User Manual

grandMA3

Version 2.2 | 2024/12/18

English

Table of Contents

1. grandMA3 User Manual

- **New in the Manual**
- **About the Help**
- **Device Overview**
- **System Overview**
- **First Steps**
- **grandMA3 onPC**
- **Show File Handling**
- **Workspace**
- **Command Syntax and Keywords**
- **Windows, Views, and Menus**
- **Networking**
- **DMX In and Out**
- **Single User and Multi User Systems**
- **Patch and Fixture Setup**
- **Operate Fixtures**
- **Scribbles**
- **Images**
- **Screenshots**
- **Video**
- **Meshes**
- **Gobos**
- **Symbols**
- **Appearances**
- **Notes**
- **Label Objects**
- **Groups**
- **Presets**
- **Worlds and Filters**
- **MAtricks and Shuffle**
- **Cues and Sequences**
- **Executors**
- **Masters**
- **Recipes**
- **Phasers**
- **Generator - Random**

- **Bitmap**
- **XYZ**
- **Tags**
- **Macros**
- **Agenda**
- **Timers**
- **Timecode Show**
- **Layouts**
- **Plugins**
- **Quickeys**
- **Data Pools**
- **System**
- **Remote In and Out**
- **Sound**
- **RDM (Remote Device Management)**
- **Local Settings**
- **Update the Software**
- **Fixture Types**
- **File Management**
- **Show Creator**
- **Control other MA Devices**
- **Troubleshooting**
- **Glossary**
- **grandMA3 List of Trademarks**

1.1. New in the Manual

Learn what is new in the latest release of the help manual.

New	Description	Topic
Recipe Workflow	The Recipe Editor Window and Edit Recipe Mode allow a better and faster workflow when creating and editing recipes.	**Recipe Editor Window** **Create Recipes with the Edit Recipe Mode**
Tags	Tags allow organizing, linking, and cross-referencing objects throughout the software.	**Tags**
Timing Masters	Timing masters can be set for playback timings instead of a numeric value.	**Timing Masters**
Timecode Show	The timecode show section was enhanced and extended.	**Timecode Show** **Timecode Viewer** **Track Groups** **Markers** **Tracks** **Time Ranges** **Events** **Timecode Slots** **Timecode Settings** **Create a Timecode Show** **External Connections**
Improvements Command Line	Notifications inform the user about changes in the message center. Additionally, it is possible to show or hide the progress bar.	**Command Line**
Clock Viewer	The clock viewer has new options for session time and time zone.	**Clock**
Keywords	New keyword topics were added.	**CopyCrashLog** **Deactivate** **EditRecipe** **HelpKeyword** **ListCrashLog** **NetworkSpeedTest** **Session** **Tag**

New	Description	Topic
Videos	Videos were added to several topics.	**Gels Pool** **Notes in Pool Objects** **Info Window**
Updated Topics in This Release	For more information on other changes, see Release Notes.	**Release Notes 2.2**

1.2. About the Help

Use this guide to learn how grandMA3 can help you to make an aMAzing show.

Start at the beginning, visit each section, use the context sensitive help, connect with the community to work your way through a show, or use our huge pdf document.

Support

Contact us for any questions about your MA product.

MA Lighting and its extensive distributor network offer unparalleled technical support. Call on our expertise for help with any problem, no matter if it is about the operation, software features, software installations, or troubleshooting.

Community

Share your knowledge and get help from other MA users.

A community can be stronger and better than an individual. Be part of the MA community! Go to **https://www.malighting.com/training-support/community**.

Central Support

If you need further assistance or manufacturer support, please fill out an email request on www.malighting.com in Training and Support. Your request will automatically be sent to **support@malighting.com** (in English or German). This email service is monitored during MA Lighting's regular business hours in Germany from 8.30 am until 5 pm (CET), Monday through Friday.

Emergency Hotline

In any case of a show critical emergency please contact the MA Lighting support hotline.

Phone +49 5251 68 88 65 99

Please note that this 24/7 hotline is strictly for emergency cases.

Intellectual Property

MA Lighting Technology has registered multiple patents, trademarks, design patents, and utility models for its products.
We will take legal action against the violation of this intellectual property.

For more information see: **https://www.malighting.com/intellectual-property/**

Subtopics

- **Open the Help in the Console**
- **Open the Help in Web Browser**
- **Open the Help as PDF**
- **Navigate in the Help**
- **Videos in the Help**

1.2.1. Open the Help in the Console

There are different ways to open the help on the console.

Open the Search Tab

To open the help pop-up on the search tab:

- Press Help and then press Please.
- Type **Help** and then press Please.
  For more information, see **Help Keyword**.

The help pop-up opens to the search tab. To search for a topic, just start typing.

Open the Last Viewed Topic

To open the help pop-up on the topic you last viewed:

- Double-tap ? in the **Control Bar**.
- Double-press Help.

Help Window

To open the help window:

- Open the Add Window dialog, tap the More tab, then tap Help. For more information, see the **Add Windows** topic.
- Store the Help view button on a view bar. For more information, see **Store a View Directly on a View Button**.

Help window

Open the Context Sensitive Help

The context sensitive help displays a help topic to a related UI element or a key. To open the context sensitive help:

- Press Help and tap the desired user interface element.
- Tap ? on the control bar and tap the user interface element.
- Press and hold Help and then press Any key.

Open a Keyword Topic

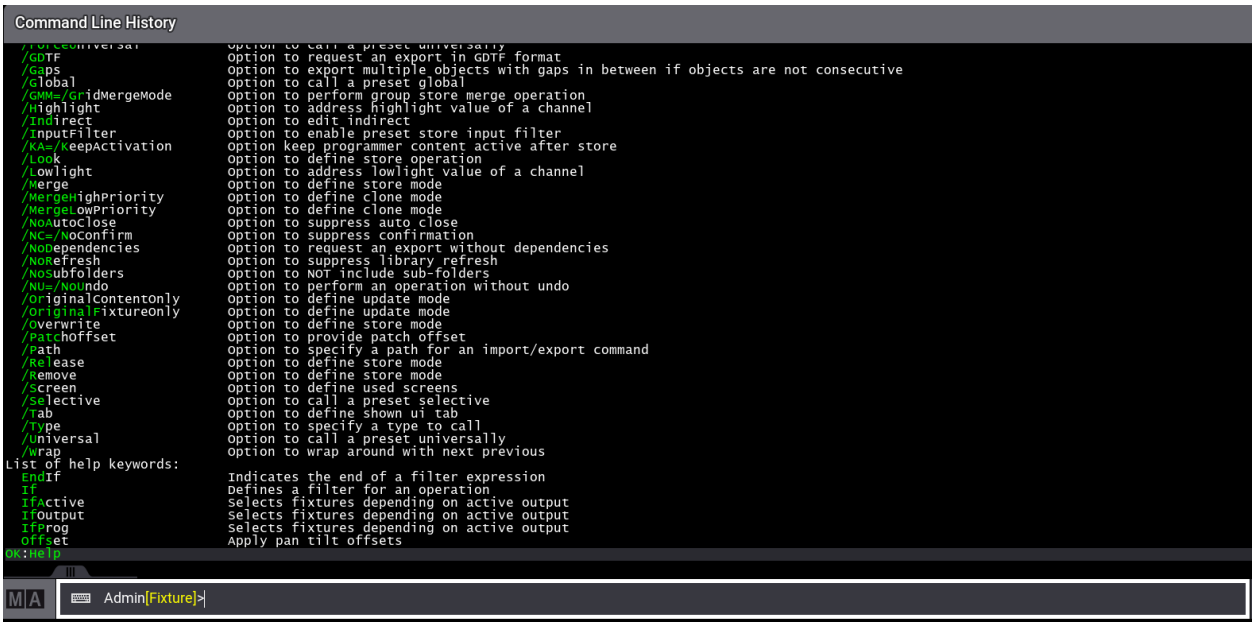
To open a keyword topic, use one of these options:

- Press Help and type the desired keyword, then press Please.
- Type **Help** and type the desired keyword, then press Please.
- Press Help + Any key + Please.

Display Keyword List

To display all keywords in the command line history:

1. Type the command **HelpKeyword** into the command line and press Please.
   For more information, see **HelpKeyword Keyword**.
2. Tap MA at the left side of the command line.



Keyword list in the command line history

To search for specific keywords, type the command **HelpKeyword** into the command line followed by the first letters of the keyword you are looking for with an asterisk, for example **HelpKeyword Fix***.

The command line history displays all keywords that begin with the letters you typed in.

The keywords can be used in full or in their abbreviated form, indicated in green.

Tooltips

Tooltips are short pieces of information such as descriptions and instructions. They are displayed in the command line in light grey text.

- To display a tooltip in grandMA3 onPC, hover the mouse over an element in the user interface.
- To display a tooltip in the console, tap and hold an element in the user interface.

	Important:
	Tap and hold in the user interface of the console often also executes the element.

Shortcuts are displayed in squared brackets at the beginning of the tooltip. For more information on shortcuts see **Shortcuts**.

Tooltip for the At overlay

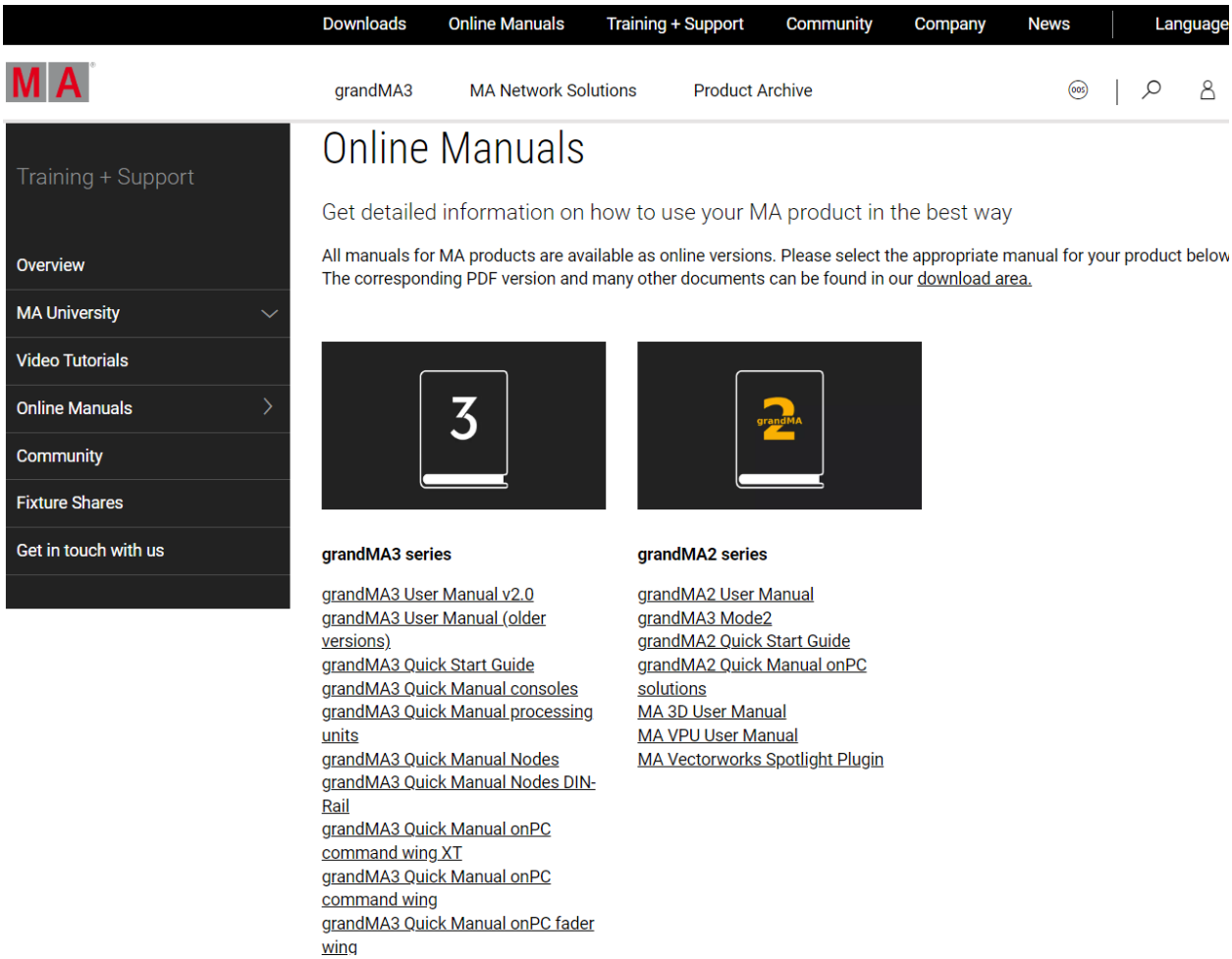
1.2.2. Open the Help in Web Browser

Please note that the help topics included in the software represent the status and volume of the help pages at the date of release.

The help system is constantly being updated and extended.

The most up-to-date and complete version of the manual is directly published on the following website:

https://www.malighting.com/training-support/online-manuals



Online manuals

The Online Manuals section is divided into four product families:

- grandMA3 series
- grandMA2 series
- dot2 series
- MA Network Switch

1.2.3. Open the Help as PDF

Download single topics or the entire manual from the **online help** as a PDF document to view them offline.

To download the contents of a topic as a PDF file:

1. Click the printer icon next to the topic title.
2. As destination choose **Save as PDF**.
3. Click Save.
   The topic PDF is saved to your Downloads folder.

To download a PDF of the entire manual:

1. Navigate to the Downloads section of the MA Lighting website:
   https://www.malighting.com/downloads
2. Click the desired product family from the list on the left side.
3. Click **Offline User Manuals + QuickMAnual**.
4. Click the desired manual to download. The download begins.

1.2.4. Navigate in the Help

Buttons in the Title Bar

To increase or decrease the size of the content displayed within the help window:

- Tap + (plus) or — (minus)
- Tap on Zoom Factor and swipe left or right
- Tap Zoom Factor and choose a percentage in the drop-down list

Tap ← (left arrow) to reverse through the history of previously viewed help topics.

Tap → (right arrow) to advance through the history of previously viewed help topics.

Scrolling in the Help

- Two-finger tap and swipe in X or Y direction to scroll through a help topic.
- Use the Screen encoder to scroll through a help topic in the XY direction. For more information, see **User Settings**.

Topics Tab

The topics tab in the Help window displays the available topics in a tree structure. Single topics are indicated by next to them. Multiple topics of the same category are sorted within a folder ( ) and are called subtopics.

Tap > to the left of any topic to show or hide any related subtopics. Tap the name of any topic to display its content in the main area of the window. Subtopics are also displayed in a bullet list at the end of the topic.

If the Help window is too small to display both the main area and the Topics and Search tabs, a burger menu ( ) appears in the upper left corner. It contains the Topic and Search tabs.

The root topics in the tree structure are divided into the different products within the family. These main sections include:

- **grandMA3 User Manual**
- **grandMA3 Quick Start Guide**
- **grandMA3 Quick Manual consoles**
- **grandMA3 Quick Manual processing**
- **grandMA3 Quick Manual Nodes**
- **grandMA3 Quick Manual Nodes DIN-Rail**

- **grandMA3 Quick Manual onPC command wing XT**
- **grandMA3 Quick Manual onPC command wing**
- **grandMA3 Quick Manual onPC fader wing**
- **grandMA3 Quick Manual viz-key**
- **grandMA3 Quick Manual I/O Nodes**
- **Release Notes**

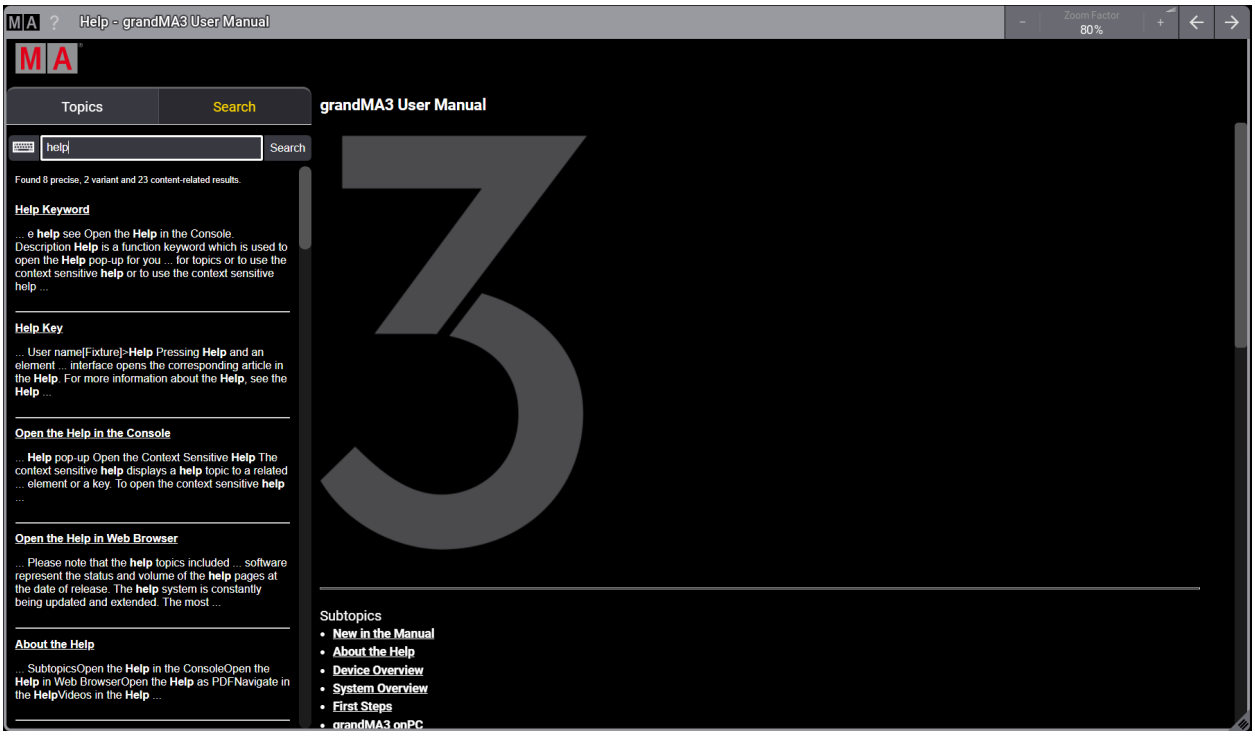
	Important:
	The complete grandMA3 documentation consists of the grandMA3 User Manual and the respective grandMA3 Quick Manual with technical specifications, safety instructions, and declaration of conformity.

Search Tab

To find topics containing a specific search term, use the search function:

1. Open the search tab in the help window.
2. Tap into the white field.
3. Type the term, such as cue, in the search box.
4. Press Please. The search results are shown.

To open a virtual keyboard in the software, tap on the keyboard icon on the left of the search box.



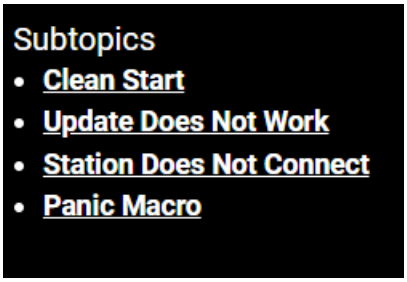
Help window - Search tab

To search for a specific product or search term, the brand name or the series name (e.g. grandMA3) is not necessary.

Listed Subtopics

A list of subtopics that are related to the current topic is dynamically generated at the bottom of the help page.

To open a related topic, tap it in the list.



Listed subtopics

Breadcrumbs

Breadcrumbs are navigation elements that provide links to the related sections of the current topic and subtopics. Breadcrumbs are located at the top of each topic.

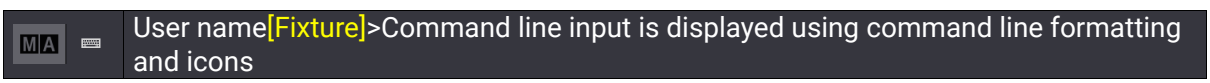
To go back in a topic, tap the corresponding section in the breadcrumbs.

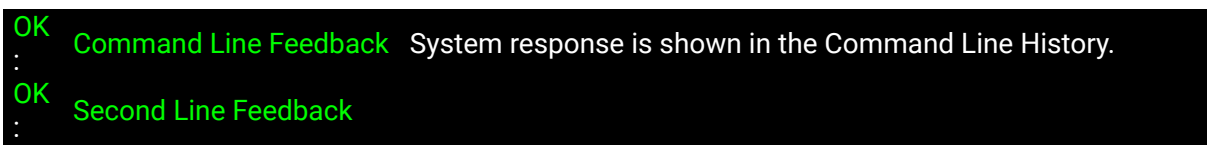
Breadcrumbs to navigate in the help

Display of Content

The user manual offers graphical assistance to add context to the contents.

- Please: Keys on the control surface of the hardware are displayed in a white font on a dark gray background.
- Dimmer: Buttons on the screen are displayed in a white font on a light gray background
- Step-by-step instructions are displayed in a numeric list:

1. This is the first step
2. This is the second step
   - One-step instructions are displayed with the help of bullet points.
   - **Menu - Settings - User**: Navigating to specific menus is displayed in bold.
   - **Assign [Object] (At) [Object]:** Syntax descriptions are underlined and displayed in bold.
   - 

   - 

The following symbols display possible danger, useful hints, and information.

For information on safety, harmonized standards, and conformity see **grandMA3 Quick Manual Consoles**.

	The warning icon indicates possible injury and hazards.
	**Restriction:**
	The restriction icon indicates known limitations of functions.
	**Important:**
	The important icon indicates essential information for console usage.
	**Hint:**
	The hint icon indicates useful tips for console usage.

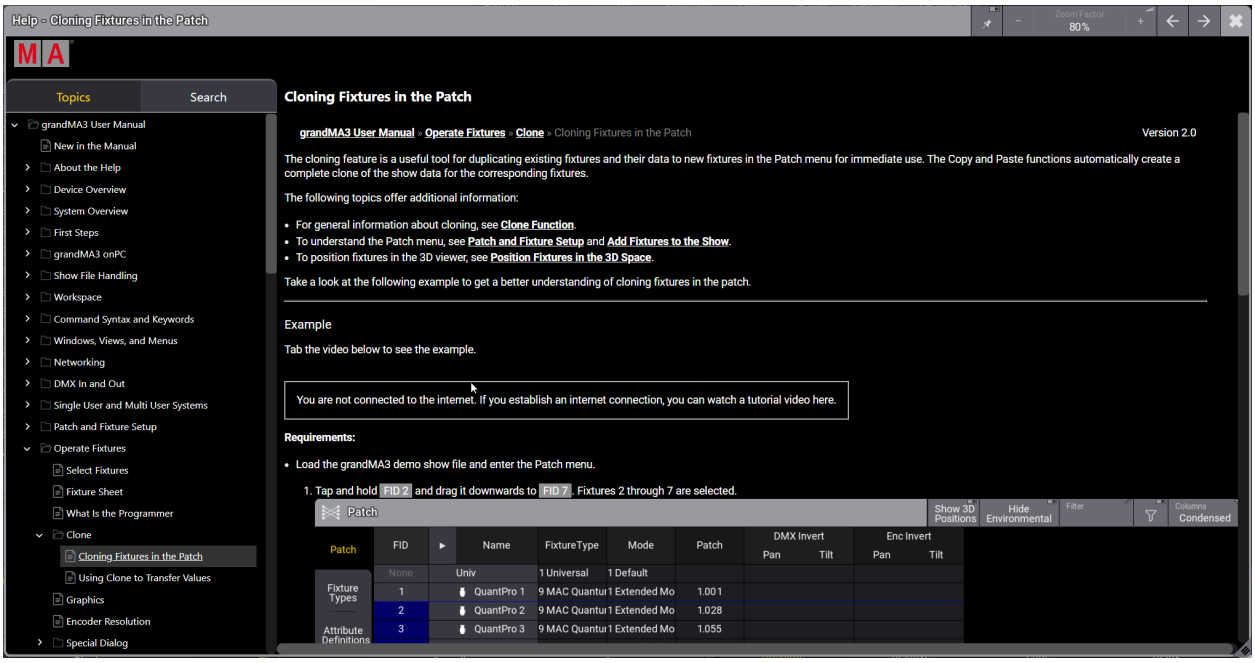
1.2.5. Videos in the Help

The Help system offers micro-videos to introduce the user to specific features or tasks in the software. The average video length is less than one minute. A topic can have several videos.

A video is indicated by a play icon (▶) .

	Hint:
	Videos in the help are only visible when an internet connection is established! For more information, see **Networking**.

When the internet connection is interrupted, a placeholder is shown instead of the video:



Help window - Video placeholder

Videos are recorded in the grandMA3 onPC software. For better visualization, mouse effects are used:

- Mouse highlight effect in a yellow color.
- Left mouse click effect in a red color.
- Right mouse click effect in a gray color.

Play a Video

To start playing a video:

- Tap the play icon (▶). The video starts.

To stop playing a video:

- Tap anywhere in the video area. The video stops.

Or

- Tap the pause button ( ) in the video control bar.

To restart a video:

- Tap the restart button (↻) after the video is finished.

Video Control Bar

The video control bar is located at the bottom of the video area.

Video control bar

- To display the video control bar in the console, tap the video area. The video control bar appears for several seconds.
- To display the video control bar in grandMA3 onPC, hover the mouse over the video area.
- To jump to a specific position in the video, tap the timeline in the control bar.

To change the playback speed of the video:

1. Tap the gear icon on the right. The settings open.
2. Tap Speed. A dropdown menu opens.
3. Tap a value to increase or decrease the speed of the video.
4. Tap to close the settings. The playback speed is changed.

Example

The following video shows an example of how to open a video and change the playback speed:

1.3. Device Overview

- All MA Lighting devices are built for touring and installation purposes and meet international standards for electronic safety and protection. The Declaration of Conformity and the Electromagnetic Compatibility regulation (EMC) can be found in the Quick Guide included with the product and available for download at **https://www.malighting.com.**
- All MA Lighting devices have a wide input voltage range and can operate from 90V to 230V.

The following chapters describe the different grandMA3 devices.

- **grandMA3 consoles**
- **grandMA3 extension**
- **grandMA3 replay unit**
- **grandMA3 processing units**
- **grandMA3 Nodes**
- **grandMA3 Nodes DIN-Rail**
- **grandMA3 I/O Node**
- **grandMA3 I/O Node DIN-Rail**
- **grandMA3 onPC command wing XT**
- **grandMA3 onPC command wing**
- **grandMA3 onPC fader wing**
- **grandMA3 onPC rack-unit**
- **Screen Allocation**
- **Shortcuts**
- **Keys**
- **Control Elements**
- **Connector Pin Assignment**
- **UPS Battery**

1.3.1. grandMA3 consoles

The grandMA3 consoles control all kinds of lighting genres such as conventional lights, moving lights, LEDs, video, and media via DMX signal or within a network environment.

All grandMA3 components, despite different hardware solutions, use the same software. For general overview of grandMA3 hardware, see **Products**.

All components are fully integrable into the network environment.

This topic describes the different sections of the grandMA3 consoles.

These sections vary from model to model.

The flagship of the grandMA3 range is the full-size or full-size CRV, respectively.

The control room version (CRV) of full-size and light are models without the monitor wing, but come with additional external monitor ports.

The grandMA3 light console or the light CRV are the work-horses of the range.

The compact and compact XT models offer the full system benefits in a compact and lightweight format.

The grandMA3 consoles number their screens in a specific order, for more information see **Screen Allocation**.

You can connect several external touch screens with a console, see **Connect External Screens**.

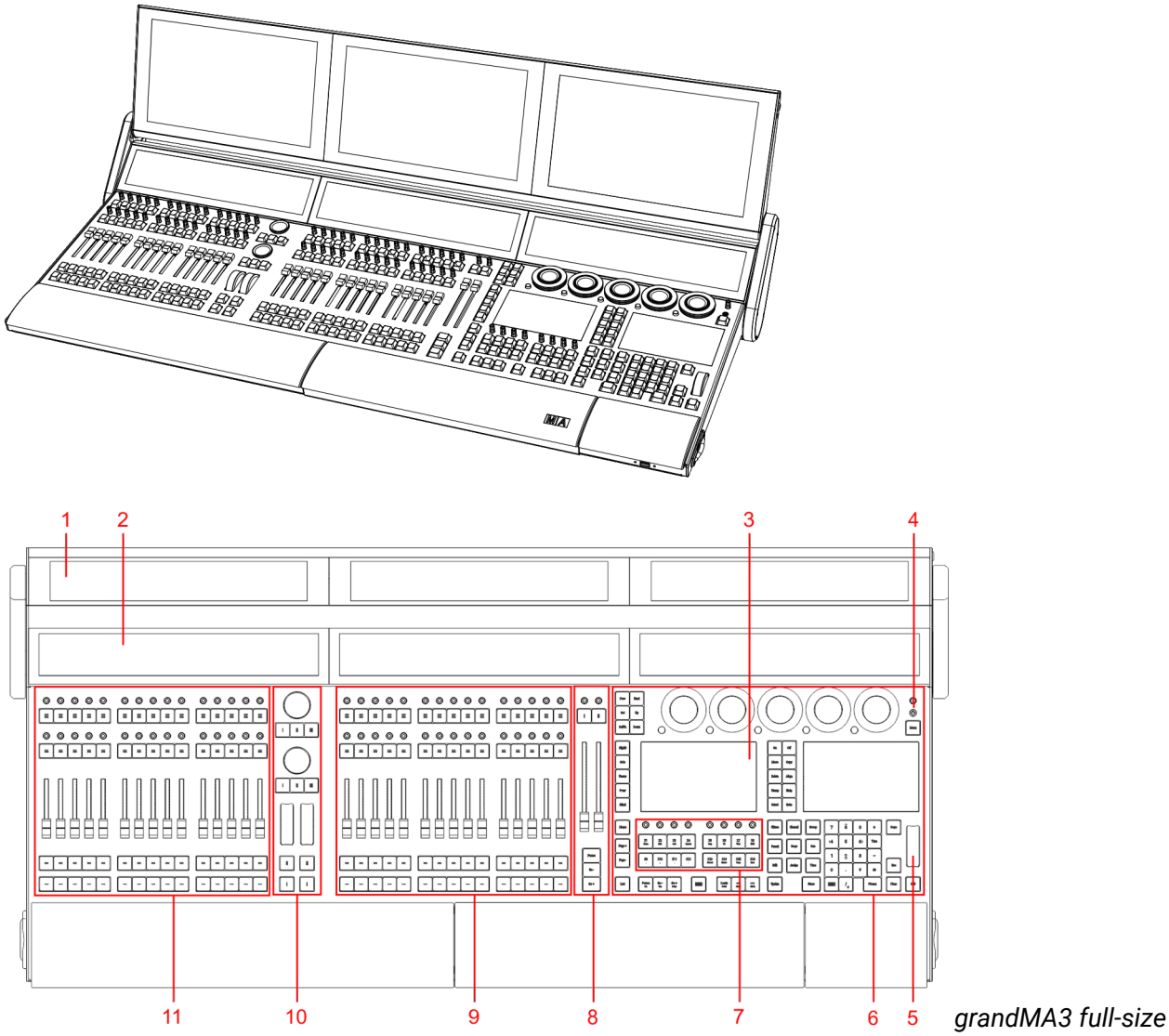
The display elements of the screens can be modified, see **Configuration of Displays**.

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

Subtopics

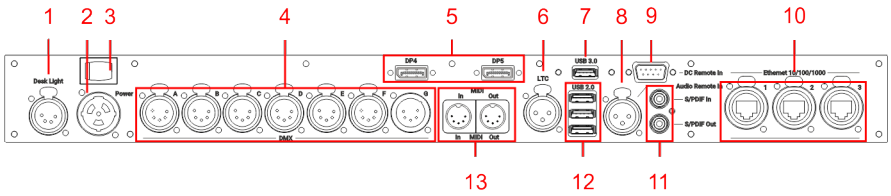
- **grandMA3 full-size**
- **grandMA3 full-size CRV**
- **grandMA3 light**
- **grandMA3 light CRV**
- **grandMA3 compact XT**
- **grandMA3 compact**

1.3.1.1. grandMA3 full-size



grandMA3 full-size front panel

1. **Screens 1-3**
2. **Letterbox screens 8-10**
3. **Command screens 6+7**
4. **Power key**
5. **Level wheel**
6. **Command area**
7. **Xkeys section**
8. **Master area**
9. **Right executor area with 3 executor sections**
10. **Custom area**
11. **Left executor area with 3 executor sections**



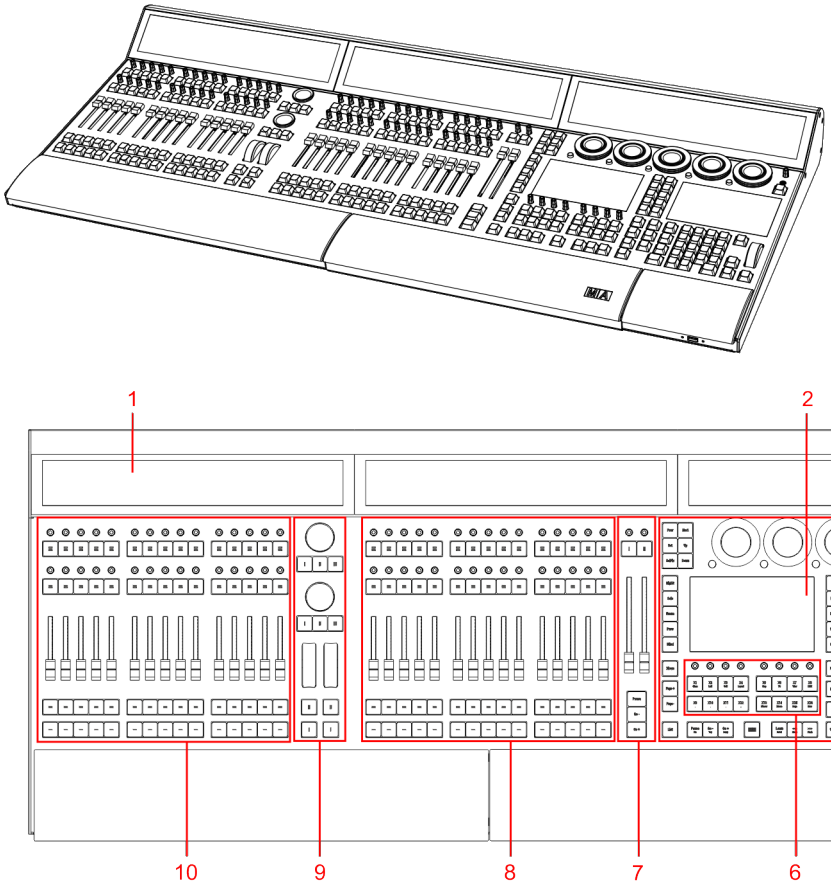
grandMA3 full-size rear panel

1. **Desk light**
2. **powerCON TRUE1**
3. Power switch
4. **DMX A-G**
5. **DisplayPort 4+5**
6. **LTC**
7. **USB 3.0**
8. **Audio Remote In**
9. **DC Remote Control**
10. **Ethernet 1-3**
11. **S/PDIF In+Out**
12. **USB 2.0**
13. **MIDI In+Out**

	Hint:
	For command line and graphical user interface access to an overlay showing the connectors on the back of the console, see the **DMX Port Configuration** topic.

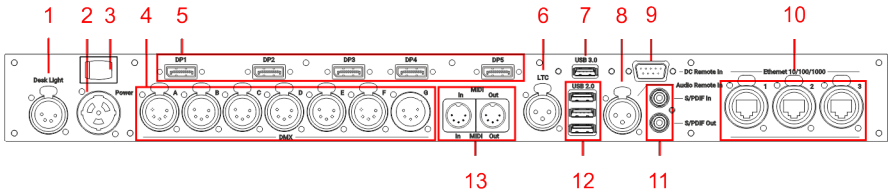
For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

1.3.1.2. grandMA3 full-size CRV



grandMA3 full-size CRV front panel

1. **Letterbox screens 8-10**
2. **Command screens 6+7**
3. **Power key**
4. **Level wheel**
5. **Command area**
6. **Xkeys section**
7. **Master area**
8. **Right executor area with 3 executor sections**
9. **Custom area**
10. **Left executor area with 3 executor sections**



grandMA3 full-size

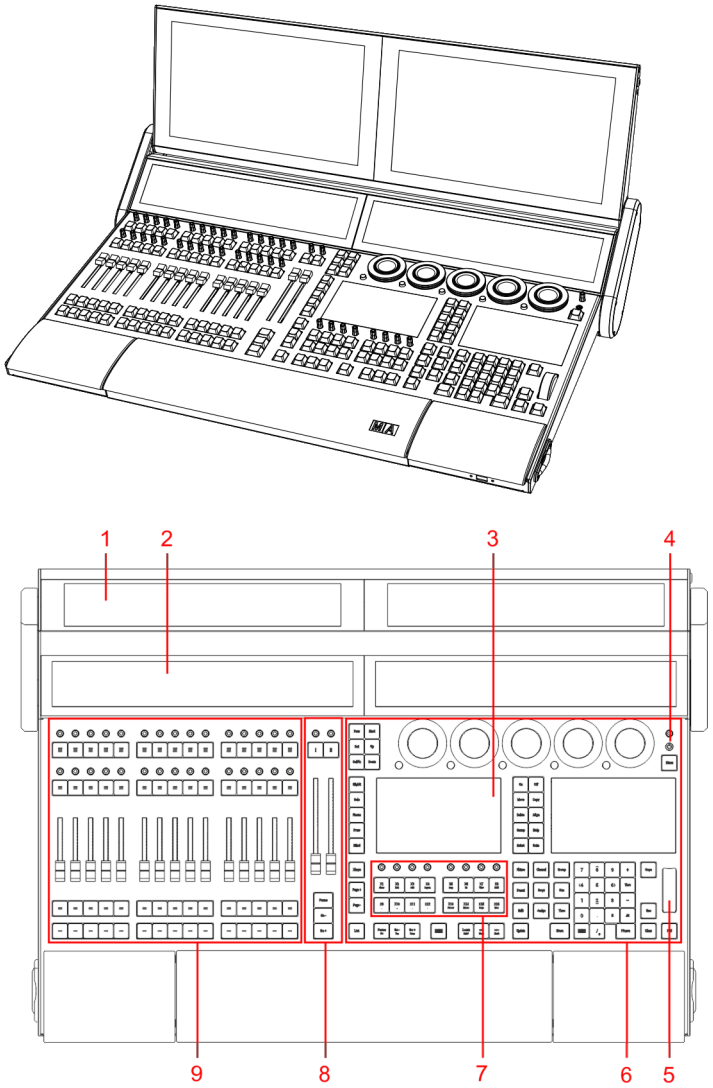
CRV rear panel

1. **Desk light**
2. **powerCON TRUE1**
3. Power switch
4. **DMX A-G**
5. **DisplayPort 1-5**
6. **LTC**
7. **USB 3.0**
8. **Audio Remote In**
9. **DC Remote Control**
10. **Ethernet 1-3**
11. **S/PDIF In+Out**
12. **USB 2.0**
13. **MIDI In+Out**

	Hint:
	For command line and graphical user interface access to an overlay showing the connectors on the back of the console, see the **DMX Port Configuration** topic.

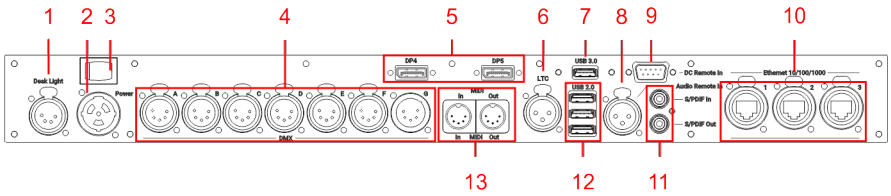
For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

1.3.1.3. grandMA3 light



grandMA3 light front panel

1. **Screens 1+2**
2. **Letterbox screens 8+9**
3. **Command screens 6+7**
4. **Power key**
5. **Level wheel**
6. **Command area**
7. **Xkeys section**
8. **Master area**
9. **Executor area with 3 executor sections**



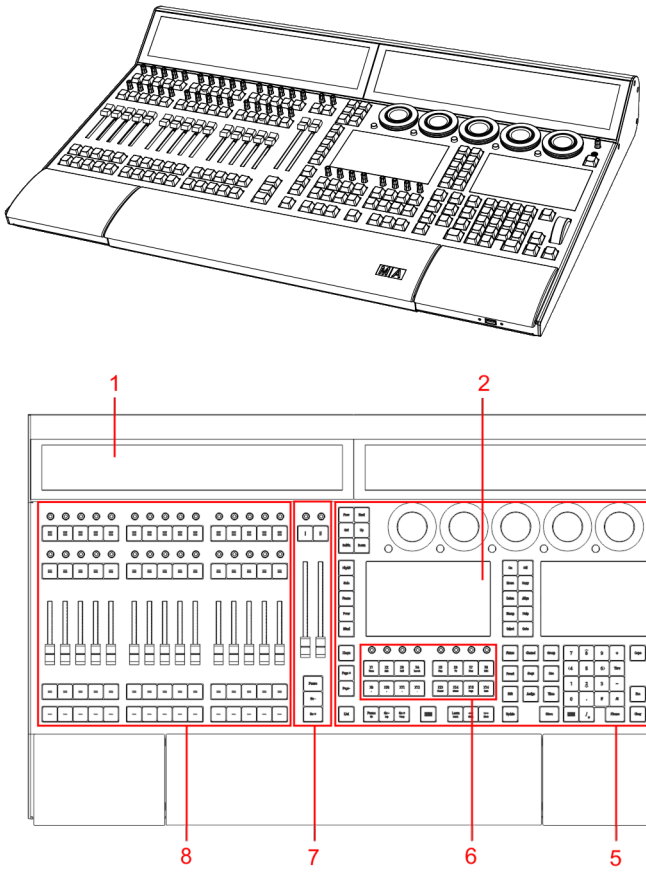
grandMA3 full-size rear panel

1. **Desk light**
2. **powerCON TRUE1**
3. Power switch
4. **DMX A-G**
5. **DisplayPort 4+5**
6. **LTC**
7. **USB 3.0**
8. **Audio Remote In**
9. **DC Remote Control**
10. **Ethernet 1-3**
11. **S/PDIF In+Out**
12. **USB 2.0**
13. **MIDI In+Out**

	Hint:
	For command line and graphical user interface access to an overlay showing the connectors on the back of the console, see the **DMX Port Configuration** topic.

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

1.3.1.4. grandMA3 light CRV



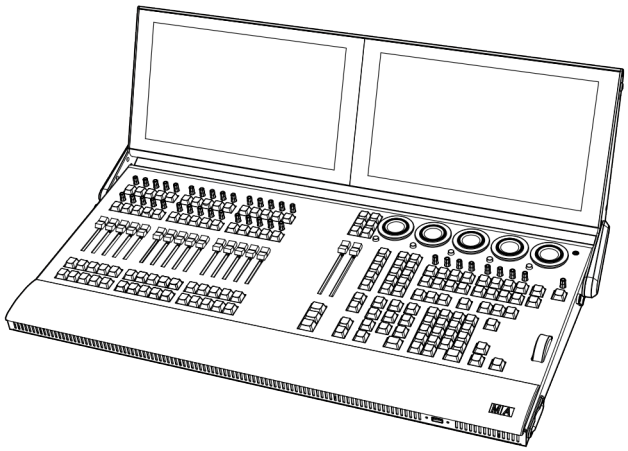
grandMA3 light CRV front panel

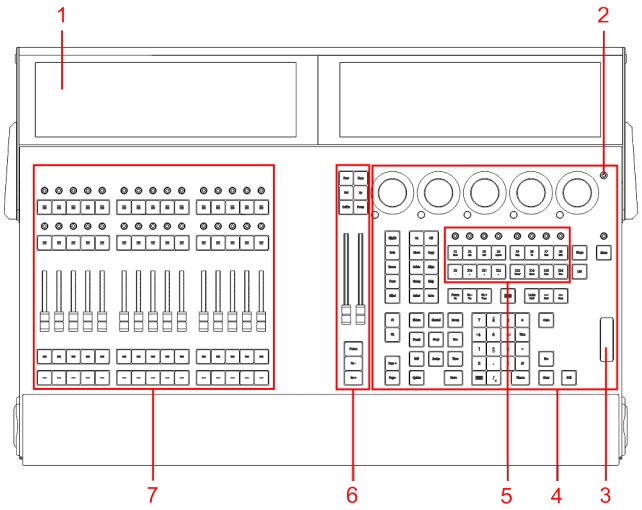
1. **Letterbox screens 8+9**
2. **Command screens 6+7**
3. **Power key**
4. **Level wheel**
5. **Command area**
6. **Xkeys section**
7. **Master area**
8. **Executor area with 3 executor sections**

	Hint:
	For command line and graphical user interface access to an overlay showing the connectors on the back of the console, see the **DMX Port Configuration** topic.

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

1.3.1.5. grandMA3 compact XT



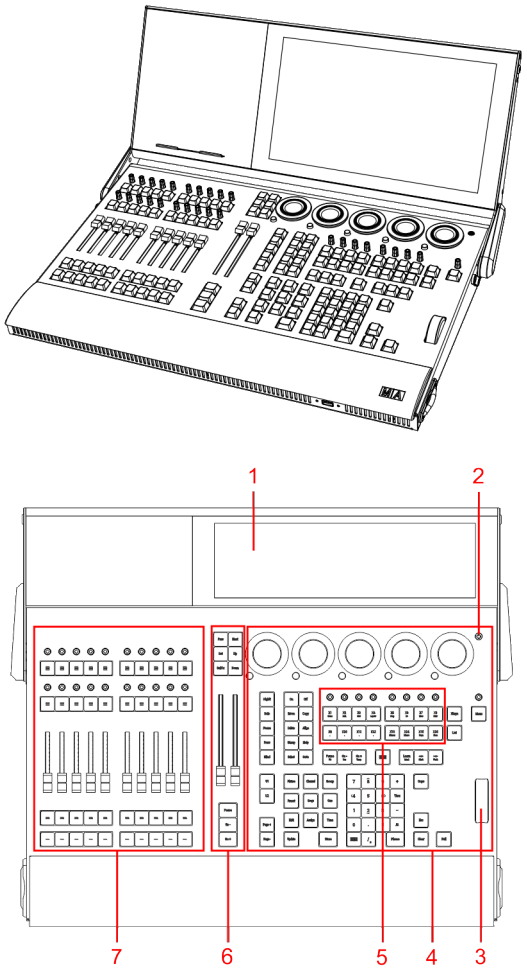


grandMA3 compact XT front panel

1. **Screens 1+2**
2. **Power key**
3. **Level wheel**
4. **Command section**
5. **Xkeys**
6. **Master section**
7. **Executor section**

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

1.3.1.6. grandMA3 compact

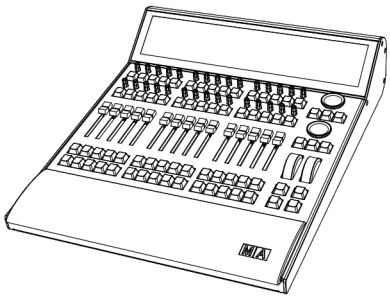


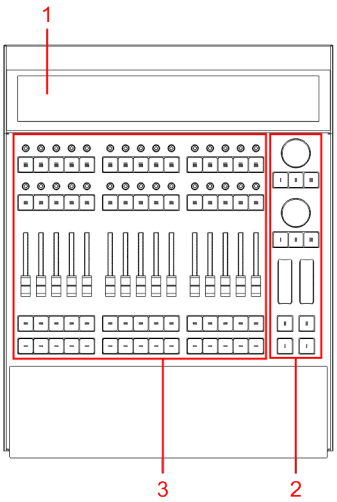
grandMA3 compact front panel

1. **Screen 1**
2. **Power key**
3. **Level wheel**
4. **Command section**
5. **Xkeys**
6. **Master section**
7. **Executor section**

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

1.3.2. grandMA3 extension

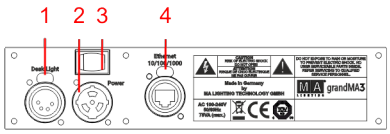




grandMA3 extension

front panel

1. **Letterbox screen 10**
2. **Custom section**
3. **Executor section**



grandMA3 extension

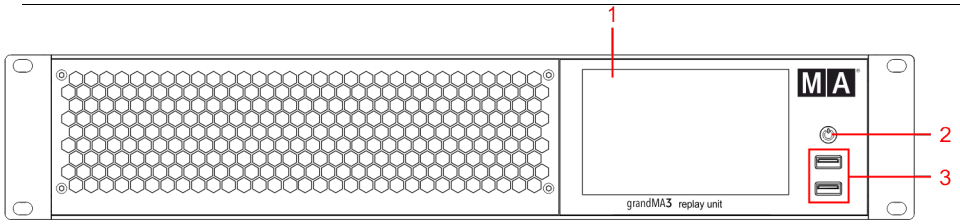
rear panel

1. **Desk light**
2. **powerCON TRUE1 connector**
3. Power switch
4. **Ethernet**

For more information, see the topics **First steps**, **Connect grandMA3 extension**.

