 1 to the number of network destination patches in the workspace.

/cue/{cue_number}/patchList

Read-only; return a list of audio patches defined for this workspace:

```
[
    {
        "patchNumber": integer,
        "patchName": string
```

```
    }
]
```

/cue/{cue_number}/playCount {number}

If `number` is given, set the play count (number of times to loop) of the specified cue to `number`. If not, return the play count of the specified cue. `number` can be any whole number greater than zero.

/cue/{cue_number}/rate {number}
/cue/{cue_number}/liveRate {number}

If `number` is given, set the rate of the specified cue to `number`. If not, return the rate of the specified cue. `number` can be any positive whole or decimal number from 0.03 to 33.0.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/setDefaultLevels

Set the audio levels of the specified cue to the workspace default levels.

/cue/{cue_number}/setSilentLevels

Set the audio levels of the specified cue to silent.

/cue/{cue_number}/sliceMarkers

Read-only; return a JSON dictionary (or array of dictionaries) listing the marker time and play count of all slices of the specified cue:

```
{
    "time": number,
    "playCount": number
}
```

Note: slices *end* with slice markers. Therefore, `time` corresponds to the end time of the slice whose `playCount` is being reported.

/cue/{cue_number}/sliceMarker {index}
/cue/{cue_number}/sliceMarker/{index}

Read-only; return a JSON dictionary listing the marker time and play count of slice `index` of the specified cue. `index` can be zero or any positive whole number.

/cue/{cue_number}/sliceMarker/{index}/time

Read-only; return the marker time of slice `index` of the specified cue.

/cue/{cue_number}/sliceMarker/{index}/playCount

Read-only; return the play count of slice `index` of the specified cue.

/cue/{cue_number}/addSliceMarker {time} {play_count}

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 not, set the play count to 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}/deleteSliceMarker {index}

/cue/{cue_number}/deleteSliceMarker/{index}

Delete slice marker `index` of the specified cue. `index` can be zero or any positive whole number.

/cue/{cue_number}/deleteSliceMarkers

Delete all slice markers of the specified cue.

/cue/{cue_number}/sliceMarker {index} {time} {play_count}
/cue/{cue_number}/sliceMarker/{index} {time} {play_count}

Set the marker time and play count of slice `index` of the specified cue. `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.

/cue/{cue_number}/sliceMarker/{index}/time {time}

Set the marker time of slice `index` of the specified cue. `time` can be any positive whole or decimal number.

/cue/{cue_number}/sliceMarker/{index}/time/+ {delta}
/cue/{cue_number}/sliceMarker/{index}/time/- {delta}

Add or subtract `delta` to/from the marker time of slice `index` in the specified cue. `delta` can be any positive whole or decimal number.

/cue/{cue_number}/sliceMarkers/time/+ {delta}
/cue/{cue_number}/sliceMarkers/time/- {delta}

Add or subtract `delta` to/from the marker time of all slices in the specified cue. `delta` can be any positive whole or decimal number.

/cue/{cue_number}/sliceMarker/{index}/playCount {play_count}

Set the play count of slice `index` of the specified cue. `play_count` can be any positive whole number, or `-1` to set the slice to infinite loop.

/cue/{cue_number}/sliceMarker/{index}/playCount/+ {delta}
/cue/{cue_number}/sliceMarker/{index}/playCount/- {delta}

Add or subtract `delta` to/from the play count of slice `index` in the specified cue. `delta` can be any positive whole number.

/cue/{cue_number}/lastSlicePlayCount {play_count}

If `play_count` is given, set the play count of the last slice of the specified cue to `play_count` . Otherwise, return the play count of the last slice of the specified cue. `play_count` can be any positive whole number, or `-1` to set the slice to infinite loop.

/cue/{cue_number}/lastSliceInfiniteLoop {number}

If `number` is given, and is any positive whole number, set the last slice of the specified cue to loop infinitely. If `number` is 0, then set the last slice of the specified cue to not loop. If `number` is not given, return the infinite loop state of the last slice of the specified cue.

/cue/{cue_number}/sliderLevel {channel} {decibel}
/cue/{cue_number}/sliderLevel/{channel} {decibel}

Get or set a single output slider volume level.

`channel` is either an integer from 0 to 64, or a string (the cue output name). 0 is the cue master slider.

`decibel` is an optional whole or decimal number. When present it is the decibel value to set.

If `decibel` is sent as a string (e.g. “-inf”) QLab will use the minimum decibel value set in workspace settings.

If no `decibel` is given, return the volume level of the specified output slider.

/cue/{cue_number}/sliderLevels

Read-only; return an array of the output levels of the specified cue, including the cue master. The array is therefore 65 numbers.

/cue/{cue_number}/startTime {number}

If `number` is given, set the start time of the specified cue to `number` seconds. If not, return the start time of the specified cue. `number` can be any whole or decimal number greater than or equal to zero.

Mic cue methods

Methods specific to Mic cues. Mic cues also respond to most Audio cue methods.

/cue/{cue_number}/channelOffset

Read-only; return the input channel offset of the specified cue, as set in the Audio Levels tab of the inspector.

/cue/{cue_number}/channels {number}

If `number` is given, set the number of input channels used by the specified cue. If not, return the number of input channels used by the specified cue. `number` can be any positive whole number.

/cue/{cue_number}/setDefaultLevels

Set the audio levels of the specified cue to the workspace default levels.

/cue/{cue_number}/setSilentLevels

Set the audio levels of the specified cue to silent.

Video cue methods

Methods specific to Video cues.

/cue/{cue_number}/cueSize

Read-only; return the natural size of the cue's video frame:

```
{
    "width": number,
    "height": number
}
```

/cue/{cue_number}/doFade {number}

Get or set the state of the integrated fade curve of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the integrated fade curve of the specified cue.

/cue/{cue_number}/doPitchShift {number}

Get or set the state of the pitch shift checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the pitch shift checkbox of the specified cue.

/cue/{cue_number}/doEffect {number}

Get or set the state of the *Apply effects* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the *Apply effects* checkbox of the specified cue.

/cue/{cue_number}/effectIndex {number}

If `number` is given, set the video effect used for the specified cue. If not, return the index number of the video effect used for the specified cue. `number` can be any of the following:

```
1  - Color Controls
2  - Exposure
3  - Gamma
4  - Sepia Monochrome
5  - Min Max Invert
6  - White point
7  - Box / Disc / Gaussian Blurs
8  - Motion Blur
9  - Sharpen Luminance
10 - Unsharp Mask
11 - Zoom Blur
12 - Pixellation
13 - Screen
14 - Bloom and Gloom
15 - CMYK Halftone
16 - Color Posterize
17 - Crystallize and Pointillize
18 - Edge Work
19 - Kaleidoscope
20 - Median and Comic Effect
21 - Noise Reduction
22 - Circle Splash / Hole Distortion
23 - Pinch / Bump Distortion
24 - Torus / Lens Distortion
25 - Twirl / Circular Wrap / Vortex
26 - Glass Lozenge
27 - Op Tile
28 - Perspective Tile
29 - Quad Tiles
30 - Reflected Tiles
31 - Rotated Tiles
32 - Titles
```

Note: `/effectIndex` replaces `/effect` which worked in earlier versions of QLab 4, but which was removed for boring technical reasons.

/cue/{cue_number}/effectSet {parameter} {value}
/cue/{cue_number}/liveEffectSet {parameter} {value}

If `value` is given, set the specified video effect's `parameter` to `value`. If not, return the value of the specified `parameter`.

`parameter` must be a string, and must match the name of a parameter of the video effect currently in use for the specified cue. For example, if the specified cue is using the "Color Controls" video effect, then `parameter` could be "Brightness", "Contrast", "Hue Angle", or "Saturation".

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/endTime {number}

If `number` is given, set the end time of the specified cue to `number` seconds. If not, return the end time of the specified cue. `number` can be any whole or decimal number greater than or equal to zero.

/cue/{cue_number}/fullScreen {number}
/cue/{cue_number}/fullSurface {number}

Get or set the full-surface state of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the full-surface state of the specified cue.

/cue/{cue_number}/gang {inChannel} {outChannel} {gang}
/cue/{cue_number}/gang/{inChannel}/{outChannel} {gang}

Get or set a single crosspoint gang.

`inChannel` is an integer from 0 to 24.

`outChannel` is either an integer from 0 to 64, or a string (the cue output name).

`gang` is an optional text string. When present it is the gang to set.

If no `gang` is given, return the gang of the specified crosspoint.

/cue/{cue_number}/holdLastFrame {number}

Get or set the state of the *hold last frame* checkbox of the specified cue. `number` is interpreted as a boolean; `0` equals false, any other number equals true. If no number is given, return the state of the *hold last frame* specified cue.

/cue/{cue_number}/infiniteLoop {number}

Get or set the infinite loop state of the specified cue. `number` is interpreted as a boolean; `0` equals false, any other number equals true. If no number is given, return the infinite loop state of the specified cue.

/cue/{cue_number}/layer {number}

If `number` is given, set the layer of the specified cue to `number`. If not, return the layer of the specified cue. `number` can be any positive whole number from 0 to 1000. Layer 0 is the “bottom” layer and layer 1000 is the “top” layer.

A number between 0 and 1000, inclusive.

/cue/{cue_number}/level {inChannel} {outChannel} {decibel}
/cue/{cue_number}/level/{inChannel}/{outChannel} {decibel}

Get or set a single crosspoint volume level.

`inChannel` is an integer from 0 to 24. 0 is the master column.