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

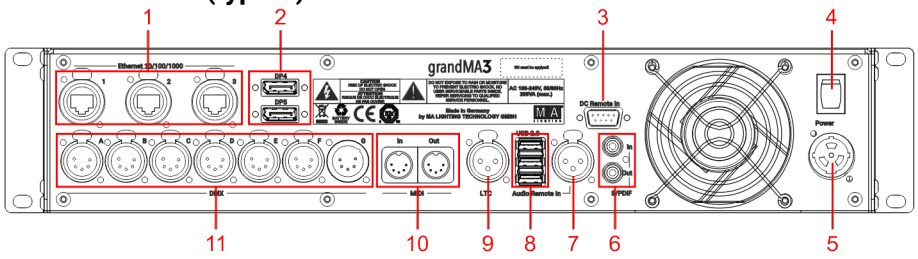
1.3.3. grandMA3 replay unit



grandMA3 replay unit

front panel

1. Internal command multi-touch screen
2. **Power key**
3. **USB 2.0 (type A)**



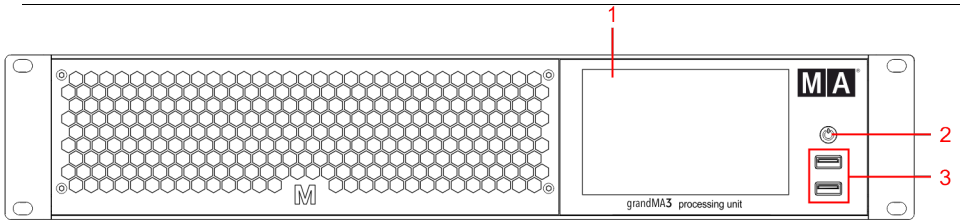
grandMA3 replay unit

rear panel

1. **etherCON/RJ45**
2. **DisplayPort 1+2**
3. **DC Remote Control**
4. Power switch
5. **powerCON TRUE1**
6. **S/PDIF In+Out**
7. **Audio Remote In**
8. **USB 2.0 (type A)**
9. **LTC**
10. **MIDI In+Out**
11. **DMX-512-A OUT (5pin XLR female)**

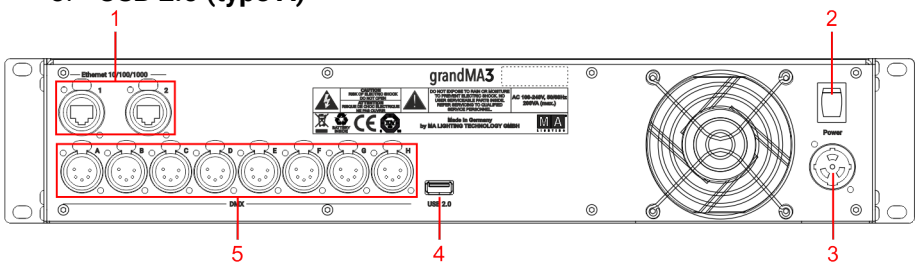
For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

1.3.4. grandMA3 processing units



grandMA3 processing unit M, L, XL front panel

1. Internal command multi-touch screen
2. **Power key**
3. **USB 2.0 (type A)**

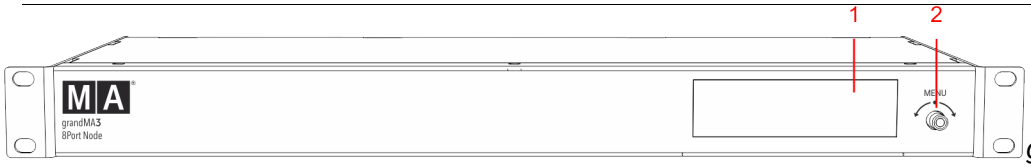


grandMA3 processing unit M, L, XL rear panel

1. **etherCON/RJ45**
2. Power switch
3. **powerCON TRUE1**
4. **USB 2.0 (type A)**
5. **DMX-512-A OUT (5pin XLR female)**

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual processing**.

1.3.5. grandMA3 Nodes

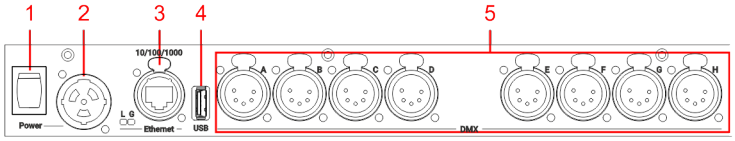


grandMA3 8Port Node front panel

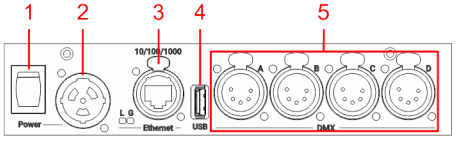
grandMA3 4Port Node front panel

grandMA3 2Port Node front panel

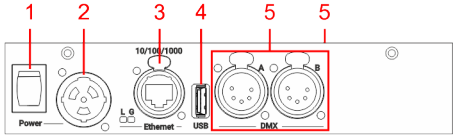
1. Display
2. Rotary knob



grandMA3 8Port Node rear panel



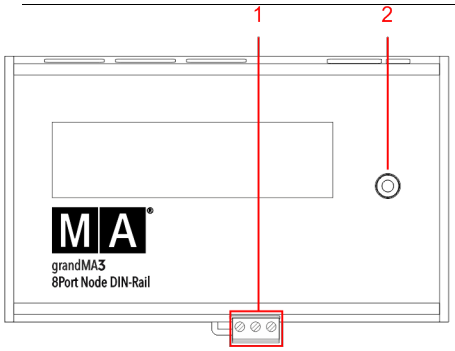
grandMA3 4Port Node rear panel



grandMA3 2Port Node rear panel

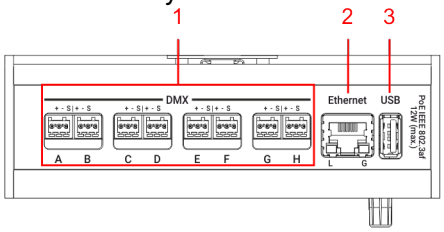
1. Power switch
2. **powerCON TRUE1**
3. **Ethernet** with **L (link) and G (gigabit) LEDs**
4. **USB port**
5. **DMX**

1.3.6. grandMA3 Nodes DIN-Rail



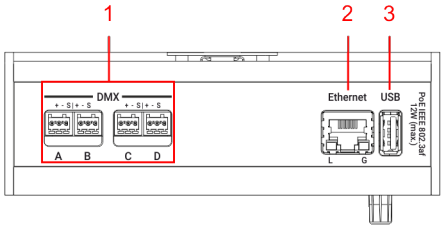
grandMA3 8Port Node DIN-Rail front panel

1. **Terminal block**
2. Rotary knob

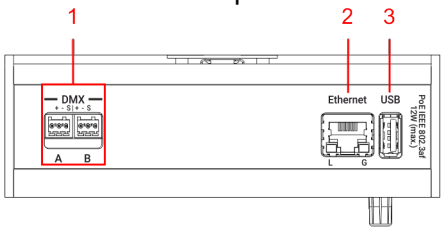


grandMA3 8Port Node DIN-Rail rear panel

1. **DMX**
2. **Ethernet** with **L (link) and G (gigabit) LEDs**
3. **USB 2.0**



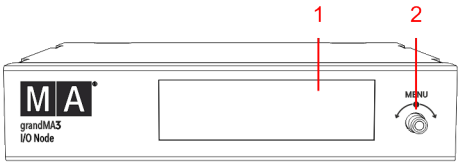
grandMA3 4Port Node DIN-Rail rear panel



grandMA3 2Port Node DIN-Rail rear panel

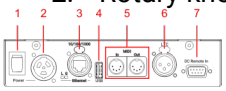
For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual Nodes DIN-Rail**.

1.3.7. grandMA3 I/O Node



grandMA3 I/O Node front panel

1. Display
2. Rotary knob

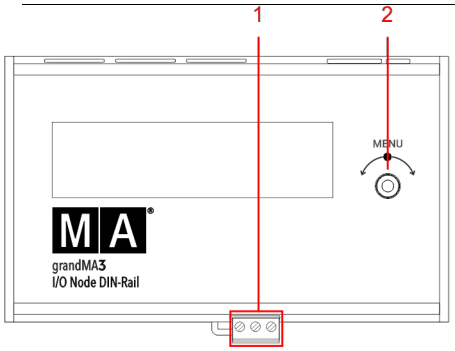


grandMA3 I/O Node rear panel

1. Power switch
2. **powerCON TRUE1**
3. **Ethernet** with **L (link) and G (gigabit) LEDs**
4. **USB 2.0**
5. **MIDI In+Out**
6. **LTC**
7. **DC Remote Control**

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual I/O Nodes**.

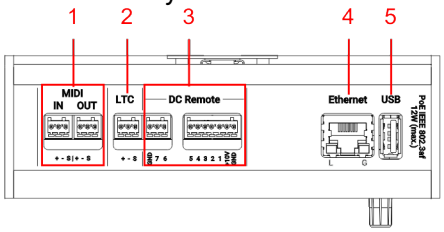
1.3.8. grandMA3 I/O Node DIN-Rail



grandMA3 I/O Node

DIN Rail front panel

1. **Terminal block**
2. Rotary knob



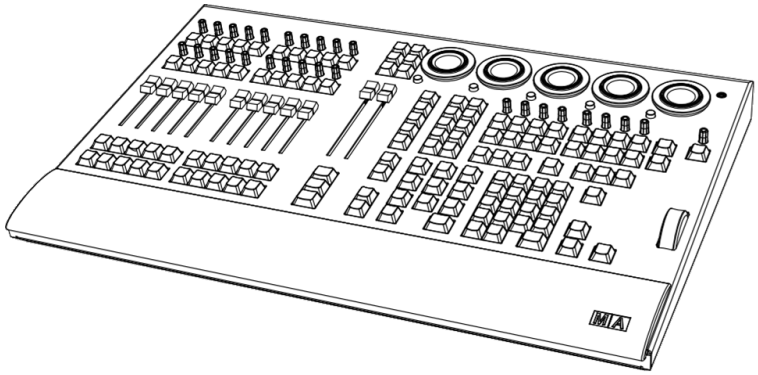
grandMA3 I/O Node

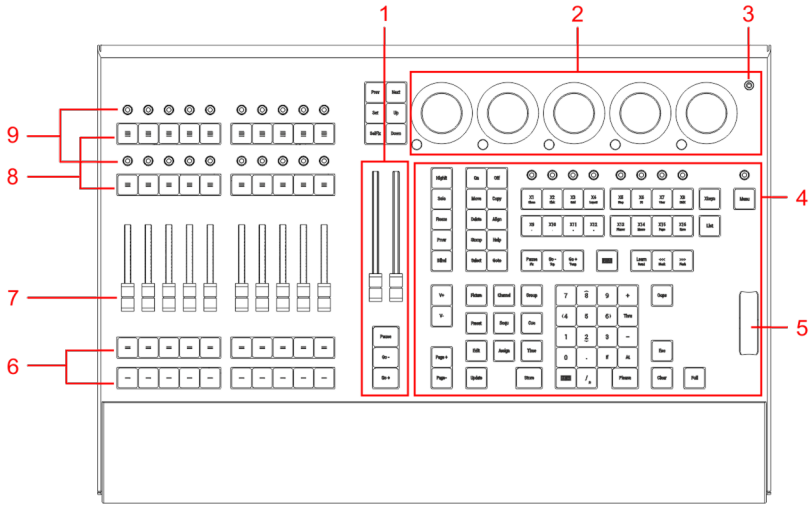
DIN Rail rear panel

1. **MIDI In+Out**
2. **LTC**
3. **DC Remote Control**
4. **Ethernet** with **L (link) and G (gigabit) LEDs**
5. **USB 2.0**

For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual I/O Nodes**.

1.3.9. grandMA3 onPC command wing XT

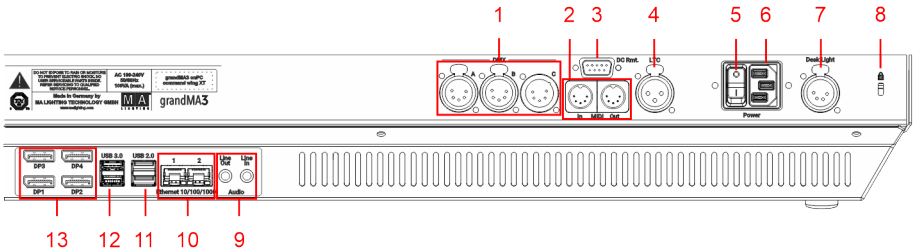




grandMA3 onPC command wing XT front panel

1. **Master area**
2. **Dual encoder section**
3. **Power key**
4. **Command area**
5. **Level wheel**
6. Executor buttons 101-190 + 201-290
7. Executor faders 201-290
8. Executor buttons 301-390 + 401-490
9. Executor knobs 301-390 + 401-490

For more information about executors, see **Executor elements**.

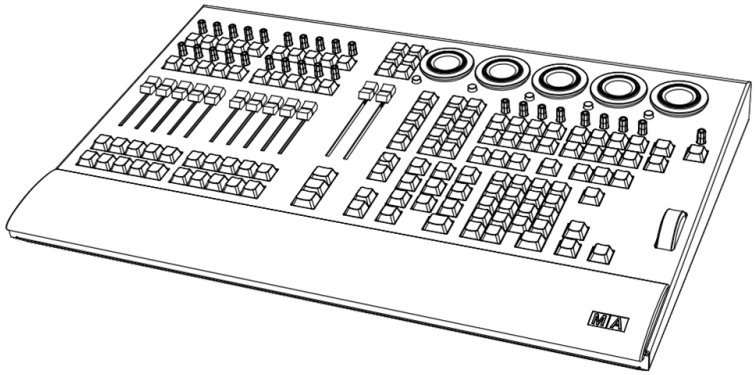


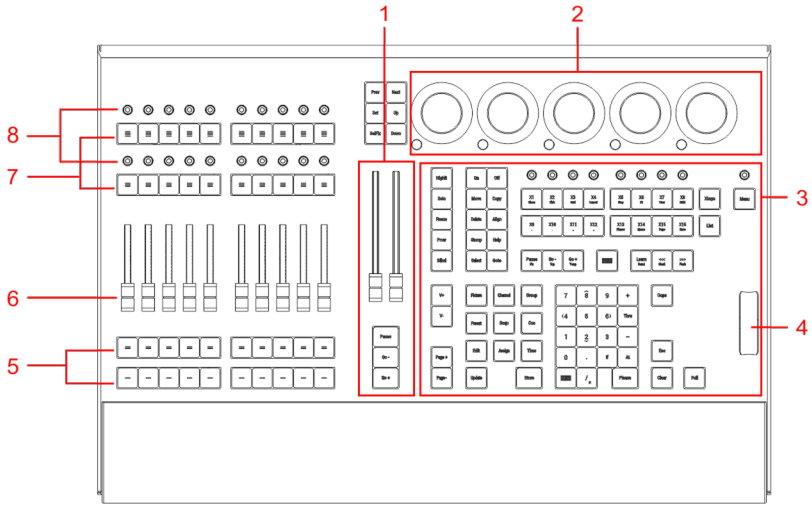
grandMA3 onPC command wing XT rear panel

1. **DMX A, B, C**
2. **MIDI In+Out**
3. **DC Remote Control**
4. **LTC**
5. Power switch
6. **IEC connector**
7. **Desk light**
8. Kensington lock
9. **Line In+Out**
10. **Ethernet 1+2**
11. **USB 2.0**
12. **USB 3.0**
13. **DisplayPort 1-4**

For technical specifications see **Technical Data** in the **grandMA3 Quick Manual onPC command wing XT**.

1.3.10. grandMA3 onPC command wing

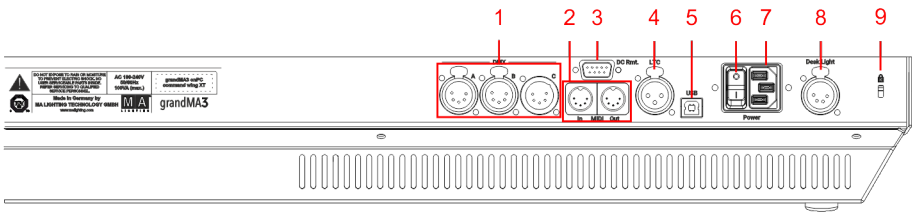




grandMA3 onPC command wing front panel

1. **Master area**
2. **Dual encoder section**
3. **Command area**
4. **Level wheel**
5. Executor buttons 101-190 + 201-290
6. Executor faders 201-290
7. Executor buttons 301-390 + 401-490
8. Executor knobs 301-390 + 401-490

For more information about executors, see **Executor elements**.

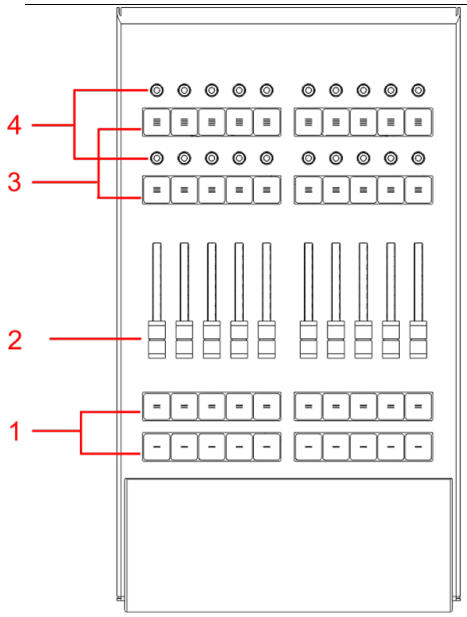


grandMA3 onPC command wing rear panel

1. **DMX A, B, C**
2. **MIDI In+Out**
3. **DC Remote Control**
4. **LTC**
5. USB
6. Power switch
7. **IEC connector**
8. **Desk light**
9. Kensington lock

For technical specifications see **Technical Data** in the **grandMA3 Quick Manual onPC command wing**.

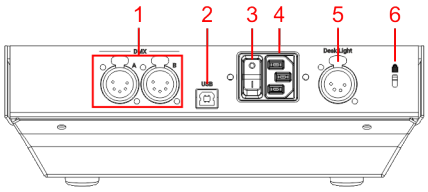
1.3.11. grandMA3 onPC fader wing



grandMA3 onPC fader wing front panel

1. Executor buttons 101-190 + 201-290
2. Executor faders 201-290
3. Executor buttons 301-390 + 401-490
4. Executor knobs 301-390 + 401-490

For more information about executors, see **Executor elements**.

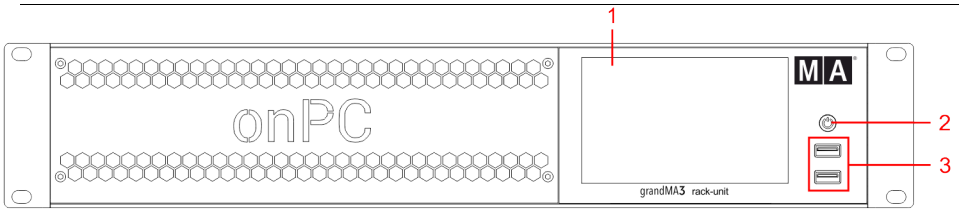


grandMA3 onPC fader wing rear panel

1. **DMX A+B**
2. USB
3. Power switch
4. **IEC connector**
5. **Desk light**
6. Kensington lock

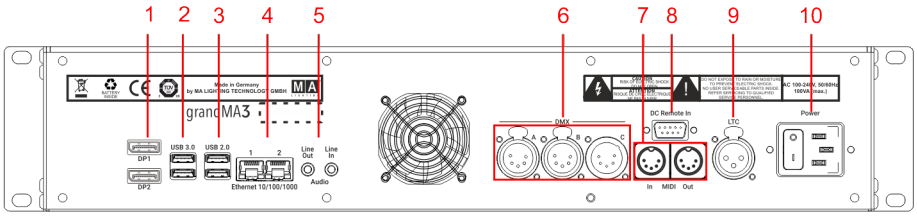
For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

1.3.12. grandMA3 onPC rack-unit



grandMA3 onPC rack-unit front panel

1. Screen
2. **Power key**
3. **USB 2.0**



grandMA3 onPC rack-unit rear panel

1. **DisplayPort 1+2**
2. **USB 3.0**
3. **USB 2.0**
4. **Ethernet 1+2**
5. **Line In+Out**
6. **DMX A-C**
7. **MIDI In+Out**
8. **DC Remote Control**
9. **LTC**
10. Power switch + **IEC connector**

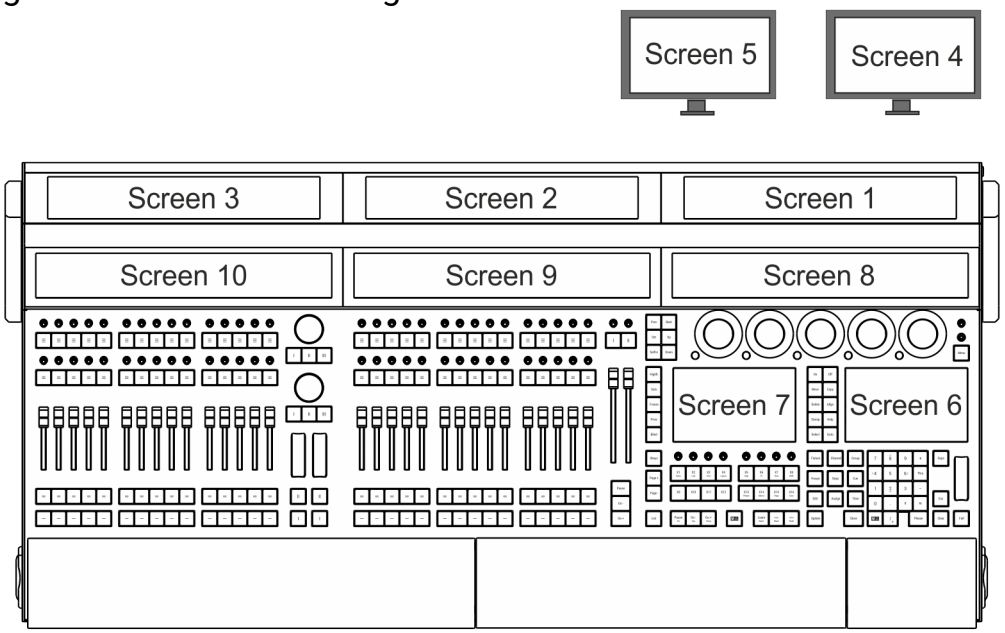
For technical specifications see **Technical Data** in the **grandMA3 Quick Manual onPC rack-unit**.

1.3.13. Screen Allocation

Numerical Order of Screens

grandMA3 consoles number their screens in a specific order. Different consoles within the range have different amounts of internal screens and connections for external monitors. This topic covers those physical differences as well as the numbering system applied to those screens.

grandMA3 full-size and light



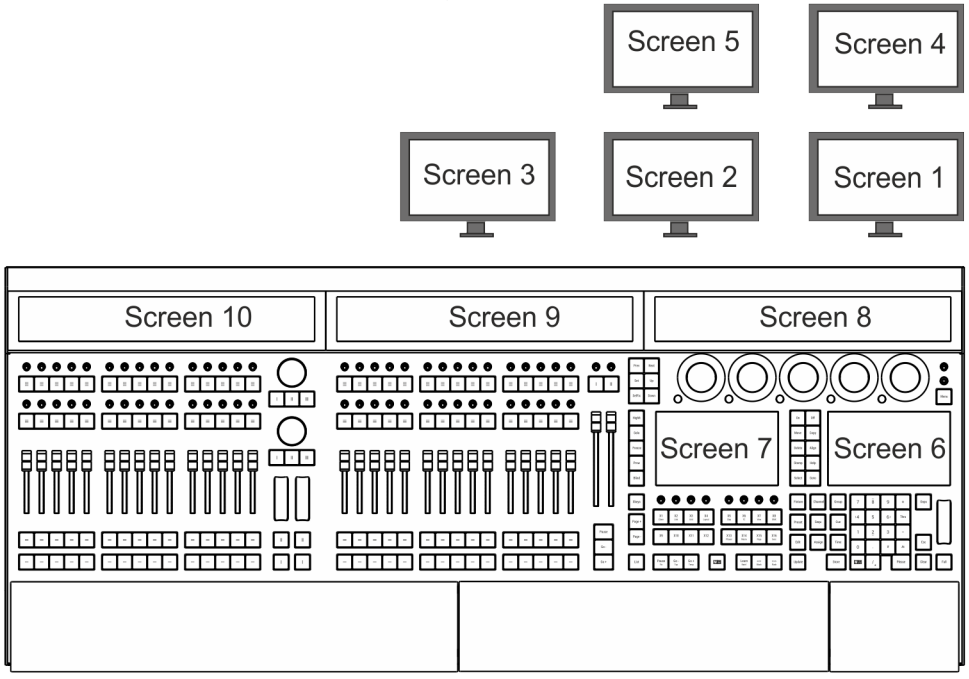
grandMA3 full-size with all screens numbered

The grandMA3 full-size includes three internal 15.6-inch screens mounted in a double-hinged monitor wing, three internal 14.9-inch letterbox screens, two internal 7-inch screens, and two connections for optional external monitors. The grandMA3 light includes two internal 15.6-inch screens mounted in a double-hinged monitor wing, two internal 14.9-inch letterbox screens, two internal 7-inch screens, and two connections for optional external monitors. The numbering of these screens is detailed in the following table:

Screen number	Size	Remark
Screen 1	15.6-inch	
Screen 2	15.6-inch	
Screen 3	15.6-inch	Only on grandMA3 full-size
Screen 4		External screen
Screen 5		External screen
Screen 6	7-inch	Right command screen

Screen 7	7-inch	Left command screen
Screen 8	14.9-inch	Letterbox encoder screen
Screen 9	14.9-inch	Right letterbox executor screen
Screen 10	14.9-inch	Left letterbox executor screen, on grandMA3 full-size or first grandMA3 extension connected to grandMA3 light
Screen 11	14.9-inch	Letterbox executor screen, on grandMA3 extension connected to grandMA3 full-size or second grandMA3 extension connected to grandMA3 light

grandMA3 full-size CRV and light CRV



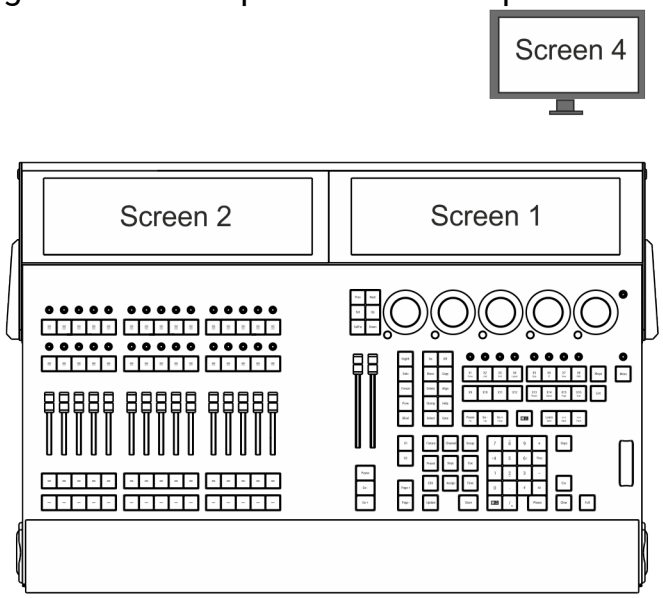
grandMA3 full-size CRV with all screens numbered

The grandMA3 full-size CRV includes three internal 14.9-inch letterbox screens, two internal 7-inch screens, and five connections for optional external monitors. The grandMA3 light CRV includes two internal 14.9-inch letterbox screens, two internal 7-inch screens, and four connections for optional external monitors. The numbering of these screens is detailed in the following table:

Screen number	Size	Remark
Screen 1		External screen
Screen 2		External screen
Screen 3		External screen (only grandMA3 full-size CRV)
Screen 4		External screen
Screen 5		External screen
Screen 6	7-inch	Right command screen
Screen 7	7-inch	Left command screen
Screen 8	14.9-inch	Letterbox encoder screen
Screen 9	14.9-	Right letterbox executor screen

	inch	
Screen 10	14.9-inch	Left letterbox executor screen, on grandMA3 full-size CRV or grandMA3 extension connected to grandMA3 light CRV
Screen 11	14.9-inch	Letterbox executor screen, on grandMA3 extension connected to grandMA3 full-size CRV or second grandMA3 extension connected to grandMA3 light CRV

grandMA3 compact XT and compact

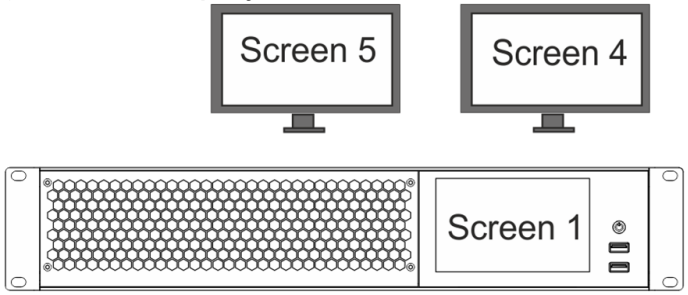


grandMA3 compact XT with all screens numbered

The grandMA3 compact XT includes two internal 15.6-inch screens mounted in a hinged monitor wing and one connection for an optional external monitor. The grandMA3 compact includes one internal 15.6-inch screen mounted in a hinged monitor wing and one connection for an optional external monitor. The numbering of these screens is detailed in the following table:

Screen number	Size	Remark
Screen 1	15.6-inch	
Screen 2	15.6-inch	Only on grandMA3 compact XT
Screen 4		External screen

grandMA3 replay unit



grandMA3 replay unit with all screens numbered

The grandMA3 replay unit includes two connections for optional external monitors. The numbering of these screens is detailed in the following table:

Screen number	Size	Remark
Screen 4		External screen
Screen 5		External screen
Screen 10	14.9-inch	Letterbox executor screen on first grandMA3 extension connected to grandMA3 replay unit
Screen 11	14.9-inch	Letterbox executor screen on second grandMA3 extension connected to grandMA3 replay unit
Screen 12	14.9-inch	Letterbox executor screen on third grandMA3 extension connected to grandMA3 replay unit

For more information on connecting external screens, see the **Connect External Screens** topic.

grandMA3 onPC rack-unit and grandMA3 onPC command wing XT

For onPC stations, the single windows for each display can be freely arranged. For more information, see the **Displays in grandMA3 onPC** topic.

1.3.14. Shortcuts

The following topics describe the use of shortcuts in the grandMA3 software.

Subtopics

- **Keyboard Shortcuts**
- **Framework Shortcuts**

1.3.14.1. Keyboard Shortcuts

Keyboard shortcuts allow fast operation of the console and onPC command wings via the (internal) keyboard.

Some of these user-editable shortcuts are for general use and override other commands. Others are related to specific windows and pop-ups.

The **KeyboardShortcut keyword** is used to edit and list the keyboard shortcuts. To modify the keyboard shortcuts, use the **Set keyword**.

Keyboard Shortcuts are part of the UserProfile. This makes it possible for every user/user profile to use their own shortcut definitions within a show file.

Turn the Keyboard Shortcuts on

- Tap ShCuts next to the command line or press F10 on your keyboard.

Keyboard shortcuts enabled The keyboard shortcuts are turned on (yellow text).

Turn the Keyboard Shortcuts off

- Tap ShCuts next to the command line or press F10 on your keyboard.

Keyboard shortcuts disabled The keyboard shortcuts are turned off (white text).

	Hint:
	In certain overlays and pop-ups where the shortcuts are not relevant, they are switched off and back on automatically. The shortcut icon turns red.

Shortcut Examples

Command section window left

Command section window right

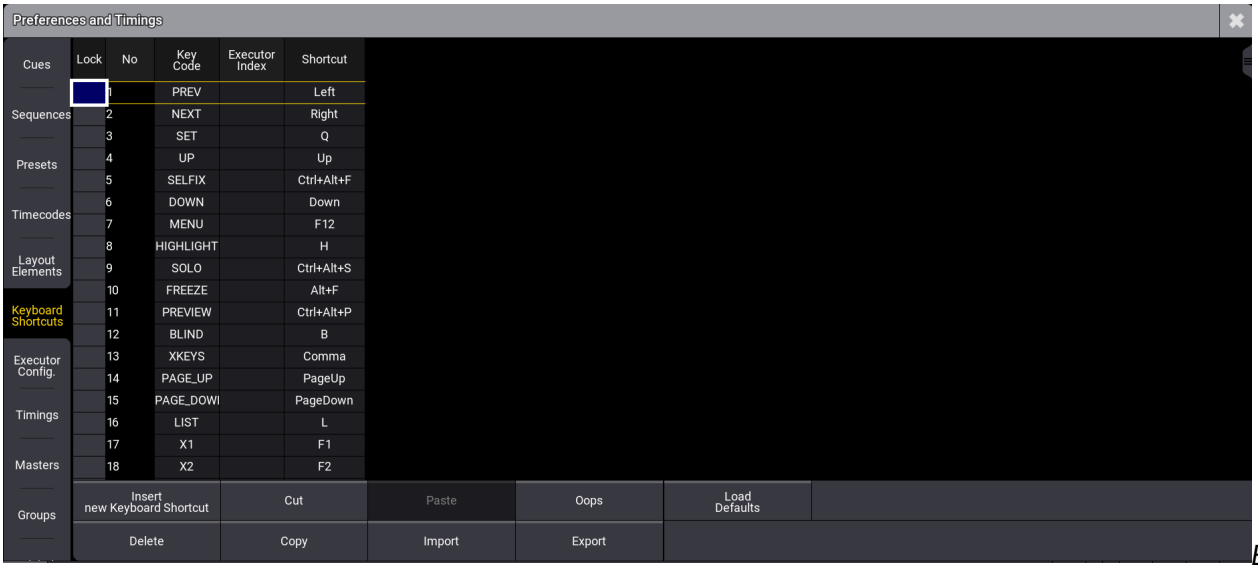
Playback controls window

The images above show the activated shortcut overlay. The overlay color can be changed in the **color theme**.

Keyboard Shortcuts Window

1. To edit the shortcuts, open the Preferences and Timings window.
2. Tap Keyboard Shortcuts.

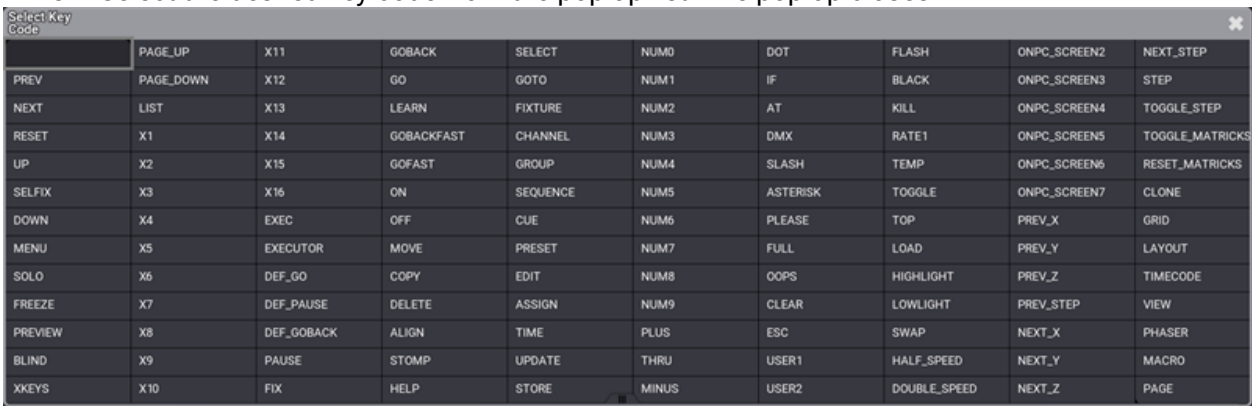
The Keyboard Shortcuts window opens:



Edit Keyboard Shortcuts window

Add Keyboard Shortcuts

1. To add a new keyboard shortcut at the end of the list, scroll down and tap New KeyboardShortcut and then tap Insert new Keyboard Shortcut. A new empty cell is generated at the end of the list.
2. Tap and hold or right click on the empty cell. The Select Key Code pop-up opens.
3. Select the desired key code from the pop-up list. The pop-up closes.



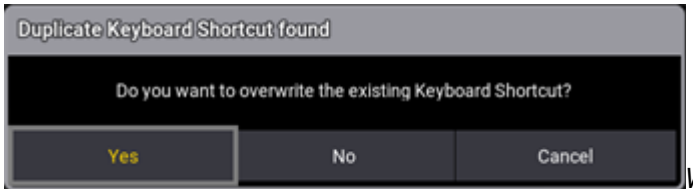
Select key code window

4. To open the Edit Keyboard Shortcut pop-up, tap and hold or right click on the empty cell in the Shortcut row.
5. To enter the desired shortcut, press the relevant keys. The pop-up closes.

Edit Keyboard shortcut pop-up As Enter can also be used as a shortcut, it is not possible to close the Edit keyboard shortcut pop-up with Enter.

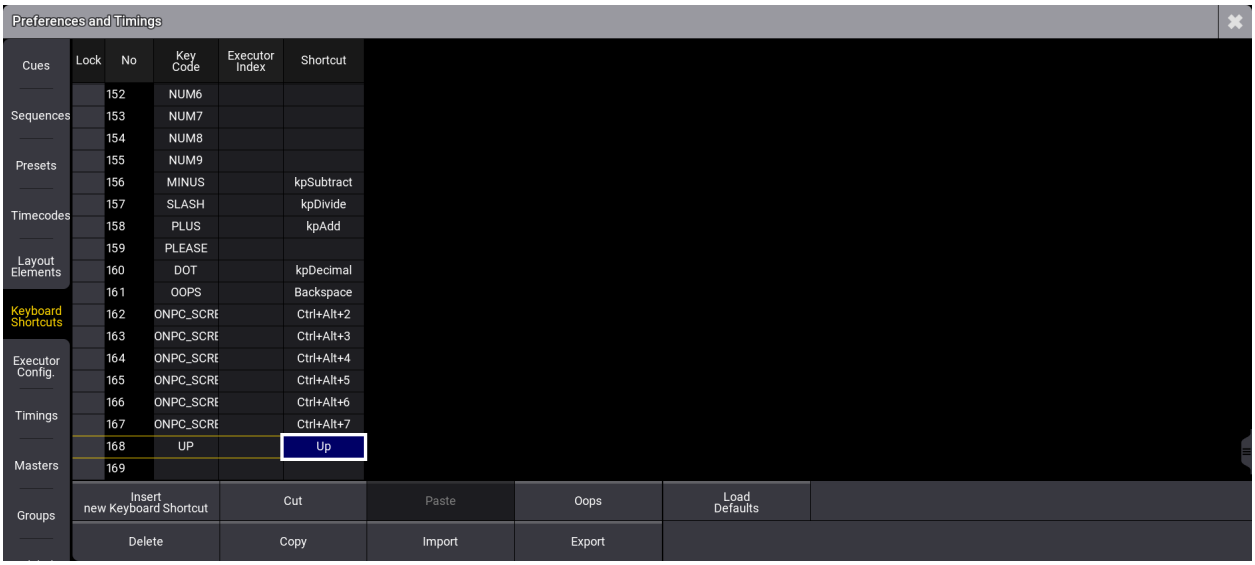
- To confirm the shortcut tap Ok.
- To clear the shortcut, tap None.
- To cancel the edit, tap X.

6. If a keyboard shortcut already exists, a warning pop-up appears. Tap Yes to save the new shortcut.



Warning pop-up

7. The new shortcut is added to the list.



New shortcut is added

Edit Existing Keyboard Shortcut

1. To edit an existing shortcut definition, tap and hold the Keyboard Shortcut you want to edit.

The Edit keyboard shortcut pop-up opens:

Edit keyboard shortcut pop-up

2. Enter the new shortcut.

Delete Keyboard Shortcuts

3. To select the keyboard shortcut to be deleted, tap the desired shortcut.
4. Tap Delete.

5. The keyboard shortcut is deleted. To undo a deletion, tap Oops.

Reset Keyboard Shortcuts to Defaults

- To reset the keyboard shortcuts to their defaults, tap Load Defaults. The keyboard shortcuts are reset to their defaults.

	**Hint:**
	The Shift keys on the keyboard correspond with the MA keys on the console.

Import / Export Keyboard Shortcuts

- To import or export the keyboard shortcuts, tap Import or Export.

By default, files will be exported to the relevant folder within the library folder structure, either on the local drive of the console or onPC station, or on a selected USB drive. For more information about this folder structure, see the **Folder Structure** topic.

For information about exporting and importing show data using command line syntax without the use of these menus, see the **Export keyword** and **Import keyword** topics.

1.3.14.2. Framework Shortcuts

The framework of the grandMA3 software uses the following keyboard shortcuts for user interface related grids, for example, in the Patch menu or the Fixture Builder.

Press key	Command	Description
Insert	Insert UIGridSelection	Executing a UI related insert function.
Ctrl + Insert	Store UIGridSelection	Executing a UI related store function.
Delete	Delete UIGridSelection	Executing a UI related delete function.
Ctrl + C	Copy UIGridSelection	Copying from a UI grid.
Ctrl + V	Paste UIGridSelection	Pasting into a UI grid.
Ctrl + X	Cut UIGridSelection	Cutting from a UI grid.
Ctrl + N	-	Toggles, for example, the "New Object Line" in a UI grid.
Ctrl + M	-	Toggles Merge children.
Ctrl + Enter	Edit UIGridSelection	Opens the Edit overlay for a selected cell.
Ctrl + Alt + D		Resets the color theme.

Hint:

On Apple® products, use control instead of Ctrl and option instead of Alt.

1.3.15. Keys

A shortcut is a series of one or several keys that invoke the software to perform a pre-programmed action.

In grandMA3, we differentiate between the term key and button.

A key is a (hard) key on the console or on the (integrated) keyboard.

A button is an element in the User Interface.

The keys F9, F10, F11, Pause and Print Screen provide system-wide shortcuts on all operating systems always available in all contexts as long as the program is running. These shortcuts cannot be edited.

The following shortcuts are available on all OS through several input methods:

Desk lock:

- Keyboard
  - F9
  - Pause
- Key
  - MA + MA + Pause | Fix
- Button
  - Lock

Toggle keyboard shortcuts:

- Keyboard
  - F10
- Button
  - ShCuts

Print screen:

- Keyboard
  - F11
  - Print

Reload UI (If the color theme is set to unreadable values):

- Key
  - MA + MA + Clear

Hint:
The Shift keys on the keyboard correspond with the MA keys on the console.

All other shortcuts can be edited by the user. For more information on see **Keyboard Shortcuts**.

This overview displays shortcuts for a quick execution of commands.

Press key	Command	Description
.	.	Enters . [Dot] into the command line. For more information see the **. [Dot] keyword**.
. .	Zero	Sets the dimmer value of the selected fixtures to 0.
MA .	Default	Sets the attributes to their default values.
<<<	<<<	Enters <<< [GoFastBackward] into the command line. For more information see the **<<< [GoFastBackward] keyword**.
Press and hold <<<		Allows several successive operations.
MA <<<	Black	Enters Black into the command line. For more information see the **Black keyword**.
Press and hold <<<		Allows several successive operations.
>>>	>>>	Enters >>> [GoFastForward] into the command line. For more information see the **>>> [GoFastBackward] keyword**.
Press and hold >>>		Allows several successive operations.
MA >>>	Flash	Enters Flash into the command line. For more information see the **Flash keyword**.
Press and hold >>>		Allows several successive operations.
-	-	Enters - [Minus] into the command line. For more information see the **- [Minus] keyword**.
- -	At Percent - 10	Decreases the intensity of the selected fixtures by 10%.
+	+	Enters + [Plus] into the command line. For more information see the **Plus keyword**.
+ +	At Percent + 10	Increases the intensity of the selected fixtures by 10%.
/	/	Enters / [Slash] into the command line. For more information see the **/ [Slash] keyword**.
MA /	*	Enters * [Asterisk] into the command line. For more information see the *** [Asterisk] keyword**.
At	At	Enters At into the command line. For more information see the **At keyword**.

Press key	Command	Description
At At	Normal	Sets the intensity to the Normal value in the selected fixtures.
MA At	Integrate	Enters Integrate into the command line. For more information see the **Integrate keyword**.
MA At At	Extract	Enters Extract into the command line. For more information see the **Extract keyword**.
Press and hold At		Opens the At Filter Overlay. For more information see **At Filter**.
Store At	/Embed	Enters /Embed into the command line. For more information see the **/Embed option keyword**.
Assign	Assign	Enters Assign into the command line. For more information see the **Assign keyword**.
Assign Assign	Label	Enters Label into the command line. For more information see the **Label keyword**.
MA Assign	Set	Enters Set into the command line. For more information see the **Set keyword**.
Press and hold Assign		Allows several successive operations.
Align	Align	Cycles through the different align modes.
Press and hold Align	Align "Off"	Sets the align mode to off.
MA Align	AlignTransition	Cycles through the different align transitions.
Press and hold MA Align	AlignTransition "Linear"	Sets the align transition to linear.
Blind	Blind	Toggles the blind mode. For more information see the **Blind keyword**.
Clear	Clear	Clears the selected fixtures. For more information see the **Clear keyword**.
Clear Clear	Clear	Clears the active values. For more information see the **Clear keyword**.
Clear Clear Clear	Clear	Clears the programmer. For more information see the **Clear keyword**.
Press and hold Clear	ClearAll	Clears the programmer and command filter. You can also press 3 x Clear. For more information see the **ClearAll keyword**.
Channel	Channel	Enters Channel into the command line. For more information see the **Channel keyword**.

Press key	Command	Description
Channel Channel	Channel Universal and other ID types in use	Pressing Channel repeatedly switches through Channel, Universal, and other ID types in use. If ID types are not assigned to patched fixtures, they are not use and hence will not be toggled in the command line. For more information see the **Channel key**, **Universal keyword**, **Command Line - Change the Default Keyword**.
Copy	Copy	Enters Copy into the command line. For more information see the **Copy keyword**.
Copy Copy	Paste	Enters Paste into the command line. For more information see the **Paste keyword**.
Copy Copy Copy	Insert	Enters Insert into the command line. For more information see the **Insert keyword**.
MA Copy	Cut	Enters cut into the command line. For more information see the **Cut keyword**.
Press and hold Copy		Allows several successive operations.
Ctrl		Press and hold to use multi-selection in sheets. To initiate a clean start press and hold when the Mode Selector pop-up appears during the boot process.
Cue	Cue	Enters Cue into the command line. For more information see the **Cue keyword**.
Cue Cue	Part	Enters Part into the command line. For more information see the **Part keyword**.
MA Cue	Programmer	Enters Programmer into the command line. For more information see the **Programmer keyword**.
Down	Down	Navigates downward in the structure of fixtures or subfixtures.
Delete	Delete	Enters Delete into the command line. For more information see the **Delete keyword**.
Delete Delete	Remove	Enters Remove into the command line. For more information see the **Remove keyword**.
Delete Delete Delete	Release	Enters Release into the command line. For more information see the

Press key	Command	Description
		Release keyword.
Press and hold Delete		Allows several successive operations.
ESC		Closes pop-ups, returning keyboard focus to the command line. If there are no open pop-ups, ESC will delete unexecuted commands in the command line.
Press and hold for 5 seconds MAESC -or- Press and hold for 5 seconds ShiftESC		Cancels operations that take a longer period of time to process. For example, such operations are copying or storing large amount of data.
Edit	Edit	Enters Edit into the command line. For more information see the **Edit keyword**.
Edit Edit	EditSetting	Enters EditSetting into the command line. For more information see the **EditSetting keyword**.
MAEdit	EditRecipe	Enters EditRecipe into the command line. For more information see the **EditRecipe keyword**.
Freeze	Freeze	Toggles the Freeze function.
Full	Full	Sets the intensity to 100% in the selected fixtures. For more information see the **Full keyword**.
Fixture	Fixture	Enters Fixture into the commnd line. For more information see the **Fixture keyword**.
Fixture Fixture	Selection	Enters Selection into the command line. For more information see **Selection keyword**.
FixtureFixture Fixture	Collection	Enters Collection into the command line. For more information see the **Collection keyword**.
Group	Group	Enters Group into the commnd line. For more information see the **Group keyword**.
Group Group	World	Enters World into the command line. For more information see the **World keyword**.
Group Group Group	Filter	Enters Filter into the command line. For more information see the **Filter keyword**.
MA Group	Tag	Enters Tag into the command line.

Press key	Command	Description
		For more information see the **Tag keyword**.
Goto	Goto	Enters Goto into the command line. For more information see the **Goto keyword**.
Goto Goto	Load	Enters Load into the command line. For more information see the **Load keyword**.
MA Goto Goto	Loaded	Enters Loaded into the command line. For more information see the **Loaded keyword**.
Go+ [Large]	Go+ Executor	Executes Go+ Executor into the command line and affects the selected sequence. For more information see the **Go+ keyword**.
MA Go+ [Large]	Go+ Loaded	Executes Go+ Loaded into the command line. For more information see the **Loaded keyword**.
Go- [Large]	Go- Executor	Executes Go- Executor into the command line and affects the selected sequence. For more information see the **Go- keyword**.
Go+ | Temp	Go+	Enters Go+ into the command line. For more information see the **Go+ keyword**.
Press and hold Go+ | Temp		Allows several successive operations.
Go+ | TempGo+ | Temp	Unpark	Enters Unpark into the command line. For more information see the **Unpark keyword**.
MA Go+ | Temp	Temp	Enters Temp into the command lilne. For more information see the **Temp keyword**.
Press and hold MAGo+ | Temp		Allows several successive operations.
MA Go+ | Temp Go+ | Temp	Toggle	Enters Toggle into the command line. For more information see the **Toggle keyword**.
Go- | Top	Go-	Enters the Go- into the command line. For more information see the **Go- keyword**.
Press and hold Go- | Top		Allows several successive operations.
MA Go- | Top	Top	Enters Top into the command line. For more information see the **Top keyword**.
Press and hold MA Go- | Top		Allows several successive operations.
MA Go- | Top Go- | Top	Kill	Enters Kill into the command line. For more information see the **Kill keyword**.

Press key	Command	Description
Help	Help	Enters Help into the command line. For more information see the **Help keyword**.
HighIt	Highlight	Toggles the highlight function on and off. For more information see the **Highlight keyword**.
MA HighIt	Lowlight	Toggles the lowlight function on and off. For more information see the **Lowlight keyword**.
If	IfOutput If EndIf	If the command line is empty, IfOutput will be entered into the command line. For more information see the **IfOutput keyword**. If any other command has already been entered, pressing If again will enter If into the command line. For more information see the **If keyword**. If If has already been entered into the command line, pressing If will enter EndIf into the command line. For more information see the **EndIf keyword**.
If If	IfActive	Enters IfActive command into the command line. For more information see the IfActive keyword.
If If If	IfProgammer	Enters IfProgrammer into the command line. For more information see the **IfProgrammer keyword**.
If If If If	If	Enters If into the command line. For more Information see the **If keyword**.
Learn | Rate1	LearnSpeed	Enters LernSpeed into the command line. For more information see the **LearnSpeed keyword**.
Press and hold Learn | Rate1		Allows several successive operations.
MA Learn | Rate1	Rate1	Enters Rate1 into the command line. For more information see the **Rate1 keyword**.
Press and hold Learn | Rate1		Allows several successive operations.
List	List	Enters List into the command line. For more information see the **List keyword**.
MA List	ListReference	Enters ListReference into the command line. For more

Press key	Command	Description
		information see the **ListReference keyword**.
Press and hold MA + any other key		Pressing and holding MA in combination with other keys provides shortcuts to other functions. It works the same as using Shift on a keyboard.
Press and hold MA		If the name of a pool object is hidden or the pool object displays a custom name, pressing and holding MA displays the name of media files. This works with the following pools: Appearance pool, Gobo pool, Image pool, Mesh pool, Sound pool, and Symbol pool. It willl also show numbers of assigned objects on view buttons and scroll thorugh the address string of the assigned object on executors if they are in other data pools.
MA Please		Sets focus to the command line.
Menu		Opens the menu pop-up.
Menu Menu	SaveShow	Saves the show file using the current name. For more information see the **SaveShow keyword**.
Move	Move	Enters Move into the command line. For more information see the **Move keyword**.
Move Move	Exchange	Enters Exchange into the command line. For more information see the **Exchange keyword**.
Press and hold Move		Allows several successive operations.
Next	Next	Enters Next into the command line. For more information see the **Next keyword**.
MA Next	Next Step	Enters Next Step into the command line. For more information see the **Step keyword**.
Oops	Oops	Undoes the last operation or selection. For more information see the **Oops keyword**.
Press and hold Oops		Opens the Oops window.

Press key	Command	Description
On	On	Enters On into the command line. For more information see the **On keyword**.
OnOn	Call	Enters Call into the command line. For more iformation see the **Call keyword**.
Press and hold On		Allows several successive operations.
Off	Off	Enters Off into the command line. For more information see the **Off keyword**.
Off Off		Opens the Off Menu.
MA Off	Deactivate	Enters Deactivate into the command line. For more information see the **Deactivate keyword**.
MA MA Off		Closes any active RemoteHID connections.
Press and hold Off		Allows several successive operations.
Pause [Large]	Pause Executor	Toggles pause on and off in the selected sequence. For more information see the **Pause keyword**.
Pause | Fix	Pause	Enters Pause into the command line. For more information see the **Pause keyword**.
Press and hold Pause | Fix		Allows several successive operations.
Pause | Fix Pause | Fix	Park	Enters Park into the command line. For more information see the **Park keyword**.
MA Pause | Fix	Fix	Enters Fix into the command line. For more information see the **Fix keyword**.
MA MA Pause | Fix		Toggles the desk lock.
Press and hold MAPause | Fix		Allows several successive operations.
Page+	Next Page	Calls the next page. For more information see the **Page keyword**.
Page-	Previous Page	Calls the previous page. For more information see the **Previous keyword**.
Press and hold Page-	Page	Calls page 1.
Please		Executes the command entered in the command line.
Please Please		Activates all attributes in the selected fixtures. If you keep on repeatedly pressing Please, the activation will be toggled.
Power		Powers up or powers down the

Press key	Command	Description
		console.
Preset + 1 , 2	Select EncoderBank 1, 2	Changes between encoder banks.
Preset	Preset	Enters Preset into the command line. For more information see the **Preset keyword**.
Preset Preset	Attribute	Enters Attribute in the command line. For more information see the **Attribute keyword**.
Preset Preset Preset	Gel	Enters Gel into the command line. For more information see the **Gel keyword**.
MA Preset	FeatureGroup	Enters FeatureGroup into the command line. For more information see the **FeatureGroup keyword**.
MA Preset Preset	DataPool	Enters DataPool into the command line. For more information see the **DataPool keyword**.
Prev	Previous	Selects the previous fixture. For more information see the **Previous keyword** .
MA Prev	Previous Step	Selects the previous step. For more information see the **Step keyword**.
Select	Select	Enters Select into the command line. For more information see the **Select keyword**.
Select Select	Collect	Enters Collect into the command line. For more information see the **Collect keyword**.
Press and hold Select		Allows several successive operations.
SelFix	SelectFixtures	Enters SelectFixtures into the command line. For more information see the **SelectFixtures keyword**.
SelFix SelFix	Shuffle	Enters Shuffle into the command line. For more information see the **Shuffle keyword**.
Press and hold SelFix		Allows several successive operations.
Sequ	Sequence	Enters Sequence into the command line. For more information see the **Sequence keyword**.
Set	Set Selection Property "Active"	Toggles MATricks on and off. For more information see the **MAtricks keyword**.
MA Set	Step Toggle	Toggles between the firstly selected step and all selected steps.
Solo	Solo	Toggles the solo function on and

Press key	Command	Description
		off. For more information see the **Solo keyword**.
Stomp	Stomp	Enters Stomp into the command line. For more information see the **Stomp keyword**.
Stomp Stomp	Capture	Enters Capture into the command line. For more information see the **Capture keyword**.
Press and hold Stomp		Allows several successive operations.
Store	Store	Enters Store into the command line. For more information see the **Store keyword**.
Press and hold Store		Opens Store Settings.
Thru	Thru	Enters Thru into the command line. For more information see the **Thru keyword**.
Time		Toggles CueFade and CueDelay, or through Relative, Fade, Delay, and Absolute. Which layers are toggled depends on two things: the Time Key Target of your user profile settings and the content of the command line. For more information see these keywords **CueFade**, **CueDelay**, **Relative**, **Fade**, **Delay** and **Absolute**. For more information on the Time Key Target see **User Settings**.
Up	Up	Navigates upward in the structure of fixtures or subfixtures.
Update	Update	Opens the Update menu.
MA Update	Cook Offset	Enters Cook into the command line. For more information see the **Cook keyword**. If there is already Store in the command line, this key combination will enter Offset into the command line. For more information see the **Offset keyword**.
X1|Clone		At the moment all Xkeys behave as executors. X1 is executor 291.
MA X1 | Clone	Clone	Enters Clone into the command line. For more information see the **Clone keyword**.
MA X1 | Clone X1 | Clone	Recast	Enters Recast into the command line. For more information see the **Recast keyword**.
X2 | Link		At the moment all Xkeys behave as executors. X2 is executor 292.
X3 | Grid		At the moment all Xkeys behave

Press key	Command	Description
		as executors. X3 is executor 293.
MA X3 | Grid	Grid	Enters Grid into the command line. For more information see the **Grid keyword**.
X4 | Layout		At the moment all Xkeys behave as executors. X4 is executor 294.
MA X4 | Layout	Layout	Enters Layout into the command line. For more information see the **Layout keyword**.
MA X4 | Layout X4 | Layout	Appearance	Enters Appearance into the command line. For more information see the **Appearance keyword**.
MA X4 | Layout X4 | Layout X4 | Layout	Scribble	Enters Scribble into the command line. For more information see the **Scribble keyword**.
X5 | Step		At the moment all Xkeys behave as executors. X5 is executor 295.
MA X5 | Step	Step	Enters Step into the command line. For more information see the **Step keyword**.
X6 | TC		At the moment all Xkeys behave as executors. X6 is executor 296.
MA X6 | TC	Timecode	Enters Timecode into the command line. For more information see the **Timecode keyword**.
MA X6 I TC X6 I TC	TimecodeSlot	Enters Timecode Slot into the command line. For more information see the **TimecodeSlot keyword**.
MA X6 I TC X6 I TC X6 I TC	Timer	Enters Timer into the command line. For more information see the **Timer keyword**.
X7 | View		At the moment all Xkeys behave as executors. X7 is executor 297.
MA X7 | View	View	Enters View into the command line. For more information see the **View keyword**.
MA X7 | View X7 | View	ViewButton	Enters ViewButton into the command line. For more information see the **ViewButton keyword**.
MA X7 | View X7 | View X7 | View	ScreenContent	Enters ScreenContent into the command line. For more information see the **ScreenContent keyword**.
X8 | DMX		At the moment all Xkeys behave as executors. X8 is executor 298.
MA X8 | DMX	DMXUniverse	Enters DMXUniverse into the command line. For more information see the **DMXUniverse keyword**.

Press key	Command	Description
MA X8 | DMX X8 | DMX	DMXAddress	Enters DMXAddress into the command line. For more information see the **DMXAddress keyword**.
X9		At the moment all Xkeys behave as executors. X9 is executor 191.
X10		At the moment all Xkeys behave as executors. X10 is executor 192.
X11		At the moment all Xkeys behave as executors. X11 is executor 193.
X12		At the moment all Xkeys behave as executors. X12 is executor 194.
X13 | Phaser		At the moment all Xkeys behave as executors. X13 is executor 195.
MA X13 | Phaser	Menu "PhaserEditorOnly"	Opens the Phaser overlay. For more information see **Phasers**.
X14 | Macro		At the moment all Xkeys behave as executors. X14 is executor 196.
MA X14 | Macro	Macro	Enters Macro into the command line. For more information see the **Macro keyword**.
MA X14 | Macro X14 | Macro	Plugin	Enters Plugin into the command line. For more information see the **Plugin keyword**.
MA X14 | Macro X14 | Macro X14 | Macro	Quickey	Enters Quickey into the command line. For more information see the **Quickey keyword**.
X15 | Page		At the moment all Xkeys behave as executors. X15 is executor 197.
MA X15 | Page	Page	Enters Page into the command line. For more information see the **Page keyword**.
X16 | Exec		At the moment all Xkeys behave as executors. X16 is executor 198.
MA X16 | Exec	Executor	Enters Executor into the command line. For more information see the **Executor keyword**.
MA X16 | Exec X16 | Exec	SpecialExecutor	Enters SpecialExecutor into the command line. For more information see the **SpecialExecutor keyword**.
MA X16 | Exec X16 | Exec X16 | Exec	FaderMaster	Enters FaderMaster into the command line. For more information see the **FaderMaster keyword**.

Subtopics

- **. [Dot]**
- **<<< [GoFastBackward] | Black**
- **>>> [GoFastForward] | Flash**

- **- [Minus]**
- **+ [Plus]**
- **/ [Slash] | * [Asterisk/Multiply]**
- **Align**
- **Assign**
- **At**
- **Blind**
- **Clear**
- **Channel**
- **Copy**
- **Ctrl**
- **Cue**
- **Delete**
- **Down**
- **Edit**
- **Esc**
- **Fixture**
- **Freeze**
- **Full**
- **Go+ | Temp**
- **Go+ [large]**
- **Go- [large]**
- **Go- | Top**
- **Goto**
- **Group**
- **Help**
- **Highlt [Highlight]**
- **If**
- **Learn | Rate1**
- **List**
- **MA**
- **Menu**
- **Move**
- **Next**
- **Numeric Keys | Arrows**
- **Off**
- **On**
- **Oops**
- **Page-**
- **Page+**
- **Pause [large]**
- **Pause | Fix**
- **Please**
- **Power**
- **Preset**
- **Prev [Previous]**
- **Prvw [Preview]**
- **Select**
- **SelFix [SelectFixtures]**
- **Sequ [Sequence]**

- **Set**
- **Stomp**
- **Solo**
- **Store**
- **Thru**
- **Time**
- **Up**
- **Update**
- **U1**
- **U2**
- **X1 | Clone**
- **X2 | Link**
- **X3 | Grid**
- **X4 | Layout**
- **X5 | Step**
- **X6 | TC**
- **X7 | View**
- **X8 | DMX**
- **X9**
- **X10**
- **X11**
- **X12**
- **X13 | Phaser**
- **X14 | Macro**
- **X15 | Page**
- **X16 | Exec**
- **Xkeys**

1.3.15.1. . [Dot]

Pressing . [Dot] enters a . into the command line.

Zero

Pressing . . enters the Zero keyword into the command line.

For more information about Zero, see the **Zero keyword**.

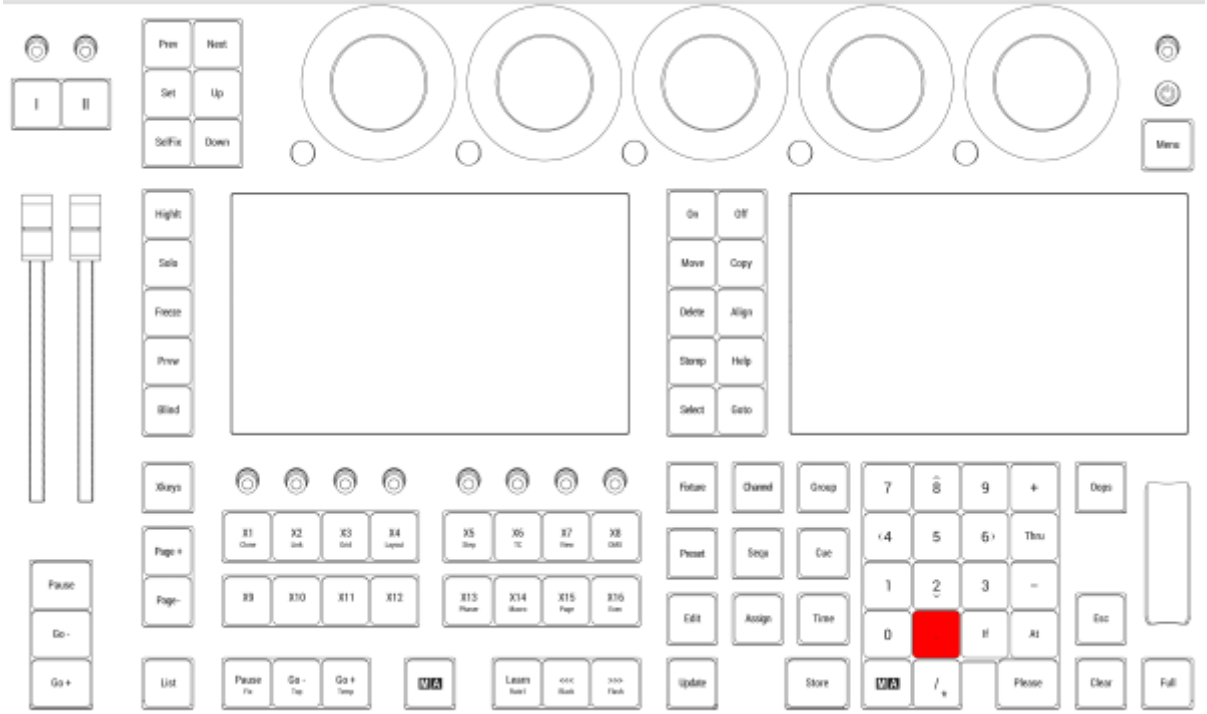
Default

Pressing MA + . enters the Default keyword into the command line.

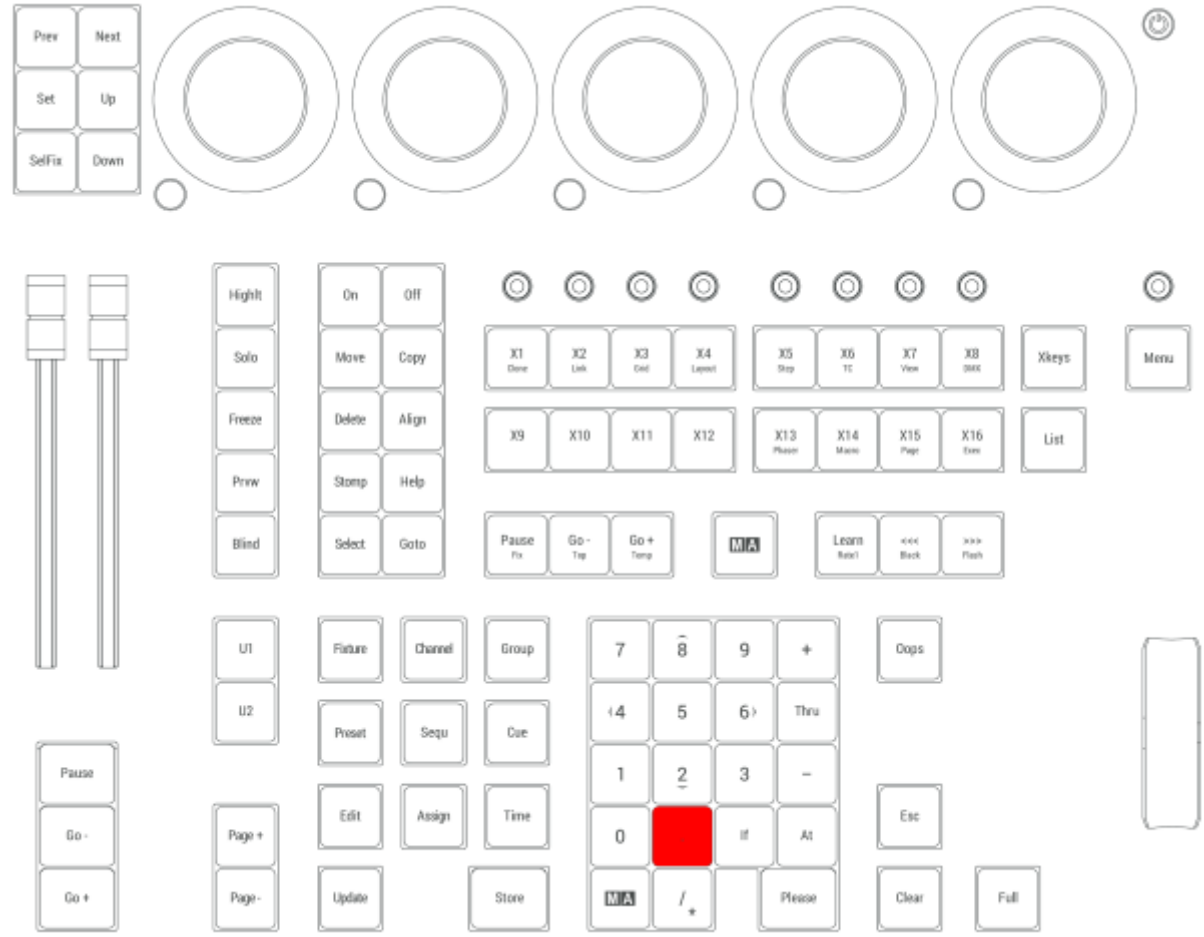
For more information about Default, see the **Default keyword**.

Location

The . [Dot] is located in the numeric keys section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.2. <<< [GoFastBackward] | Black

Pressing <<< enters the <<< keyword into the command line.

For more information about <<<, see the **GoFastBackward keyword**.

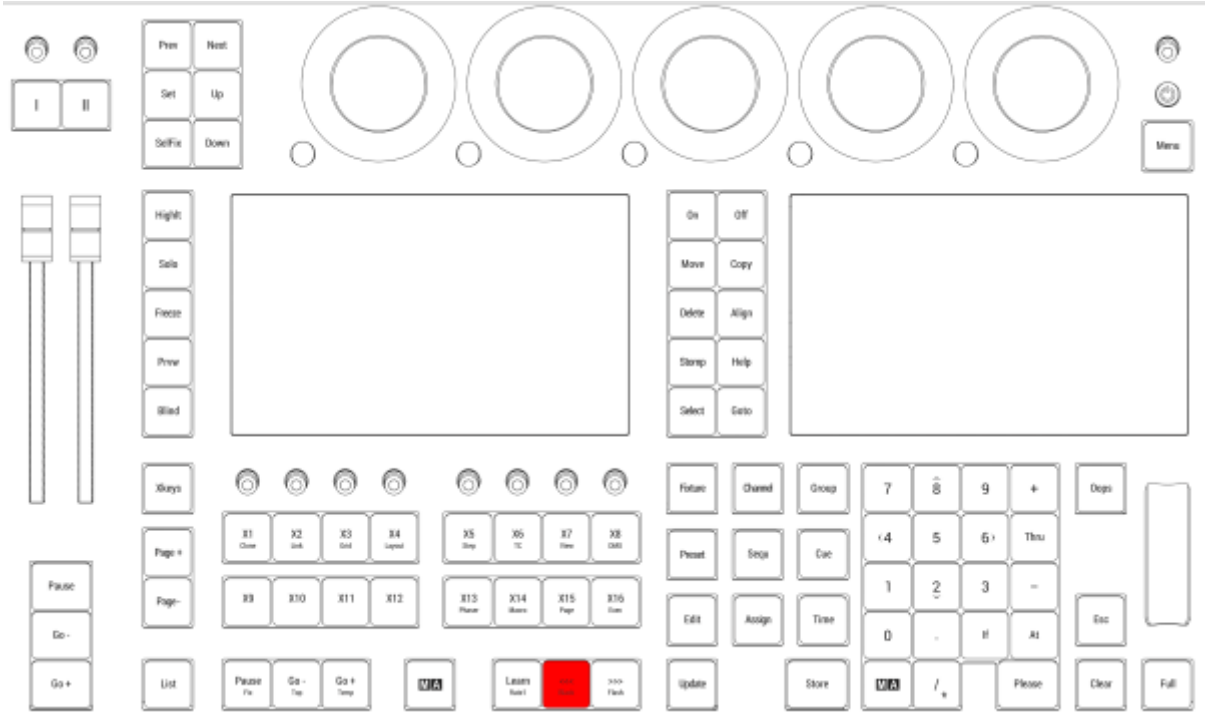
Black

Pressing and holding MA + <<< enters the Black keyword into the command line.

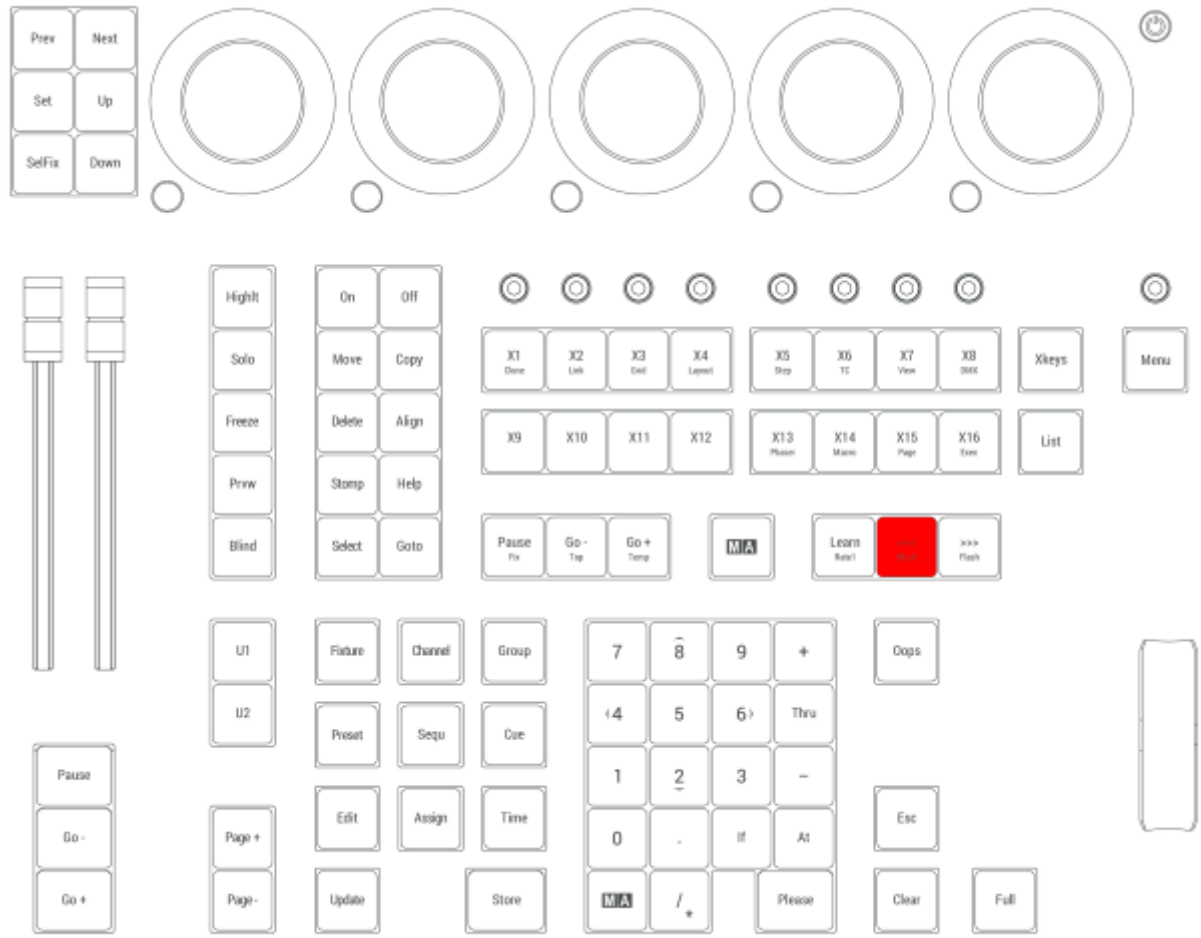
For more information about Black, see the **Black keyword**.

Location

<<< is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.3. >>> [GoFastForward] | Flash

Pressing >>> enters the >>> keyword into the command line.

For more information about >>>, see the **GoFastForward keyword**.

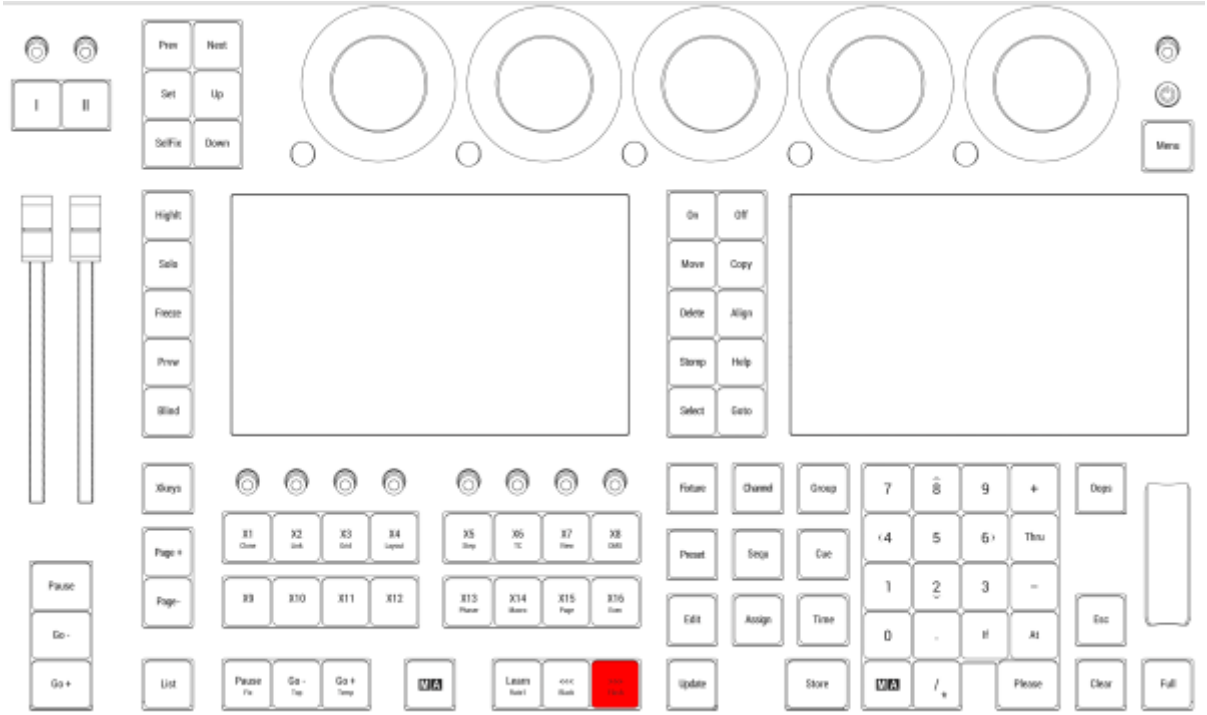
Flash

Pressing and holding MA + >>> enters the Flash keyword into the command line.

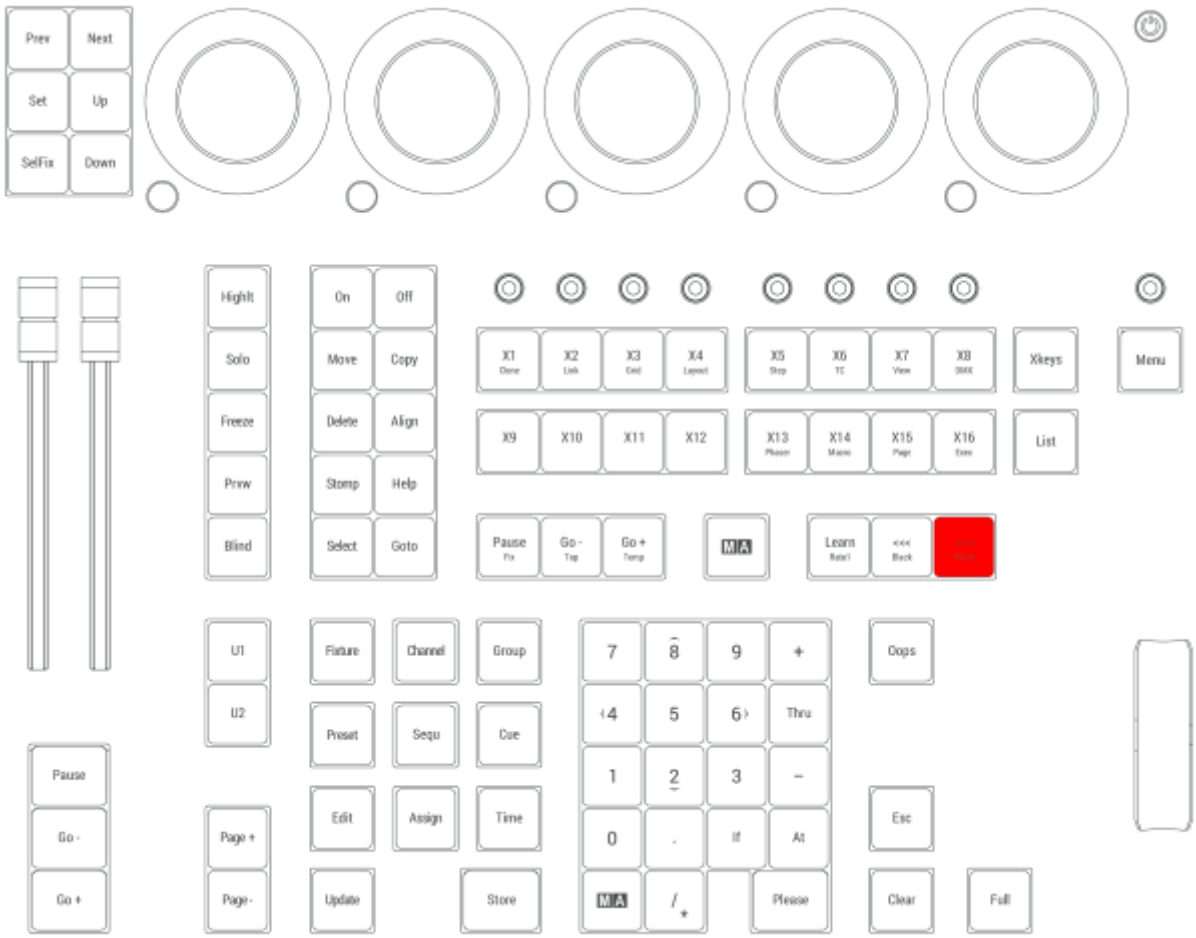
For more information about Flash, see the **Flash keyword**.

Location

>>> is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.4. - [Minus]

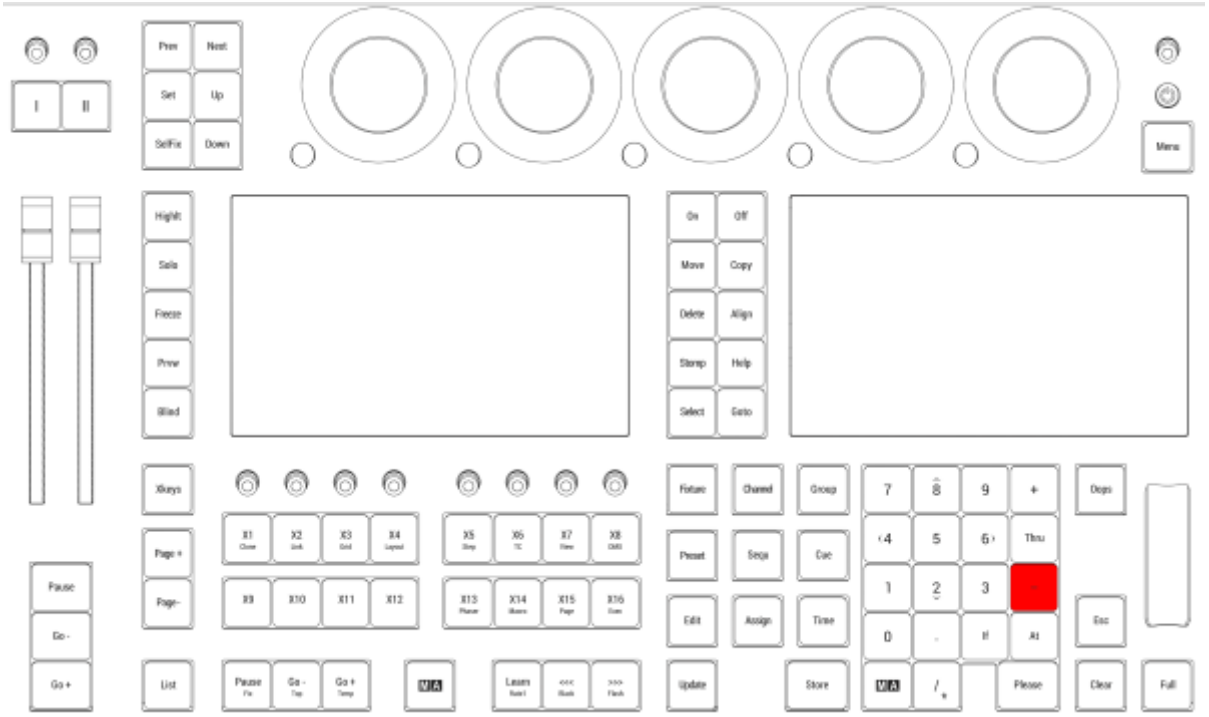
Pressing - enters a - into the command line.

At - 10

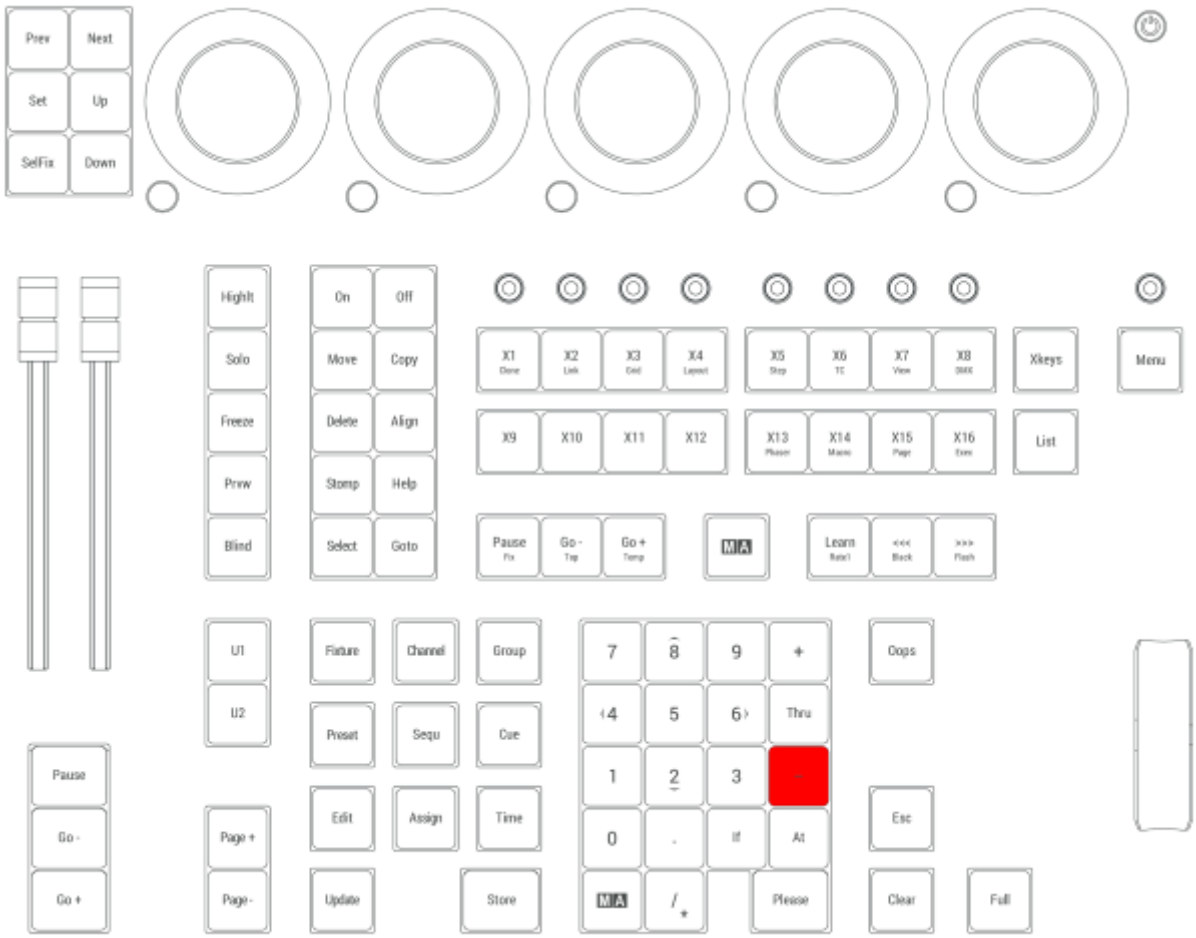
Pressing -- decreases the dimmer level of the selected fixtures by 10%.

Location

- is located in the numeric keys section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.5. + [Plus]

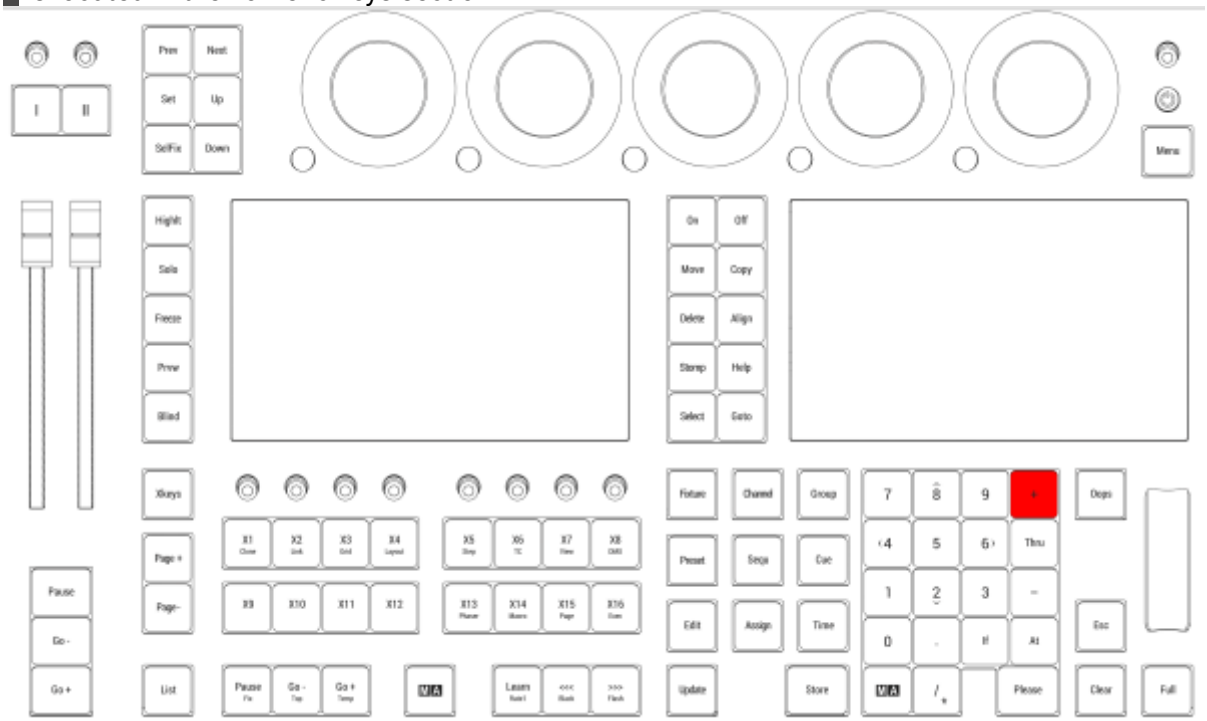
Pressing + enters the + into the command line.

At + 10

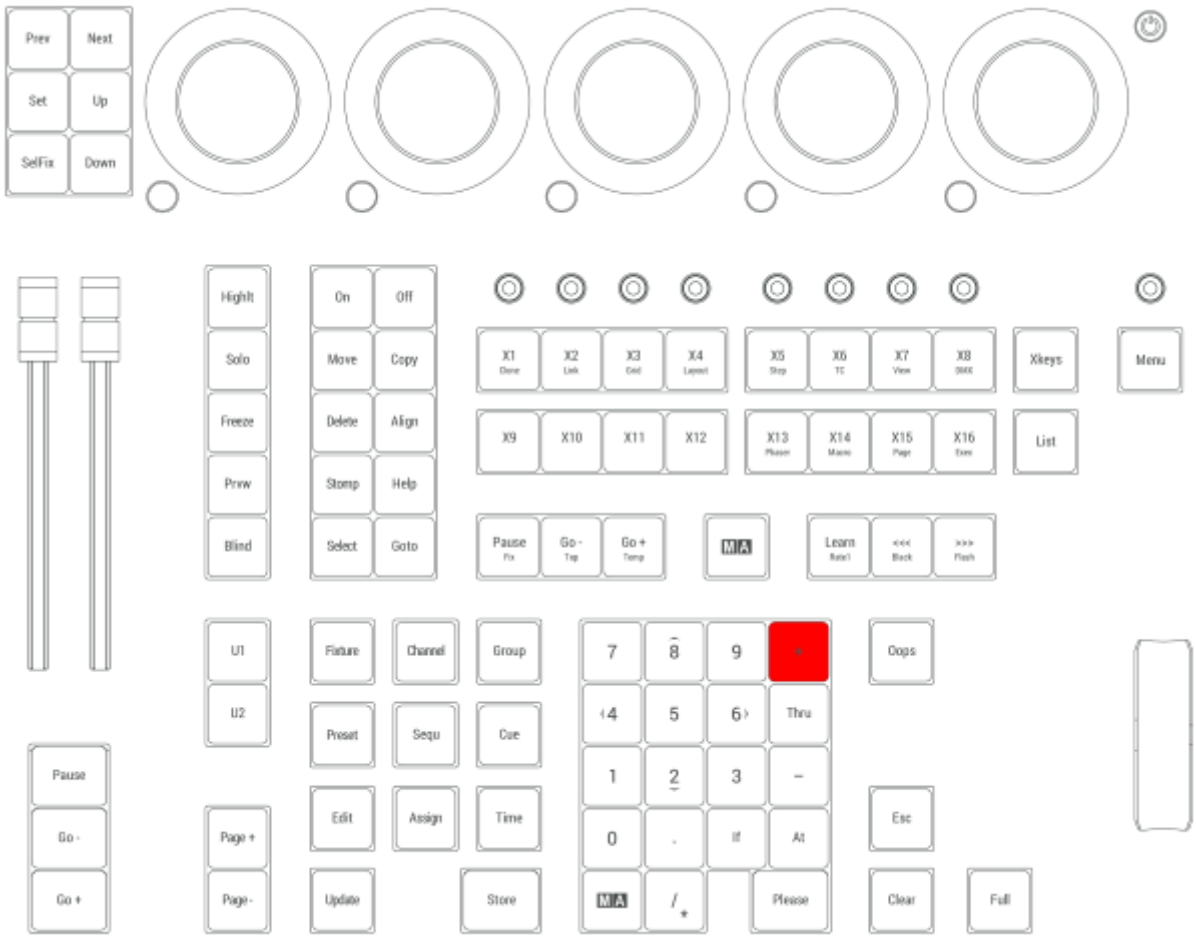
Pressing + + increases the dimmer level of the selected fixtures by 10%.