`outChannel` is either an integer from 0 to 64, or a string (the cue output name). 0 is the master row.

`decibel` is an optional whole or decimal number. When present it is the decibel value to set.

If `decibel` is sent as a string (e.g. “-inf”) QLab will use the minimum decibel value set in workspace settings.

If no `decibel` is given, return the volume level of the specified crosspoint.

/cue/{cue_number}/levels

Read-only; return all the audio levels currently available in the specified cue’s inspector. The levels are returned as an array of arrays, like so: `[row0Array, row1Array, row2Array, ...]`

Row 0 of `/levels` is equivalent to the results of the `/sliderLevels` method.

/cue/{cue_number}/liveAverageLevel/{outputChannel} {low} {high}

Read-only; return the RMS level of `outputChannel`. `low` and `high` are optional values to re-scale the output of this method. For example, sending the message `/cue/1/liveAverageLevel/1 0 100` will return a `0` if the level of output 1 is silent, and `100` if the level of output 1 is as loud as is possible.

/cue/{cue_number}/lockFadeToCue {number}

Get or set the state of the *Lock fade to start/end* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the *Lock fade to start/end* checkbox of the specified cue.

/cue/{cue_number}/numChannelsIn

Read-only; return the number of input channels in the specified cue.

/cue/{cue_number}/opacity {number}

If `number` is given, set the opacity of the specified cue to `number`. If not, return the opacity of the specified cue. `number` can be any decimal number between 0 and 1.

/cue/{cue_number}/originX {number}

If `number` is given, set the X-axis anchor point of the specified cue to `number`. If not, return the X-axis anchor point of the specified cue. `number` can be any decimal number.

/cue/{cue_number}/originY {number}

If `number` is given, set the Y-axis anchor point of the specified cue to `number`. If not, return the Y-axis anchor point of the specified cue. `number` can be any decimal number.

/cue/{cue_number}/origin {x} {y}

Set the anchor point of the specified cue to `(x,y)`. `x` and `y` can be any decimal numbers.

/cue/{cue_number}/patch {number}

If `number` is given, set the patch of the specified cue. If not, return the patch of the specified cue. `number` must be a whole number from 1 to 8.

/cue/{cue_number}/patchList

Read-only; return a list of audio patches defined for this workspace:

```
[
    {
        "patchNumber": integer,
        "patchName": string
    }
]
```

/cue/{cue_number}/playCount {number}

If `number` is given, set the play count (number of times to loop) of the specified cue to `number`. If not, return the play count of the specified cue. `number` can be any whole number greater than zero.

/cue/{cue_number}/preserveAspectRatio {number}

Get or set the state of the *Preserve aspect ratio* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the *Preserve aspect ratio* checkbox of the specified cue.

/cue/{cue_number}/quaternion {number number number number}

If four `numbers` are given, set the rotation of the specified cue. If not, return an array of four numbers representing the cue's rotation as a quaternion. `number` can be any decimal number. **Caution:** you need to understand quaternion math to make any meaningful use of this method. Quaternion math is really hard. Good luck!

/cue/{cue_number}/rate {number}

If `number` is given, set the rate of the specified cue to `number`. If not, return the rate of the specified cue. `number` can be any positive whole or decimal number from 0.03 to 33.0.

/cue/{cue_number}/liveRate {number}

If `number` is given, set the rate of the specified cue to `number`. If not, return the rate of the specified cue. `number` can be any positive whole or decimal number from 0.03 to 33.0.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/resetRotation

Reset the rotation of the specified cue.

/cue/{cue_number}/rotateX {number}
/cue/{cue_number}/rotateY {number}
/cue/{cue_number}/rotateZ {number}

Add `number` to the current quaternion rotation of the specified cue. `number` can be any decimal number.

/cue/{cue_number}/liveRotation/Xaxis {number}
/cue/{cue_number}/liveRotation/Yaxis {number}
/cue/{cue_number}/liveRotation/Zaxis {number}
/cue/{cue_number}/liveRotation/x {number}
/cue/{cue_number}/liveRotation/y {number}
/cue/{cue_number}/liveRotation/z {number}

Rotate the specified cue by `number` degrees in the X, Y, or Z axis. These are equivalent to using a Fade cue in single-axis mode to rotate the specified cue.

These "live" methods have no non-live counterparts. They operate on the active, "live" value of a running cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/scaleX {number}
/cue/{cue_number}/liveScaleX {number}

If `number` is given, set the X-axis scale of the specified cue to `number`. If not, return the X-axis scale of the specified cue. `number` can be any decimal number.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/scaleY {number}
/cue/{cue_number}/liveScaleY {number}

If `number` is given, set the Y-axis scale of the specified cue to `number`. If not, return the Y-axis scale of the specified cue. `number` can be any decimal number.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/scale {x} {y}
/cue/{cue_number}/liveScale {x} {y}

If `x` and `y` are given, set the scale of the specified cue to `(x,y)`. If not, return the current scale of the specified cue. `x` and `y` can be any decimal numbers.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/setDefaultLevels

Set the audio levels of the specified cue to the workspace default levels.

/cue/{cue_number}/setSilentLevels

Set the audio levels of the specified cue to silent.

/cue/{cue_number}sliceMarkers

Read-only; return a JSON dictionary (or array of dictionaries) listing the marker time and play count of all slices of the specified cue:

```
{
    "time": number,
    "playCount": number
}
```

Note: slices *end* with slice markers. Therefore, `time` corresponds to the end time of the slice whose `playCount` is being reported.

/cue/{cue_number}/sliceMarker {index}
/cue/{cue_number}/sliceMarker/{index}

Read-only; return a JSON dictionary listing the marker time and play count of slice `index` of the specified cue. `index` can be zero or any positive whole number.

/cue/{cue_number}/sliceMarker/{index}/time

Read-only; return the marker time of slice `index` of the specified cue.

/cue/{cue_number}/sliceMarker/{index}/playCount

Read-only; return the play count of slice `index` of the specified cue.

/cue/{cue_number}/addSliceMarker {time} {play_count}

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 not, set the play count to 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}/deleteSliceMarker {index}
/cue/{cue_number}/deleteSliceMarker/{index}

Delete slice marker `index` of the specified cue. `index` can be zero or any positive whole number.

/cue/{cue_number}/deleteSliceMarkers

Delete all slice markers of the specified cue.

/cue/{cue_number}/sliceMarker {index} {time} {play_count}
/cue/{cue_number}/sliceMarker/{index} {time} {play_count}

Set the marker time and play count of slice `index` of the specified cue. `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.

/cue/{cue_number}/sliceMarker/{index}/time {time}

Set the marker time of slice `index` of the specified cue. `time` can be any positive whole or decimal number.

/cue/{cue_number}/sliceMarker/{index}/time/+ {delta}
/cue/{cue_number}/sliceMarker/{index}/time/- {delta}

Add or subtract `delta` to/from the marker time of slice `index` in the specified cue. `delta` can be any positive whole or decimal number.

/cue/{cue_number}/sliceMarkers/time/+ {delta}
/cue/{cue_number}/sliceMarkers/time/- {delta}

Add or subtract `delta` to/from the marker time of all slices in the specified cue. `delta` can be any positive whole or decimal number.

/cue/{cue_number}/sliceMarker/{index}/playCount {play_count}

Set the play count of slice `index` of the specified cue. `play_count` can be any positive whole number, or `-1` to set the slice to infinite loop.

/cue/{cue_number}/sliceMarker/{index}/playCount/+ {delta}
/cue/{cue_number}/sliceMarker/{index}/playCount/- {delta}

Add or subtract `delta` to/from the play count of slice `index` in the specified cue. `delta` can be any positive whole number.

/cue/{cue_number}/lastSlicePlayCount {play_count}

If `play_count` is given, set the play count of the last slice of the specified cue to `play_count` . Otherwise, return the play count of the last slice of the specified cue. `play_count` can be any positive whole number, or `-1` to set the slice to infinite loop.

/cue/{cue_number}/lastSliceInfiniteLoop {number}

If `number` is given, and is any positive whole number, set the last slice of the specified cue to loop infinitely. If `number` is 0, then set the last slice of the specified cue to not loop. If `number` is not given, return the infinite loop state of the last slice of the specified cue.

/cue/{cue_number}/sliderLevel {channel} {decibel}
/cue/{cue_number}/sliderLevel/{channel} {decibel}

Get or set a single output slider volume level.

`channel` is either an integer from 0 to 64, or a string (the cue output name). 0 is the cue master slider.

`decibel` is an optional decimal number. When present it is the decibel value to set.

If `decibel` is sent as a string (e.g. "-inf") QLab will use the minimum decibel value set in workspace settings.

If no `decibel` is given, return the volume level of the specified output slider.

/cue/{cue_number}/sliderLevels

Read-only; return an array of the output levels of the specified cue, including the cue master. The array is therefore 65 numbers.

/cue/{cue_number}/startTime {number}

If `number` is given, set the start time of the specified cue to `number` seconds. If not, return the start time of the specified cue. `number` can be any whole or decimal number greater than or equal to zero.

/cue/{cue_number}/surfaceID {number}

If `number` is given, set the surface of the specified cue. If not, return the surface ID of the surface of the specified cue.

/cue/{cue_number}/surfaceList

Read-only; return a list of surfaces defined for this workspace:

```
[
    {
        "surfaceName": string,
        "surfaceID": number
    }
]
```

/cue/{cue_number}/surfaceName {string}

If `string` is given, assign the specific cue to surface `string`. If not, return the name of the surface to which the specified cue is assigned. `string` must be the name of a surface in this workspace.