Location

+ is located in the numeric keys section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.6. / [Slash] | * [Asterisk/Multiply]

Pressing / enters the / into the command line.

For more information about /, see the **/[Slash] keyword**.

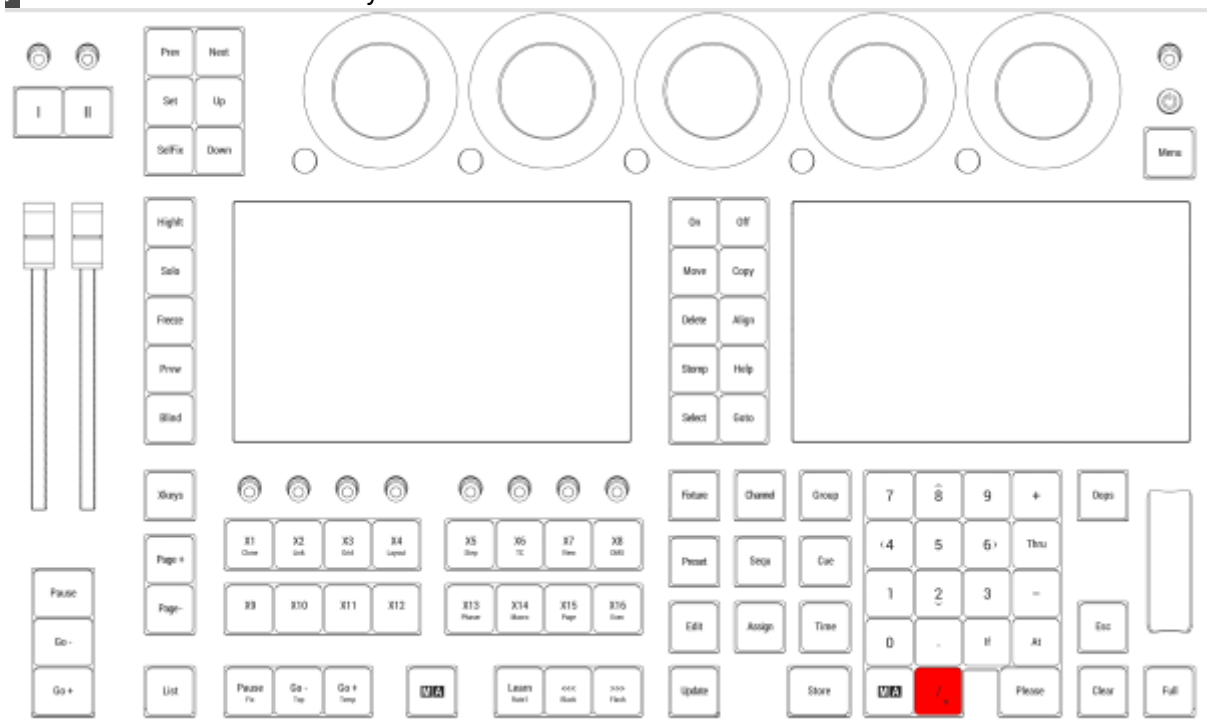
Asterisk

Pressing MA+/ enters the * keyword into the command line.

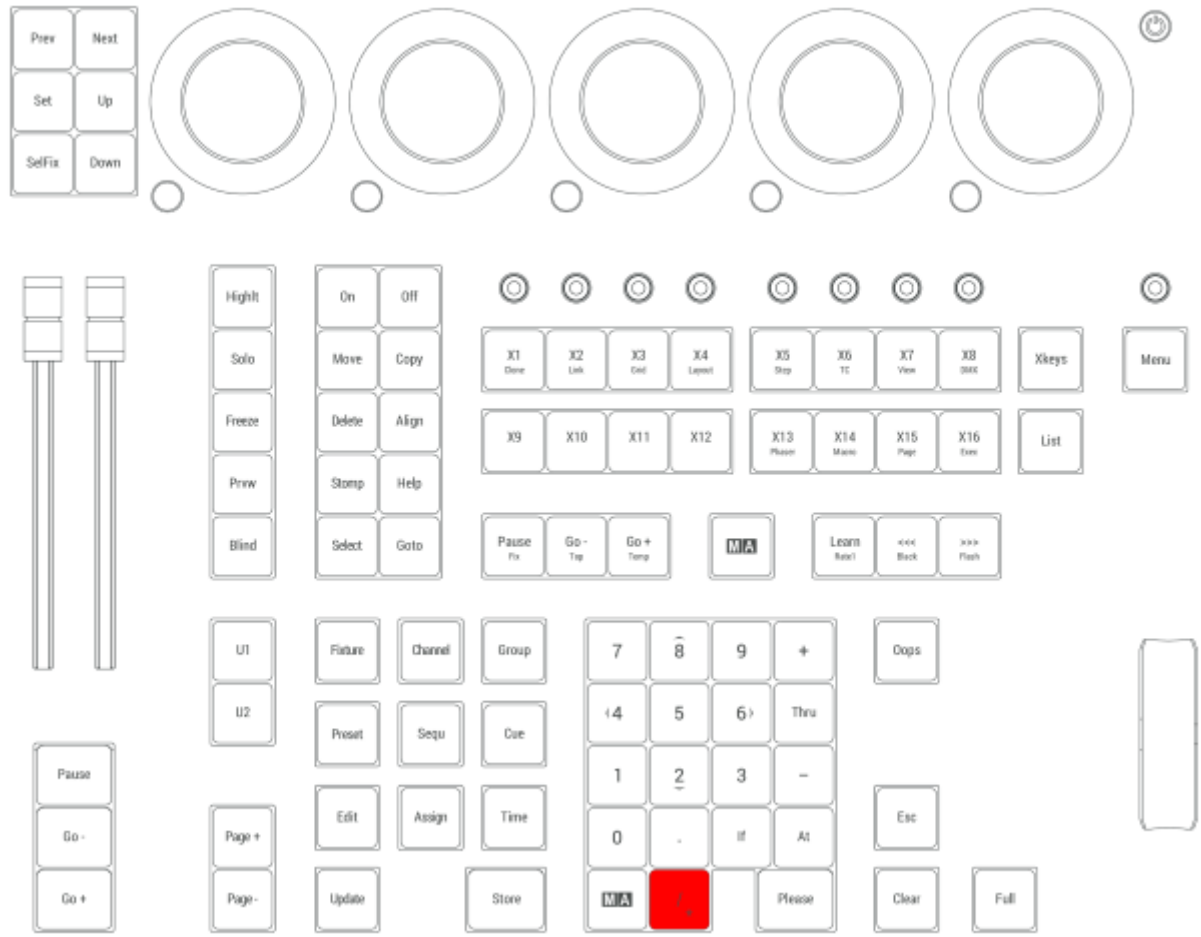
For more information about *, see the *** [Asterisk] keyword**.

Location

/ is located in the numeric keys section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.7. Align

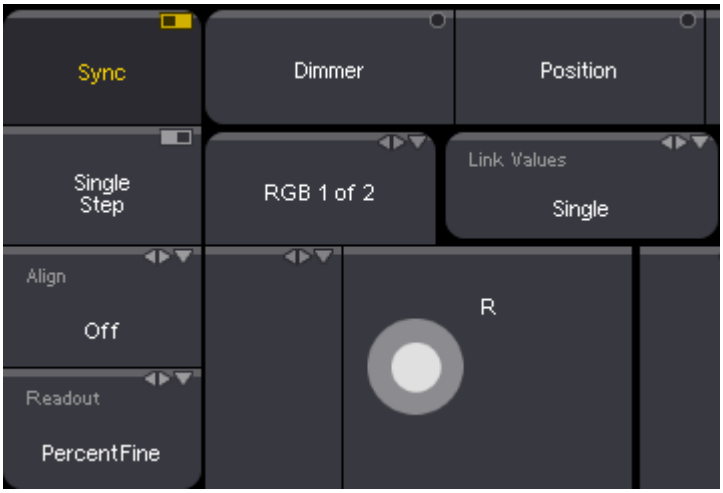
Pressing Align toggles between 6 different Align modes.

By default, the Align mode is disabled.

Pressing MA + Align toggles between four different transition modes.

By default, the transition mode is linear.

The Align mode is displayed in the default encoder bar.



Default encoder bar

The align mode will be disabled when changing the value of a different attribute and the transition mode becomes linear.

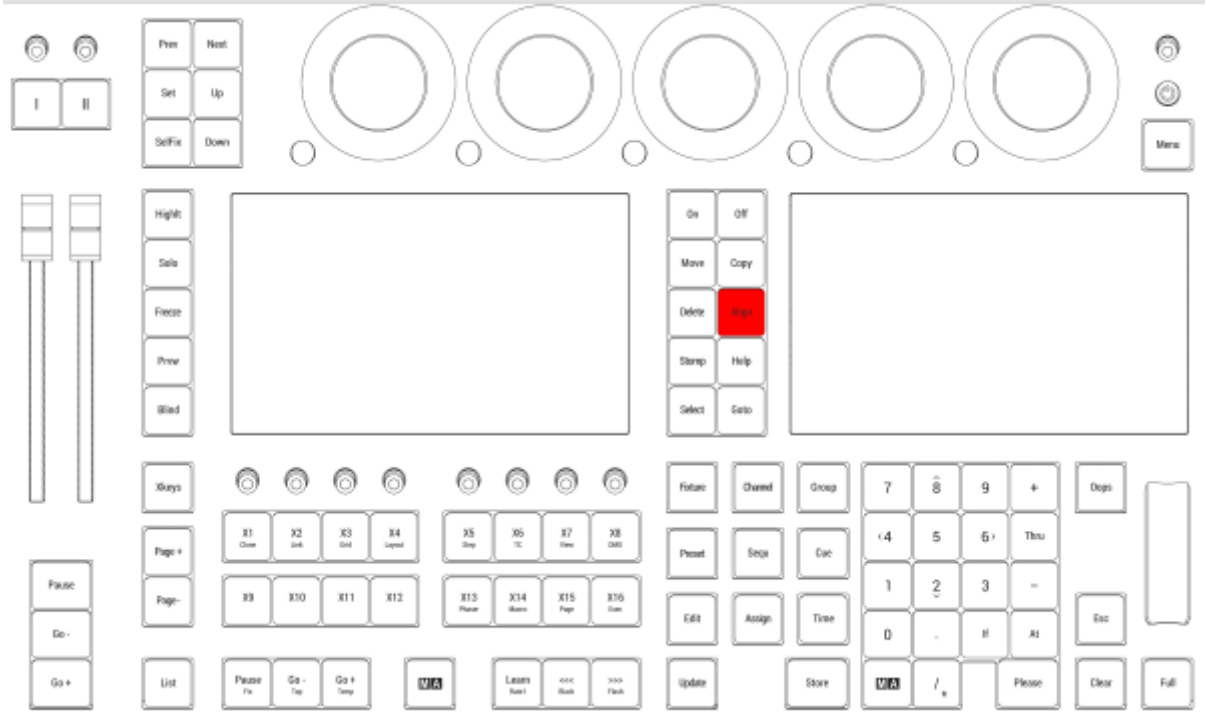
Pressing and holding Align resets the align mode.

Pressing and holding MA + Align resets the transition mode.

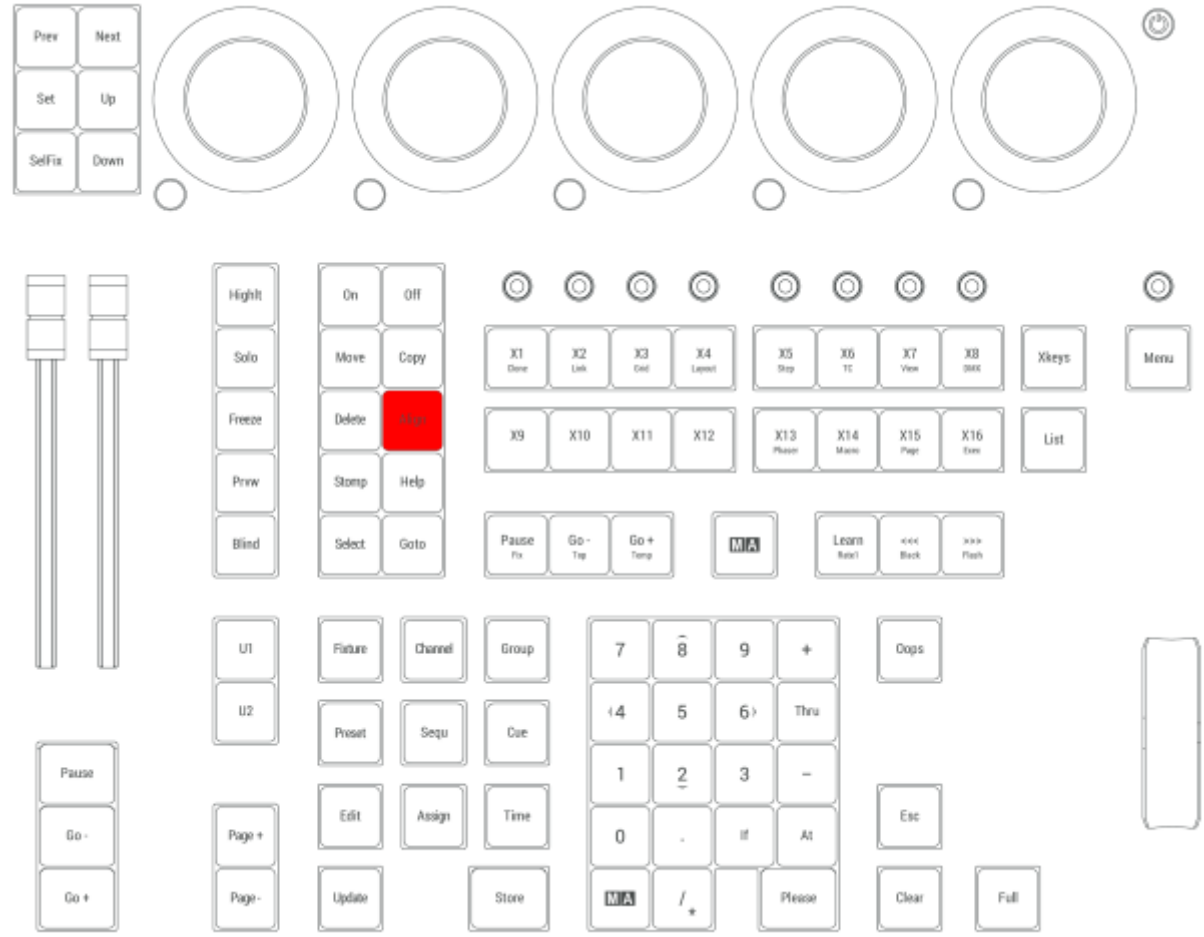
For more information see **Align**.

Location

Align is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.8. Assign

Pressing Assign enters the Assign keyword into the command line.

For more information about Assign, see **Assign keyword**.

Label

Pressing Assign Assign enters the Label keyword into the command line.

For more information about Label, see **Label keyword**.

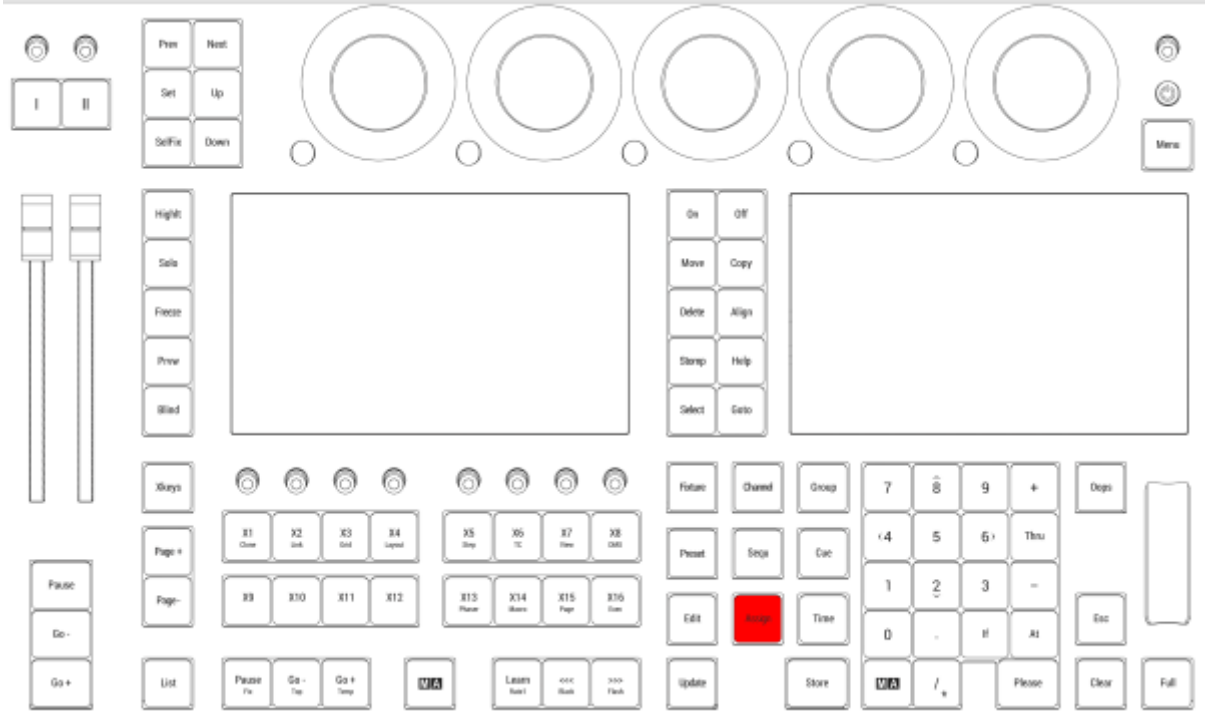
Set

Pressing MA + Assign enters the Set keyword into the command line.

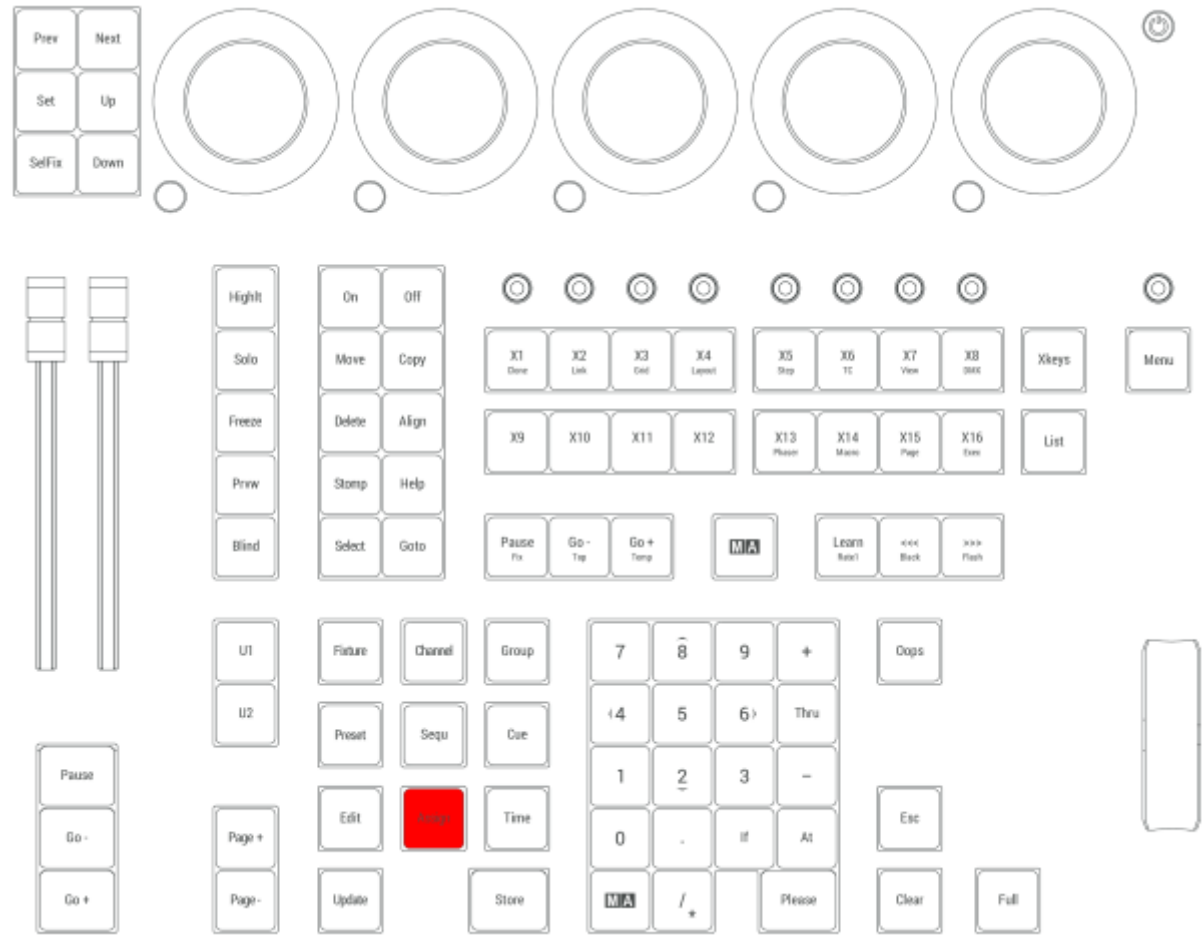
For more information about Set, see **Set keyword**.

Location

Assign is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.9. At

Pressing At enters the At keyword into the command line.

For more information about At, see the **At keyword**.

Normal

Pressing At At executes the **Normal keyword** in the command line in the selected fixtures.

For more information see the **Normal keyword**.

Integrate

Pressing and holding MA + At enters the Integrate keyword into the command line.

For more information about Integrate, see the **Integrate keyword**.

Extract

Pressing and holding MA + At + At enters the Extract keyword into the command line.

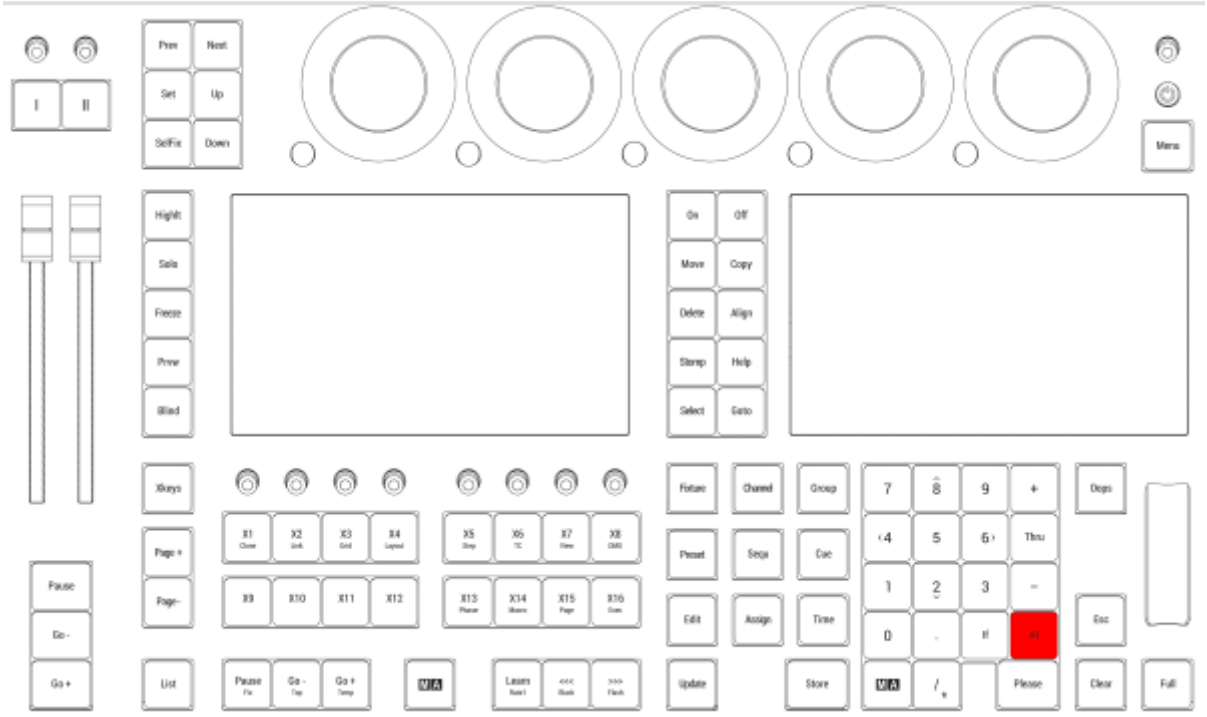
For more information about Extract, see the **Extract keyword**.

At Filter Menu

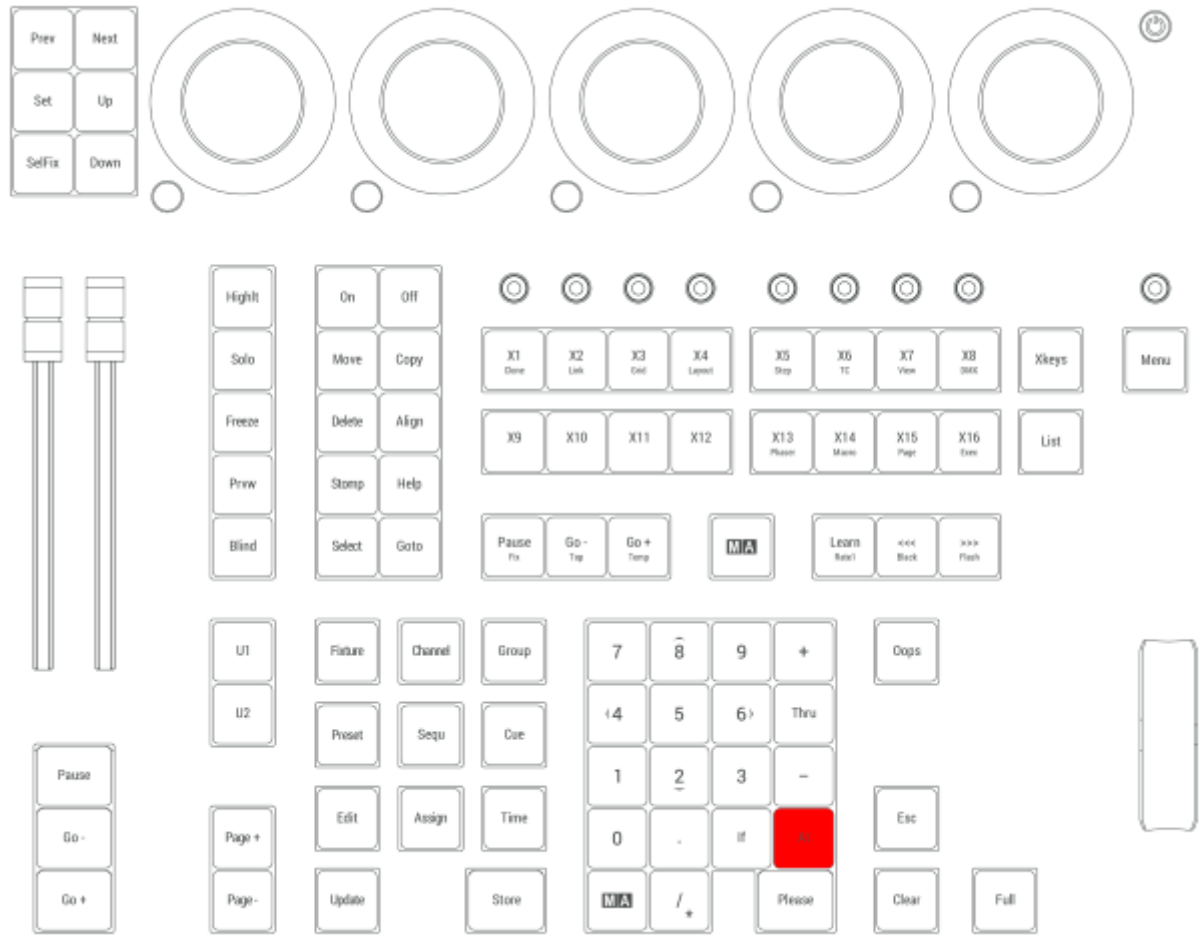
Pressing and holding At opens the At Filter menu.

Location

At is located in the numeric keys section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.10. Blind

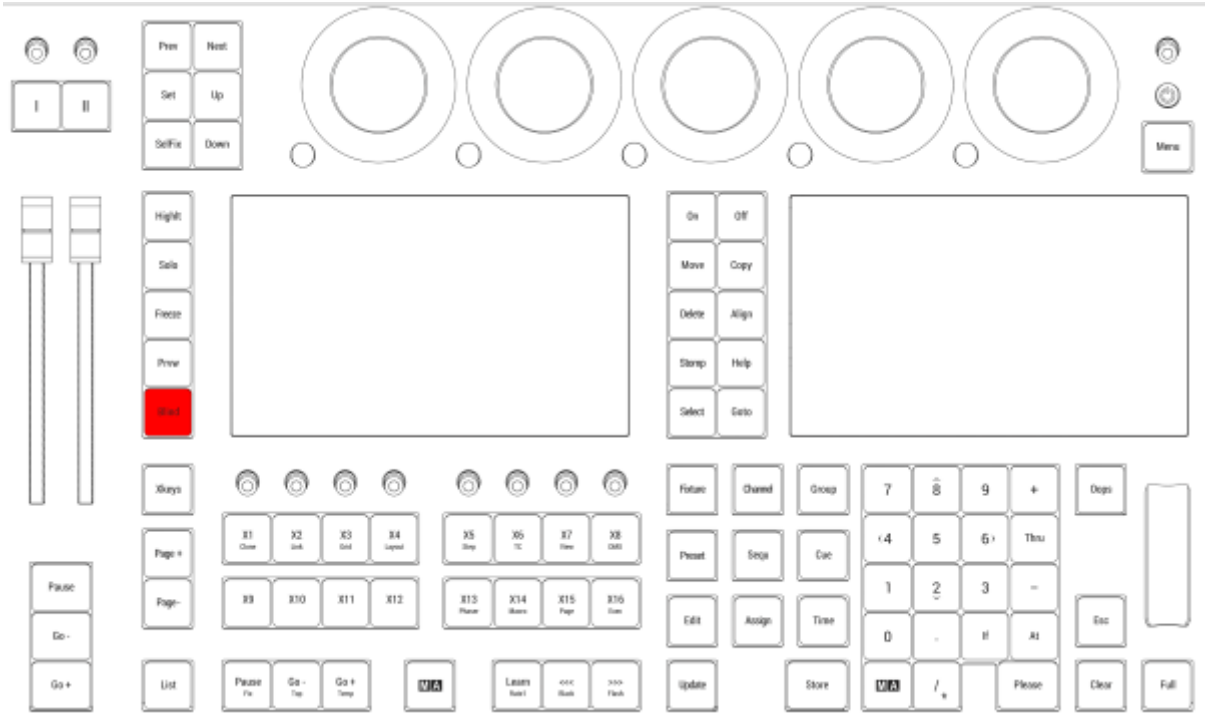
Pressing Blind toggles the Blind function.

```
OK
:   Blind
```

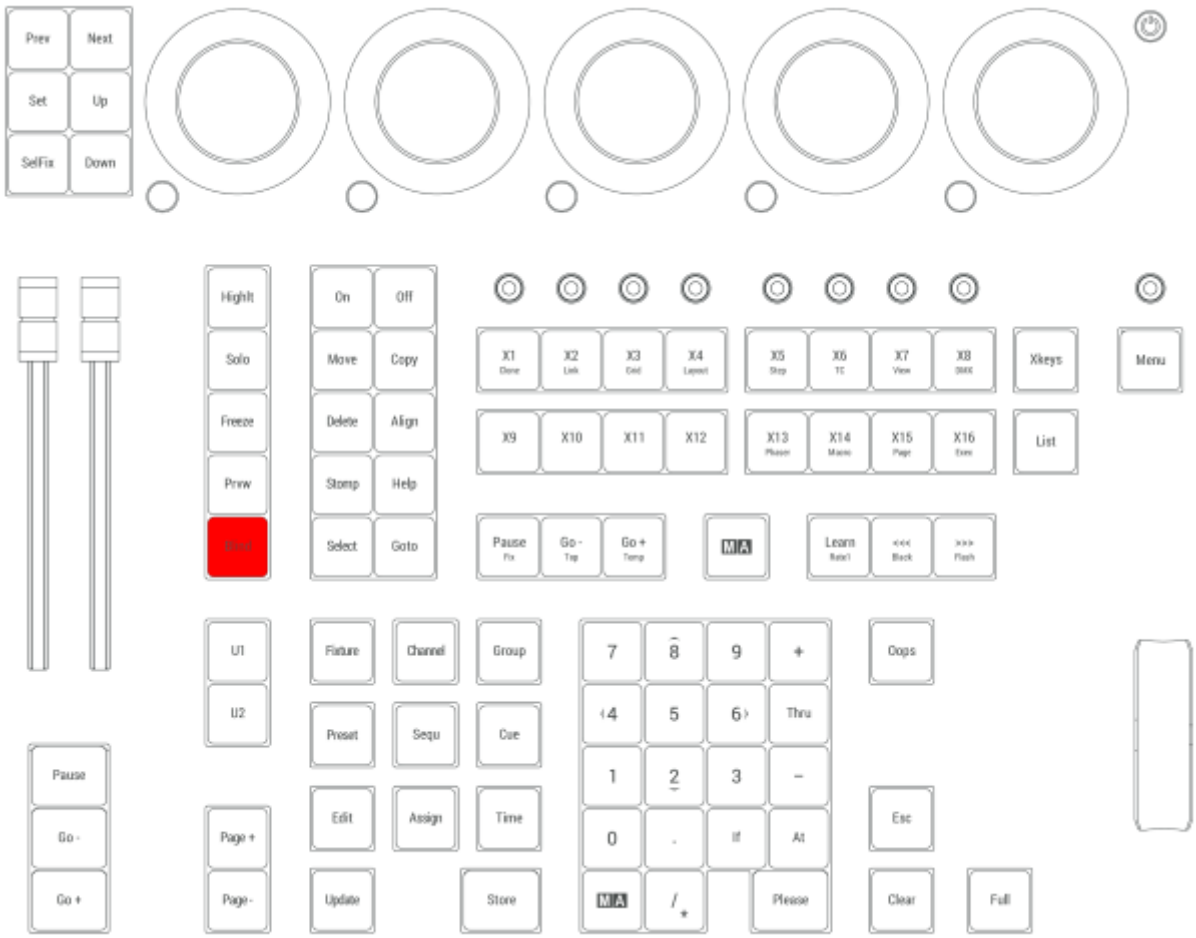
For more information about Blind, see the **Blind keyword**.

Location

Blind is located in the Command Section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.11. Clear

If there are values in the programmer, Clear behaves like this:

- Pressing Clear executes the **ClearSelection keyword**.
- Pressing Clear Clear executes the **ClearActive keyword**.
- Pressing Clear Clear Clear executes the **ClearAll keyword**.

If there are no values in the programmer, pressing Clear deletes unexecuted commands in the command line and executes the ClearActive keyword

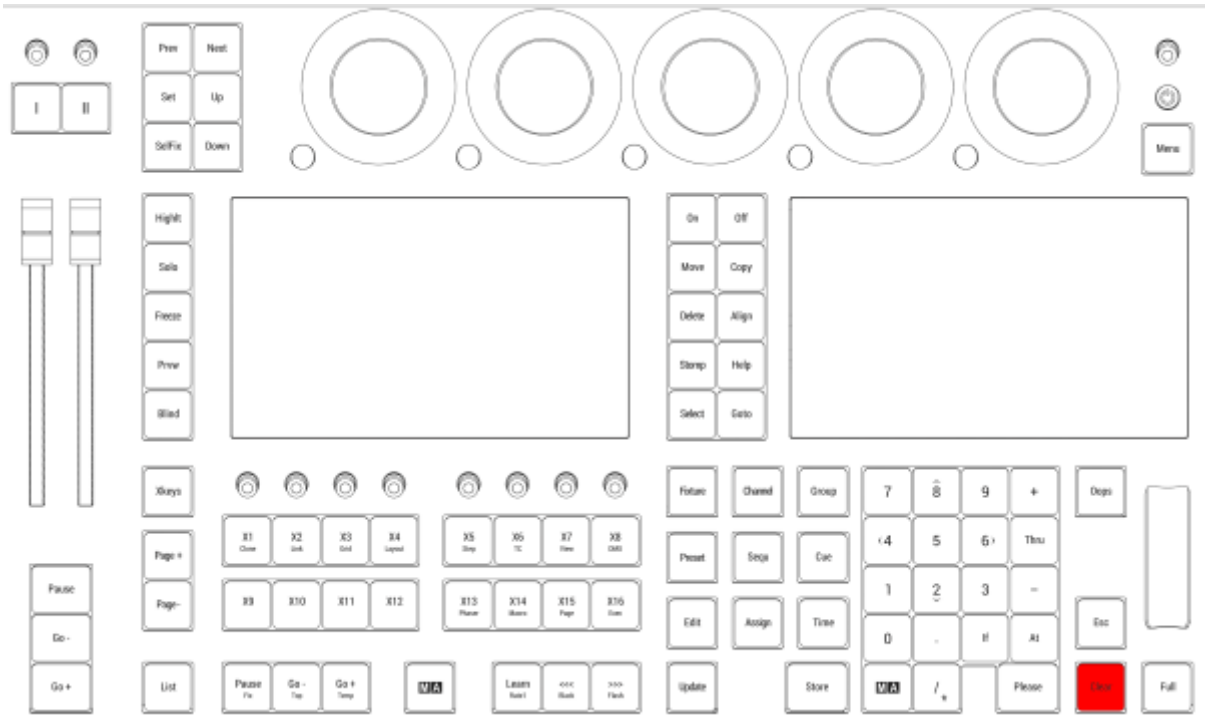
Pressing and holding Clear executes the ClearAll keyword.

Pressing Clear, no matter how often, always displays a Clear in the Command Line Feedback.

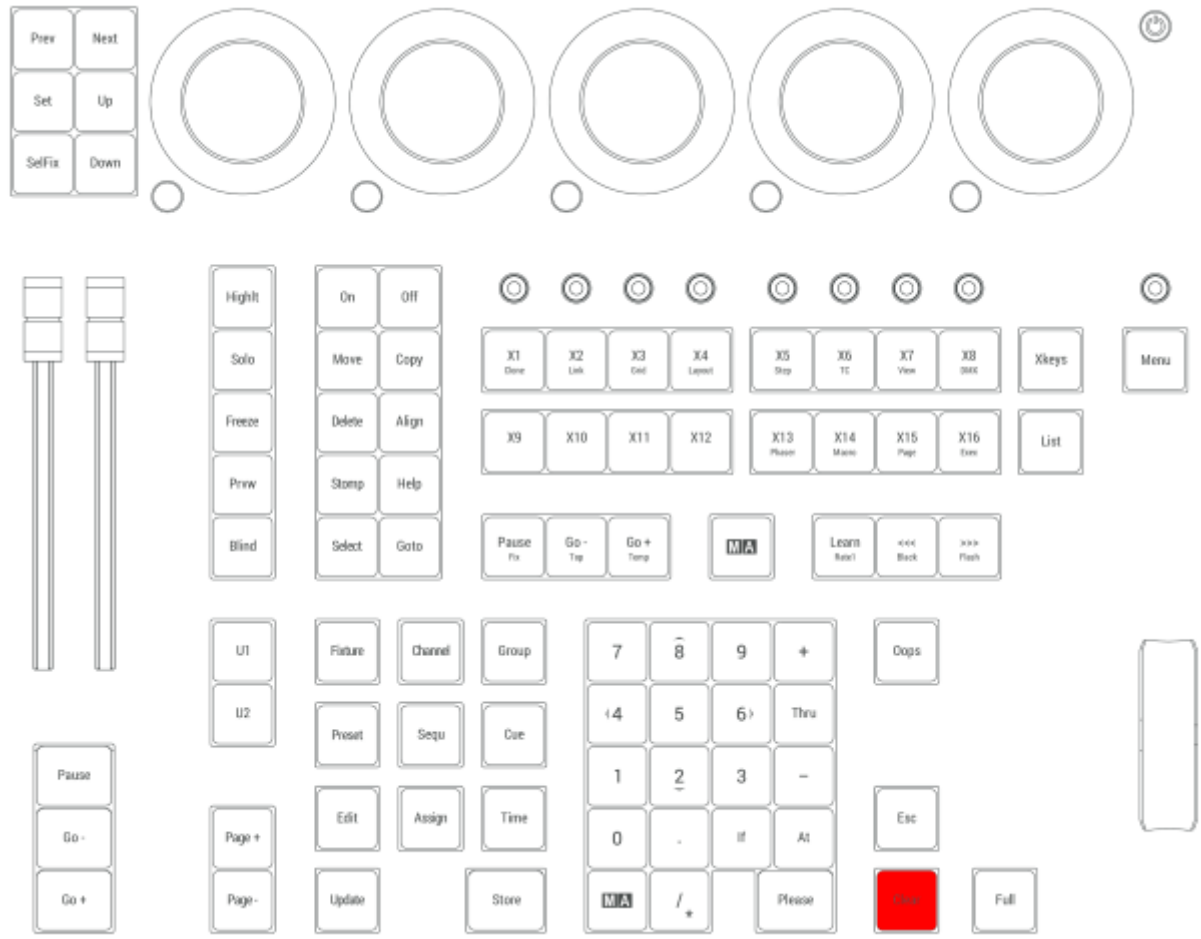
For more information about Clear, see the **Clear keyword**.

Location

Clear is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.12. Channel

Pressing Channel enters the Channel keyword in the command line.

For more information, see the **Channel keyword**.

Universal

Pressing Channel Channel enters the Universal keyword in the command line.

For more information, see the **Universal keyword**.

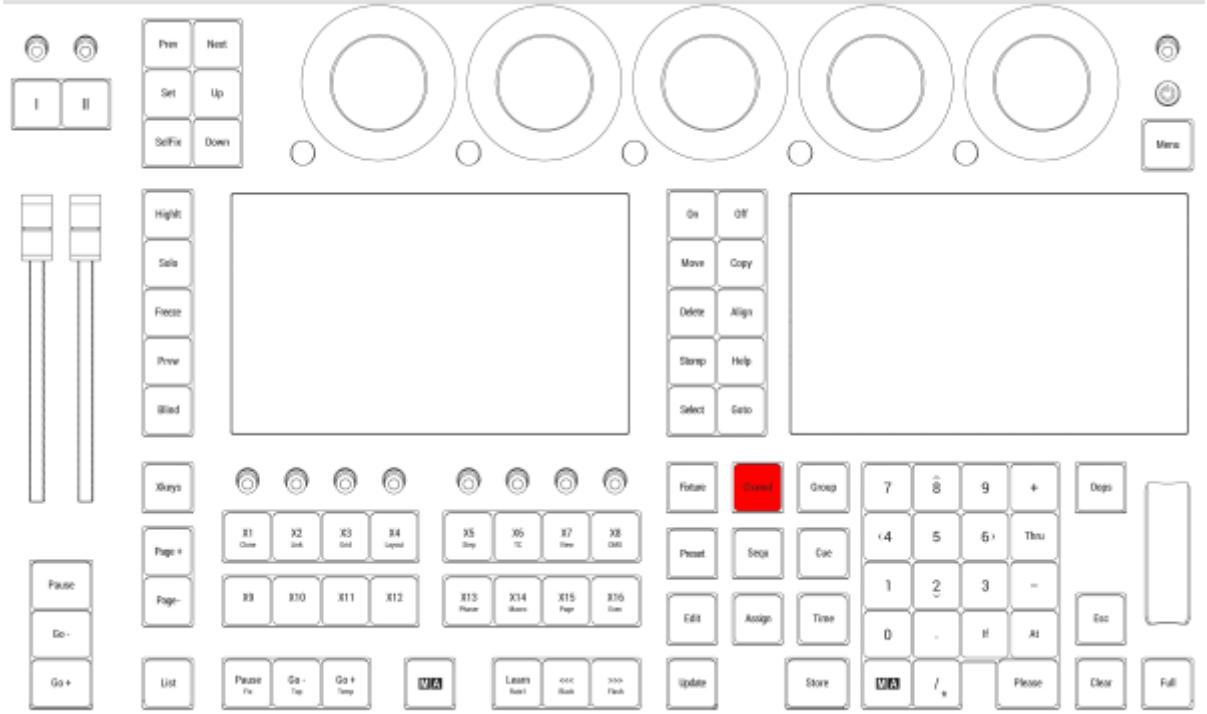
ID Types

Pressing Channel repeatedly switches through Channel, Universal, and ID types in use. ID types where no patched fixtures are assigned are not showing in the command line.

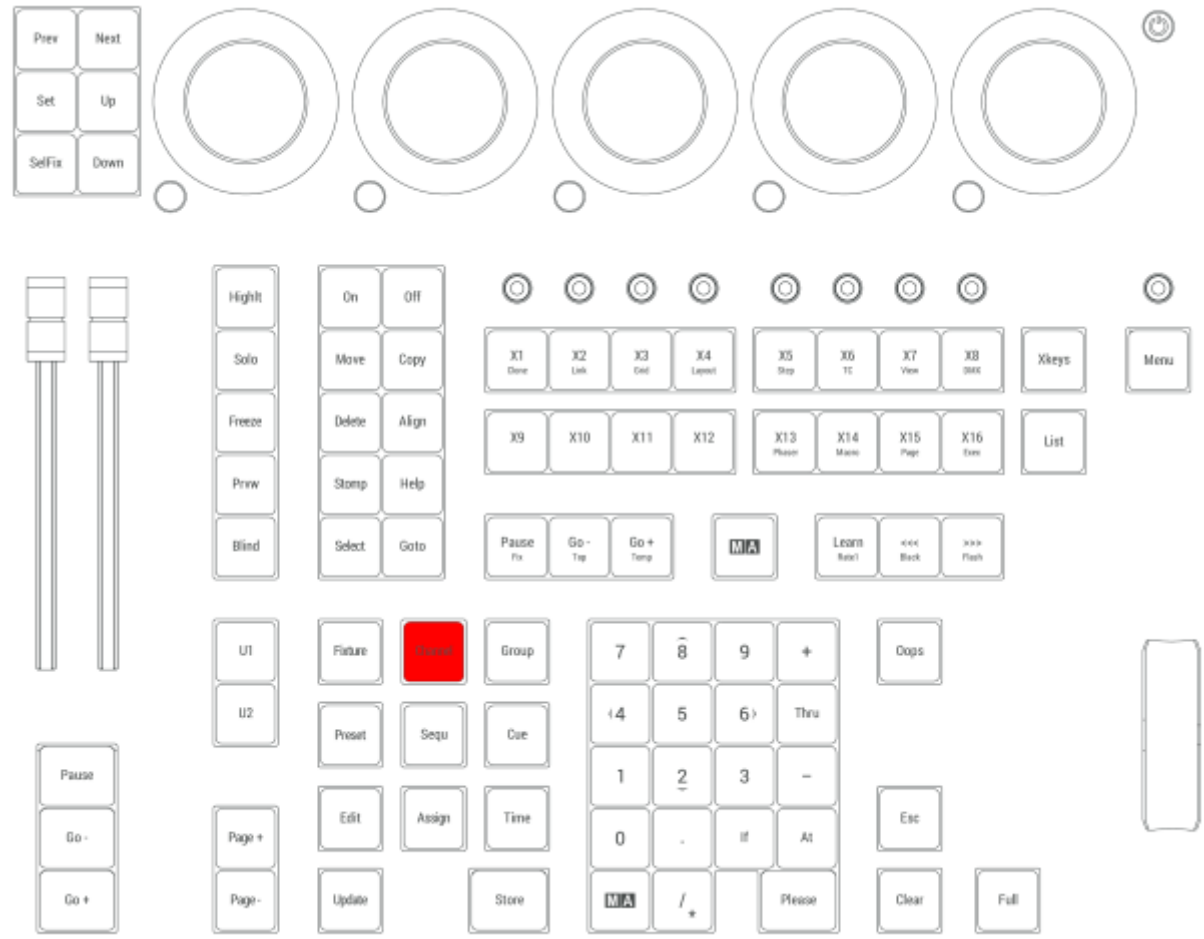
For more information, see the **Add Fixtures to the Show**.

Location

Channel is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.13. Copy

Pressing Copy enters the copy keyword into the command line.

For more information about Copy, see the **Copy keyword**.

Paste

Pressing Copy Copy enters the Paste keyword into the command line.

For more information about Paste, see the **Paste keyword**.

Insert

Pressing Copy Copy Copy enters the Insert keyword into the command line.

For more information about Insert, see the **Insert keyword**.

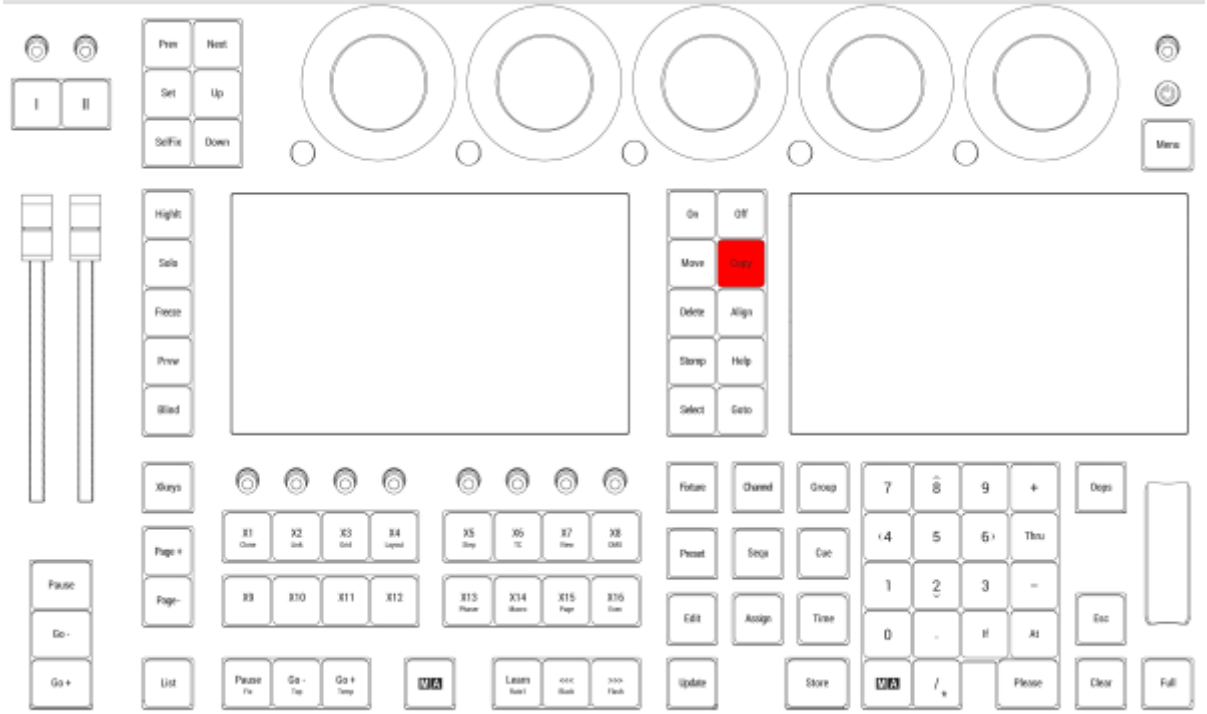
Cut

Pressing MA Copy enters the Cut keyword into the command line.

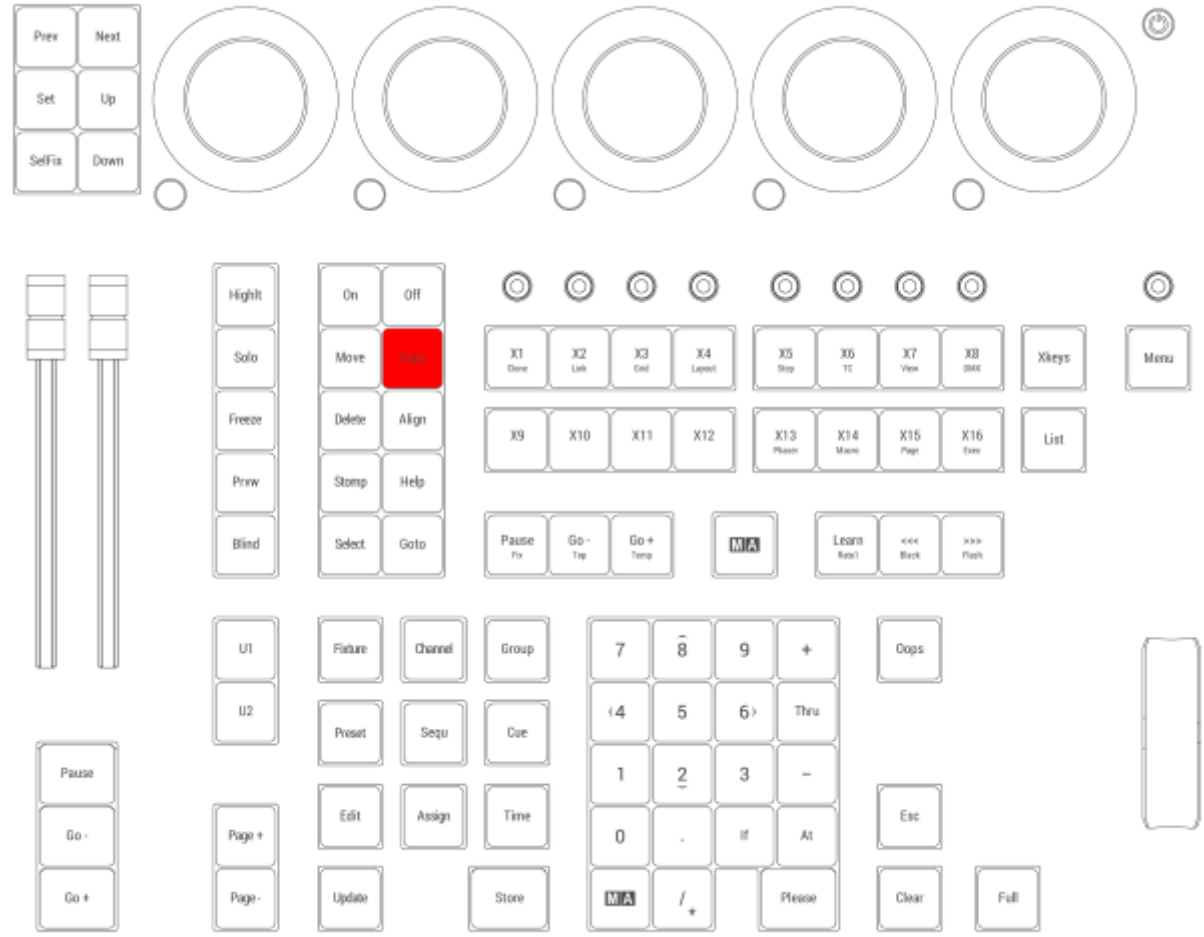
For more information about Cut, see the **Cut keyword**.

Location

Copy is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.14. Ctrl

Multi Select

To select multiple cells in a sheet, hold Ctrl and tap or click each cell.

Clean Start

To perform a clean start, hold Ctrl when the mode selection pop-up appears during the boot or reboot process and tap the desired mode. For more information about clean start, see **Clean Start**.

Location

Ctrl is located in two places on the bottom row of the alpha-numeric keyboard.

Location on grandMA3 full-size and grandMA3 light console

1.3.15.15. Cue

Pressing Cue enters the Cue keyword into the command line.

For more information about Cue, see the **Cue keyword**.

Part

Pressing Cue Cue enters the Part keyword into the command line.

For more information about Part, see the **Part keyword**.

	Important:
	If there is already **Cue** in the command line right in front of the cursor, pressing Cue once will display **Part** in the command line.

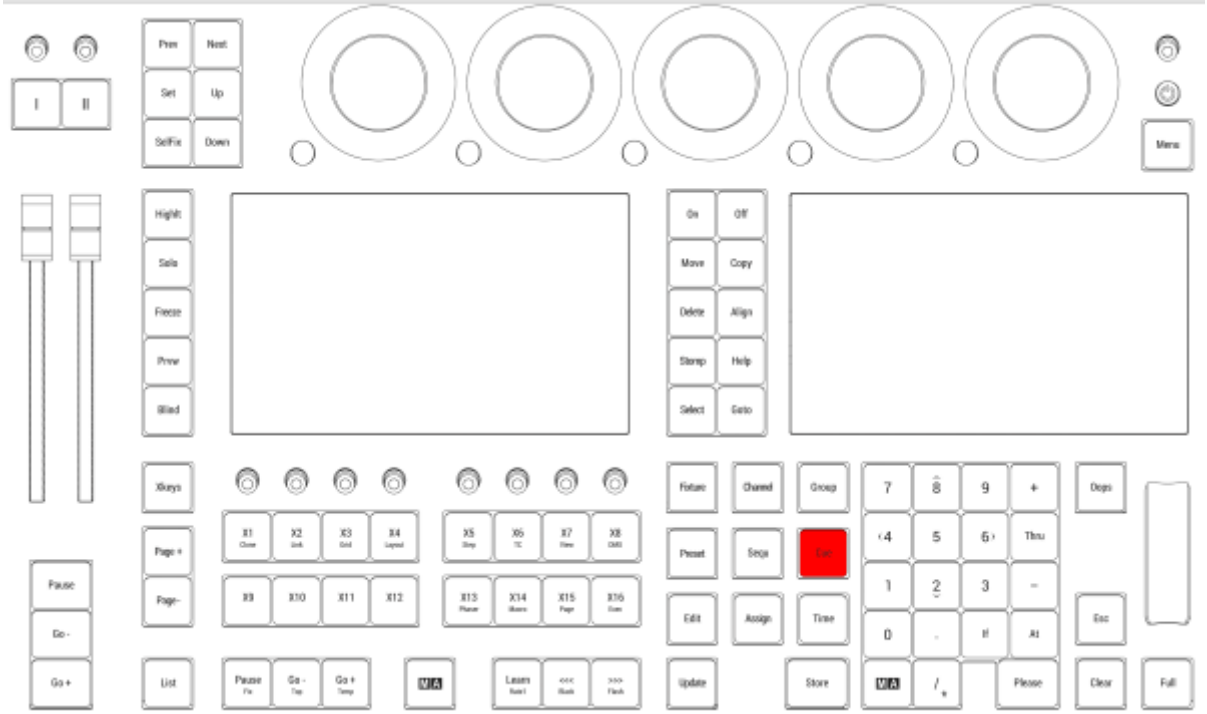
Programmer

Pressing MA +Cue enters the Programmer keyword into the command line.

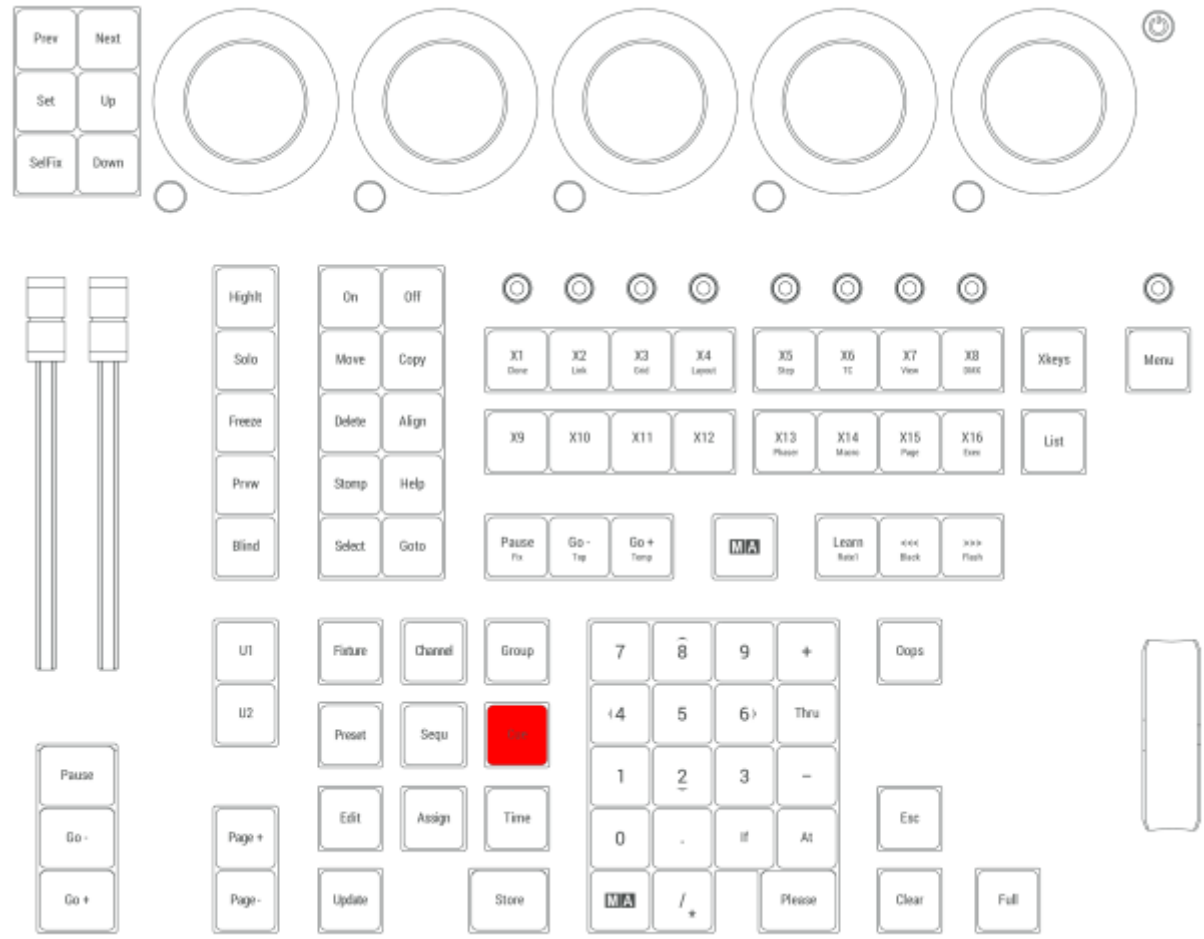
For more information about Programmer, see the **Programmer keyword**.

Location

Cue is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.16. Delete

Pressing Delete enters the Delete keyword into the command line.

For more information about Delete, see the **Delete keyword**.

Remove

Pressing Delete Delete enters the Remove keyword into the command line.

For more information about Remove, see the **Remove keyword**.

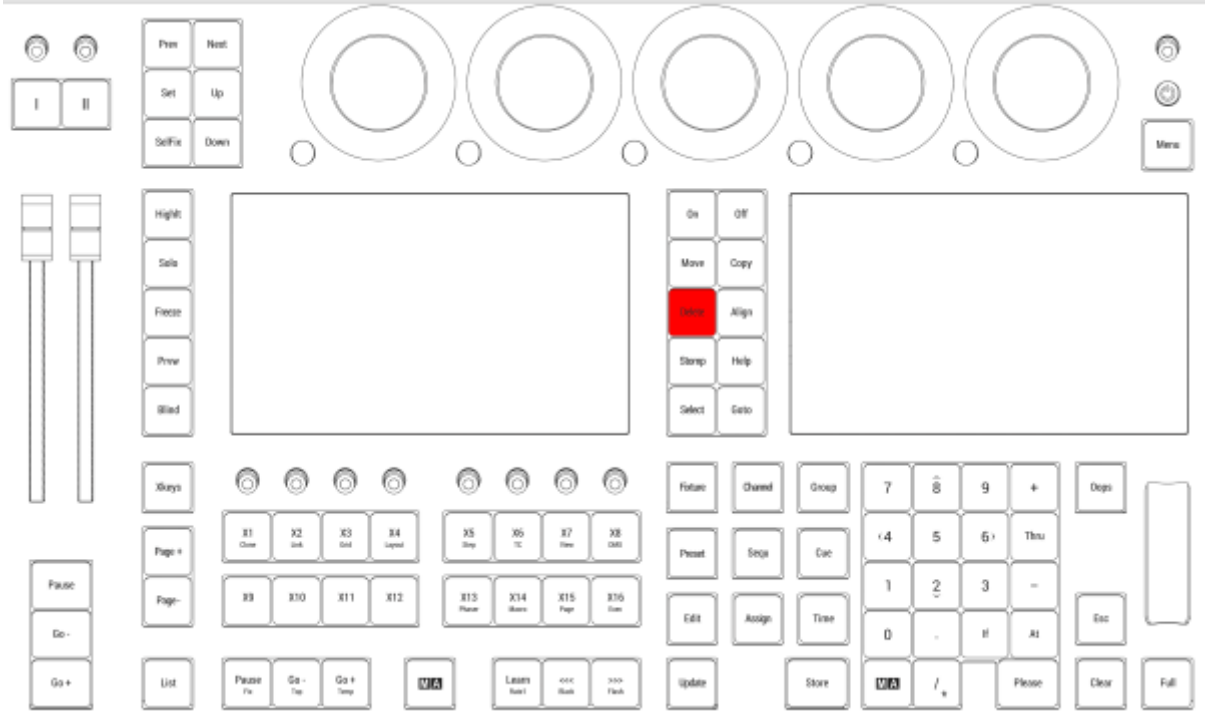
Release

Pressing Delete Delete Delete enters the Release keyword into the command line.

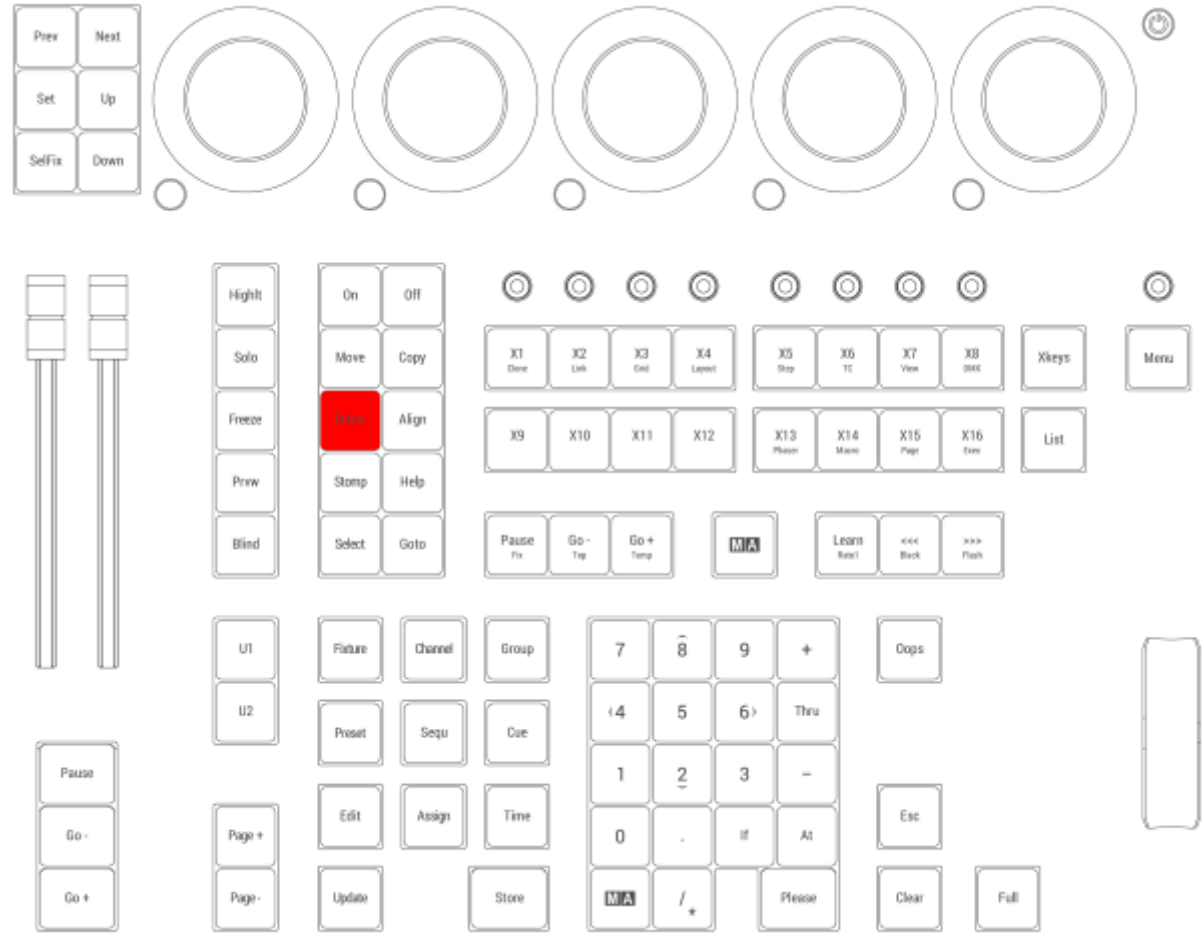
For more information about Release, see the **Release keyword**.

Location

Delete is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



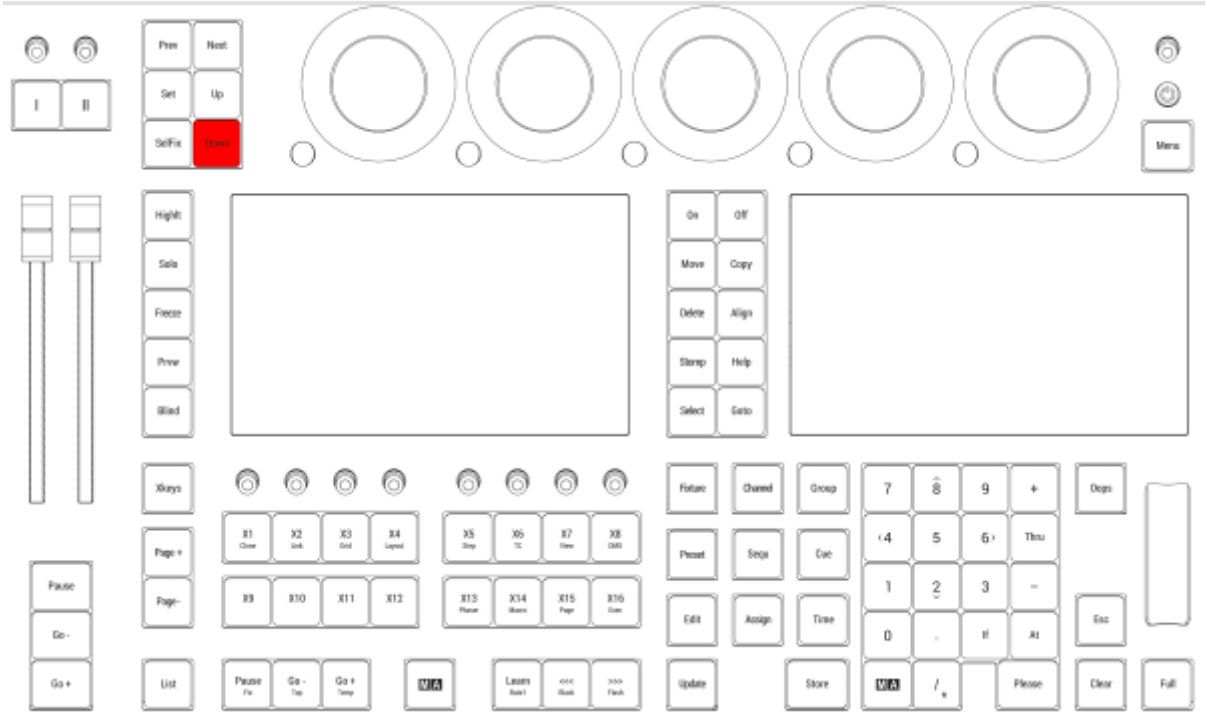
Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.17. Down

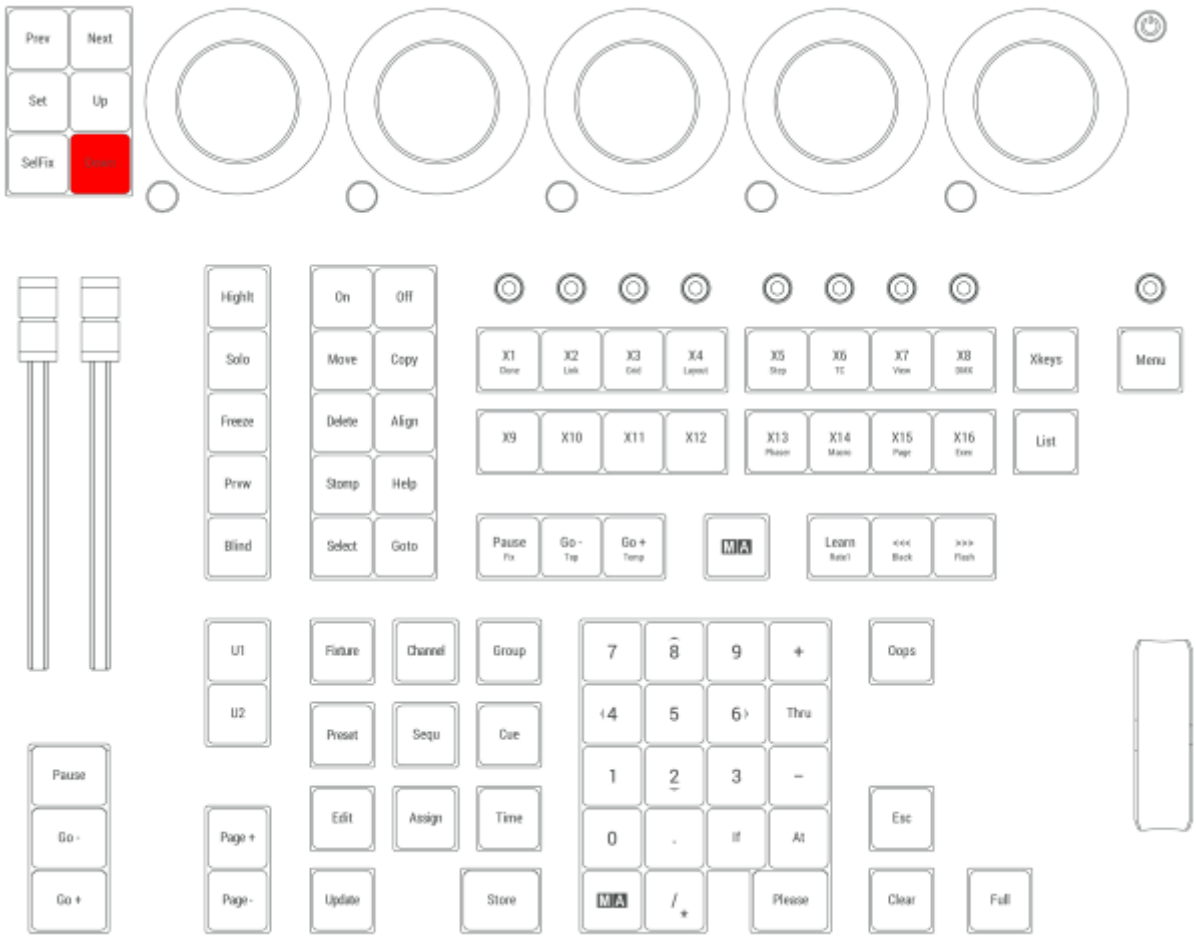
Pressing Down acesses one level deeper in the fixture structure.

Location

Down is located in the command section on the left side of the five dual encoders.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.18. Edit

Pressing Edit enters the Edit keyword into the command line.

For more information about Edit, see the **Edit keyword**.

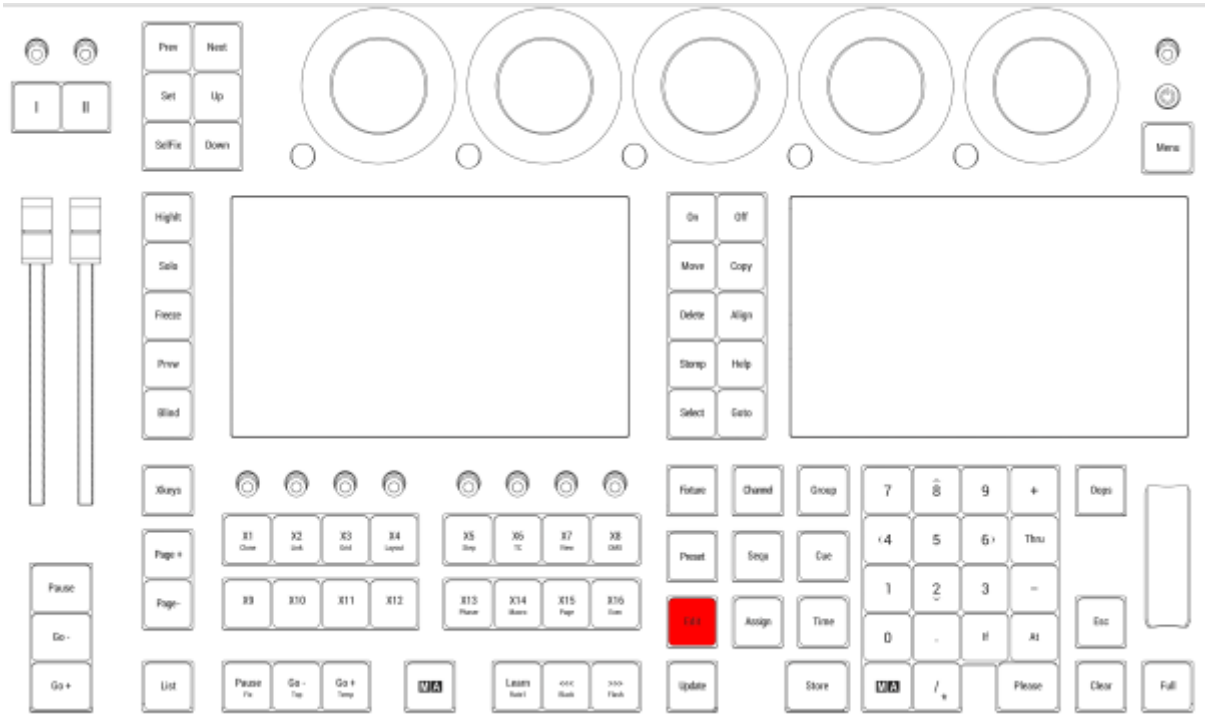
EditSetting

Pressing Edit Edit enters the EditSetting keyword into the command line.

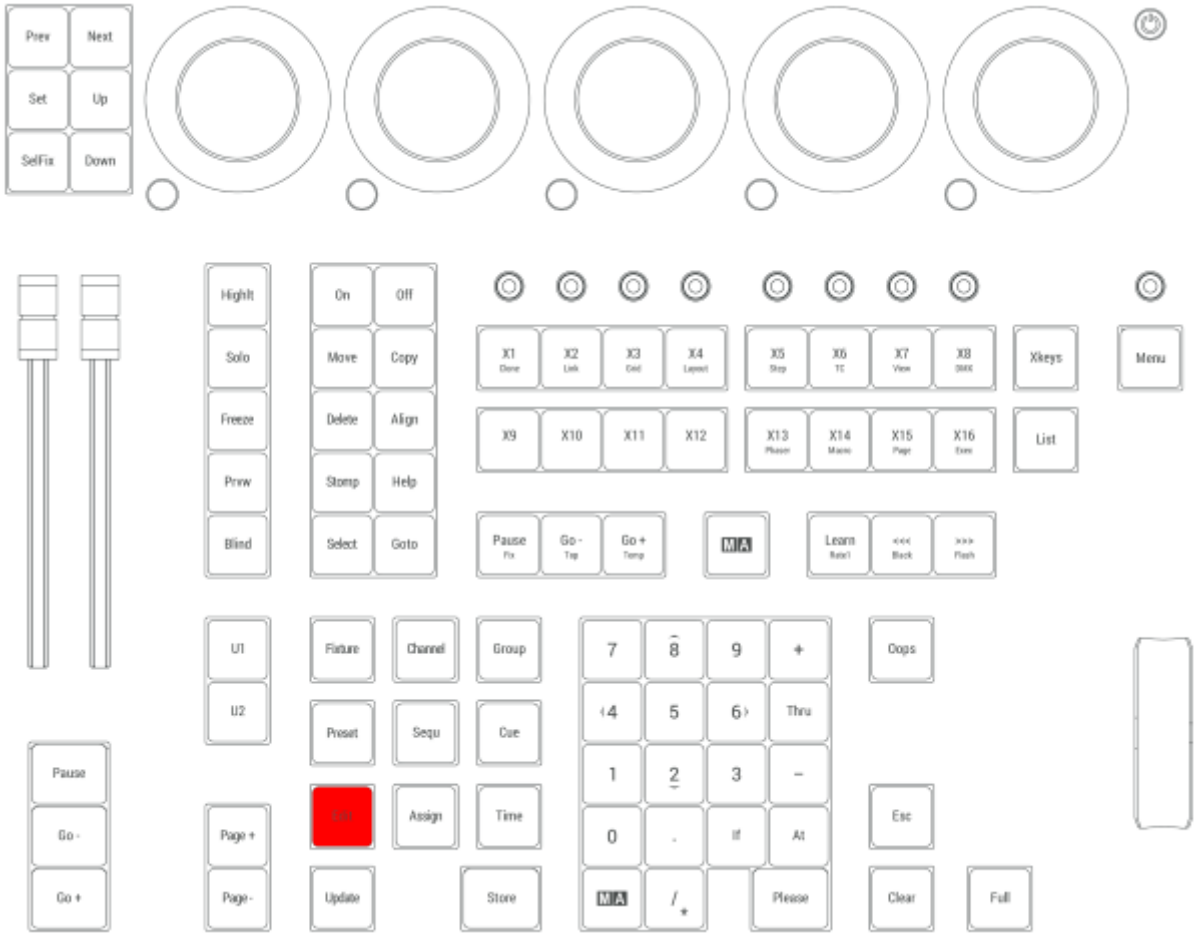
For more information about EditSetting, see the **EditSetting keyword**.

Location

Edit is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



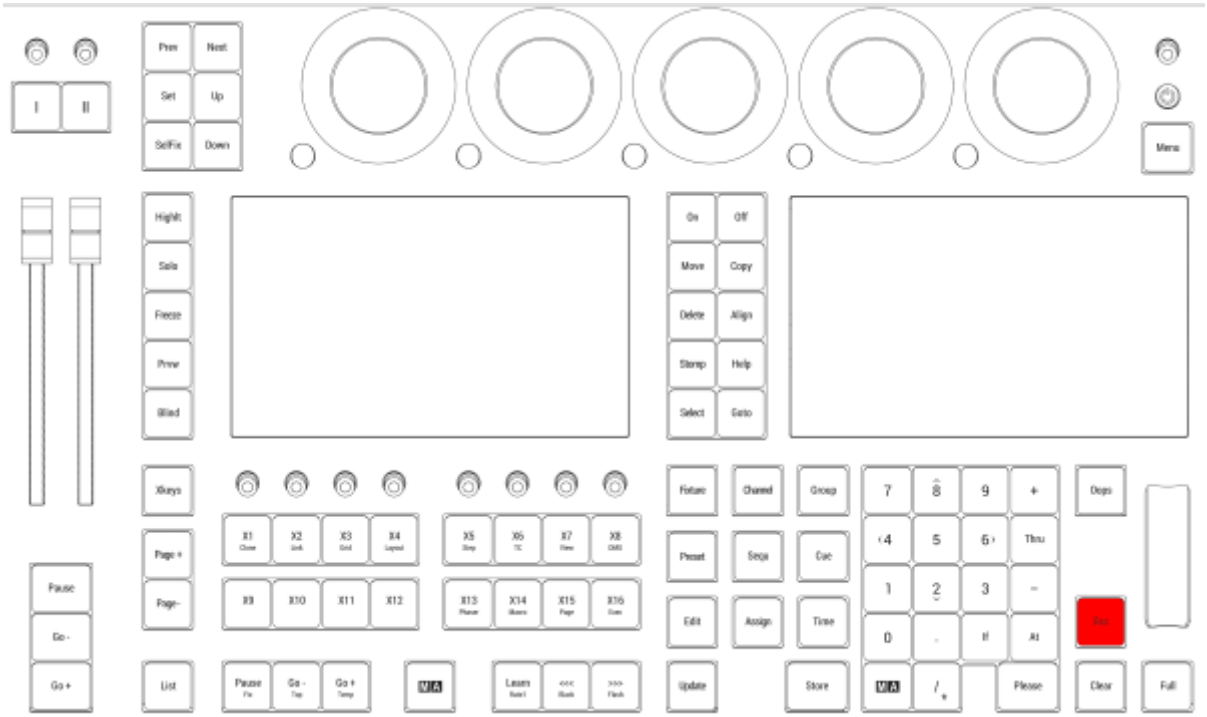
Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.19. Esc

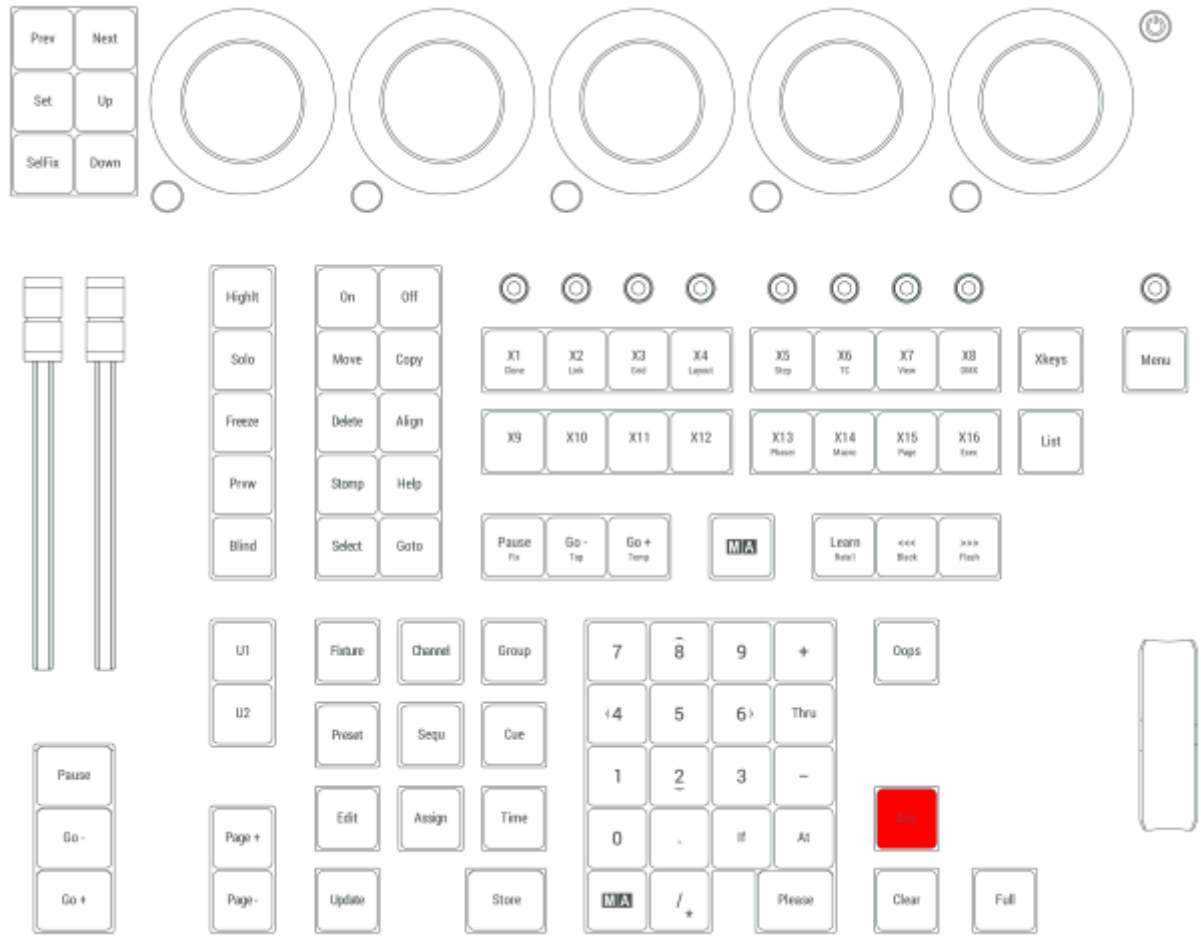
Pressing Esc deletes commands that were not executed in the command line and closes pop-ups.

Location

Esc is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.20. Fixture

Pressing Fixture enters the Fixture keyword into the command line.

For more information about Fixture, see the **Fixture keyword**.

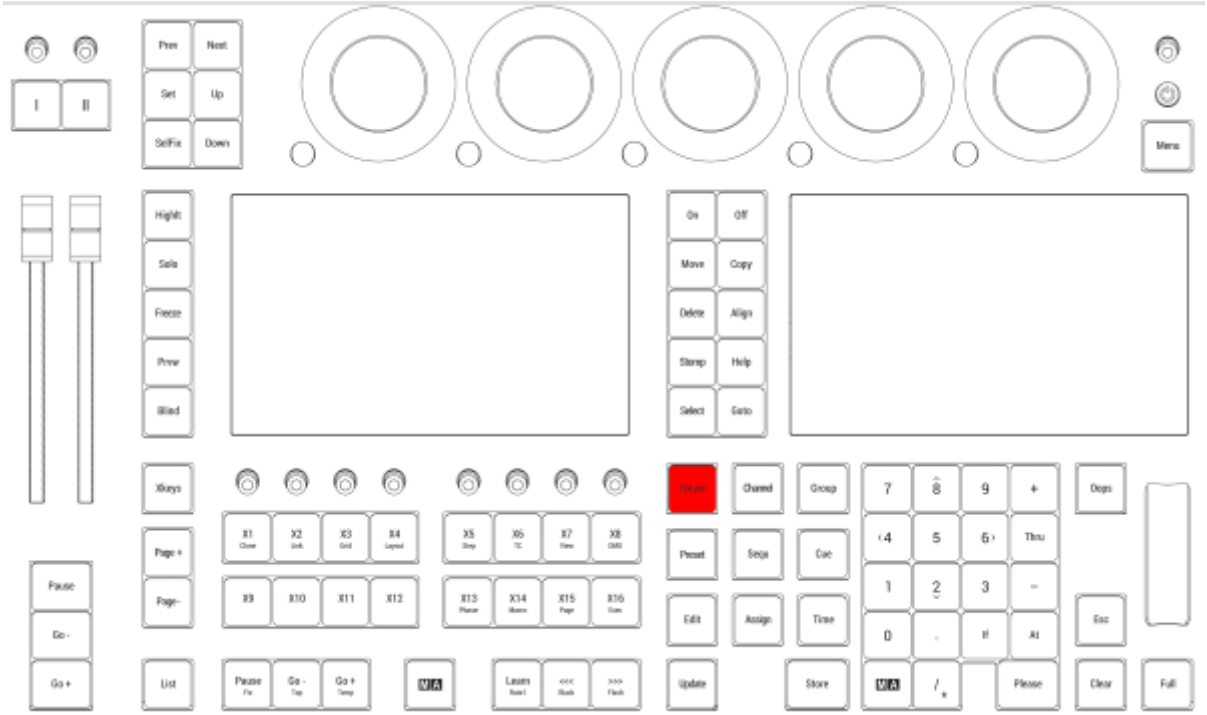
Selection

Pressing Fixture Fixture enters the Selection keyword into the command line.

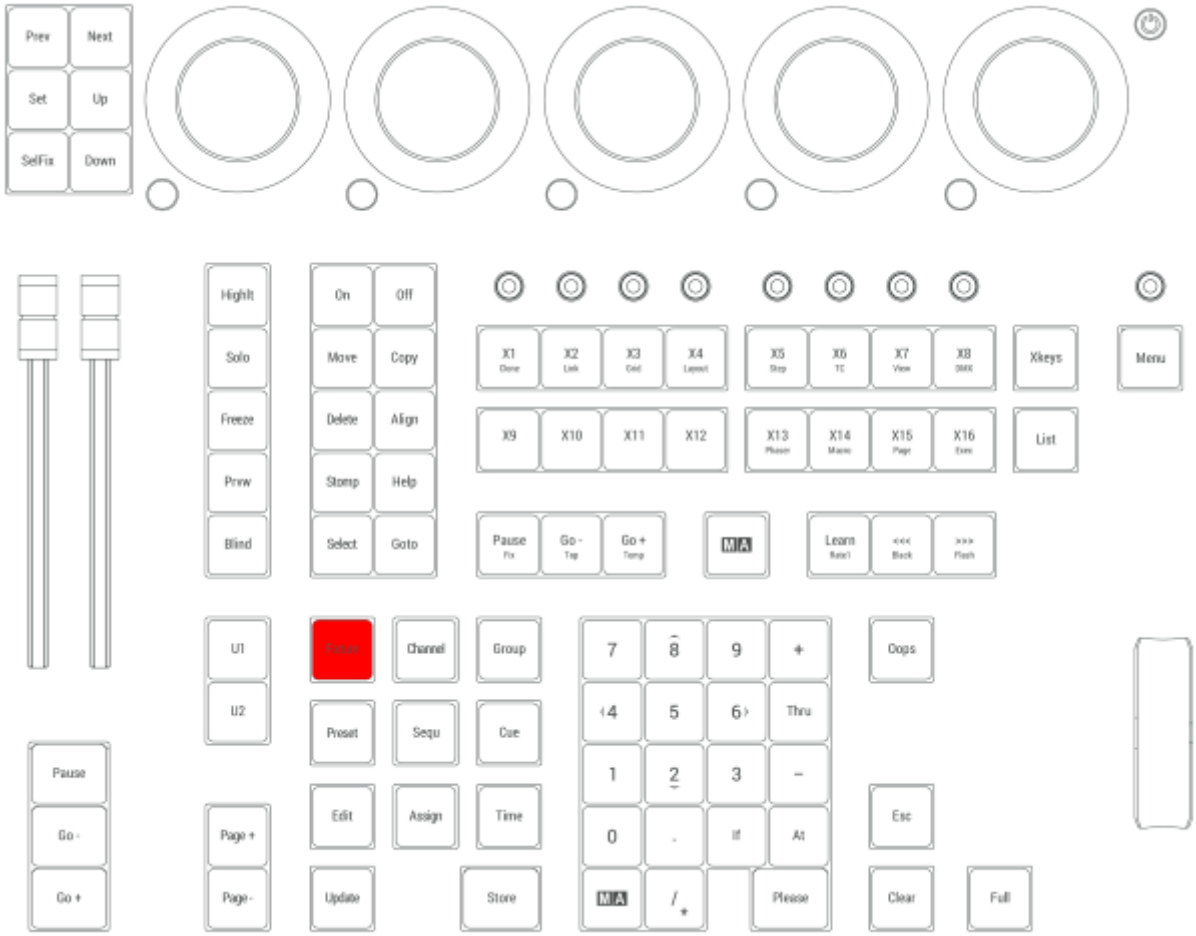
For more information about Selection, see the **Selection keyword**.