/cue/{cue_number}/surfaceSize

Read-only; returns the size of the cue's display surface:

```
{
    "width": number,
    "height": number
}
```

/cue/{cue_number}/translationX {number}
/cue/{cue_number}/liveTranslationX {number}

If `number` is given, set the X-axis translation of the specified cue to `number`. If not, return the X-axis translation of the specified cue. `number` can be any decimal number.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/translationY {number}
/cue/{cue_number}/liveTranslationY {number}

If `number` is given, set the Y-axis translation of the specified cue to `number`. If not, return the Y-axis translation of the specified cue. `number` can be any decimal number.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/translation {x} {y}
/cue/{cue_number}/liveTranslation {x} {y}

If `x` and `y` are given, set the translation of the specified cue to `(x,y)`. If not, return the current translation of the specified cue. `x` and `y` can be any decimal numbers.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

Camera cue methods

Methods specific to Camera cues. Camera cues also respond to most Video cue methods.

/cue/{cue_number}/cameraPatch

If `number` is given, set the camera patch of the specified cue. If not, return the camera patch of the specified cue. `number` must be a whole number from 1 to 8.

Text cue methods

Methods specific to Text cues. Text cues also respond to most Video cue methods.

/cue/{cue_number}/fixedWidth {number}

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. If no number is given, return the width of the specified cue. `number` can be any positive whole or decimal number.

/cue/{cue_number}/text {string}
/cue/{cue_number}/liveText {string}

If `string` is given, set the text of the specified cue to `string`. If not, return the text of the specified cue. When setting text, the formatting will match the first character of the existing text.

"Live" methods are the same as their non-live counterparts, but operate on the active, "live" value of a running cue, rather than changing the "start state" of the cue. Invoking these methods does not cause the document to have unsaved changes.

/cue/{cue_number}/text/format {json_string}

If `json_string` is given, this command can be used to set the formatting of one or more substrings of the specified cue. For example:

```
[
  {
    "fontFamily": string,
    "fontStyle": string,
    "fontName": string,
    "fontSize": number,
    "lineSpacing": number,
    "color": array of numbers representing RGBA percentage values,
    "range": optional array of numbers/percent strings representing a substring at [index, length]
  },
  { ... },
  { ... }
]
```

If `json_string` is not 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.

/cue/{cue_number}/text/format/alignment {alignment}

If `alignment` is given, set the text alignment of the specified cue to `alignment`. If not, return the alignment of the specified cue. `alignment` may be `left`, `center`, `right`, or `justify`.

/cue/{cue_number}/text/format/fontFamily

Read-only; return the font family name ("Helvetica", "Courier New", etc.) used in the specified cue.

You may optionally send this command in the form: `/cue/{cue_number}/text/format/fontFamily/{index}/{length}`, in which `index` is a whole number or a percentage string (e.g. `53%`) which lets you specify the start of a substring, and `length` is a number or percentage string which specifies the length of that substring. The first character is index `0`. For example:

`/cue/1/text/format/fontFamily/4/12` will return the font family name used for characters 5 through 17 of cue 1.

You may set length to `-1` to specify "to end the of the string."

You may also optionally send this command in the form: `/cue/{cue_number}/text/format/fontFamily/word/{word_index}`, in which `word_index` is a whole number which specifies a single word within the text of the cue. The first word is word `0`.

/cue/{cue_number}/text/format/fontStyle

Read-only; return the style ("Bold Oblique", "Regular", etc.) used in 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}

If `family` and `style` are given, set the font family and style of the text of the specified cue to `family` and `style`. Otherwise, return the font family and style for the specified cue. 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}

If `name` is given, set the font of the text of the specified cue. If not, return the name of the font used for 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}

If `number` is specified, set the font size of the text of the specified cue to `number`. If not, return the font size of the text of the specified cue.

This command may use increment and decrement syntax as described above.

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/lineSpacing {number}

If `number` is specified, set the line spacing of the text of the specified cue to `number`. If not, return the line spacing of the text of the specified cue.

This command may use increment and decrement syntax as described above.

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/color {red} {green} {blue} {alpha}
/cue/{cue_number}/text/format/backgroundColor {red} {green} {blue} {alpha}
/cue/{cue_number}/text/format/strikethroughColor {red} {green} {blue} {alpha}
/cue/{cue_number}/text/format/underlineColor {red} {green} {blue} {alpha}

If `red`, `green`, `blue`, and `alpha` are specified, set the given color attribute of the specified cue to the corresponding color. If not, return the color for the given attribute of the specified cue. `red`, `green`, `blue`, and `alpha` can be decimal numbers between 0.0 and 1.0, 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/strikethroughStyle {style}
/cue/{cue_number}/text/format/underlineStyle {style}

If `style` is given, set the strikethrough or underline style of the text of the specified cue to `style`. Otherwise, return the strikethrough or underline style for the specified cue. `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}/liveText/outputSize

Read-only; return a two-item array containing the width and height of the text or liveText of the specified cue.

“Live” methods are the same as their non-live counterparts, but operate on the active, “live” value of a running cue, rather than changing the “start state” of the cue. Invoking these methods does not cause the document to have unsaved changes.

Light cue methods

Methods specific to Light cues.

/cue/{cue_number}/alwaysCollate {number}

Get or set the state of the *Always collate* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the *Always collate* specified cue.

/cue/{cue_number}/collateAndStart

Collate and start the specified cue.

/cue/{cue_number}/lightCommandText {string}

If `string` is given, set the full command text of the specified cue to `string`. If not, return the full command text of the specified cue.

/cue/{cue_number}/prune
/cue/{cue_number}/pruneCommands

Prune the command text of the specified cue. Pruning removes any commands which have no effect. These two commands are equivalent.

/cue/{cue_number}/removeLightCommandsMatching {string}

If the specified cue contains a command that matches `string`, remove that command. Otherwise, do nothing.

/cue/{cue_number}/replaceLightCommand {old_command} {new_command}

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.

This method replaces `/updateLightCommand` starting in QLab 4.4.

/cue/{cue_number}/safeSort
/cue/{cue_number}/safeSortCommands

Lexically (alphabetically) sort the command text of the specified cue, as long as sorting doesn’t change the cue’s output. These two commands are equivalent.

/cue/{cue_number}/setLight {string} {setting}

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}/updateLightCommand {string} {number}

If the specified cue contains a command for the instrument or group `string`, then update that command to the specified value `number`. If the specified cue does not contain a command for the instrument or group `string`, then add a command in the form `string = number`. `string` can be the name of an instrument or group, with or without a parameter. `number` must be an acceptable value for the specified instrument or group.

This method is deprecated starting in QLab 4.4, and is replaced by `/replaceLightCommand`.

Fade cue methods

Methods specific to Fade cues.

/cue/{cue_number}/doOpacity {number}
/cue/{cue_number}/doRate {number}
/cue/{cue_number}/doRotation {number}
/cue/{cue_number}/doScale {number}
/cue/{cue_number}/doTranslation {number}

Get or set the state of the geometry parameter checkboxes of the specified cue. `number` is interpreted as a boolean; `0` equals false, any other number equals true. If no number is given, return the state of the relevant geometry parameter checkbox of the specified cue.

/cue/{cue_number}/gang inChannel outChannel {gang}
/cue/{cue_number}/gang/inChannel/outChannel {gang}

If `gang` is given, set the gang of crosspoint `(inChannel, outChannel)` of the specified cue to `gang`. If not, return the gang of crosspoint `(inChannel, outChannel)` of the specified cue.

`inChannel` must be a whole number between 0 and 24; `outChannel` must be a whole number between 0 and 64. 0 is the master column or row.

`decibel`, if given, must be a whole number, decimal number, or a string. If `decibel` is sent as a string (e.g. "-inf") QLab will use the minimum decibel value set in workspace settings.

/cue/{cue_number}/geoMode {number}

If `number` is given, set the geometry fade mode of the specified cue. Mode `0` is absolute fade, mode `1` is relative fade. If `number` is not given, return the geometry fade mode of the specified cue.

/cue/{cue_number}/level inChannel outChannel {decibel}
/cue/{cue_number}/level/inChannel/outChannel {decibel}

If `decibel` is given, set the level of crosspoint `(inChannel, outChannel)` of the specified cue to `decibel`. If not, return the level of crosspoint `(inChannel, outChannel)` of the specified cue.

`inChannel` must be a whole number between 0 and 24; `outChannel` must be a whole number between 0 and 64. 0 is the master column or row.

`decibel`, if given, must be a whole number, decimal number, or a string. If `decibel` is sent as a string (e.g. "-inf") QLab will use the minimum decibel value set in workspace settings.

/cue/{cue_number}/levels

Read-only; return all the audio levels currently available in the specified cue's inspector. The levels are returned as an array of arrays, like so: `[row0Array, row1Array, row2Array, ...]`

Row 0 of `/levels` is equivalent to the results of the `/sliderLevels` method.

/cue/{cue_number}/mode {number}

If `number` is given, set the fade mode of the specified cue. Mode 0 is absolute fade, mode 1 is relative fade. If `number` is not given, return the fade mode of the specified cue.

/cue/{cue_number}/opacity {number}

If `number` is given, set the opacity of the specified cue to `number`. If not, return the opacity of the specified cue. `number` can be any decimal number between 0 and 1.

/cue/{cue_number}/preserveAspectRatio {number}

Get or set the state of the *Preserve aspect ratio* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the *Preserve aspect ratio* checkbox of the specified cue.