Location

Fixture is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.21. Freeze

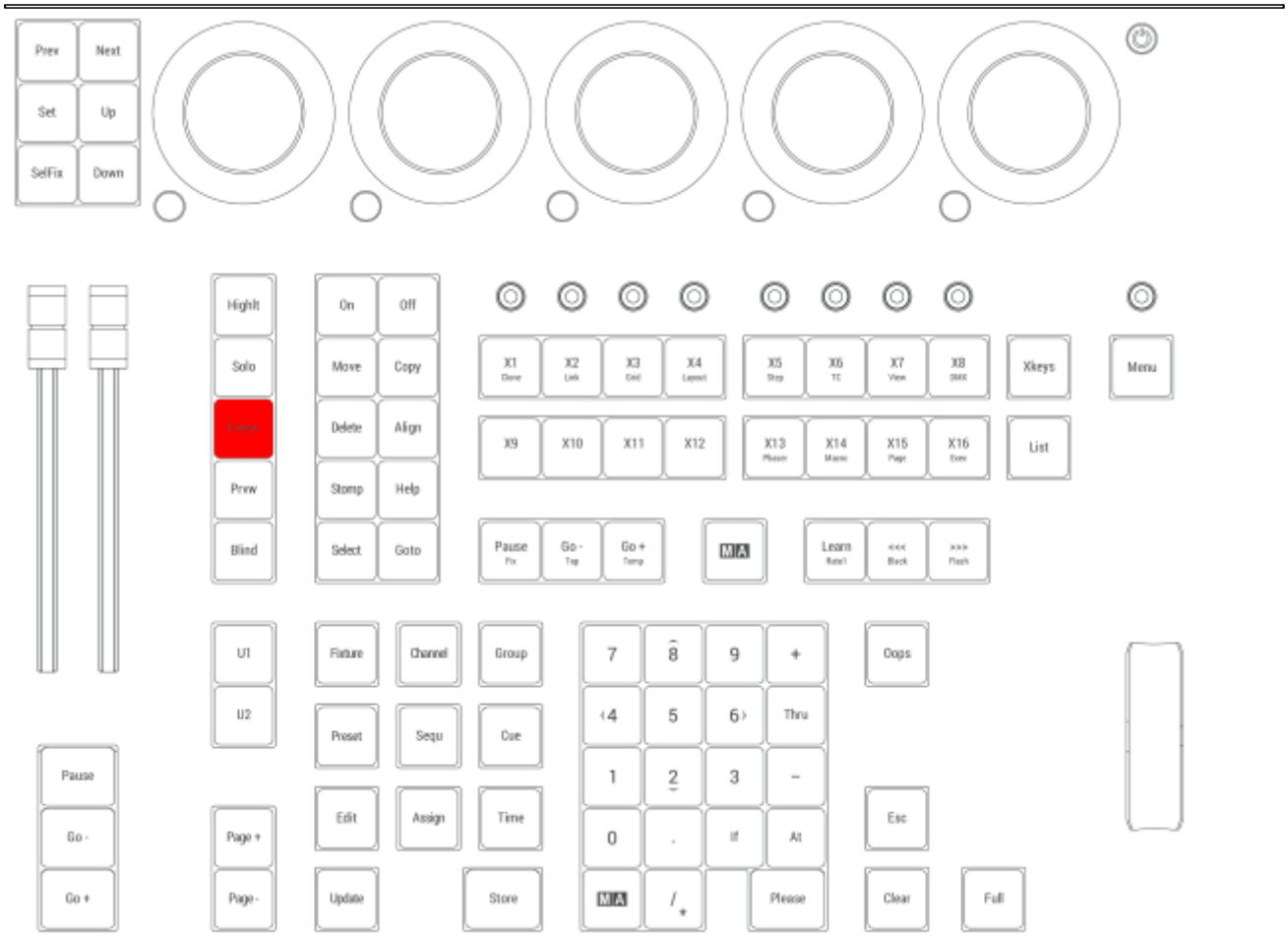
Pressing Freeze toggles the Freeze function.

For more information about Freeze, see the **Freeze keyword**.

Location

Freeze is located in the command section.

Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.22. Full

Pressing Full enters the Full keyword into the command line.

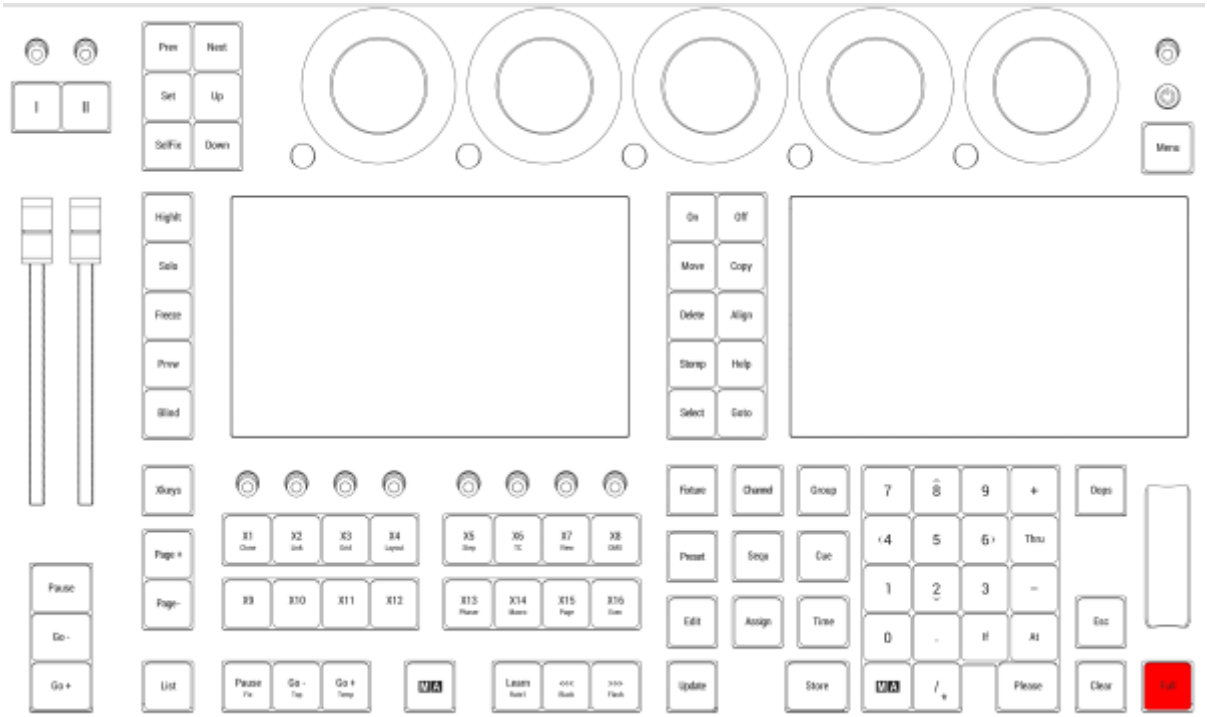
Full sets the dimmer values to 100% in the selected fixtures.

```
MA  User name[Fixture]>Full
```

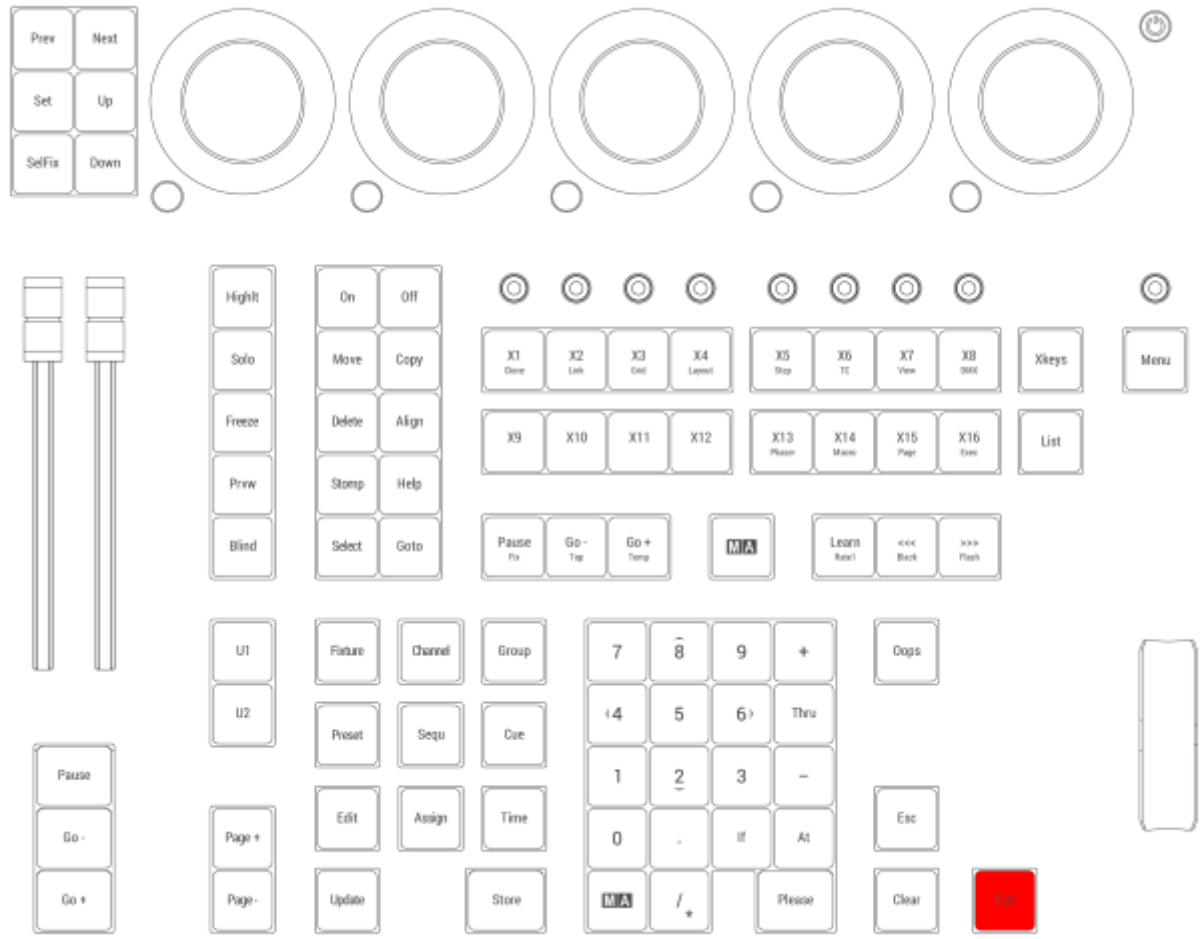
For more information about Full, see the **Full keyword**.

Location

Full is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.23. Go+ | Temp

Pressing Go+ enters the Go+ keyword into the command line.

For more information about Go+, see the **Go+ keyword**.

Temp

Pressing and holding MA + Go+ enters the Temp keyword into the command line.

For more information about Temp, see the **Temp keyword**.

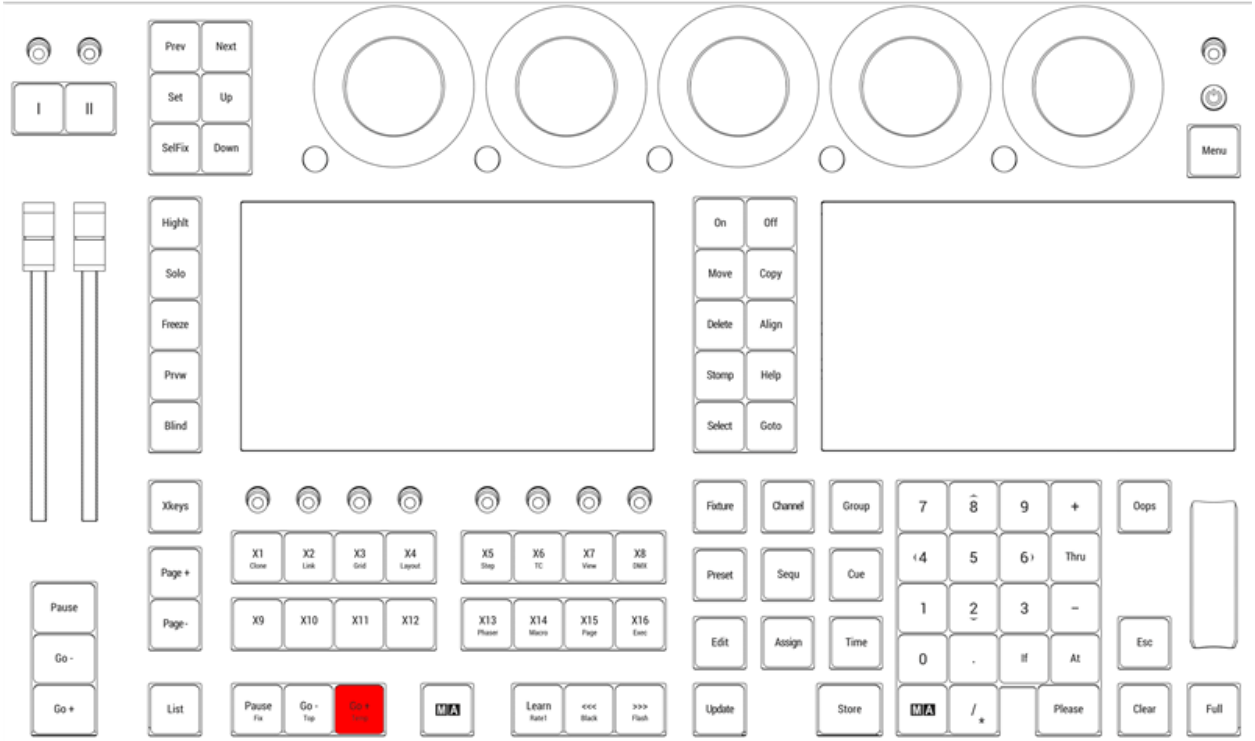
Toggle

Pressing and holding MA + Go+ Go+ enters the Toggle keyword into the command line.

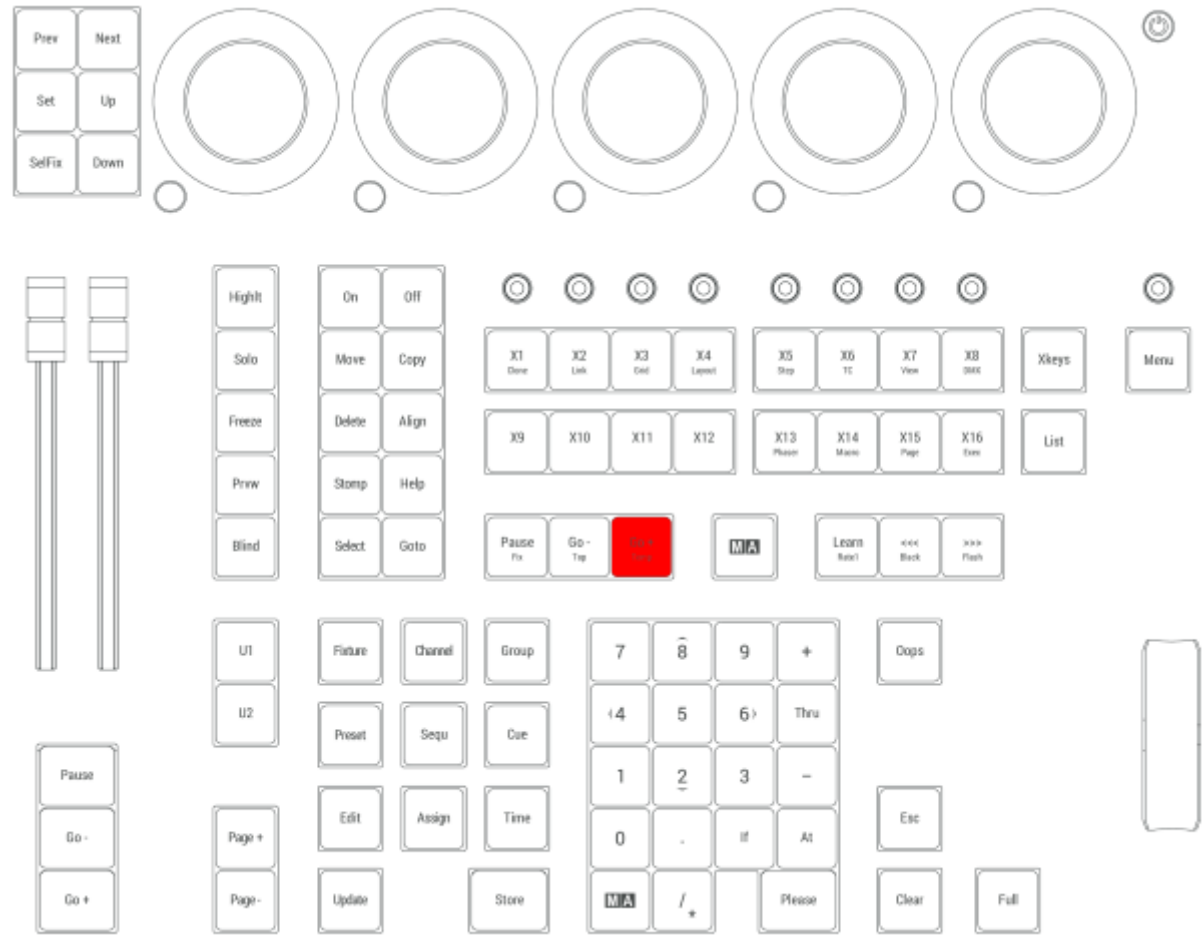
For more information about Toggle, see the **Toggle keyword**.

Location

Go+ is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.24. Go+ [large]

Go+ Executor is executed immediately in the command line when pressing Go+ [large].

For more information about Go+, see the **Go+ keyword**.

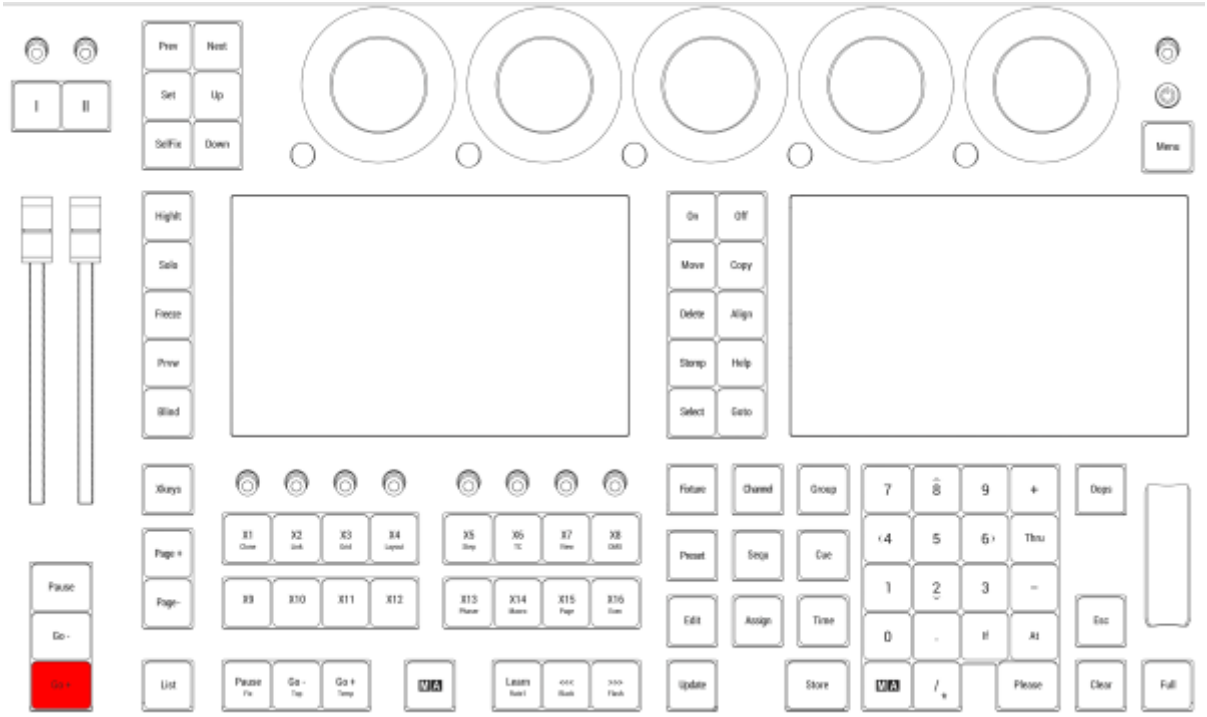
Go+ Loaded

Pressing MA + Go+ [large] executes the Go+ Loaded keyword in the command line.

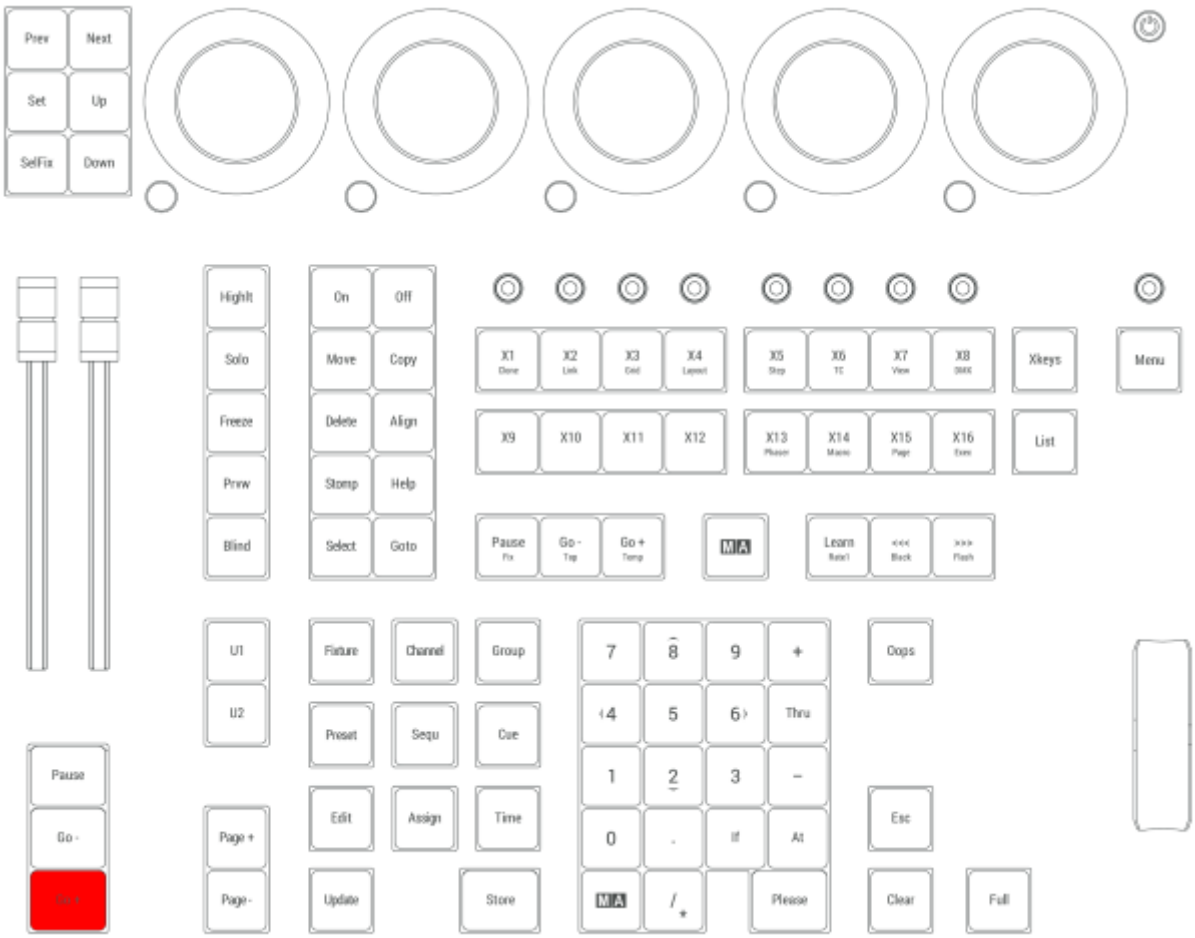
For more information about Go+, see the **Go+ keyword**.

Location

Go+ [large] is located in the master section below the two master faders.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.25. Go- [large]

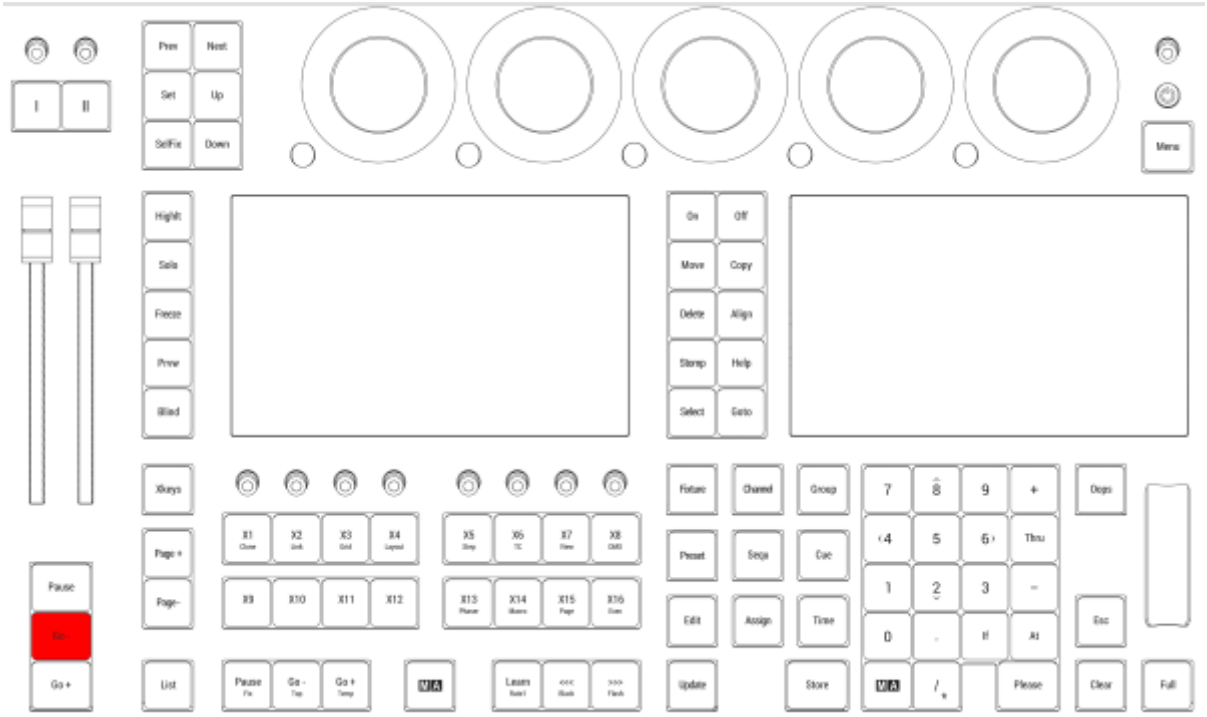
Go- Executor is executed immediately in the command line when pressing Go- [large].

```
OK
:   Go- Executor
```

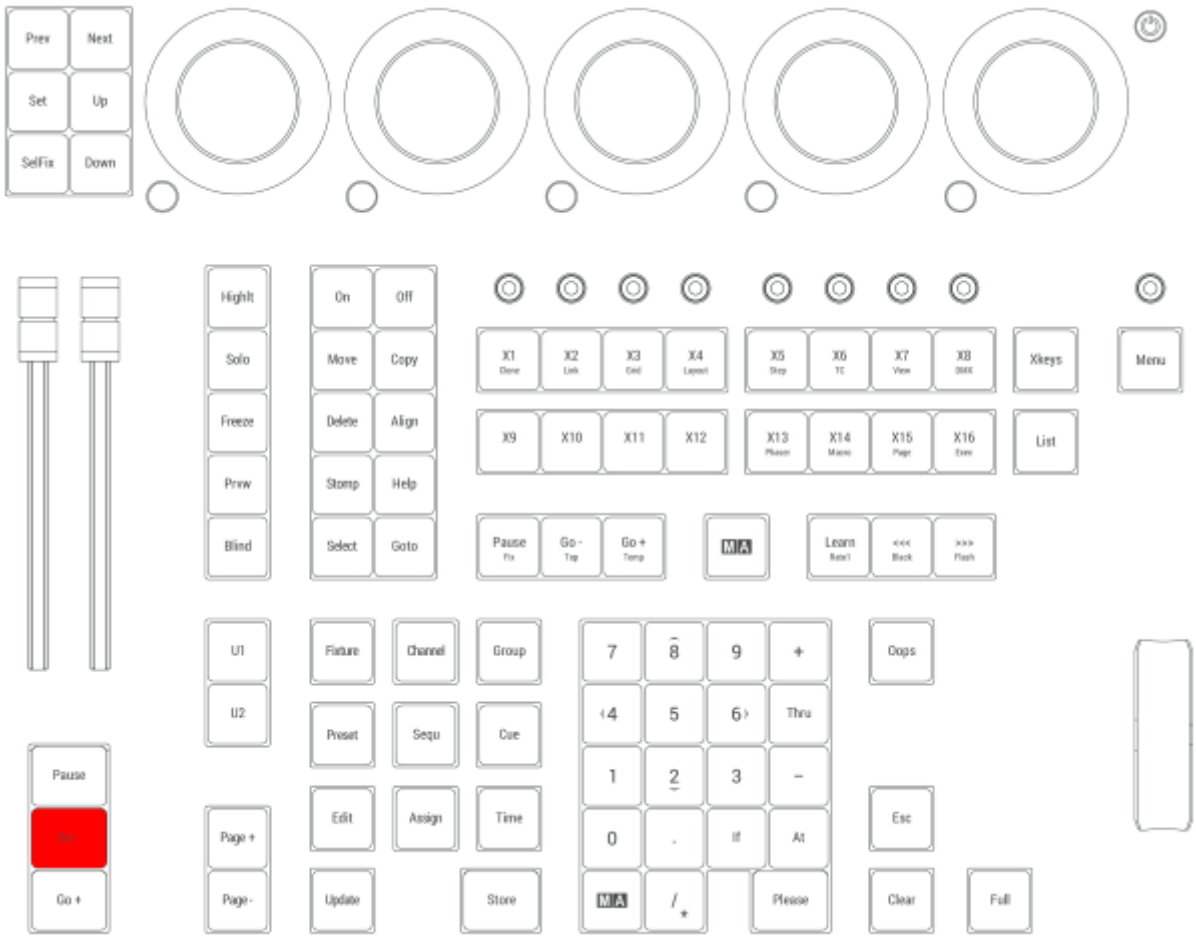
For more information about Go-, see the **Go- keyword**.

Location

Go- [large] is located in the master section under the two master faders.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.26. Go- | Top

Pressing Go- enters the Go- keyword into the command line.

For more information about Go-, see the **Go- keyword**.

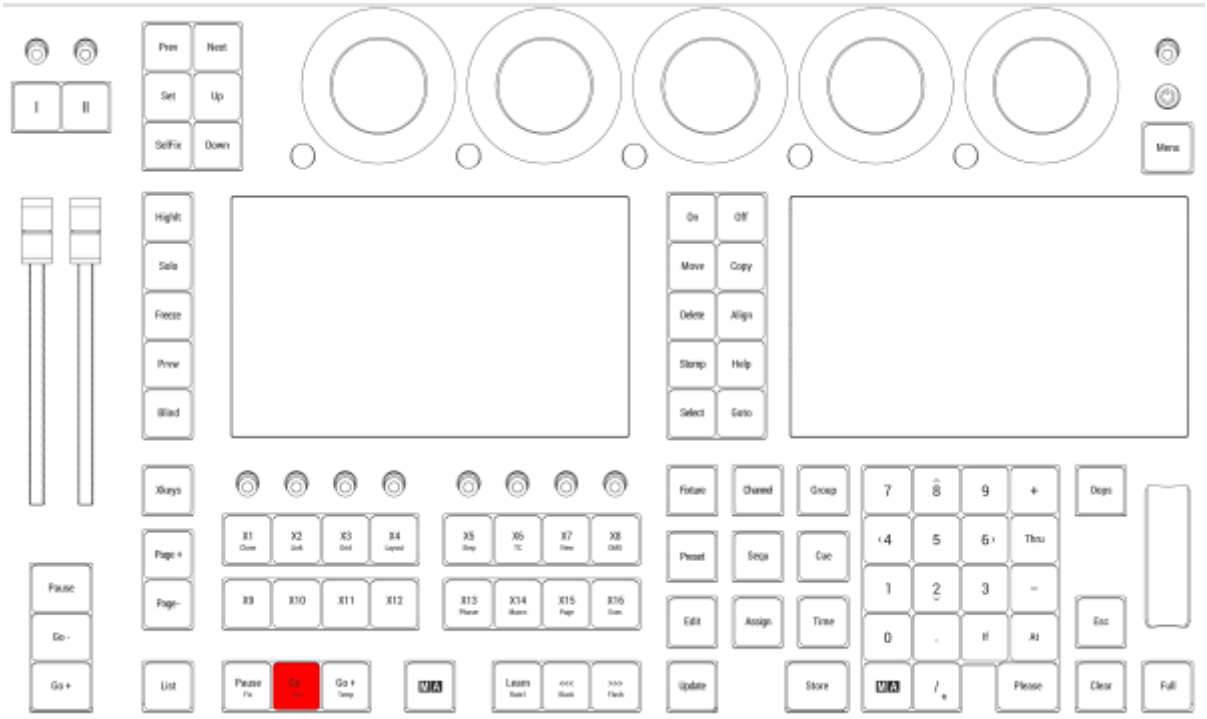
Top

Pressing and holding MA + Go- enters the Top keyword into the command line.

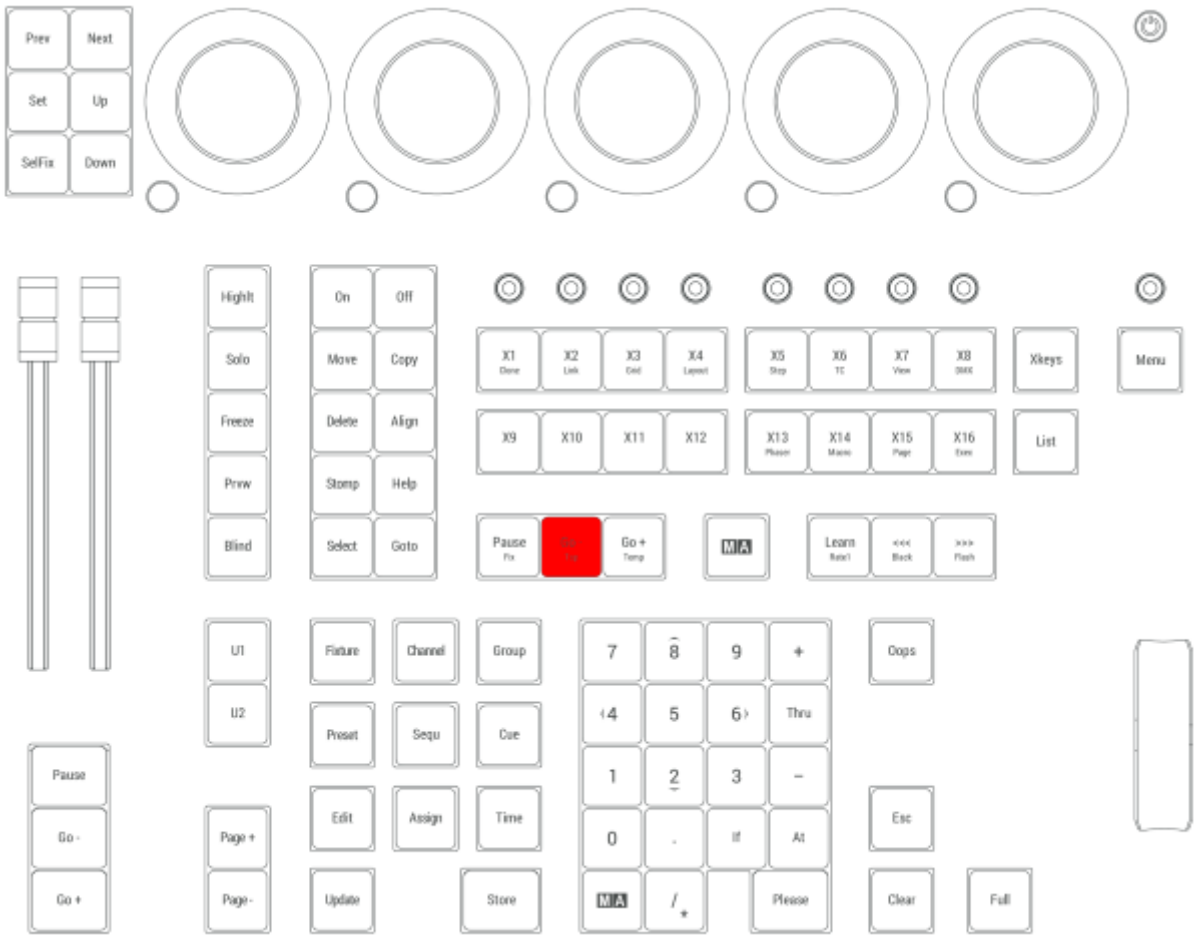
For more information about Top, see the **Top keyword**.

Location

Go- is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.27. Goto

Pressing Goto enters the Goto keyword into the command line.

For more information about Goto, see the **Goto keyword**.

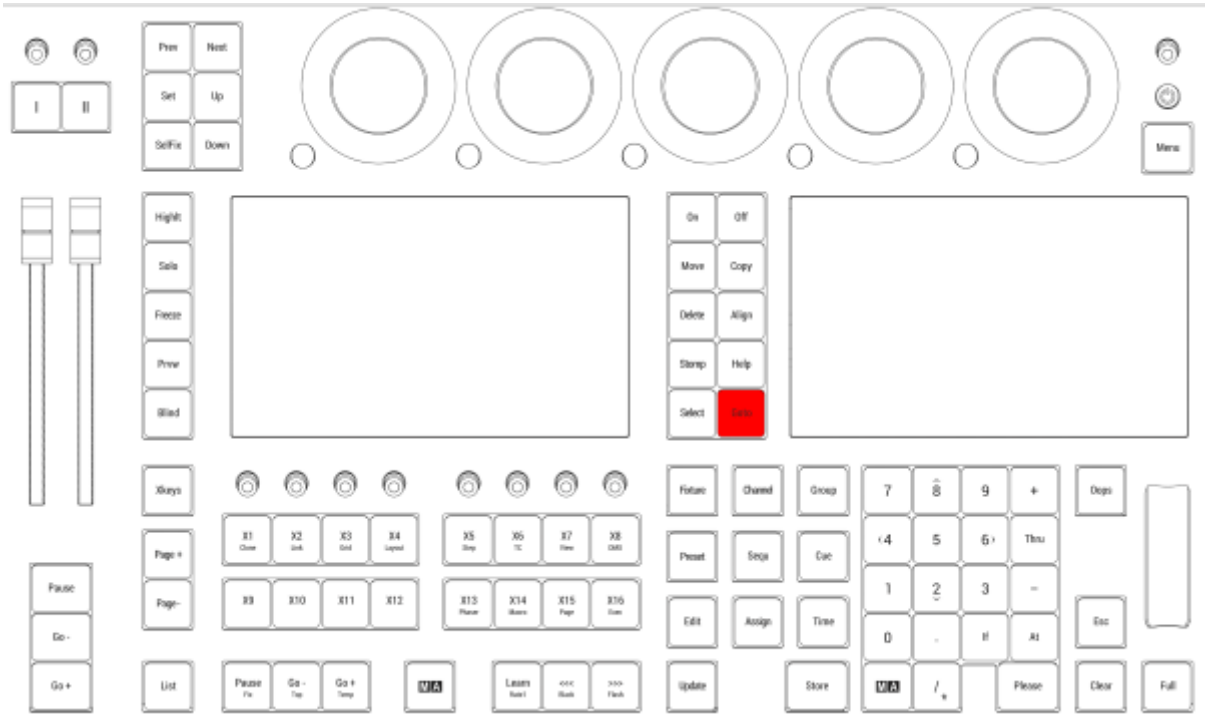
Load

Pressing Goto Goto enters the Load keyword into the command line.

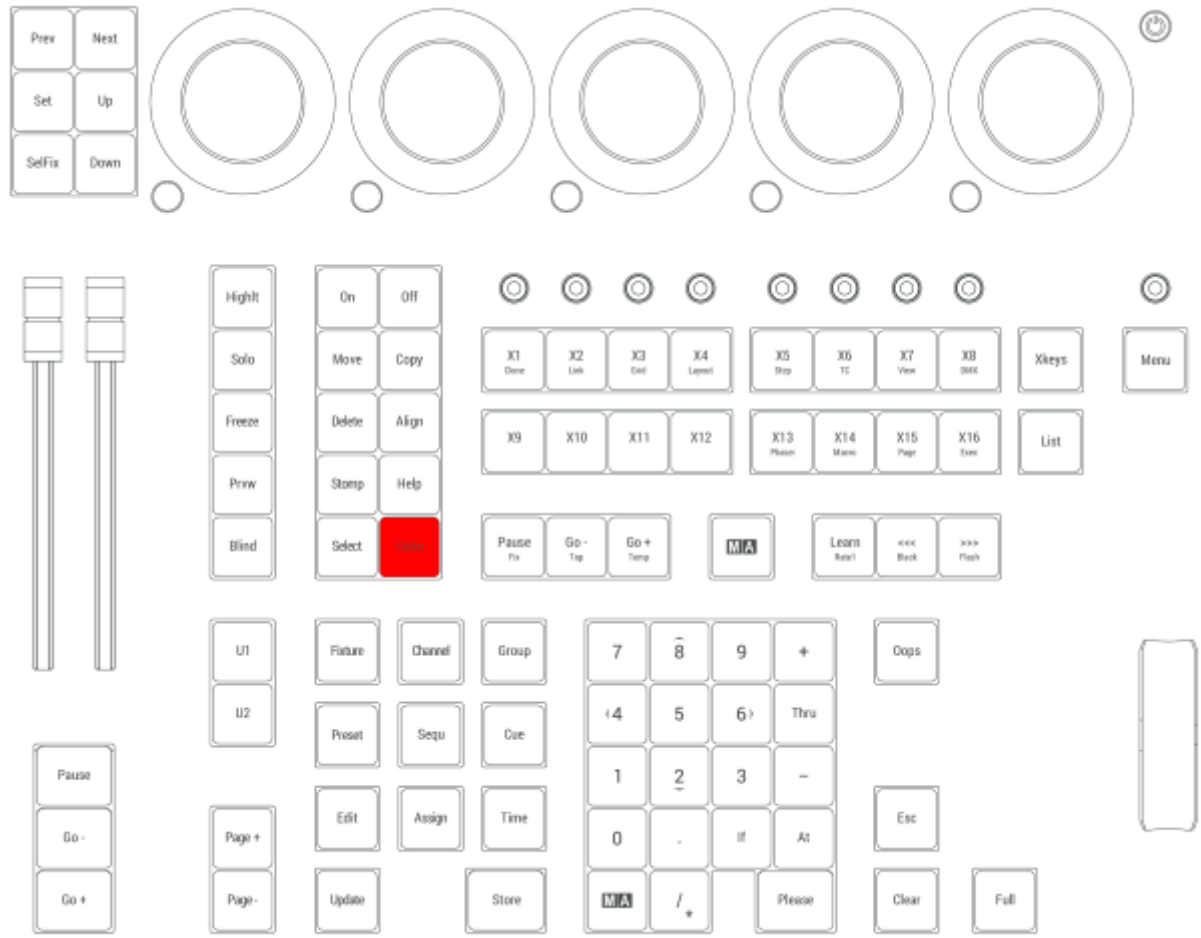
For more information about Load, see the **Load keyword**.

Location

Goto is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.28. Group

Pressing Group enters the Group keyword into the command line.

For more information about Group, see the **Group keyword**.

World

Pressing Group Group enters the World keyword into the command line.

For more information about World, see the **World keyword**.

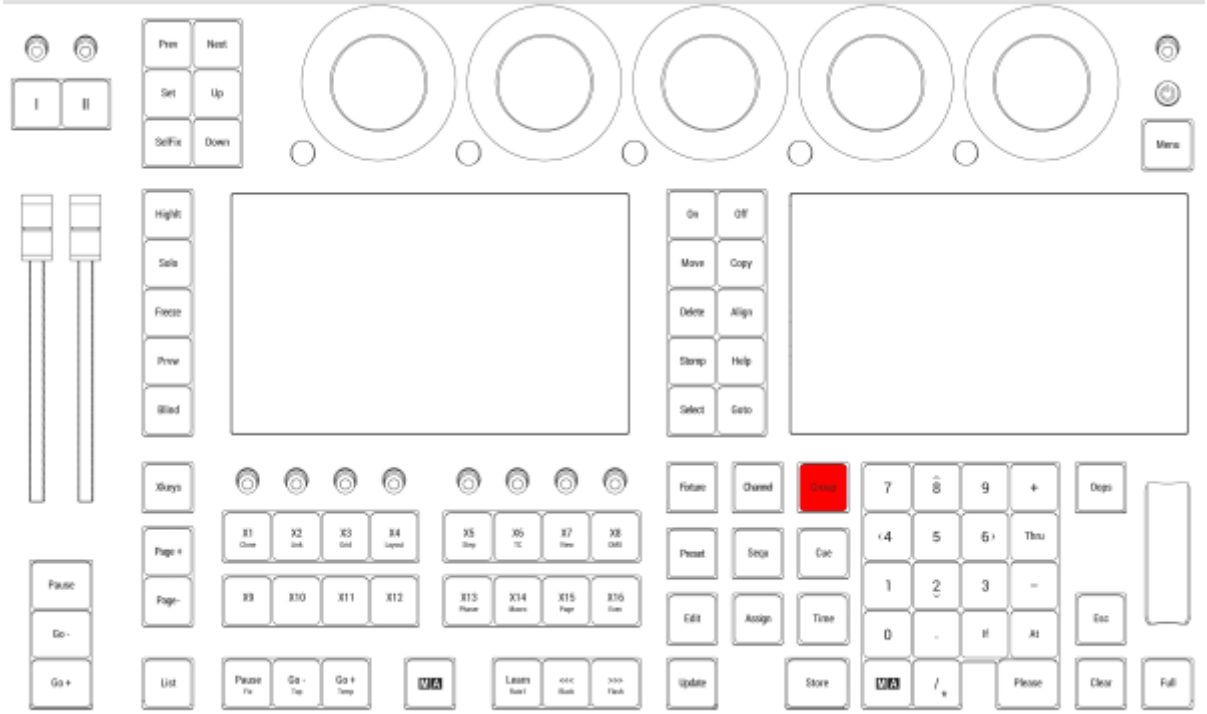
Filter

Pressing Group Group Group enters the Filter keyword into the command line.

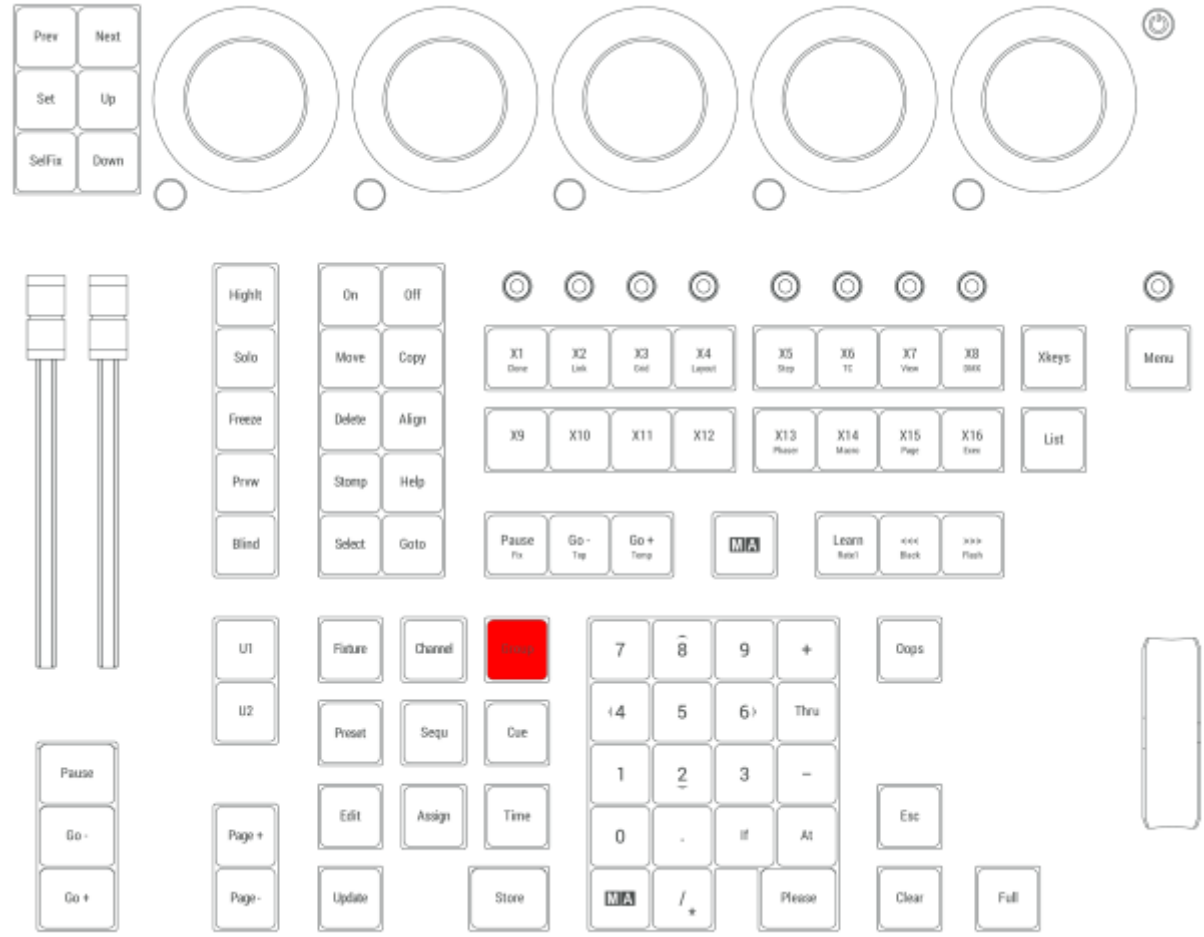
For more information about Filter, see the **Filter keyword**.

Location

Group is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.29. Help

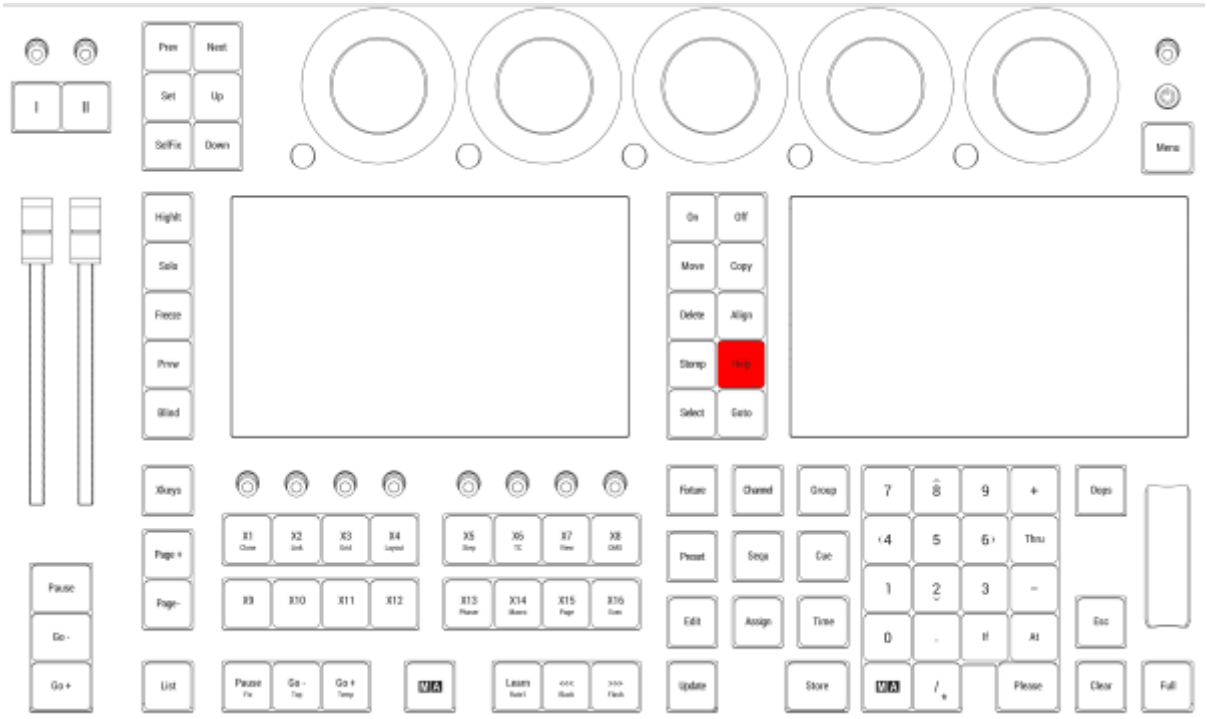
Pressing Help enters the Help keyword into the command line.

Pressing Help and an element in the user interface opens the corresponding article in the Help.

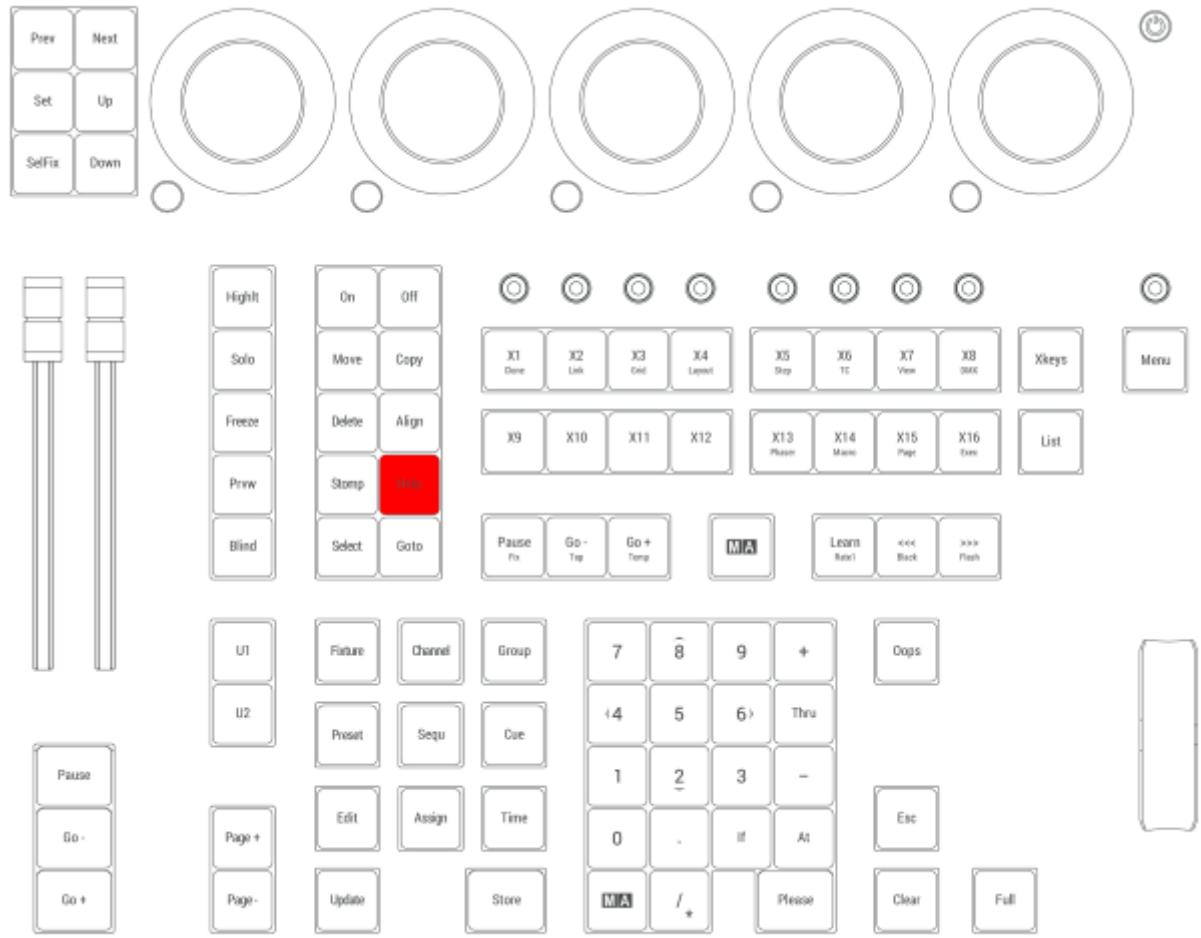
For more information about the Help, see the **Help keyword** and **Open the Help in the Console**.

Location

Help is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.30. Highlt [Highlight]

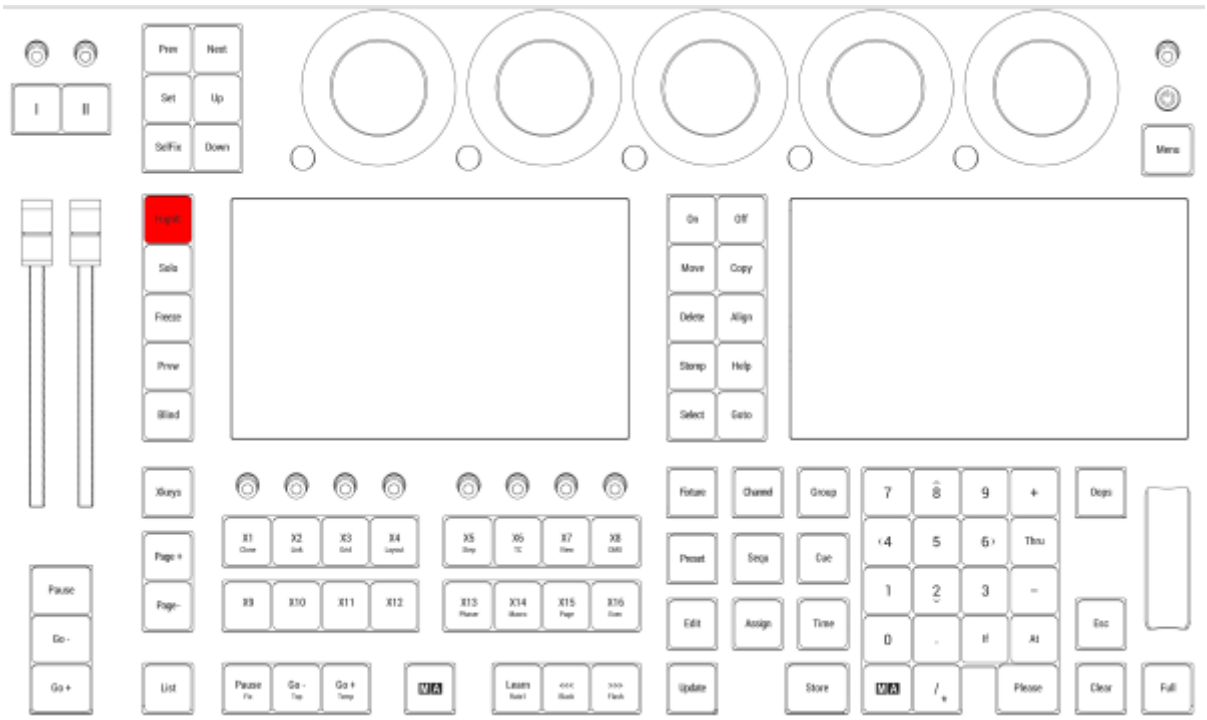
Pressing Highlt toggles the highlight function.

For more information about Highlight, see the **Highlight keyword**.

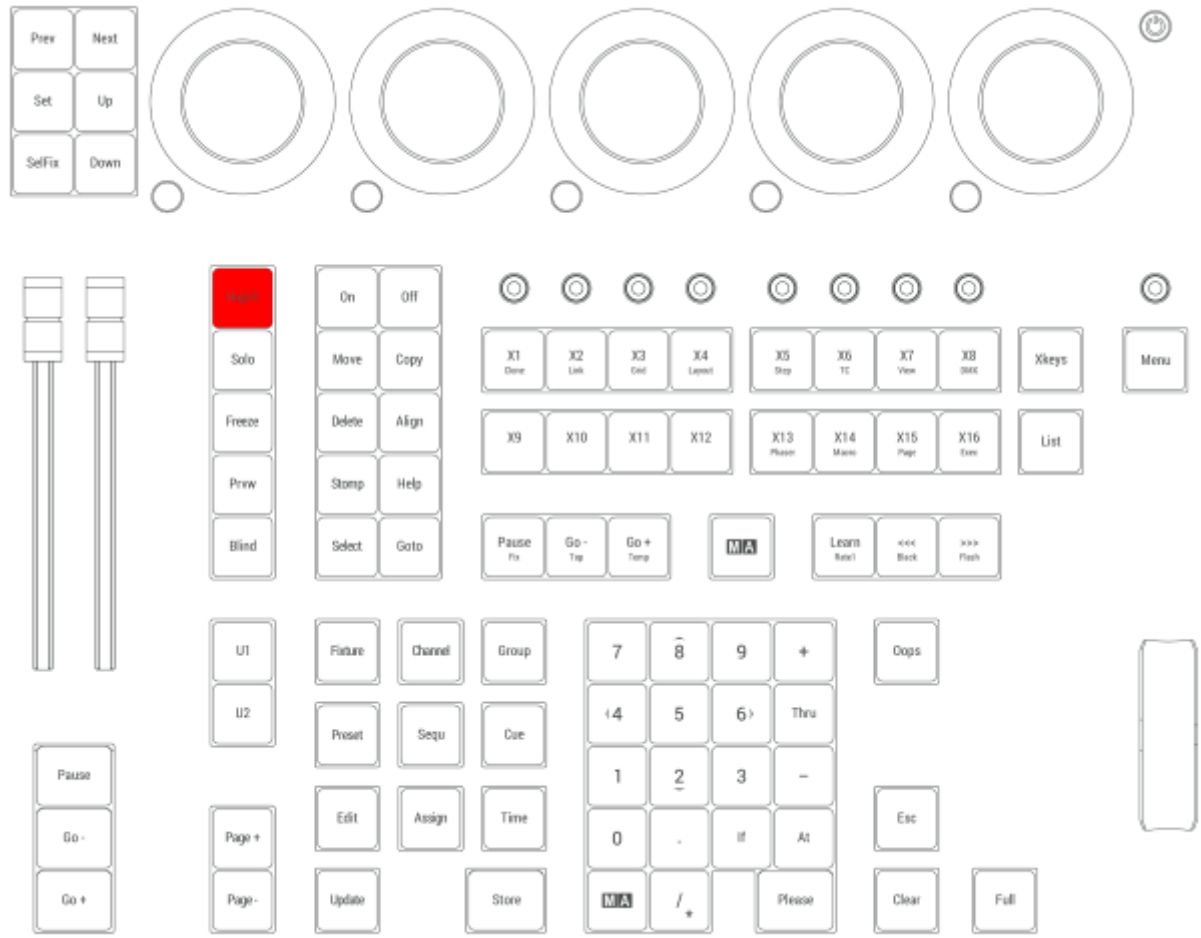
Pressing MA + Highlt toggles the lowlight function. For more information about Lowlight, see the **Lowlight keyword**.

Location

Highlt is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.31. If

IfOutput

Pressing If enters the IfOutput keyword into the command line.

For more information about IfOutput, see the **IfOutput keyword**.

IfActive

Pressing If If enters the IfActive keyword into the command line.

For more information about IfActive, see the **IfActive keyword**.

IfProg

Pressing If If If enters the IfProg keyword into the command line.

For more information about IfProg, see the **IfProg keyword**.

If

Pressing If If If If enters the If keyword into the command line.

For more information about If, see the **If keyword**.

EndIf

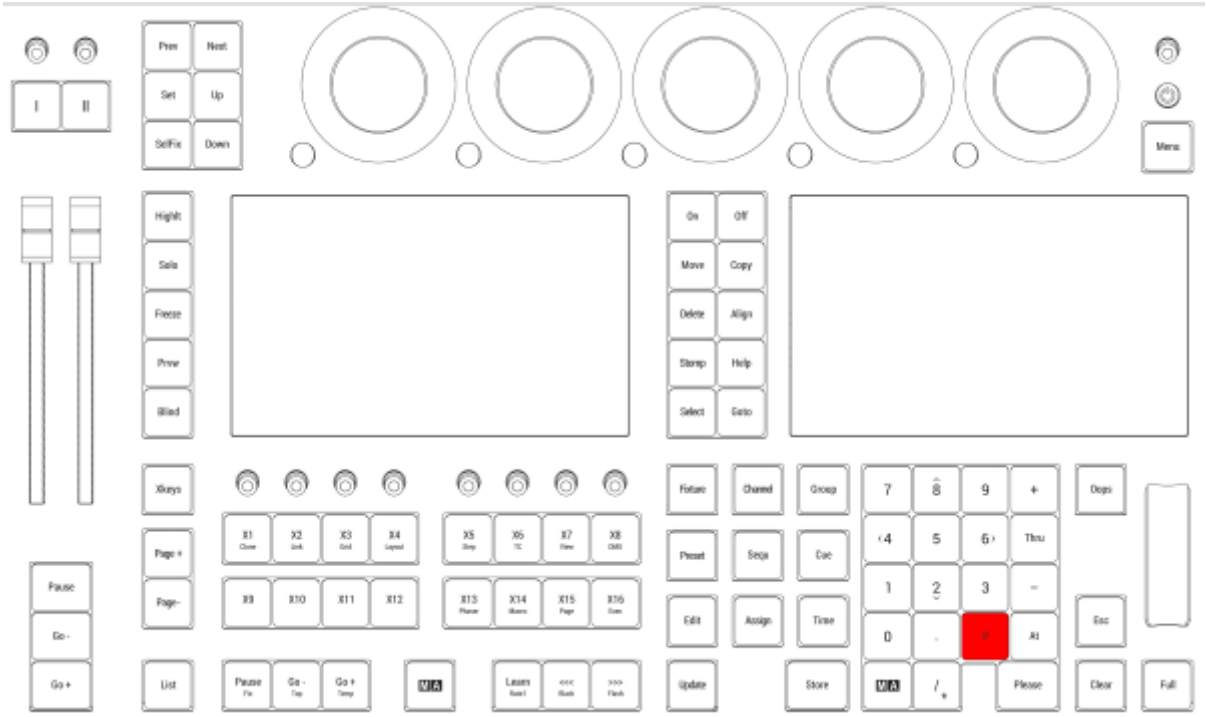
Pressing If again once the If command is already in the command line enters the EndIf keyword into the command line.

User name[Fixture]>Store If Group 5 EndIf Preset 1.1

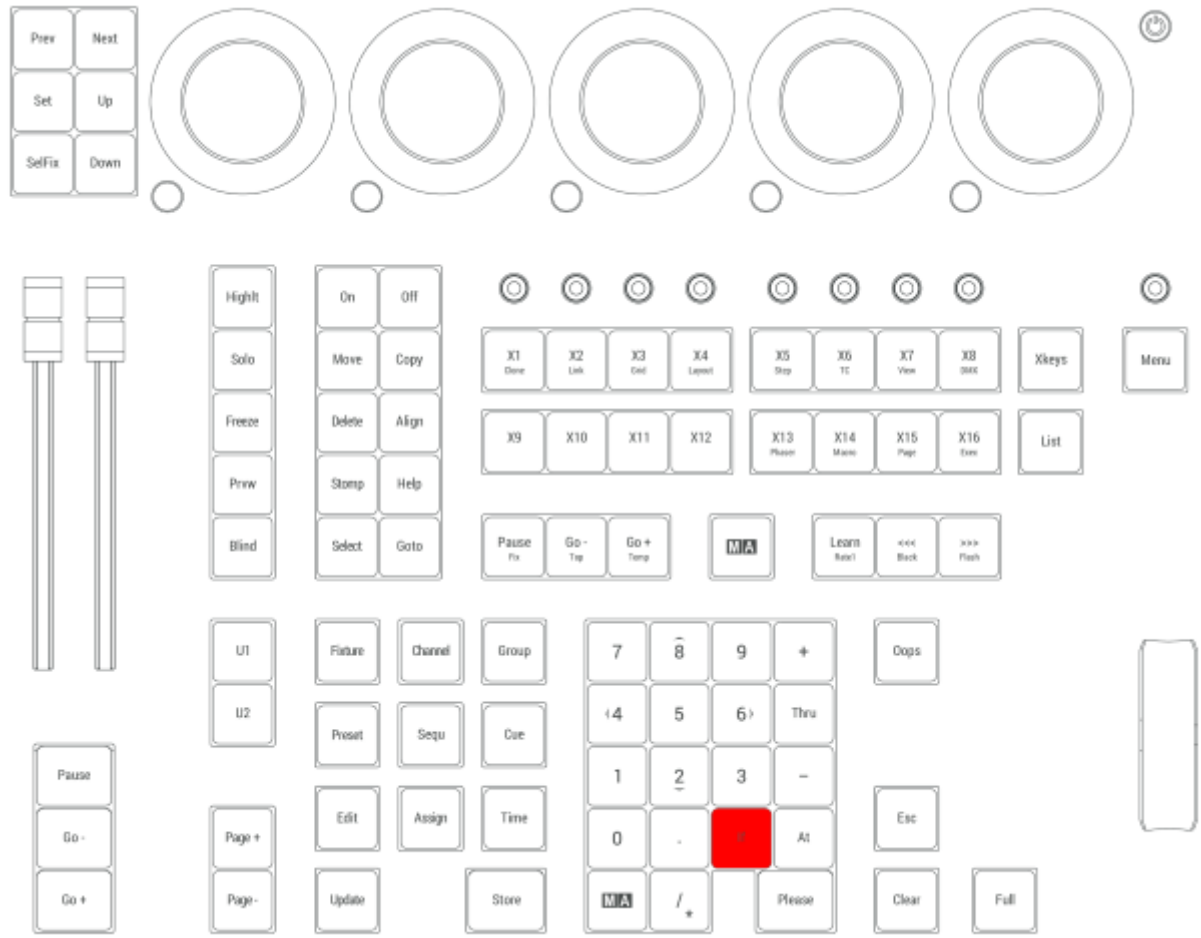
For more information about EndIf, see the **EndIf keyword**.

Location

If is located in the numeric keys section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.32. Learn | Rate1

Pressing Learn enters the Learn keyword into the command line.

For more information about Learn, see the **Learn keyword**.

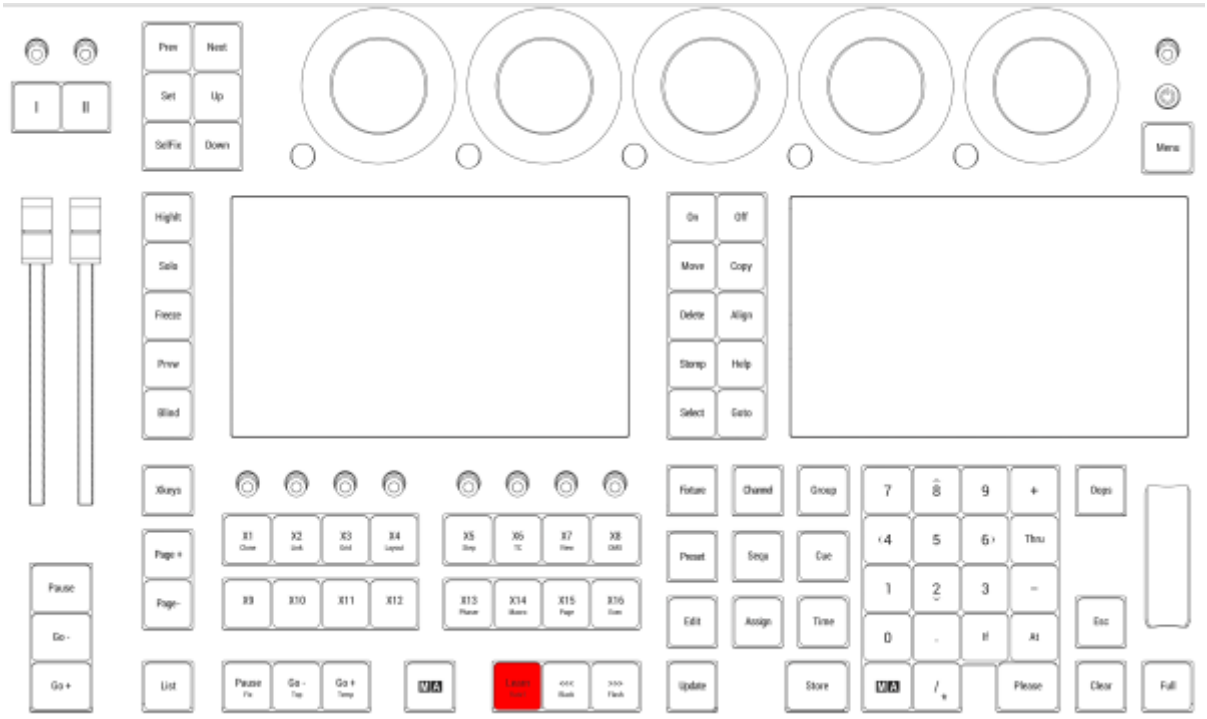
Rate1

Pressing and holding MA + Learn enters the Rate1 keyword into the command line.

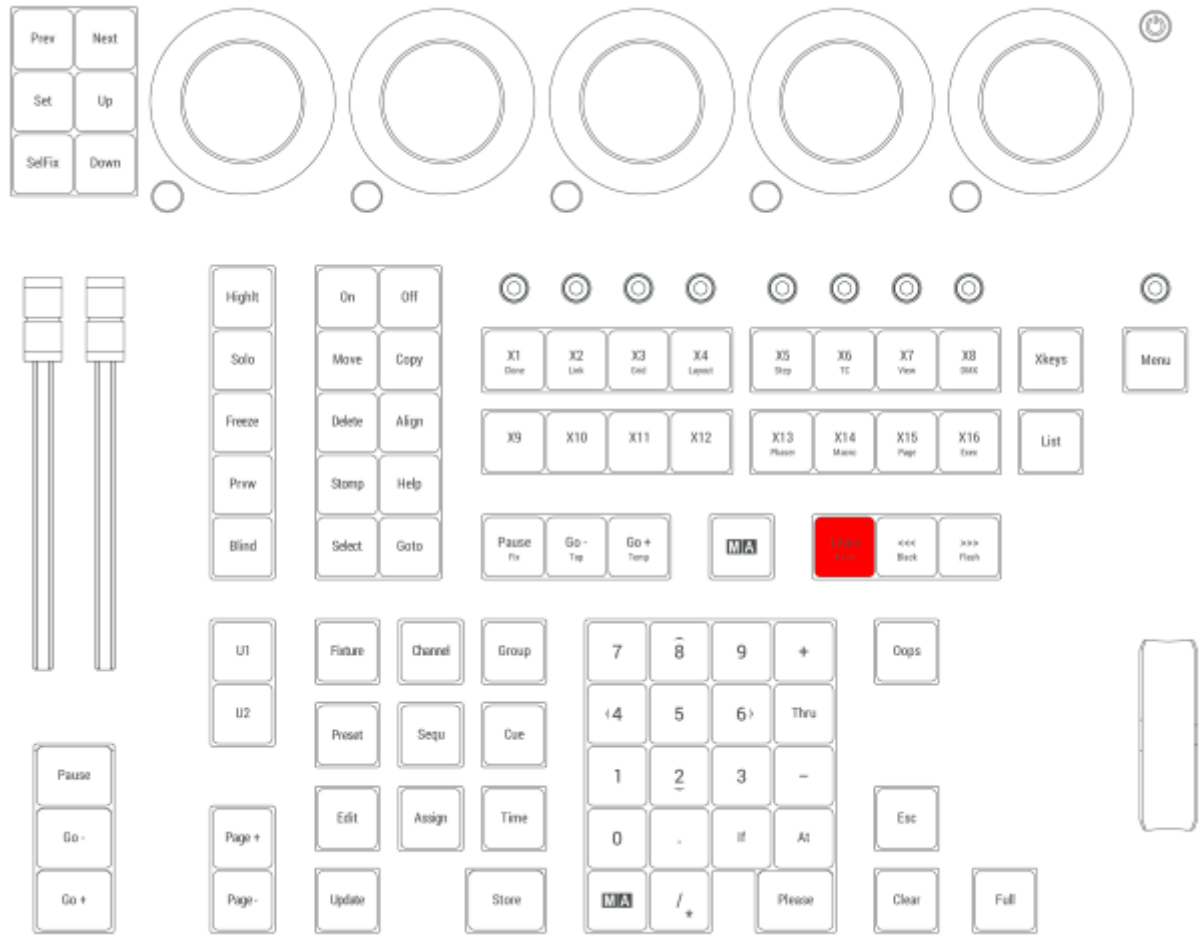
For more information about Rate1, see the **Rate1 keyword**.

Location

Learn is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.33. List

Pressing List enters the List keyword into the command line.

For more information about List, see the **List keyword**.

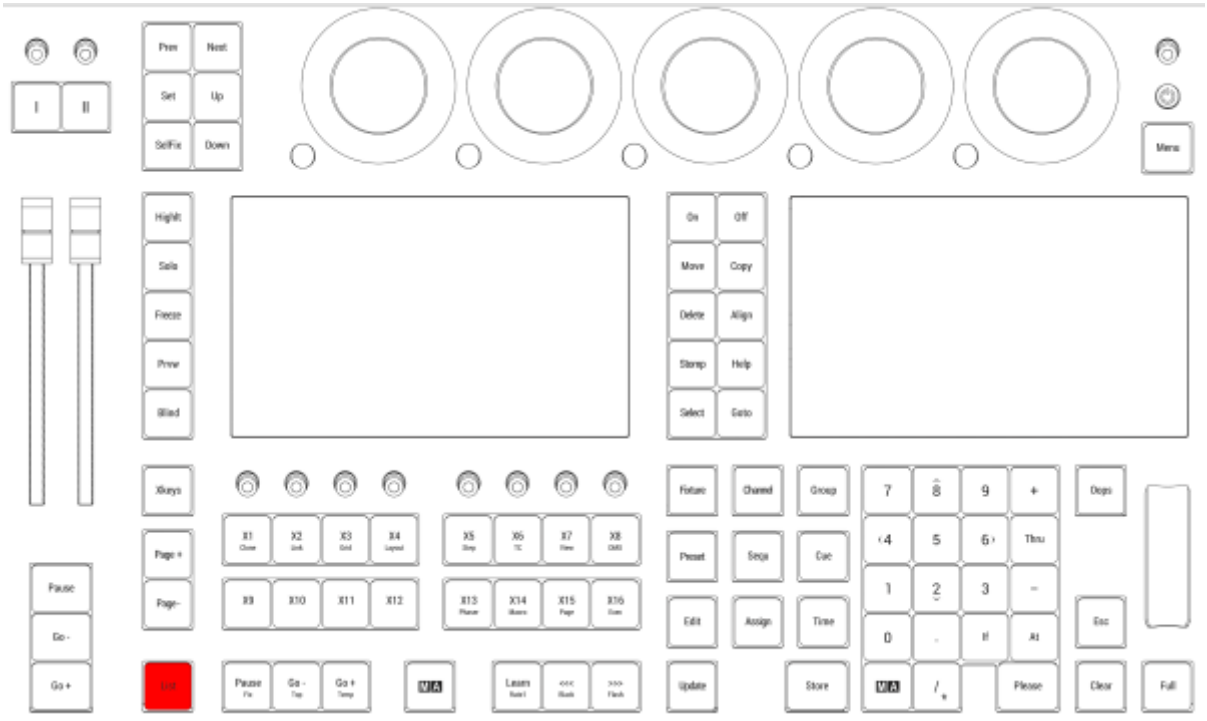
ListRef

Pressing and holding MA + List enters the ListRef keyword into the command line.

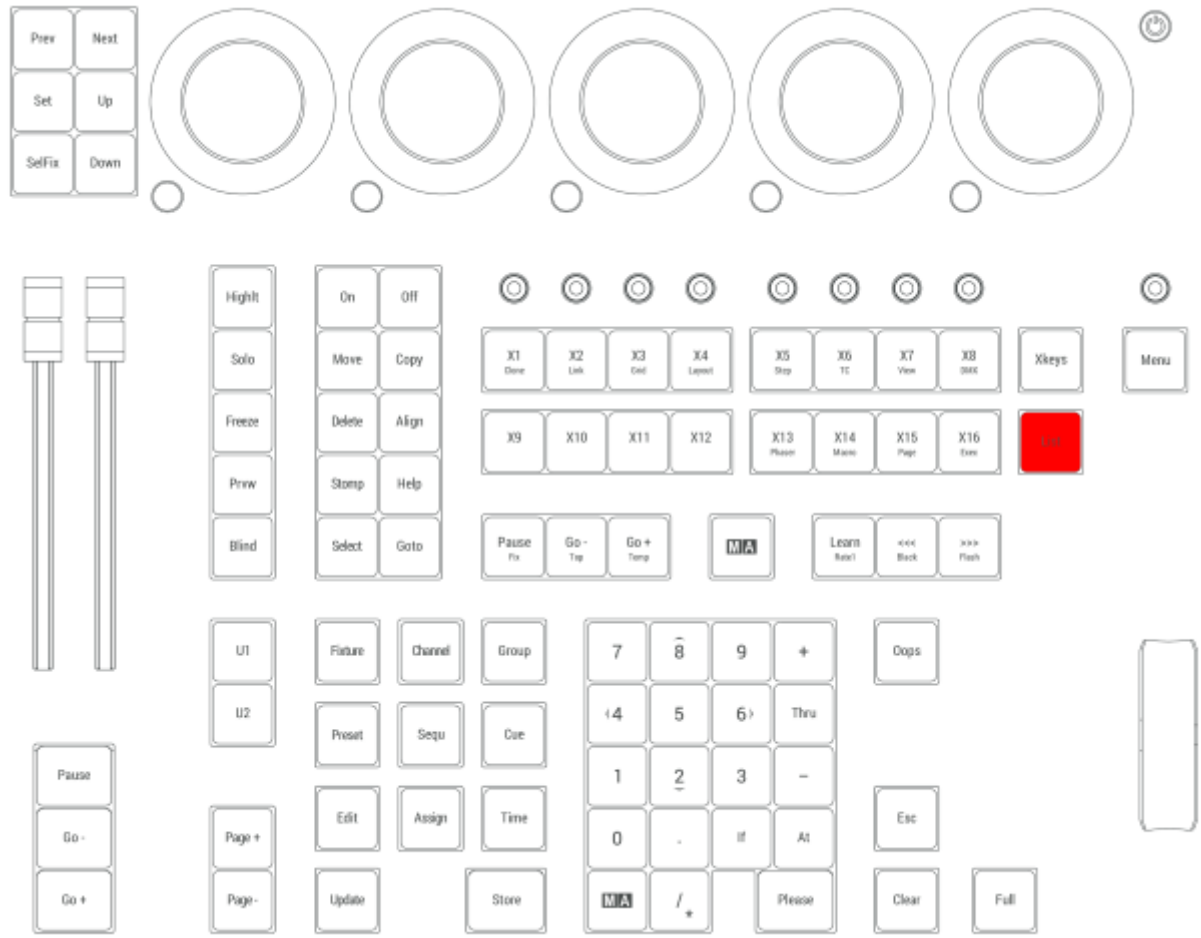
For more information about List, see the **ListRef keyword**.

Location

List is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.34. MA

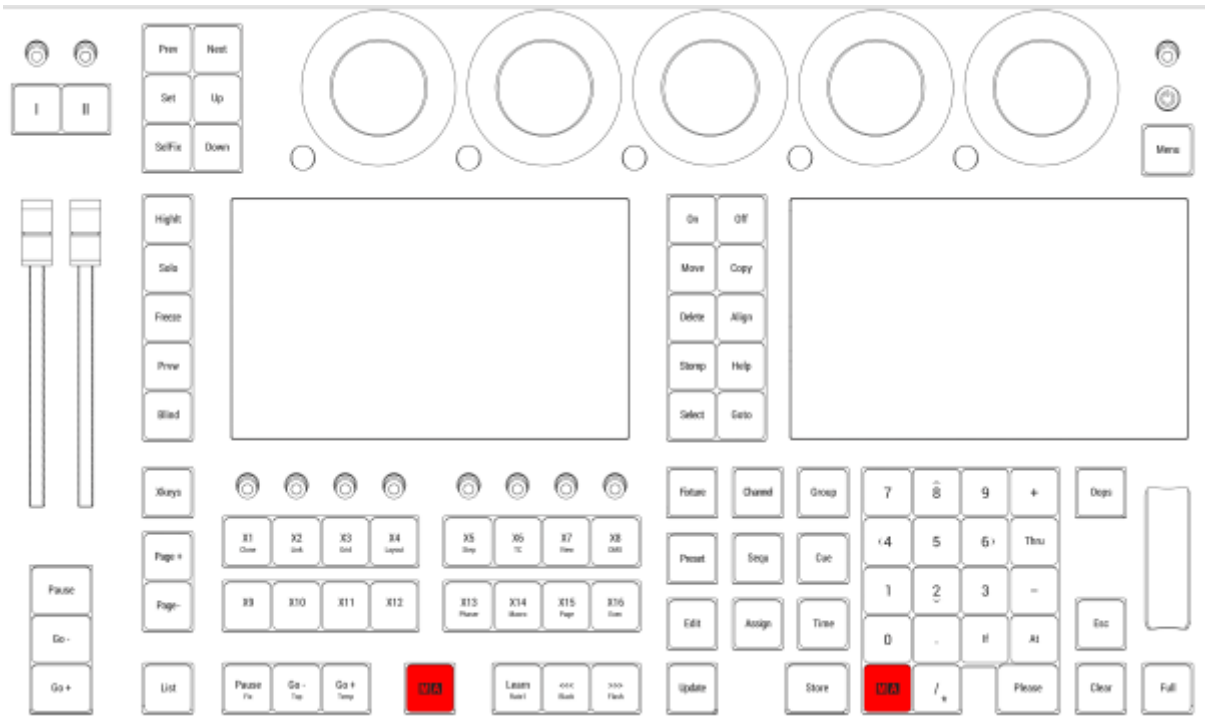
Pressing and holding MA in combination with other keys, provides shortcuts to other functions. There are two MA keys there – on both layouts of the console. They are identical and have the same function.

Pressing MA + Please sets the focus to the command line.

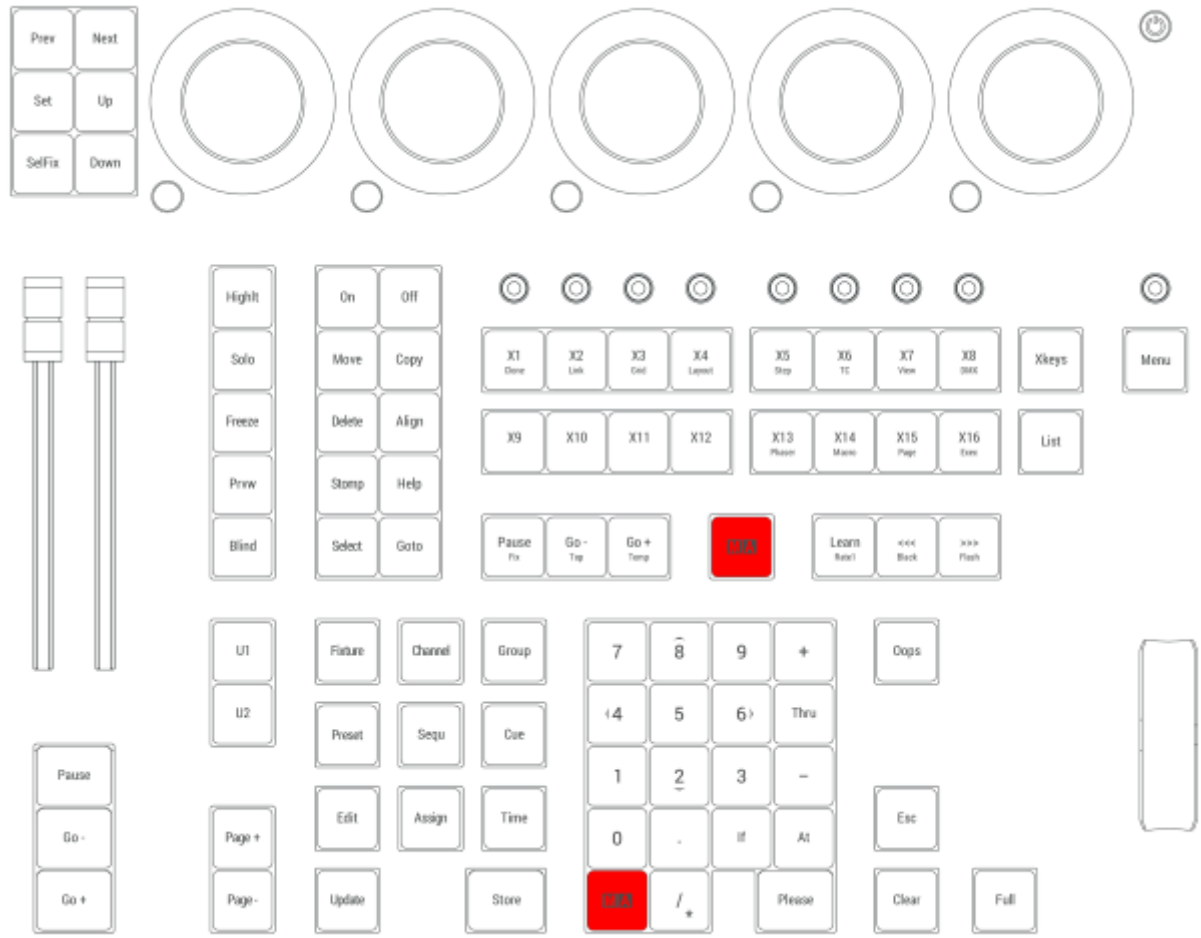
To check out the complete list of functions, see **Keys**.

Location

MA is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

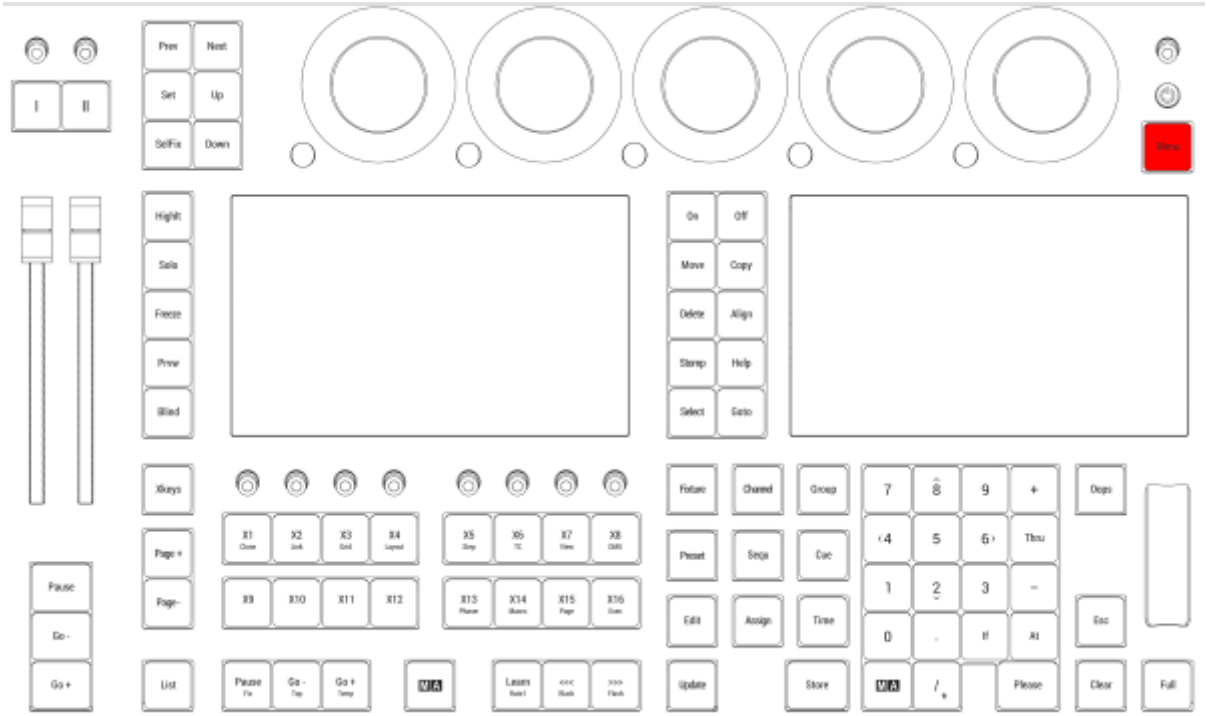
1.3.15.35. Menu

Pressing Menu opens the Menu pop-up.

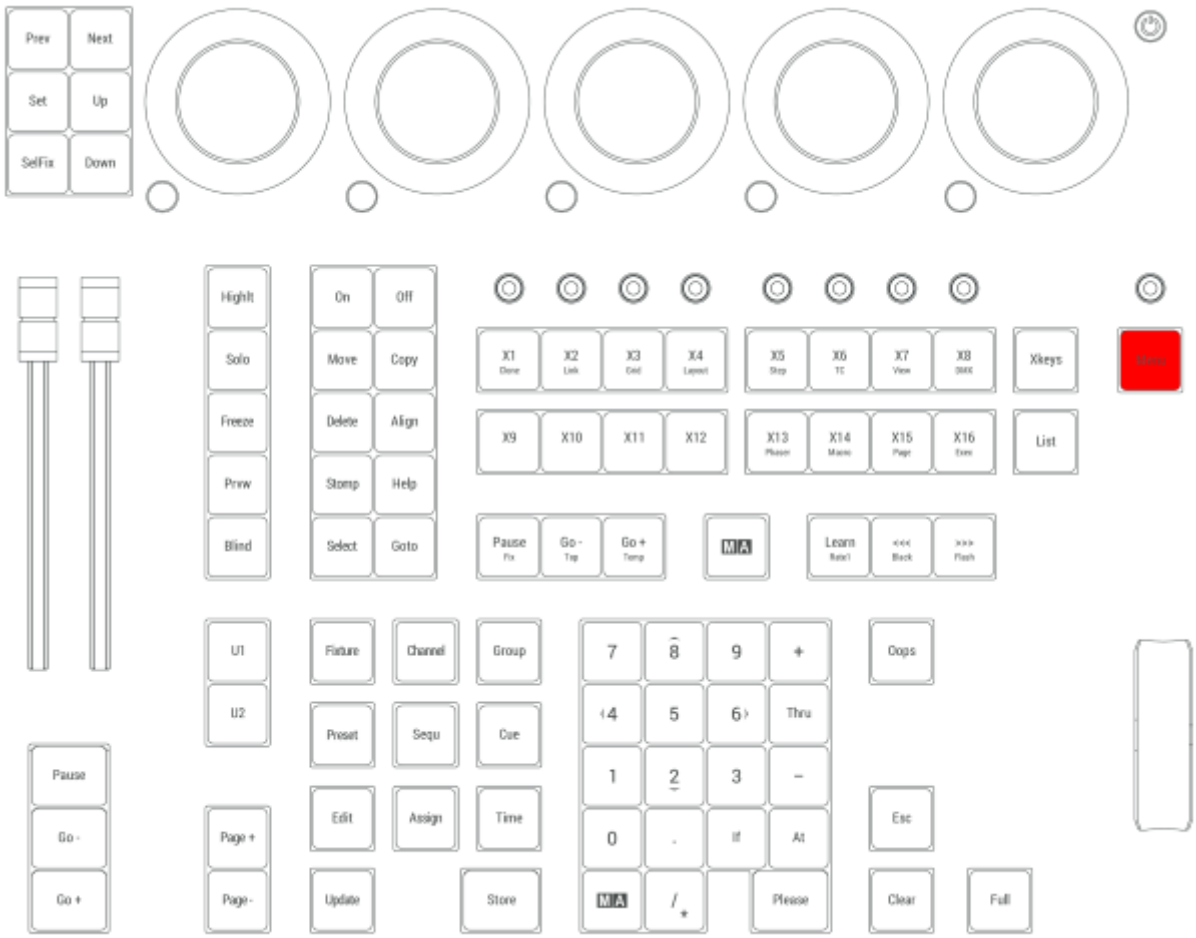
For more information about Menu, see the **Menu keyword**.

Location

Menu is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.36. Move

Pressing Move enters the Move keyword into the command line.

For more information about Move, see the **Move keyword**.