/cue/{cue_number}/quaternion {number number number number}

If four `numbers` are given, set the rotation of the specified cue. If not, return an array of four numbers representing the cue's rotation as a quaternion. `number` can be any decimal number. **Caution:** you need to understand quaternion math to make any meaningful use of this method. Quaternion math is really hard. Good luck!

/cue/{cue_number}/rate {number}

If `number` is given, set the rate of the specified cue to `number`. If not, return the rate of the specified cue. `number` can be any positive whole or decimal number from 0.03 to 33.0.

/cue/{cue_number}/resetRotation

Reset the rotation of the specified cue. If the specified cue is using single-axis rotation, this has no visible effect; use `/rotation` instead.

/cue/{cue_number}/rotateX {number}
/cue/{cue_number}/rotateY {number}
/cue/{cue_number}/rotateZ {number}

If the specified cue is using 3D orientation, add `number` to the current quaternion rotation of the specified cue. `number` can be any decimal number. If the specified cue is using single-axis rotation, this method has no effect; use `/rotation` instead.

/cue/{cue_number}/rotation {number}

If `number` is given, and if the specified cue is using single-axis rotation, set the rotation in degrees to `number`. If no `number` is given, return the current rotation of the specified cue. For 3D orientation, use `/quaternion` or `/rotate{N}`.

/cue/{cue_number}/rotationType {number}

If `number` is given, set the rotation type of the specified cue to `number`. If not, return the rotation type of the specified cue. Valid rotation types are:

```
0 - 3D orientation
1 - X rotation
2 - Y rotation
3 - Z rotation
```

/cue/{cue_number}/scaleX {number}

If `number` is given, set the X-axis scale of the specified cue to `number`. If not, return the X-axis scale of the specified cue. `number` can be any decimal number.

/cue/{cue_number}/scaleY {number}

If `number` is given, set the Y-axis scale of the specified cue to `number`. If not, return the Y-axis scale of the specified cue. `number` can be any decimal number.

/cue/{cue_number}/scale {x} {y}

If `x` and `y` are given, set the scale of the specified cue to `(x,y)`. If not, return the current scale of the specified cue. `x` and `y` can be any decimal numbers.

/cue/{cue_number}/setLevelsFromTarget

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 then this command has no effect.

/cue/{cue_number}/setSilentLevels

Set the audio levels of the specified cue to silent.

/cue/{cue_number}/setGeometryFromTarget

Set the geometry of the specified cue to the geometry of its target cue. If the target cue has no geometry then this command has no effect.

/cue/{cue_number}/sliderLevel {channel} {decibel}
/cue/{cue_number}/sliderLevel/{channel} {decibel}

Get or set a single output slider volume level.

`channel` is either an integer from 0 to 64, or a string (the cue output name). 0 is the cue master slider.

`decibel` is an optional whole or decimal number. When present it is the decibel value to set.

If `decibel` is sent as a string (e.g. "-inf") QLab will use the minimum decibel value set in workspace settings.

If no `decibel` is given, return the volume level of the specified output slider.

/cue/{cue_number}/sliderLevels

Read-only; return an array of the output levels of the specified cue, including the cue master. The array is therefore 65 numbers.

/cue/{cue_number}/stopTargetWhenDone {number}

Get or set the state of the *Stop target when done* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the *Stop target when done* checkbox of the specified cue.

/cue/{cue_number}/translationX {number}

If `number` is given, set the X-axis translation of the specified cue to `number`. If not, return the X-axis translation of the specified cue. `number` can be any decimal number.

/cue/{cue_number}/translationY {number}

If `number` is given, set the Y-axis translation of the specified cue to `number`. If not, return the Y-axis translation of the specified cue. `number` can be any decimal number.

/cue/{cue_number}/translation {x} {y}

If `x` and `y` are given, set the translation of the specified cue to `(x,y)`. If not, return the current translation of the specified cue. `x` and `y` can be any decimal numbers.

/cue/{cue_number}/willFade {row} {column} {number}

If `number`, `row` and `column` are all given, set the active state (i.e. yellow or grey) of crosspoint { `row`, `column` }. `number` is interpreted as a boolean; `0` equals false, any other number equals true. `row` and `column` must be whole numbers.

If `row` and `column` are given, but not `number`, return the active state of crosspoint { `row`, `column` }. 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, ...]`

Network cue methods

Methods specific to Network cues.

/cue/{cue_number}/customString {string}

If `string` is given, set the OSC message of the specified cue to `string`. If not, return the OSC message of the specified cue.

/cue/{cue_number}/messageType {number}

If `number` is given, set the message type of the specified cue to `number`. If not, return the message type of the specified cue. Valid message types are:

```
1 - QLab message
2 - OSC message
3 - UDP message
```

/cue/{cue_number}/qlabCommand {number}

If `number` is given, set the QLab command of the specified cue to `number`. If not, return the message type of the specified cue. Valid QLab commands are:

```
1  - start
2  - stop
3  - hardStop
4  - pause
5  - resume
6  - togglePause
7  - load
8  - preview
9  - reset
10 - panic
11 - number
12 - name
13 - notes
14 - cueTargetNumber
15 - preWait
16 - duration
17 - postWait
18 - continueMode
19 - flagged
20 - armed
21 - colorName
```

/cue/{cue_number}/patch {number}

If `number` is given, set the OSC patch of the specified cue to `number`. If not, return the OSC patch of the specified cue. `number` must be a whole number from 1 to 16, inclusive.

/cue/{cue_number}/qlabCueNumber {string}

If `string` is given, set the target cue number for the command of the specified cue to `string`. If not, return the target cue number of the command of the specified cue.

/cue/{cue_number}/qlabCueParameters {string}

If `string` is given, set the parameters for the command of the specified cue to `string`. If not, return the parameters of the command of the specified cue.

/cue/{cue_number}/rawString {string}
/cue/{cue_number}/udpString {string}

If `string` is given, set the UDP string of the specified cue to `string`. If not, return the UDP string of the specified cue.

MIDI cue methods

Methods specific to MIDI cues.

/cue/{cue_number}/byte1 {number}

If `number` is given, set byte 1 of the MIDI message of the specified cue to `number`. If not, return byte 1 of the MIDI message of the specified cue. `number` must be a whole number from 0 to 127.

/cue/{cue_number}/byte2 {number}

If `number` is given, set byte 2 of the MIDI message of the specified cue to `number`. If not, return byte 2 of the MIDI message of the specified cue. `number` must be a whole number from 0 to 127.

/cue/{cue_number}/byteCombo {number}

If `number` is given, set both bytes of the MIDI message of the specified cue based on `number`. If not, return the value of bytes 1 and 2 (as a single number) of the MIDI message of the specified cue. `number` must be a whole number from 0 to 16383.

/cue/{cue_number}/channel {number}

If `number` is given, set the MIDI channel of the specified cue to `number`. If not, return the MIDI channel of the specified cue. `number` must be a whole number from 1 to 16.

/cue/{cue_number}/command {number}

If `number` is given, set the MSC command of the specified cue to `number`. If not, return the MSC command of the specified cue. `number` must be a whole number from 0 to 127, but only the following values are meaningful:

```
1  - GO
2  - STOP
3  - RESUME
4  - TIMED_GO
5  - LOAD
6  - SET
7  - FIRE
8  - ALL_OFF
9  - RESTORE
10 - RESET
11 - GO_OFF
16 - GO/JAM_CLOCK
17 - STANDBY_+
18 - STANDBY_-
19 - SEQUENCE_+
20 - SEQUENCE_-
21 - START_CLOCK
22 - STOP_CLOCK
23 - ZERO_CLOCK
24 - SET_CLOCK
25 - MTC_CHASE_ON
26 - MTC_CHASE_OFF
27 - OPEN_CUE_LIST
28 - CLOSE_CUE_LIST
29 - OPEN_CUE_PATH
30 - CLOSE_CUE_PATH
```

/cue/{cue_number}/commandFormat {number}

If `number` is given, set the MSC command format of the specified cue to `number`. If not, return the MSC command format of the specified cue. `number` must be a whole number from 0 to 127, but only the following values are meaningful:

```
127 - All Types
1   - Lighting (General)
2   - Moving Lights
3   - Color Changers
4   - Strobes
5   - Lasers
6   - Chasers
16  - Sound (General)
17  - Music
18  - CD Players
19  - EPROM Playback
20  - Audio Tape Machines
21  - Intercoms
22  - Amplifiers
23  - Audio Effects Devices
24  - Equalizers
32  - Machinery (General)
33  - Rigging
34  - Flys
35  - Lifts
36  - Turntables
37  - Trusses
38  - Robots
39  - Animation
40  - Floats
41  - Breakaways
42  - Barges
48  - Video (General)
49  - Video Tape Machines
50  - Video Cassette Machines
51  - Video Disc Players
52  - Video Switchers
53  - Video Effects
54  - Video Character Generators
55  - Video Still Stores
56  - Video Monitors
64  - Projection (General)
65  - Film Projectors
66  - Slide Projectors
67  - Video Projectors
68  - Dissolvers
69  - Shutter Controls
80  - Process Control (General)
81  - Hydraulic Oil
82  - H2O
83  - CO2
```

```
84  - Compressed Air
85  - Natural Gas
86  - Fog
87  - Smoke
88  - Cracked Haze
96  - Pyrotechnics (General)
97  - Fireworks
98  - Explosions
99  - Flame
100 - Smoke Pots
```

/cue/{cue_number}/controlNumber {number}

If `number` is given, set the MSC control number of the specified cue to `number`. If not, return the MSC control number of the specified cue. `number` must be a whole number from 0 to 16383.

/cue/{cue_number}/controlValue {number}

If `number` is given, set the MSC control value of the specified cue to `number`. If not, return the MSC control value of the specified cue. `number` must be a whole number from 0 to 16383.

/cue/{cue_number}/deviceID {number}

If `number` is given, set the outgoing MSC device ID of the specified cue to `number`. If not, return the outgoing MSC device ID of the specified cue. `number` must be a whole number from 0 to 127.

/cue/{cue_number}/doFade {number}

Get or set the state of the MIDI *fade* checkbox of the specified cue. `number` is interpreted as a boolean; `0` equals false, any other number equals true. If no number is given, return the state of the MIDI fade checkbox of the specified cue.

/cue/{cue_number}/endValue {number}

If `number` is given, set the fade ending value of the MIDI message of the specified cue to `number`. If not, return the fade ending value of the MIDI message of the specified cue. `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}

If `number` is given, set the appropriate section of the MSC timecode of the specified cue to `number`. If not, return the appropriate section of MSC timecode hours of the specified cue.

/cue/{cue_number}/macro {number}

If `number` is given, set the MSC macro of the specified cue to `number`. If not, return the MSC macro of the specified cue. `number` must be a whole number from 0 to 127.

/cue/{cue_number}/messageType {number}

If `number` is given, set the message type of the specified cue to `number`. If not, return the message type of the specified cue. Valid message types are:

```
1 - MIDI Voice Message ("Musical MIDI")
2 - MIDI Show Control Message (MSC)
3 - MIDI SysEx Message
```

/cue/{cue_number}/patch {number}

A number from 1 to 8.

/cue/{cue_number}/qList {string}

If `number` is given, set the outgoing MSC cue list number of the specified cue to `number`. If not, return the outgoing MSC cue list number of the specified cue.

/cue/{cue_number}/qNumber {string}

If `number` is given, set the outgoing MSC cue number of the specified cue to `number`. If not, return the outgoing MSC cue number of the specified cue.

/cue/{cue_number}/qPath {string}

If `number` is given, set the outgoing MSC cue path number of the specified cue to `number`. If not, return the outgoing MSC cue path number of the specified cue.

/cue/{cue_number}/rawString {string}

If `string` is given, set the MIDI SysEx of the specified cue to `string`. If not, return the MIDI SysEx string of the specified cue. `string` must be a valid SysEx string, formatted in hexadecimal, and omitting the starting `F0` and ending `F7`.

/cue/{cue_number}/status {number}

If `number` is given, set the MIDI message type of the specified cue to `number`. If not, return the MIDI message type of the specified cue. 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}

If `number` is given, set the MSC timecode format of the specified cue to `number`. If not, return the MSC timecode format of the specified cue. Valid formats are:

```
0 - 24 fps
1 - 25 fps
2 - 30 fps drop
3 - 30 fps non-drop
```

/cue/{cue_number}/timecodeString {string}

If `number` is given, set the MSC timecode string of the specified cue to `number`. If not, return the MSC timecode string of the specified cue.

MIDI file cue methods

Methods specific to MIDI File cues.

/cue/{cue_number}/patch {number}

If `number` is given, set the patch of the specified cue. If not, return the patch of the specified cue. `number` must be a whole number from 1 to 8.

/cue/{cue_number}/rate {number}

If `number` is given, set the rate of the specified cue. If not, return the rate of the specified cue.

Devamp cue methods

Methods specific to Devamp cues.

/cue/{cue_number}/startNextCueWhenSliceEnds {number}

Get or set the state of the *Start next cue when slice ends* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the *Start next cue when slice ends* checkbox of the specified cue.

/cue/{cue_number}/stopTargetWhenSliceEnds {number}

Get or set the state of the *Stop target when slice ends* checkbox of the specified cue. `number` is interpreted as a boolean; 0 equals `false`, any other number equals `true`. If no number is given, return the state of the *Stop target when slice ends* checkbox of the specified cue.

Script cue methods

Methods specific to Script cues.

/cue/{cue_number}/compileSource

Compile the script of the specified cue.

/cue/{cue_number}/scriptSource

Read-only; return the contents of the script of the specified cue.

OSC Queries

OSC querying is a powerful new capability of the Network cue which allows you to utilize the current, live value of any property that can be accessed via OSC 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 what 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:

```
/device/standby #/cue/selected/number#
```

The hashmarks indicate that the message contained with them is a query, and when the message is sent, the query is replaced with the *result* of that query. You could think of it 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 the Network 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 for as long as it was running.

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.

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.

The dictionary is 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.

Commands

clear

v

clear *light dashboard* : The Light Dashboard you want to clear.

Example use

```
tell application id "com.figure53.QLab.4" to tell front workspace
  set theDashboard to current light dashboard
  tell theDashboard to clear
end tell
```

Example use, alternate syntax

```
tell application id "com.figure53.QLab.4" to tell front workspace
  set theDashboard to current light dashboard
  clear theDashboard
end tell
```

---

collapse

v : Collapse the Group cue in the cue list.

collapse *Group cue* : the Group cue that will collapse in the cue list.

---

collateAndStart

v

collateAndStart light cue : The Light cue you want to collate and start.

---

compile

v

compile script cue : The Script cue whose source you want to recompile.

---

expand

v : Expand the Group cue in the cue list.

expand Group cue : the Group cue that will expand in the cue list.

---

getGang

v

getGang cue : The cue for which you want to adjust the gang.
row integer : The row of the level matrix. Row 0 = the master and output levels.
column integer : The column of the level matrix. Column 0 = the master and input levels.
returns text : The value of the gang at the specified location in the matrix.

---

getLevel

v

getLevel cue : The cue for which you want to get the volume level.
row integer : The row of the level matrix. Row 0 = the master and output levels.
column integer : The column of the level matrix. Column 0 = the master and input levels.
returns real : The value in decibels of the level at the specified location in the matrix.

---

go

v : Make a workspace GO.

go specifier : The workspace to GO.

---

hardStop

v : Hard stop one or more cues or workspaces.

hardStop specifier : The cue(s) or workspace(s) to stop.

---

load

v : Load a cue or workspace to a given time.

load specifier : The cue(s) or workspace(s) to load. Because “load” is both a noun and a verb in QLab (load vs. Load cue), you need to place the specifier within parentheses.
[**time** real] : Load time.

Example uses:

- `load (cue "2")` - loads the cue with cue number 2.
- `load (cue 2)` - loads cue with cue ID 2.
- `load cue 2` - **does not work!** AppleScript interprets this as a reference to a Load cue with an extraneous, and thus error-causing, “2” hanging around.
- `load (cue "2") time 10` - loads cue number 2 to ten seconds from the beginning. If cue number 2 has no duration, then the `time 10` part is ignored.

---

make

v

make workspace
type text : Name of the kind of cue you want to make. (Audio, Video, Camera, MIDI, etc.) Pass “cue list” to make a new cue list, or “cue cart” to make a new cue cart.

---

movePlayheadDown

v : Move the playhead to the next cue.

movePlayheadDown specifier : The workspace whose playhead will change.

---

movePlayheadDownASequence

v : Move the playhead to the top of the next cue sequence.

movePlayheadDownASequence specifier : The workspace whose playhead will change.

---

movePlayheadUp

v : Move the playhead to the previous cue.

movePlayheadUp specifier : The workspace whose playhead will change.

---

movePlayheadUpASequence

v : Move the playhead to the top of the previous cue sequence.

movePlayheadUpASequence specifier : The workspace whose playhead will change.

---

moveSelectionDown

v : Select the next cue.

moveSelectionDown specifier : The workspace whose selection will change.

---

moveSelectionUp

v : Select the previous cue.

moveSelectionUp specifier : The workspace whose selection will change.

---

newCueWithAll

v

newCueWithAll light dashboard : The Light Dashboard to which you want to add a new cue.

---

newCueWithChanges

v

newCueWithChanges light dashboard : The Light Dashboard to which you want to add a new cue.

---

panic

v : Panic one or more cues or workspaces.

panic specifier : The cue(s) or workspace(s) to panic.

---

pause

v : Pause one or more cues or workspaces.

pause specifier : The cue(s) or workspace(s) to pause.

---

preview

v : Preview one or more cues.

Previewing starts only the action of the cue, skipping any pre-wait and not continuing to other cues.
preview specifier : The cue(s) to preview.

---

prune

v

prune light cue : The Light cue whose command text you want to prune.

---

recordAllToLatest

v

recordAllToLatest light dashboard : The Light Dashboard to which you want to add a new cue.

---

recordAllToSelected

v

recordAllToSelected light dashboard : The Light Dashboard to which you want to add a new cue.

---

redo

v

redo specifier.

---

removeLightCommandsMatching

v : Remove existing light commands in the specified cue matching the command provided.

removeLightCommandsMatching cue : The cue for which you want to remove light level commands.
command text : The full text of the light level command you want to remove.

---

replaceLightCommand

v : Replace the provided light command text in the specified cue with new command text.

updateLightCommand cue : The old light level command text that will be replaced.
oldCommandText text : The instrument or group name (left side) of the light level command.
newCommandText text : The new light level command text that will replace the old one.

---

reset

v : Reset one or more cues or workspaces.

reset specifier : The cue(s) or workspace(s) to reset.

---

revert

v

revert light dashboard : The Light Dashboard in which you want to revert changes.

---

setGang

v

setGang cue : The cue for which you want to adjust the gang.
row integer : The row of the level matrix. Row 0 = the master and output levels.
column integer : The column of the level matrix. Column 0 = the master and input levels.
gang text : The gang value to set.

---

setLevel

v

setLevel cue : The cue for which you want to adjust the volume level.
row integer : The row of the level matrix. Row 0 = the master and output levels.
column integer : The column of the level matrix. Column 0 = the master and input levels.
db real : The decibel value for the volume level.

---

setLight

v : Add a light command to the specified cue or Light Dashboard in the form ‘selector’ = ‘value’. Selector can be the name of an instrument or group, with or without a value parameter.

setLight specifier : The cue or Light Dashboard for which you want to add a light level command.
selector text : The instrument or group name (left side) of the light level command.
value real or text : Optional parameter value (right side) of the light level command.

Example use

```
tell application id "com.figure53.QLab.4" to tell front workspace
  set theDashboard to current light dashboard

  -- set instrument "1" to 10% in the dashboard
  setLight theDashboard selector "1" value 10

  -- set instrument "backlight" to 50% over 5.3 seconds in the dashboard
  -- note: fade time always resets back to 0 after calling setLight
  set theFadeTime to 5.3
  set dashboard fade time of theDashboard to theFadeTime
  setLight theDashboard selector "backlight" value 50

  -- set instrument "5" to 25% in cue "30"
  setLight cue "30" selector "5" value "25"

end tell
```

---

start

v : Start one or more cues or workspaces.

start specifier : The cue(s) or workspace(s) to start. For a workspace, “start” means to unpause all paused cues.

---

stop

v : Stop one or more cues or workspaces.

stop specifier : The cue(s) or workspace(s) to stop.

---

undo

v

undo specifier.

---

updateLatestCue

v

updateLatestCue light dashboard : The Light Dashboard you want to update.

---

updateLightCommand

v : Update an existing light command in the specified cue. If the specified cue does not contain a selector for the instrument or group provided, then add the command.

> **NOTE:** starting with QLab 4.4, use `replaceLightCommand` instead of `updateLightCommand`. updateLightCommand is deprecated and will be removed in a future version of QLab.

updateLightCommand cue : The cue for which you want to update a light level command.
selector text : The instrument or group name (left side) of the light level command.
value real : Optional parameter value (right side) of the light level command.

---

updateOriginatingCues

v

updateOriginatingCues light dashboard : The Light Dashboard you want to update.

---

updateSelectedCues

v

updateSelectedCues light dashboard : The Light Dashboard you want to update.

Classes

audio cue

n [inherits from *cue*] : An Audio Cue.

Properties

- **patch** (integer) : Audio device patch number.
- **start time** (real) : Time in the file where playback begins.
- **end time** (real) : Time in the file where playback ends.
- **play count** (integer) : Number of times the audio between the start and end times plays. Always >= 1.
- **infinite loop** (boolean) : Does the cue loop infinitely?
- **rate** (real) : Playback rate of the audio cue.
- **integrated fade** (enabled|disabled) : State of the integrated fade - enabled or disabled.
- **lock fade to cue** (enabled|disabled) : Whether the integrated fade should lock to the start/end times of the cue (enabled or disabled).
- **audio input channels** (integer, r/o) : The number of audio input channels for the cue. (i.e. The number of distinct channels in the file.)
- **pitch shift** (enabled|disabled) : Whether pitch shifting is enabled or disabled.
- **slice markers** (list of *slice marker record*) : List of slice marker records.
- **last slice play count** (integer) : Number of times the final slice plays. Always >= 1.
- **last slice infinite loop** (boolean) : Does the last slice loop infinitely?

Responds to

- getLevel, setLevel, getGang, setGang.

---

camera cue

n [inherits from *cue*] : A Camera Cue.

Properties

- **camera patch** (integer) : Camera patch number.
- **layer** (integer) : Display layer of the video.
- **full screen** (boolean) : Is the cue displaying in full surface mode?
- **full surface** (boolean) : Is the cue displaying in full surface mode?
- **preserve aspect ratio** (boolean) : Does the cue preserve aspect ratio?
- **opacity** (real) : Video opacity.
- **translation x** (real) : Translation along the x axis.
- **translation y** (real) : Translation along the y axis.
- **scale x** (real) : Scale along the x axis.
- **scale y** (real) : Scale along the y axis.
- **do video effect** (boolean) : Apply a video effect?
- **custom quartz file** (file) : The custom Quartz Composer file used to render this cue.

---

cue

n : A cue.

Elements

- contains cues; contained by workspaces, cues.

Properties

- **uniqueID** (text, r/o) : The unique ID of the cue.
- **parent** (cue, r/o) : The parent cue of this cue.
- **parent list** (cue, r/o) : The parent cue list (or cue cart) that contains this cue.
- **q type** (text, r/o) : The name of this kind of cue, e.g. “Audio”, “Video”, “MIDI”, etc.
- **q number** (text) : The number of the cue. Unique if present.
- **q name** (text) : The name of the cue. Not unique.
- **q list name** (text, r/o) : The name of the cue as displayed in the cue list. (i.e. Might be a default name.)
- **q display name** (text, r/o) : The name of the cue as displayed in the standby view. (i.e. Never empty.)
- **q default name** (text, r/o) : The name QLab would give the cue by default, if any.
- **q color** (text) : The name of the cue color, or ‘none’ if not set.
- **notes** (text) : The notes for this cue.
- **cue target** (cue) : The cue this cue targets, if any.
- **file target** (file) : The file this cue targets, if any.
- **pre wait** (real) : The time in seconds before the action is triggered.
- **duration** (real) : The duration of the cue’s action in seconds. Not editable for all cue types.
- **temp duration** (real) : The temporary duration of the cue’s action in seconds. Not all cues allow editing the duration. Setting the temporary duration does not mark the document as edited. Reset the cue to go back to the original duration.
- **current duration** (real, r/o) : The current active duration of the cue’s action in seconds. This property reflects the temporary duration, if it has been set. Otherwise it returns the cue’s duration.
- **post wait** (real) : The time in seconds until continuing on to the next cue.
- **continue mode** (do_not_continue, auto_continue, auto_follow) : Continue mode of the cue.
- **flagged** (boolean) : Is this cue flagged?
- **autoload** (boolean) : Does this cue auto load?
- **armed** (boolean) : Is this cue armed?
- **hotkey trigger** (enabled|disabled) : State of the HotKey trigger.
- **midi trigger** (enabled|disabled) : State of the MIDI trigger.
- **midi command** (note_on, note_off, program_change, control_change, key_pressure, channel_pressure, pitch_bend) : Type of MIDI command that will trigger the cue. (NOTE: pitch_bend messages are NOT accepted as remote MIDI triggers.)
- **midi byte one** (integer) : Byte 1 of the MIDI trigger.
- **midi byte two** (integer) : Byte 2 of the MIDI trigger.
- **midi byte one string** (text) : Display String of Byte 1 of the MIDI trigger.
- **midi byte two string** (text) : Display String of Byte 2 of the MIDI trigger.
- **timecode trigger** (enabled|disabled) : State of the timecode trigger.
- **timecode show as timecode** (boolean) : True if timecode trigger is shown as Timecode, false if shown as Real Time.
- **timecode hours** (integer) : Hours field of the timecode trigger.
- **timecode minutes** (integer) : Minutes field of the timecode trigger.
- **timecode seconds** (integer) : Seconds field of the timecode trigger.
- **timecode frames** (integer) : Frames field of the timecode trigger.
- **timecode bits** (integer) : Bits field of the timecode trigger.
- **wall clock trigger** (enabled|disabled) : State of the wall clock trigger.
- **wall clock hours** (integer) : Hours field of the wall clock trigger.
- **wall clock minutes** (integer) : Minutes field of the wall clock trigger.
- **wall clock seconds** (integer) : Seconds field of the wall clock trigger.

- **loaded** (boolean, r/o) : Is this cue loaded?
- **running** (boolean, r/o) : Is this cue running?
- **paused** (boolean, r/o) : Is this cue paused?
- **broken** (boolean, r/o) : Is this cue broken?
- **pre wait elapsed** (real, r/o) : The time in seconds that have elapsed on the pre wait.
- **action elapsed** (real, r/o) : The time in seconds that have elapsed in the action of the cue.
- **post wait elapsed** (real, r/o) : The time in seconds that have elapsed on the post wait.
- **percent pre wait elapsed** (real, r/o) : The percent of the pre wait that has elapsed.
- **percent action elapsed** (real, r/o) : The percent of the cue's action that has elapsed.
- **percent post wait elapsed** (real, r/o) : The percent of the post wait that has elapsed.

Responds to

- load, preview, start, pause, stop, hardStop, reset, panic.

---

cue list

n [inherits from group cue > cue] : A cue list.

Elements

- contained by workspaces.

Properties

- **playback position** (cue) : The playback position (playhead) of this cue list is the cue that will start at the next GO.
- **playback position** (cue) : The playhead (playback position) of this cue list is the cue that will start at the next GO.
- **sync to timecode** (enabled|disabled) : Sync the cues in this cue list to incoming timecode.
- **sync mode** (mtc, ltc) : Which kind of incoming timecode this cue list listens for.
- **smpte format** (fps_24, fps_25, fps_30_drop, fps_30_non_drop) : SMPTE format of the incoming timecode.
- **mtc sync source name** (text) : Name of the MIDI device which feeds us MTC timecode.
- **ltc sync channel** (integer) : Audio channel that carries the LTC signal.

---

devamp cue

n [inherits from *cue*] : A Devamp Cue.

Properties

- **fire next cue when slice ends** (boolean) : start the next cue at the moment the target slice ends?
- **stop target when slice ends** (boolean) : Stop the target at the moment the target slice ends?

---

fade cue

n [inherits from *cue*] : A Fade Cue.

Properties

- **fade mode** (absolute, relative) : Absolute or relative mode.
- **audio fade mode** (absolute/relative) : Audio Levels absolute or relative mode.
- **video fade mode** (absolute/relative) : Video Geometry absolute or relative mode.
- **stop target when done** (boolean) : Stop the target when this cue completes?
- **opacity** (real) : Video opacity.
- **translation x** (real) : Translation along the x axis.
- **translation y** (real) : Translation along the y axis.
- **scale x** (real) : Scale along the x axis.
- **scale y** (real) : Scale along the y axis.
- **preserve aspect ratio** (boolean) : Does the cue preserve aspect ratio?
- **rotation type** (integer) : 3D orientation (0), x-axis (1), y-axis (2), or z-axis (3).
- **rotation** (real) : Rotation in degrees when the cue 'rotation type' is set to a single-axis value (1, 2, or 3). Returns 0.0 when 'rotation type' is 3D orientation (0).
- **do opacity** (boolean) : Does the cue animate opacity?
- **do translation** (boolean) : Does the cue animate translation?
- **do scale** (boolean) : Does the cue animate scale?
- **do rotation** (boolean) : Does the cue animate rotation?

Responds to

- getLevel, setLevel, getGang, setGang.

---

group cue

n [inherits from cue] : A Group Cue.

Properties

- **mode** (cue_list, timeline, fire_first_enter_group, fire_first_go_to_next_cue, fire_random) : The start style of this group.

---

light cue

n [inherits from *cue*] : A Light Cue.

Properties

- **command text** (text) : The light cue command text.
- **always collate** (boolean) : Flag for whether this cue should always collate the effects of previous light cues in the same list when it runs.

Responds to

- flatten, prune, safeSort, collateAndStart, setLight, replaceLightCommand, updateLightCommand, removeLightCommandsMatching.

---

light dashboard

n [inherits from *item*]

Properties

- **dashboard visibility** (boolean) : Whether the Light Dashboard is currently visible.
- **dashboard mode** (sliders/tiles) : The view mode of the Light Dashboard.
- **dashboard fade time** (real) : 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 light command.

Responds to

- setLight, clear, updateLatestCue, updateOriginatingCues, updateSelectedCues, newCueWithAll, newCueWithChanges, recordAllToLatest, recordAllToSelected, revert, redo, undo.

Example use

```
tell application id "com.figure53.QLab.4" to tell front workspace
  set theDashboard to current light dashboard

  -- change display mode
  set theMode to dashboard mode of theDashboard
  set dashboard mode of theDashboard to sliders
  set dashboard mode of theDashboard to tiles

  -- toggle visibility
  set theVisibility to dashboard visibility of theDashboard
  set dashboard visibility of theDashboard to not theVisibility

end tell
```

---

load cue

n [inherits from *cue*] : A Load Cue.

Properties

- **load time** (real) : Load target cue to this time.

---

mic cue

n [inherits from *cue*] : A Microphone Cue.

Properties

- **patch** (integer) : Audio device patch number.
- **audio input channels** (integer, r/o) : The number of audio input channels for the cue.

Responds to

- getLevel, setLevel, getGang, setGang.

---

midi cue

n [inherits from *cue*] : A MIDI Cue.

Properties

- **patch** (integer) : MIDI device patch number.
- **message type** (voice, msc, sysex) : The type of MIDI message.
- **command** (note_on, note_off, program_change, control_change, key_pressure, channel_pressure, pitch_bend) : The MIDI command.
- **channel** (integer) : MIDI channel number.
- **byte one** (integer) : First byte of the message.
- **byte two** (integer or string) : Second byte of the message. Since QLab accepts greater-than and less-than expressions, and “any”, as acceptable values for the second byte of some MIDI messages, this property can be sent as a string. If the current value of the property is a string, you must use a string in AppleScript to set the value.
- **byte combo** (integer) : Value when first and second bytes are interpreted as parts of one number. Used for pitch bend messages.
- **start value** (integer, r, o) : The start value for the MIDI fade.
- **end value** (integer) : The end value for the MIDI fade.
- **fade** (enabled|disabled) : State of the MIDI fade.
- **deviceID** (integer) : MIDI Show Control device ID.
- **command format** (integer) : MIDI Show Control command format.
- **command number** (integer) : MIDI Show Control command.
- **q_number** (text) : Q Number message parameter.
- **q_list** (text) : Q List message parameter.
- **q_path** (text) : Q Path message parameter.
- **macro** (integer) : MSC macro.
- **control number** (integer) : MSC control number.
- **control value** (integer) : MSC control value.
- **hours** (integer) : MSC hours parameter.
- **minutes** (integer) : MSC minutes parameter.
- **seconds** (integer) : MSC seconds parameter.
- **frames** (integer) : MSC frames parameter.
- **subframes** (integer) : MSC subframes parameter.
- **smpte format** (fps_24, fps_25, fps_30_drop, fps_30_non_drop) : SMPTE format of the timecode parameters.
- **send time with set** (boolean) : Send the timecode parameters with the SET command?
- **sysex message** (text) : The raw SysEx message. Use only hexadecimal characters and whitespace. Omit the starting F0 and the ending F7.

---

midi file cue

n [inherits from *cue*] : A MIDI File Cue.

Properties

- **patch** (integer) : MIDI device patch number.
- **rate** (real) : Playback rate of the MIDI File cue.

---

network cue

n [inherits from *cue*] : A Network Cue.

Properties

- **custom message** (text) : The custom OSC message, for custom type messages.
- **osc message type** (text) : The type of OSC message. Can be: qlab, custom, or udp.
- **patch** (integer) : Network destination patch number.
- **q_command** (number) : The QLab OSC command, for QLab type messages.
- **q_num** (text) : The QLab cue number, for QLab type messages.
- **q_params** (text) : The QLab command parameters, for QLab type messages. Not all messages have parameters.
- **udp message** (text) : The raw UDP message, for udp type messages.

---

overrides

n [inherits from *item*]

Properties

- **overrides visibility** (boolean) : Whether the Override Controls window is currently visible.
- **midi input enabled** (boolean) : Allow musical MIDI input (default ON)
- **midi output enabled** (boolean) : Allow musical MIDI output (default ON)
- **msc output enabled** (boolean) : Allow MSC output (default ON)
- **msc input enabled** (boolean) : Allow MSC input (default ON)
- **sysex input enabled** (boolean) : Allow SysEx (other than MSC and MTC) input (default ON)
- **sysex output enabled** (boolean) : Allow SysEx (other than MSC and MTC) output (default ON)
- **osc input enabled** (boolean) : Allow OSC input (default ON)
- **osc output enabled** (boolean) : Allow OSC output (default ON)
- **timecode input enabled** (boolean) : Allow timecode input (default ON)
- **timecode output enabled** (boolean) : Allow timecode output (default ON)
- **artnet enabled** (boolean) : Allow Art-Net input and output (default ON)

Example use

```
tell application id "com.figure53.QLab.4"
    set midi input enabled of overrides to false
end tell
```

---

preferences

n [inherits from *item*]

Properties

- **live fade preview** (boolean) : Live Fade Preview

Example use

```
tell application id "com.figure53.QLab.4"
    set live fade preview of preferences to true
end tell
```

---

script cue

n [inherits from *cue*] : A Script Cue.

Properties

- **script source** (text) : AppleScript source for the cue. The script will be recompiled when set.

Responds to

- compile.

---

target cue

n [inherits from *cue*] : A Target Cue.

Properties

- **assigned number** (text) : Number of cue to assign. The cue with this number will be assigned as the new target.

---

text cue

n [inherits from *video cue* > *cue*] : A Text Cue.

Properties

- **text** (text) : Text of the Text cue.
- **live text** (text) : Live text of the Text cue. Setting this does not mark the workspace as edited.
- **text format** (list of *text format record*) : The list of text formats in the text.
- **live text format** (list of *text format record*) : The list of text formats in the live text. Setting this does not mark the workspace as edited.
- **text alignment** (text) : Text alignment of the text cue. Possible values are “left”, “center”, “right”, and “justify”.
- **live text alignment** (text) : Text alignment of the live text of the text cue. Possible values are “left”, “center”, “right”, and “justify”. Setting this does not mark the workspace as edited.
- **layer** (integer) : Display layer of the video.
- **full screen** (boolean) : Is the cue displaying in full surface mode?
- **full surface** (boolean) : Is the cue displaying in full surface mode?
- **preserve aspect ratio** (boolean) : Does the cue preserve aspect ratio?
- **opacity** (real) : Video opacity.
- **translation x** (real) : Translation along the x axis.
- **translation y** (real) : Translation along the y axis.
- **scale x** (real) : Scale along the x axis.
- **scale y** (real) : Scale along the y axis.
- **do video effect** (boolean) : Apply a video effect?
- **custom quartz file** (file) : The custom Quartz Composer file used to render this cue.
- **fixed width** (number) : Fixed width of the text cue. Setting this to 0 specifies ‘auto’ width.
- **text output size** (list of number, r/o) : A 2-item list representing the width and height of the text of the text cue.
- **live text output size** (list of number, r/o) : A 2-item list representing the width and height of the live text of the text cue.

---

timecode cue

n [inherits from *cue*] : A Timecode Cue to generate MTC or LTC timecode.

Properties

- **patch** (integer) : MIDI destination patch number.
- **smpte format** (fps_24, fps_25, fps_30_drop, fps_30_non_drop) : SMPTE format of the outgoing timecode.
- **start time offset** (real) : Time in seconds where the MTC clock begins counting.

---

video cue

n [inherits from *cue*] : A Video Cue.

Properties

- **patch** (integer) : Audio device patch number.
- **start time** (real) : Time in the file where playback begins.
- **end time** (real) : Time in the file where playback ends.
- **infinite loop** (boolean) : Does the cue loop infinitely?
- **rate** (real) : Playback rate of the video cue.
- **layer** (integer) : Display layer of the video.
- **full screen** (boolean) : Is the cue displaying in full screen mode?
- **preserve aspect ratio** (boolean) : Does the cue preserve aspect ratio?
- **opacity** (real) : Video opacity.
- **translation x** (real) : Translation along the x axis.
- **translation y** (real) : Translation along the y axis.
- **scale x** (real) : Scale along the x axis.
- **scale y** (real) : Scale along the y axis.
- **do video effect** (boolean) : Apply a video effect?
- **custom quartz file** (file) : The custom Quartz Composer file used to render this cue.
- **audio input channels** (integer, r/o) : The number of audio input channels for the cue. (i.e. The number of distinct channels in the file.)
- **integrated fade** (enabled|disabled) : State of the integrated fade (enabled or disabled).
- **lock fade to cue** (enabled|disabled) : Whether the integrated fade should lock to the start/end times of the cue (enabled or disabled).
- **pitch shift** (enabled|disabled) : Whether pitch shifting is enabled or disabled.
- **hold at end** (boolean) : Should the final frame of the video be left visible when playback reaches the end of the file?
- **slice markers** (list of *slice marker record*) : List of slice marker records.
- **last slice play count** (integer) : Number of times the final slice plays. Always >= 1.
- **last slice infinite loop** (boolean) : Does the last slice loop infinitely?

Responds to

- getLevel, setLevel, getGang, setGang.

---

workspace

n [inherits from *document*] : A QLab workspace.

Elements

- contains cue lists, cues.

Properties

- **unique id** (text, r/o) : The unique ID of the workspace.
- **current cue list** (cue list) : The currently visible cue list for the workspace.
- **selected** (list of cue) : The currently selected cue(s) in the currently visible cue list.
- **active cues** (list of cue, r/o) : The list of active cues (running or paused) in this workspace.
- **edit mode** (boolean) : Is the workspace currently in edit mode?
- **show mode** (boolean) : Is the workspace currently in show mode?

Responds to

- make, load, go, start, pause, stop, hardStop, reset, panic, moveSelectionUp, moveSelectionDown.

Enumerations

absolute relative

enum : The fade mode of a Fade cue or the audio or video component of a Fade cue.

- **absolute** : absolute fade mode
- **relative** : relative fade mode

---

continue modes

enum : The continue mode of a cue.

- **do_not_continue** : Do not automatically continue to the next cue.
- **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.

---

enabled disabled

enum : Any of several cue properties.

- **enabled** : Enable the property/check the checkbox.
- **disabled** : Disable the property/uncheck the checkbox.

---

group modes

enum : The mode of a Group cue.

- **cue_list** : The group is a cue list.
- **fire_first_enter_group** : start first child and enter into group.
- **fire_first_go_to_next_cue** : start first child and go to next cue.
- **fire_random** : start a random child cue and then go to the next cue.
- **timeline** : Timeline - start all children simultaneously.

---

light dashboard view mode

enum

- **sliders** : Slider view.
- **tiles** : Tile view.

---

midi command

enum

- **note_on** : Note on.
- **note_off** : Note off.
- **program_change** : Program change.
- **control_change** : Control change.
- **key_pressure** : Key pressure (aftertouch).
- **channel_pressure** : Channel pressure.
- **pitch_bend** : Pitch bend (pitch wheel).

---

midi type

enum

- **voice** : MIDI Voice message (“Musical MIDI”).
- **msc** : MIDI Show Control message.
- **sysex** : MIDI Sysex message.

---

mtc ltc

enum

- **mtc** : MIDI Timecode
- **ltc** : Linear / Logitudinal Timecode

---

smpte format

enum : SMPTE timecode format.

- **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)

Records

range record

n : A 2-item record representing the offset and length of a substring.

Properties

- **rangeOffset** (integer or text) : The 1-indexed location of the starting character of a substring range.
- **rangeLength** (integer or text) : The length of the substring range.

---

rgba color record

n : A 4-item record representing red, green, blue, and alpha percent values of a color.

Properties

- **red** (real)
- **green** (real)
- **blue** (real)
- **alpha** (real)

---

slice marker record

n

Properties

- **time** (real) : The slice marker time (which is also the end time of a slice).
- **playCount** (integer) : The number of times a slice will play. Play count `-1` = infinite loop.

---

text format record

n

Properties

- **fontFamily** (text) : The font family for this format. (e.g. “Helvetica”, “Courier New”)
- **fontStyle** (text) : The font style (face) for this format. (e.g. “Regular”, “Light Oblique”)
- **fontName** (text) : The font name for this format. (e.g. “CourierNewPS-BoldItalicMT”)
- **fontSize** (real) : The font size for this format.
- **lineSpacing** (real) : The line spacing for this format.
- **rgbaColor** (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.
- **backgroundRgbaColor** (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.
- **strikethroughStyle** (text) : The strikethrough style of this format. Possible values are “none”, “single”, and “double”.
- **strikethroughRgbaColor** (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.
- **underlineStyle** (text) : The underline style of this format. Possible values are “none”, “single”, and “double”.
- **underlineRgbaColor** (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.
- **range** (range record) : A range record representing the index and length for the substring that has this format.
- **wordIndex** (integer) : An optional 1-indexed word number to which this format should be applied. When used, the “range” property will be ignored. (setting only)

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 and useful. What follows here is a collection of annotated examples to help get you started. Remember, you can always 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" way to do things.

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 cue or sent from another program or device.

Adjusting Audio Levels

The following AppleScript and OSC examples both set the master level of all selected cues to `-10` dB. You could easily replace `-10` with some other number, but remember that you always need to explicitly specify whether a level is positive or negative when using AppleScript or OSC.

AppleScript

```
tell application id "com.figure53.qlab.4"
  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.4` is how you instruct the AppleScript interpreter to send the contents of this block of code to QLab 4.

`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. `theCue` creates this variable, which is just a blank container waiting to be used. `in (selected of front workspace as 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 code once for each item in the list, replacing `theCue` with each successive item.

`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 stop running the script. Just skip over the rest of this block and then keep running. This allows the script to handle a 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 every single cue is appropriate; if we select five Audio cues and one Memo cue, the script should adjust the Audio 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 represents each item in the list of selected cues, `row 0` is the master output row of the audio levels matrix mixer, `column 0` is the master 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 should be adjusted, which in this case is row 0, column 0.

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 that has a number. Remember that cue numbers in QLab don't have to be numbers; they can be any text. So cue "panda" works too. Since cue numbers have to be unique across a workspace, they are the best identifier to use with scripting.

AppleScript

```
tell application id "com.figure53.qlab.4" 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 each workspace could have a cue numbered "4". So we need to specify which workspace we want to address. If we wanted to address a specific workspace, whether or not it was in front, we could instead write:

```
tell application id "com.figure53.qlab.4" to tell workspace "Hamlet.qlab4"
```

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 the fourth cue in the workspace, rather than the cue whose cue number is "4". It's a small difference that makes a big 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.

Create Fade-in Cues

This is the most complex of these examples: create fade-in cues for every selected cue, fading the master audio level to the level that the source cue is set to, and if the source cue is a Video cue, fade the opacity in as well.

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 messages, and sometimes those messages can be chained together. But as you'll see, this example gets information from 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.4"
  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" then
        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" 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 "Camera" or 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 behaves, so the try block is more of a safety net than anything else, guarding against unexpected outcomes.

`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, and we're going to replace `sourceCue` with each successive cue on each repetition.

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 sourceCue is "Video" then`

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, `sourceCue` represents one of the selected cues in the cue list. So the `if` statement looks at that cue, and if that cue is an Audio, Mic, or Video 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 skipped over.

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 master level of `sourceCue`. Later, we're going to plug that level into a new Fade cue, so that we can fade this cue up to the level it was set to.

Once we've stored the level, we can set `sourceCue` to silent by setting its master level to `-120`, which is the lowest possible audio level in QLab: `sourceCue setLevel row 0 column 0 db -120`

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 cue should be made in.

`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 master audio level of the new Fade cue to the level we fetched from `sourceCue` and stored in the variable `sourceCueLevel`.

Now, we dive one level deeper with another `if` block: `if q type of sourceCue is "Video" 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 level of the new Fade cue, and we check to see if `sourceCue` is a Video cue which needs its opacity 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 stored it, we can `set opacity of sourceCue to 0` so that it's ready to fade in.

`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` checks the Opacity checkbox in the Fade cue.

`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` statement asks "is `sourceCue` an Audio, Mic, or Video cue?" and if the answer is *yes*, the code inside the `if` block executes. If the answer is *no*, then the script skips down to this `else` statement, which asks "OK, well is `sourceCue` a Camera or Text cue?" If the answer is still *no*, this block is also skipped over. But if the answer is *yes*, then the code is executed.

The reason we need the `else` statement is that Camera and Text cues have no audio component. The code inside the `else` block is the opacity-related second half of the block above, since Camera and Text cues have opacity, but no audio levels.

`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.