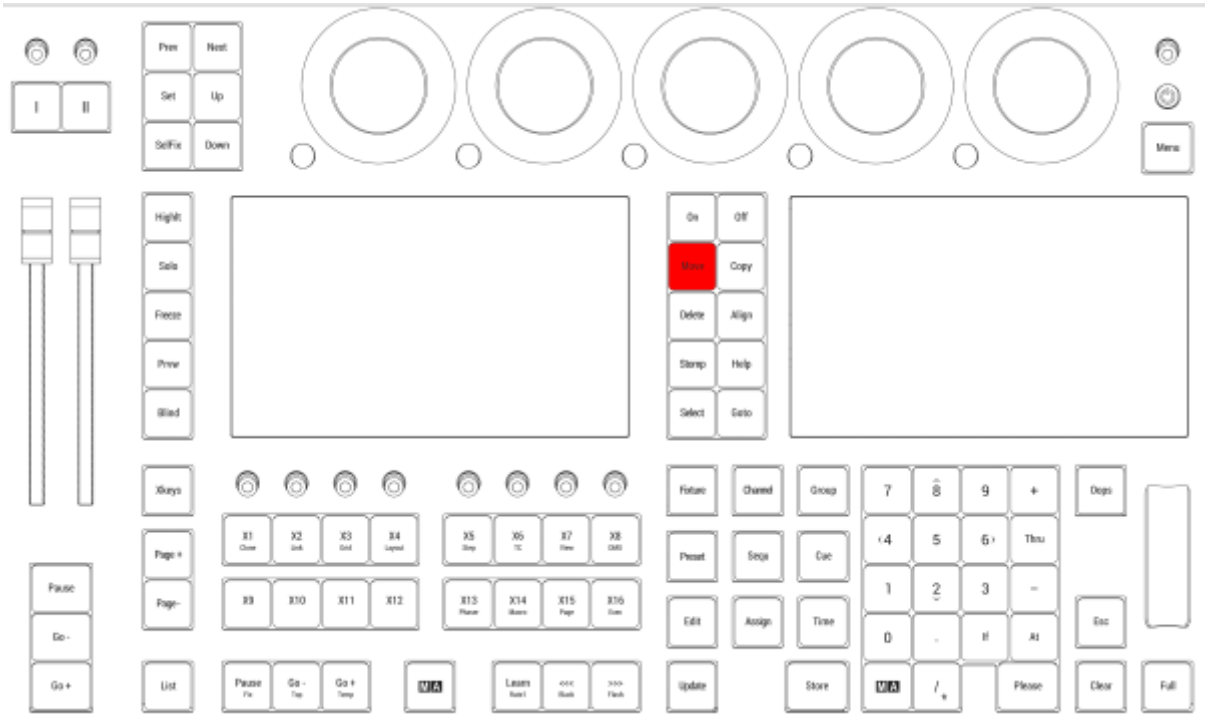
Exchange

Pressing Move Move enters the Exchange keyword into the command line.

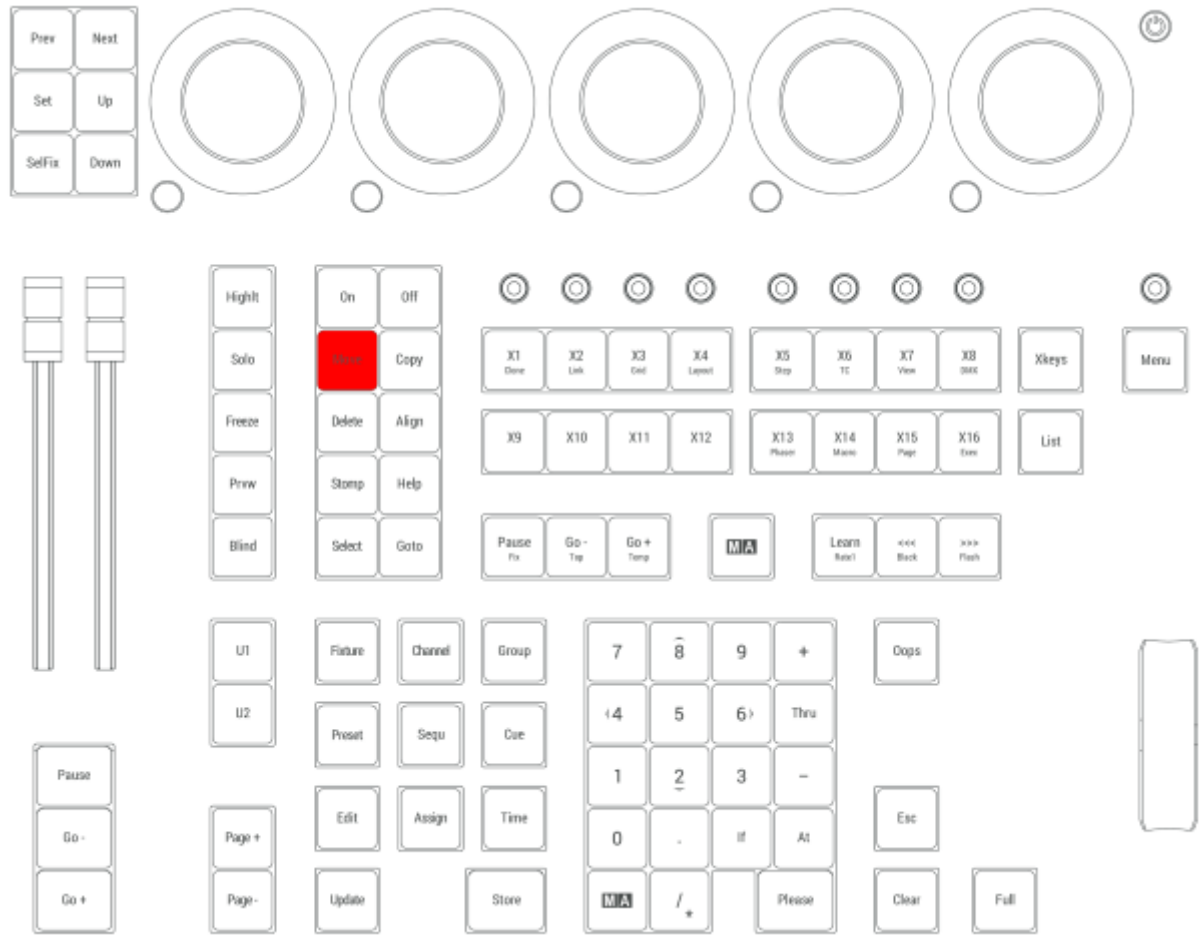
For more information about Exchange, see the **Exchange Keyword**.

Location

Move is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.37. Next

Pressing Next executes the Next keyword in the command line.

For more information about Next, see the **Next keyword**.

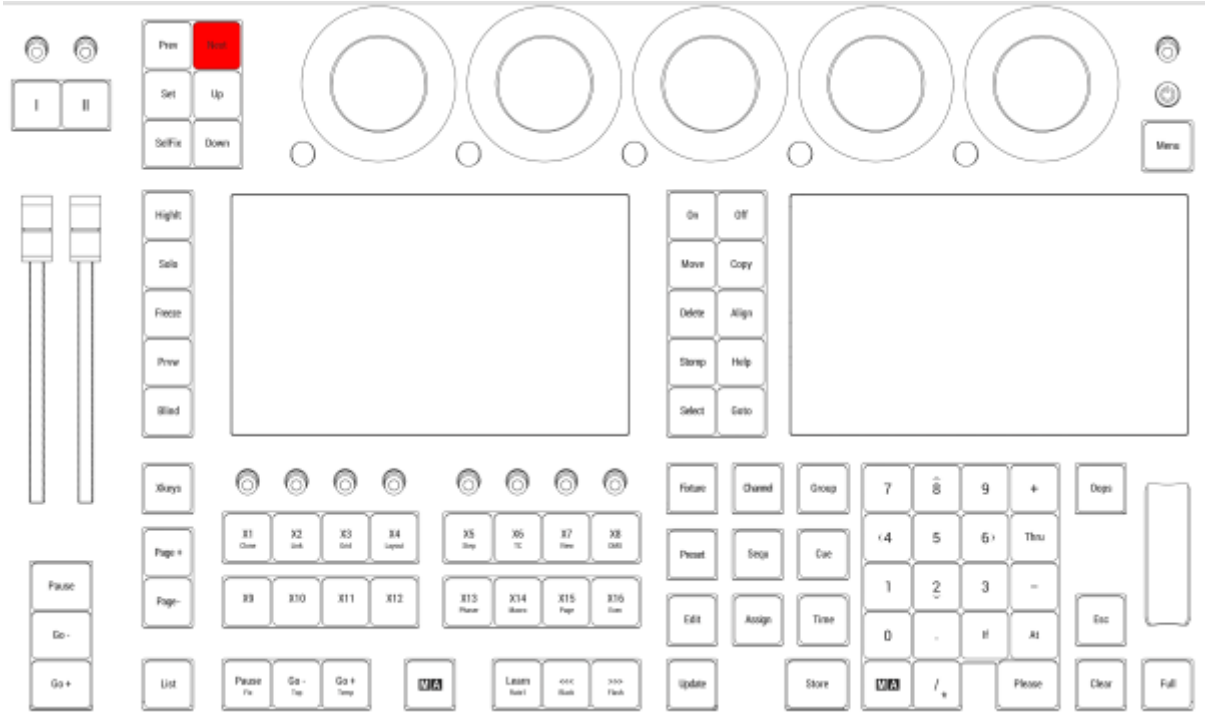
Next Step

Pressing MA + Next executes the Next Step command in the command line.

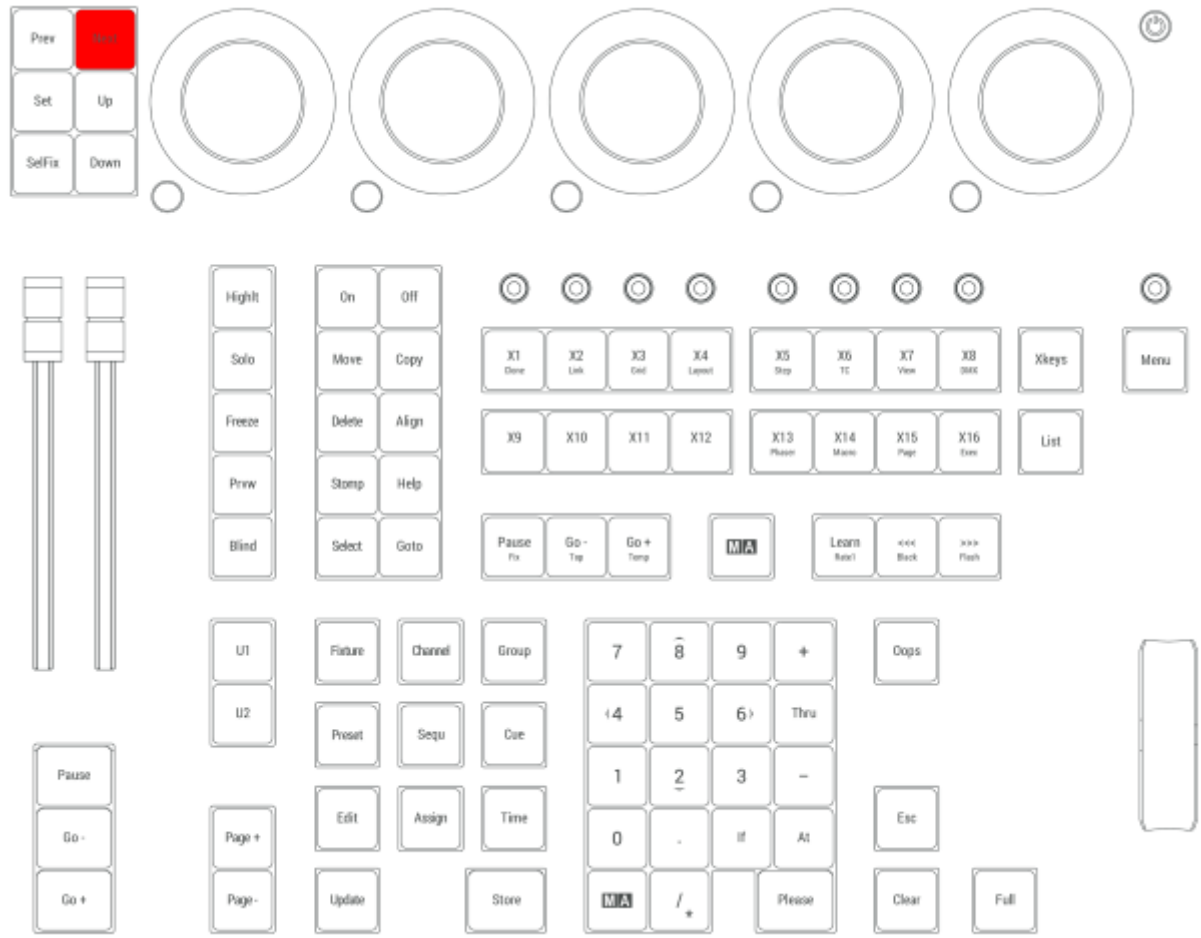
For more information about Next Step, see the **Next keyword** and the **Step keyword**. For more information on steps see **Phaser**.

Location

Next is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.38. Numeric Keys | Arrows

Pressing a numeric key enters the number into the command line.

Move Cursor

Pressing MA + 8 moves the cursor one row upward.

Pressing MA + 2 moves the cursor one row downward.

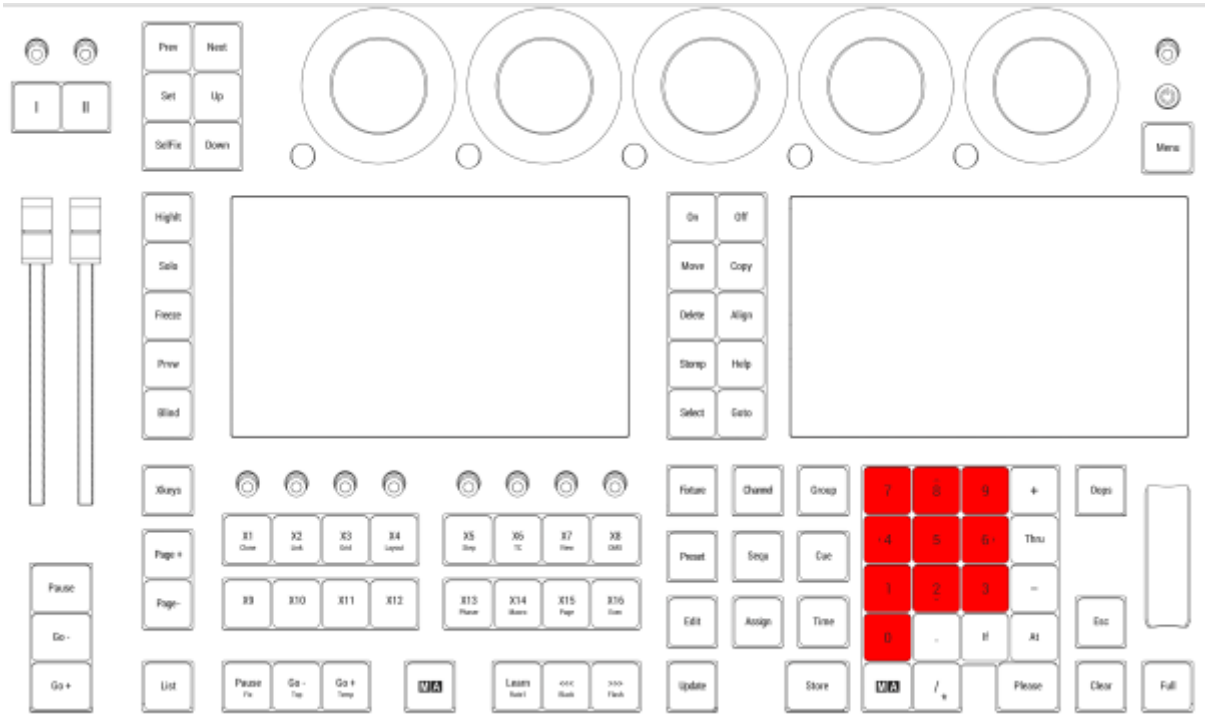
Pressing MA + 4 moves the cursor one column to the left.

Pressing MA + 6 moves the cursor one column to the right.

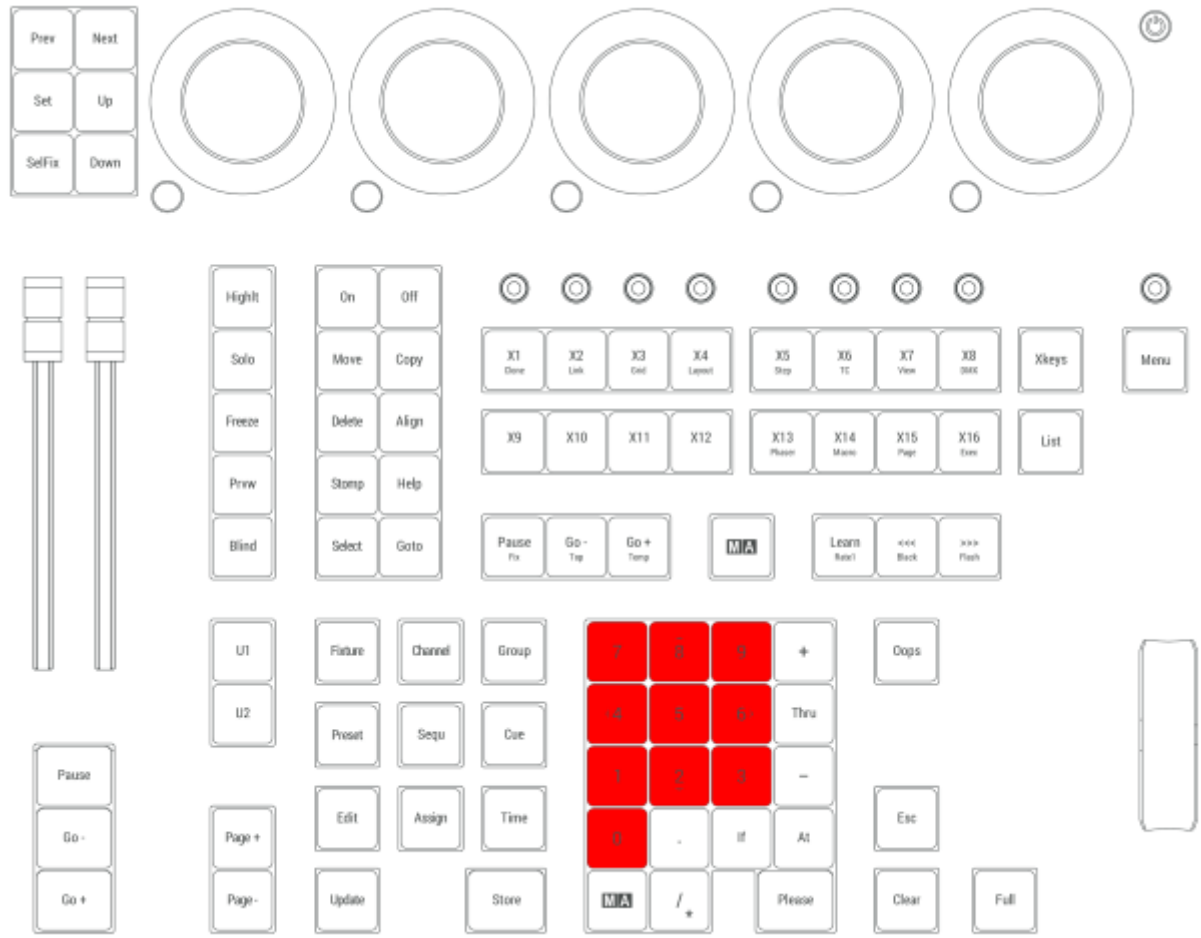
The arrow keys have the same functions on a keyboard.

Location

The numeric keys are located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.39. Off

Pressing Off enters the Off keyword into the command line.

For more information about Off, see the **Off keyword**.

Off Menu

Pressing Off Off opens the Off menu.

Read more about the Off menu **here**.

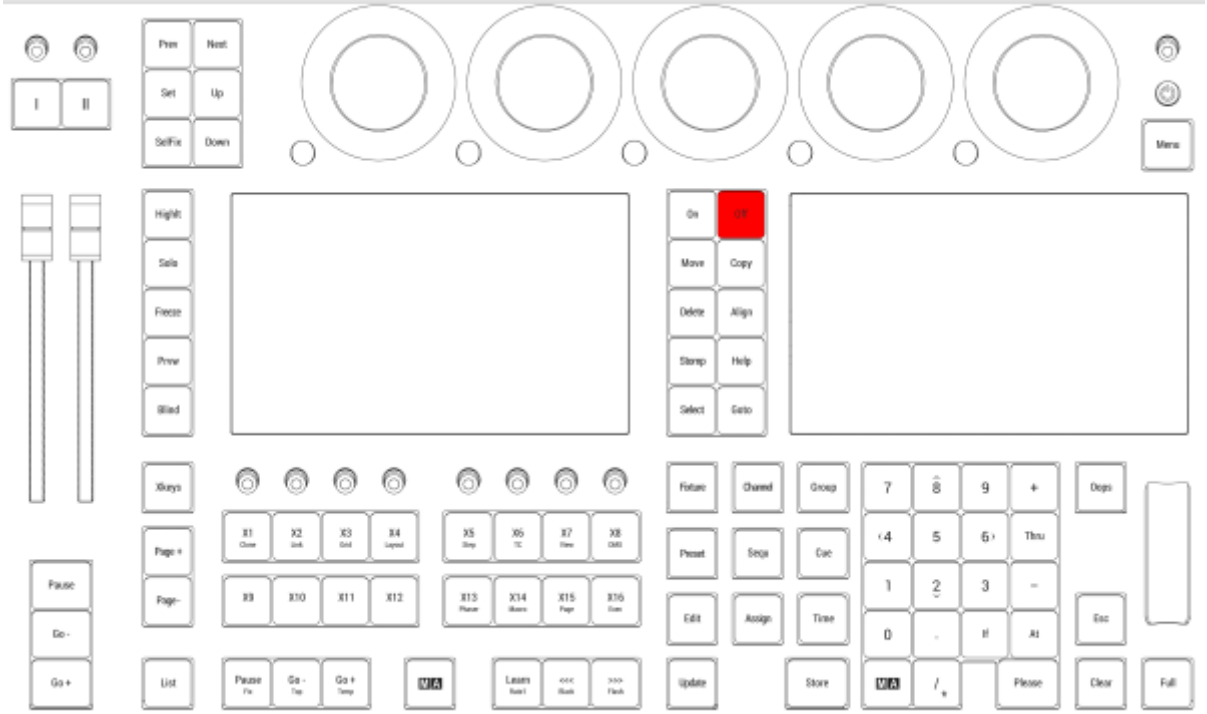
Close RemoteHID Connection

Pressing MA + MA + Off closes every RemoteHID connection that is active.

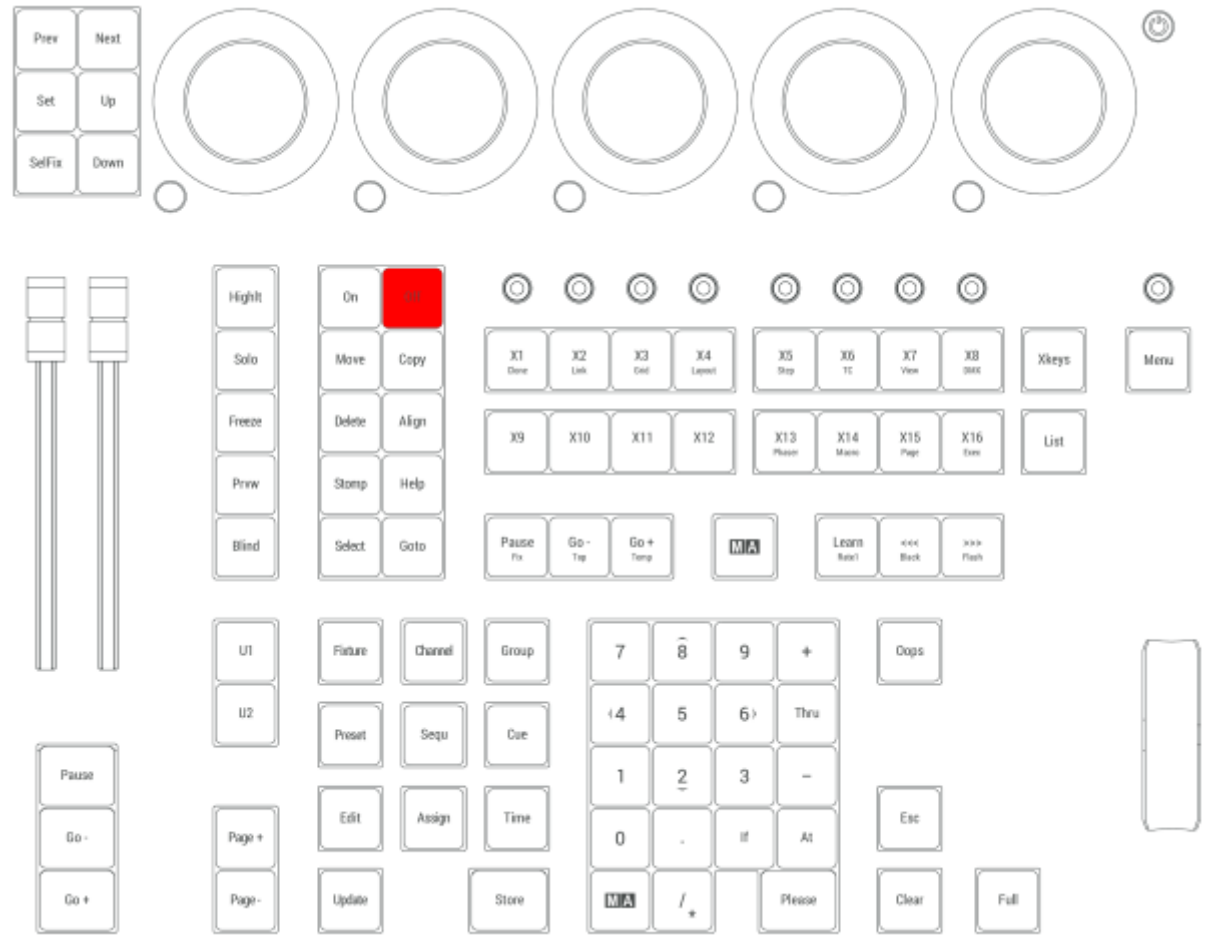
For more information about RemoteHID connections, see the **RemoteHID**.

Location

Off is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.40. On

Pressing On enters the On keyword into the command line.

For more information about On, see the **On keyword**.

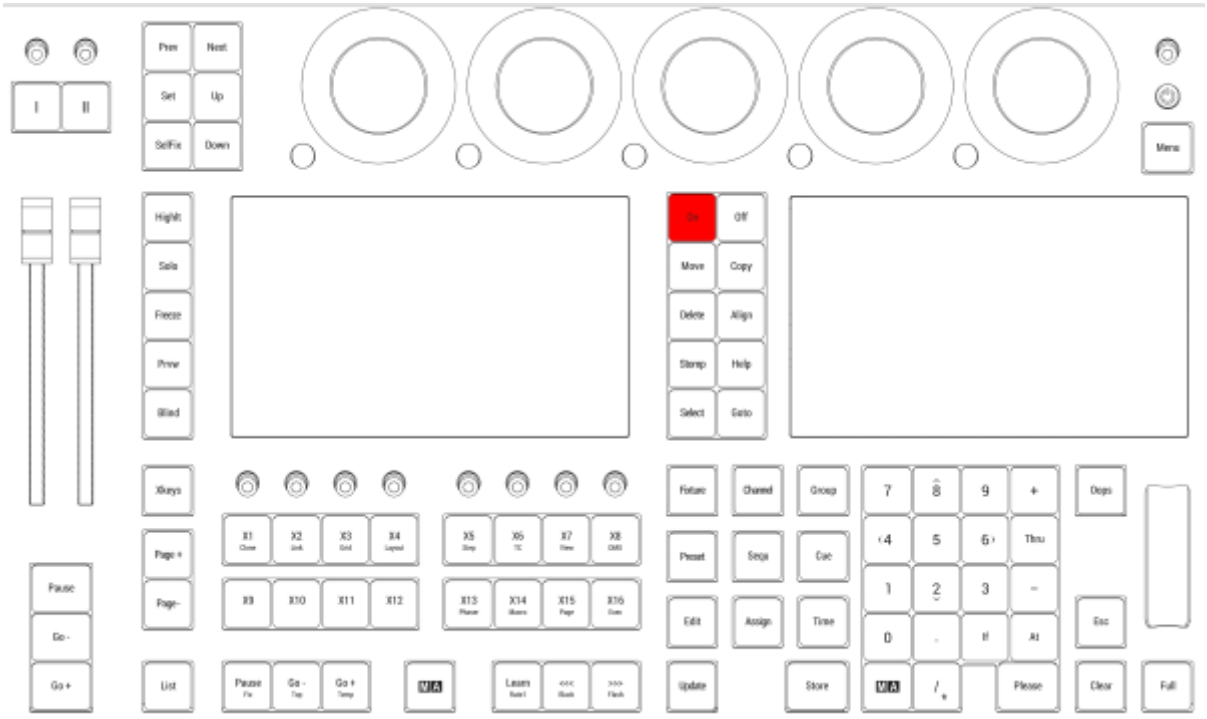
Call

Pressing On On enters the Call keyword into the command line.

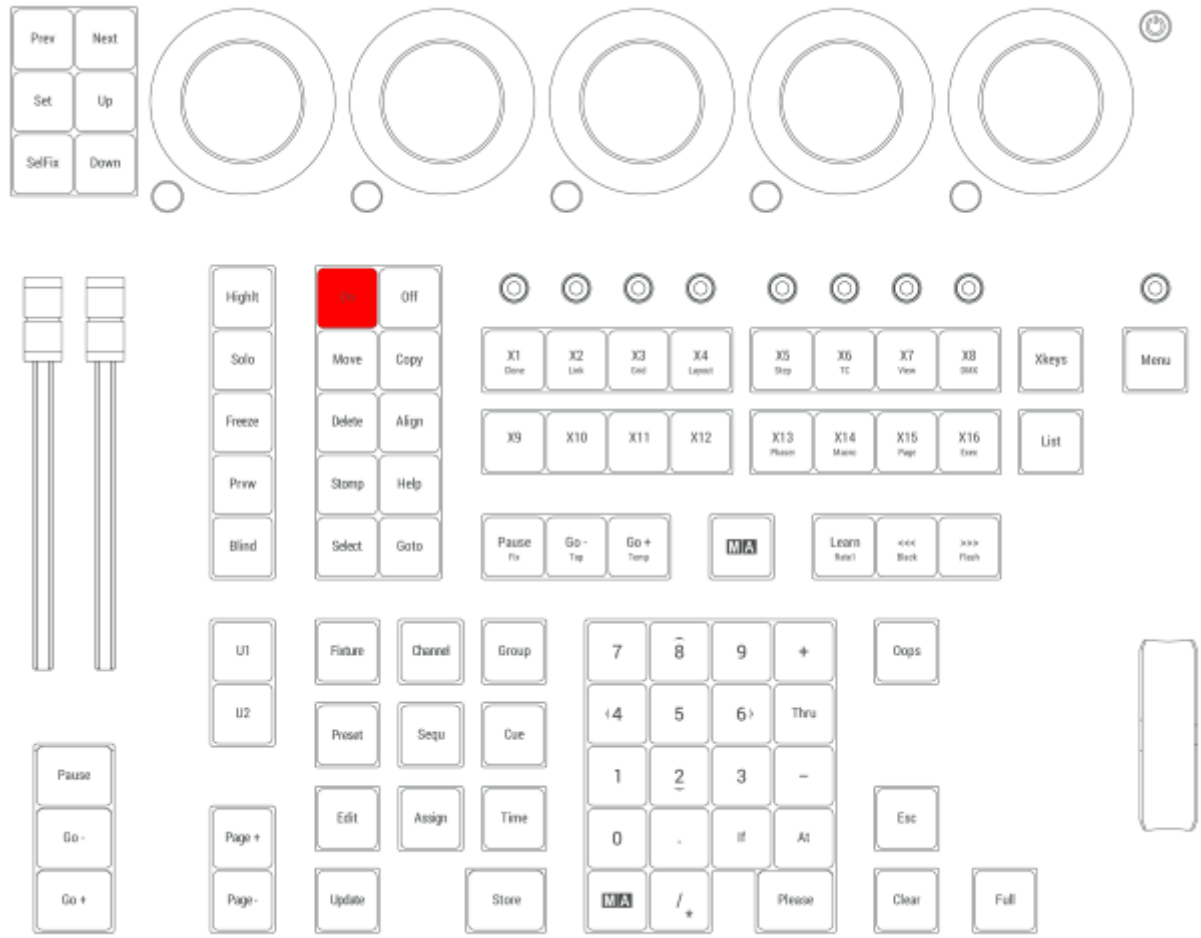
For more information about Call, see the **Call keyword**.

Location

On is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.41. Oops

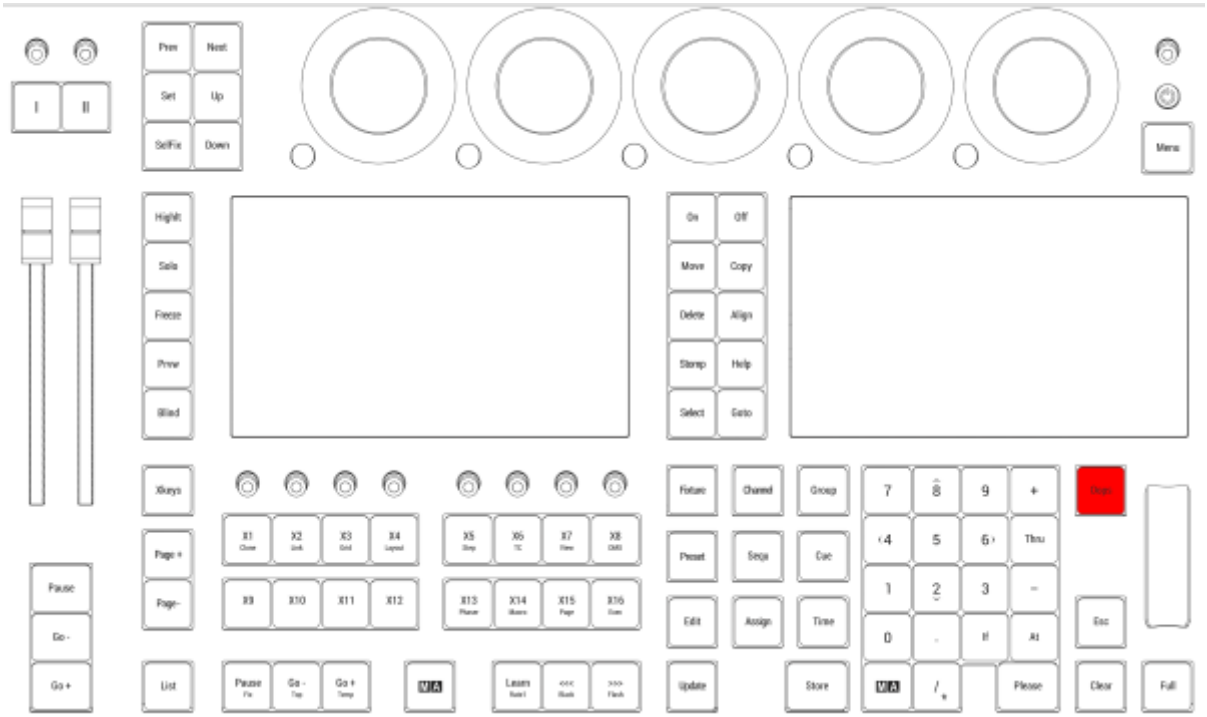
Pressing Oops executes the Oops command.

```
OK
:   Oops
```

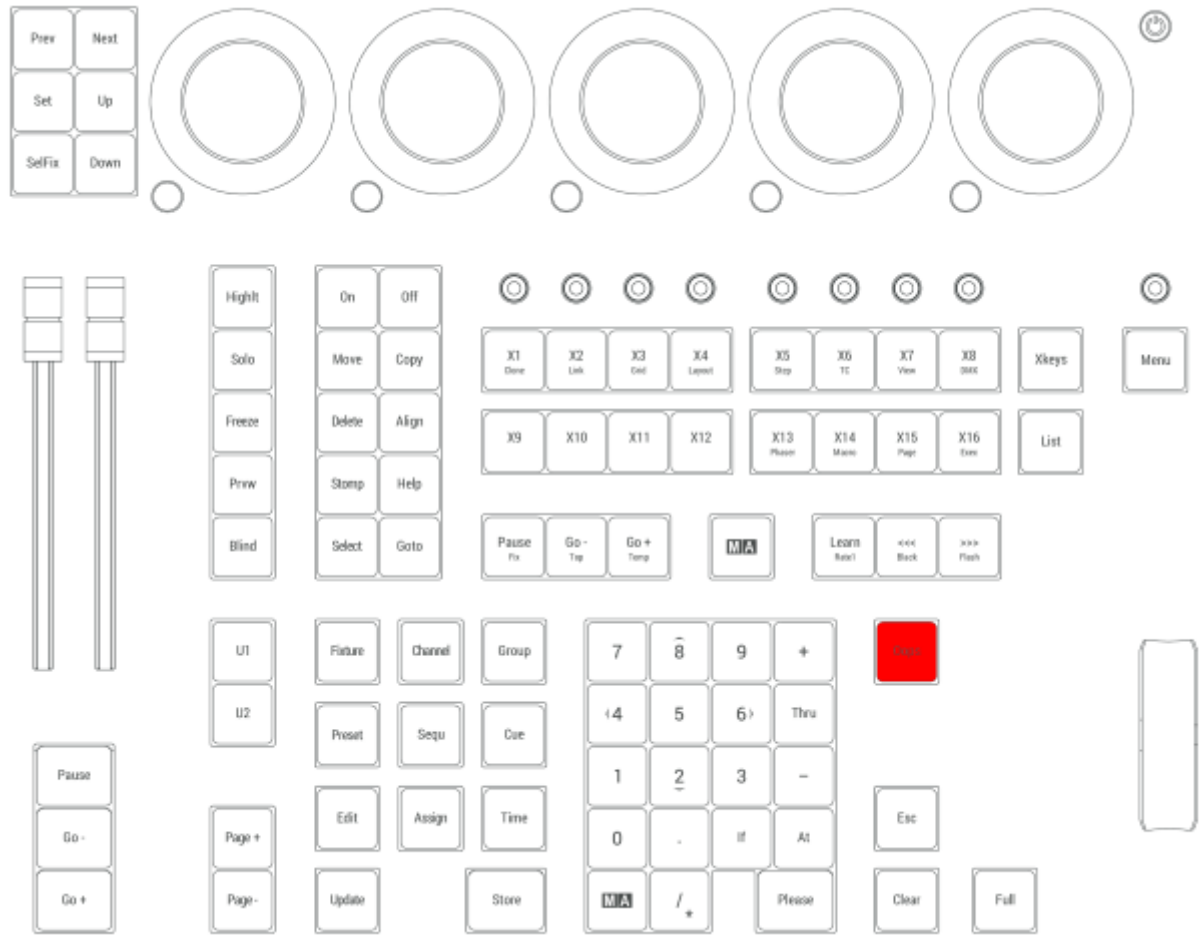
For more information about Oops, see the **Oops keyword**.

Location

Oops is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.42. Page-

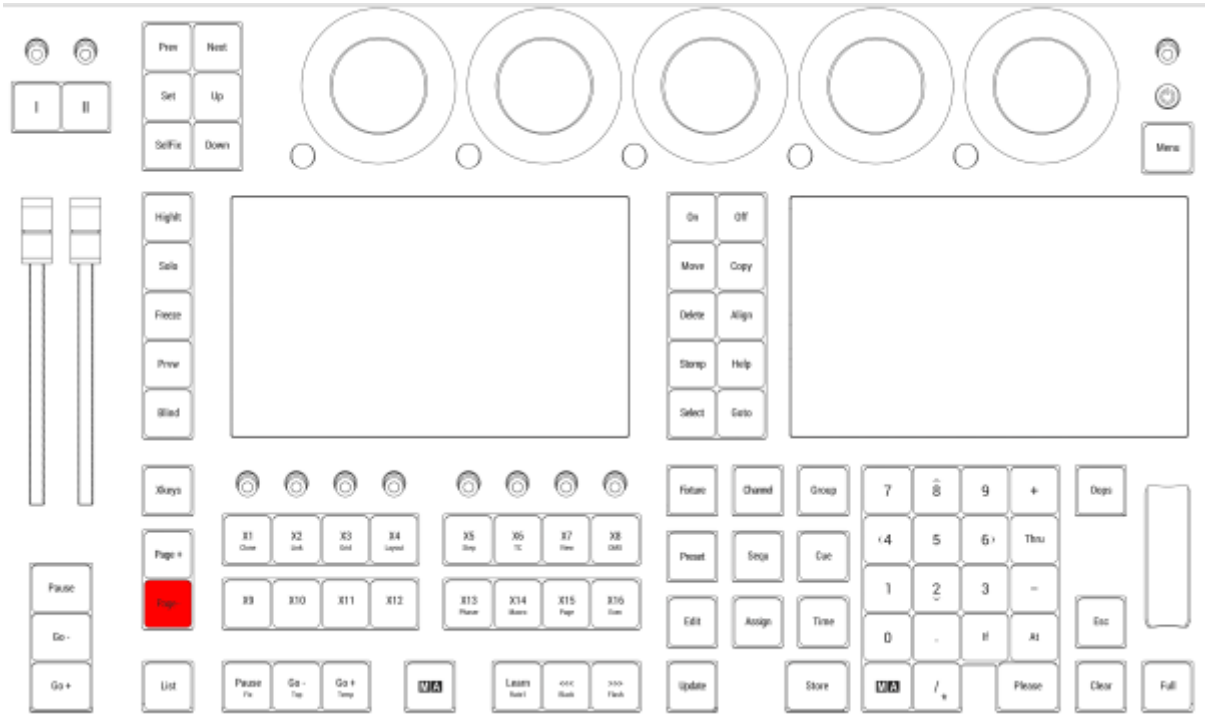
Pressing Page- executes the command **Previous Page** in the command line.

```
OK
:  Previous Page
```

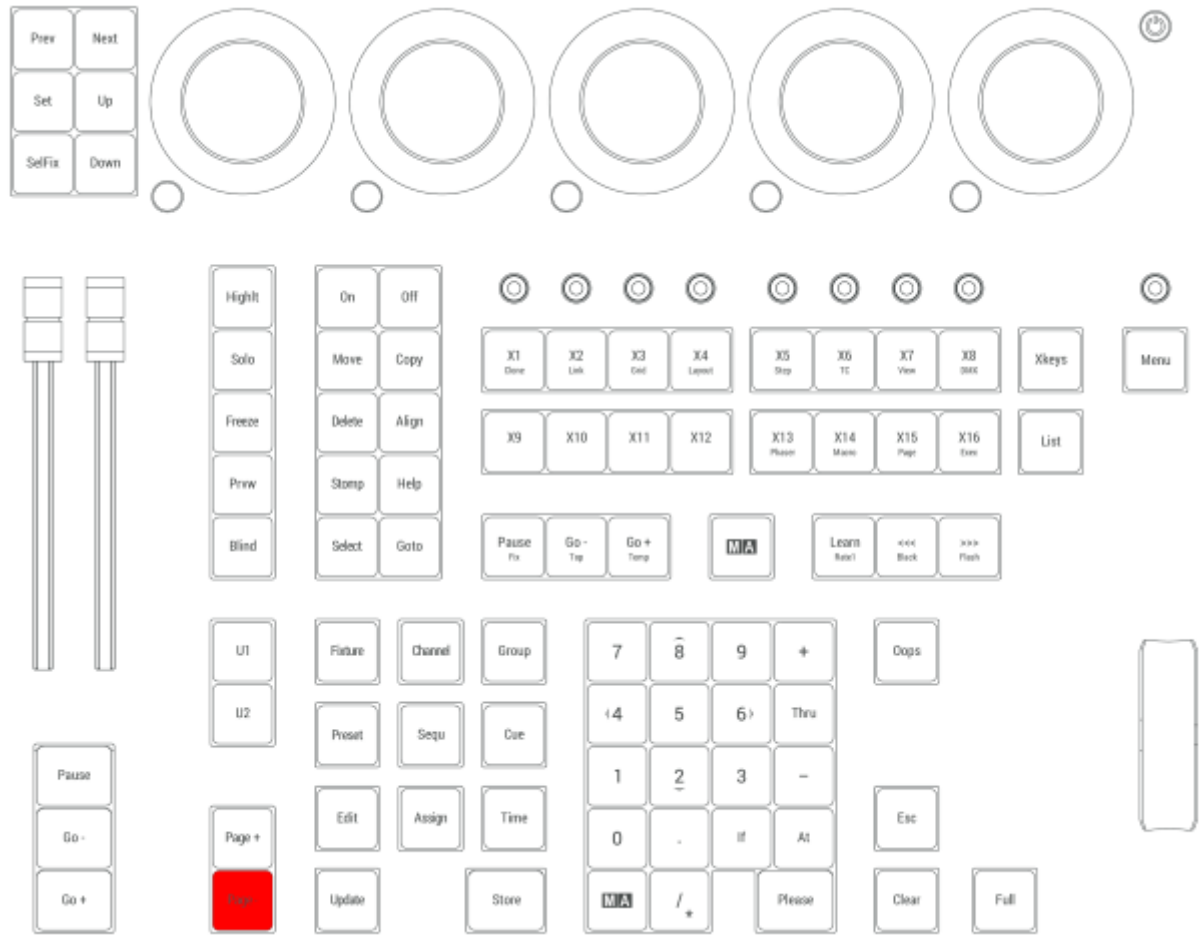
Pressing and holding Page- for over a 1 second sets the page to page 1.

Location

Page- is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

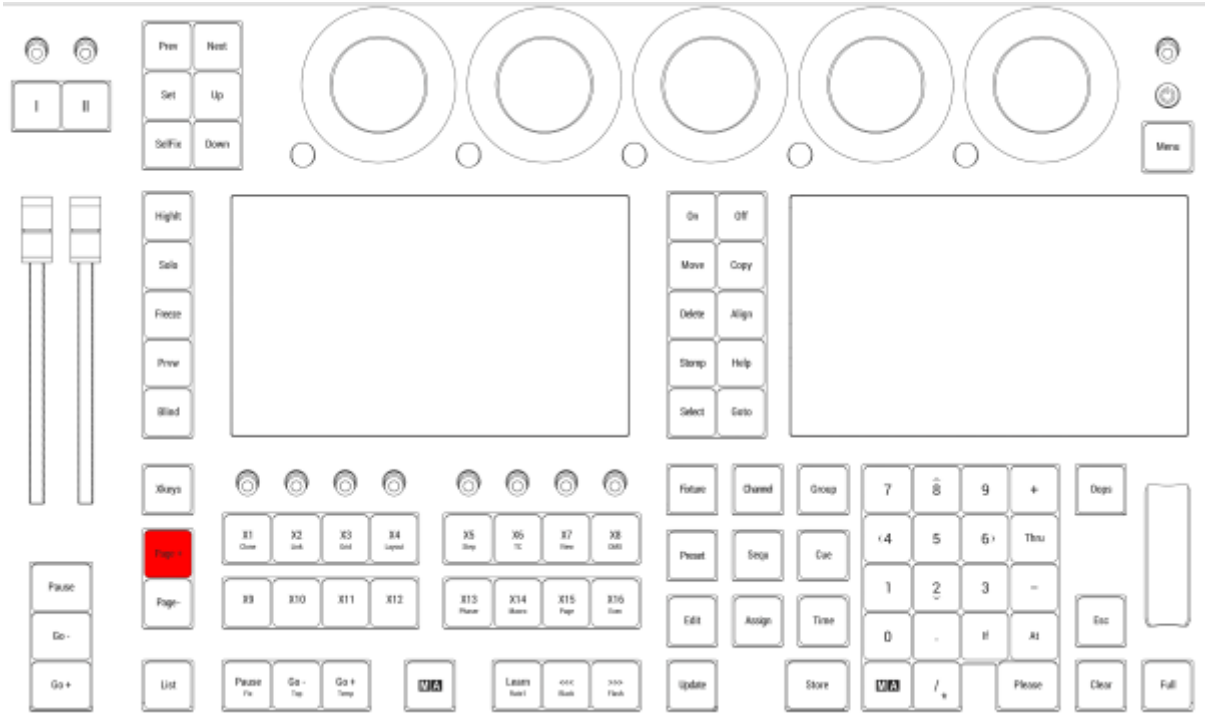
1.3.15.43. Page+

Pressing Page+ executes the command **Next Page** in the command line.

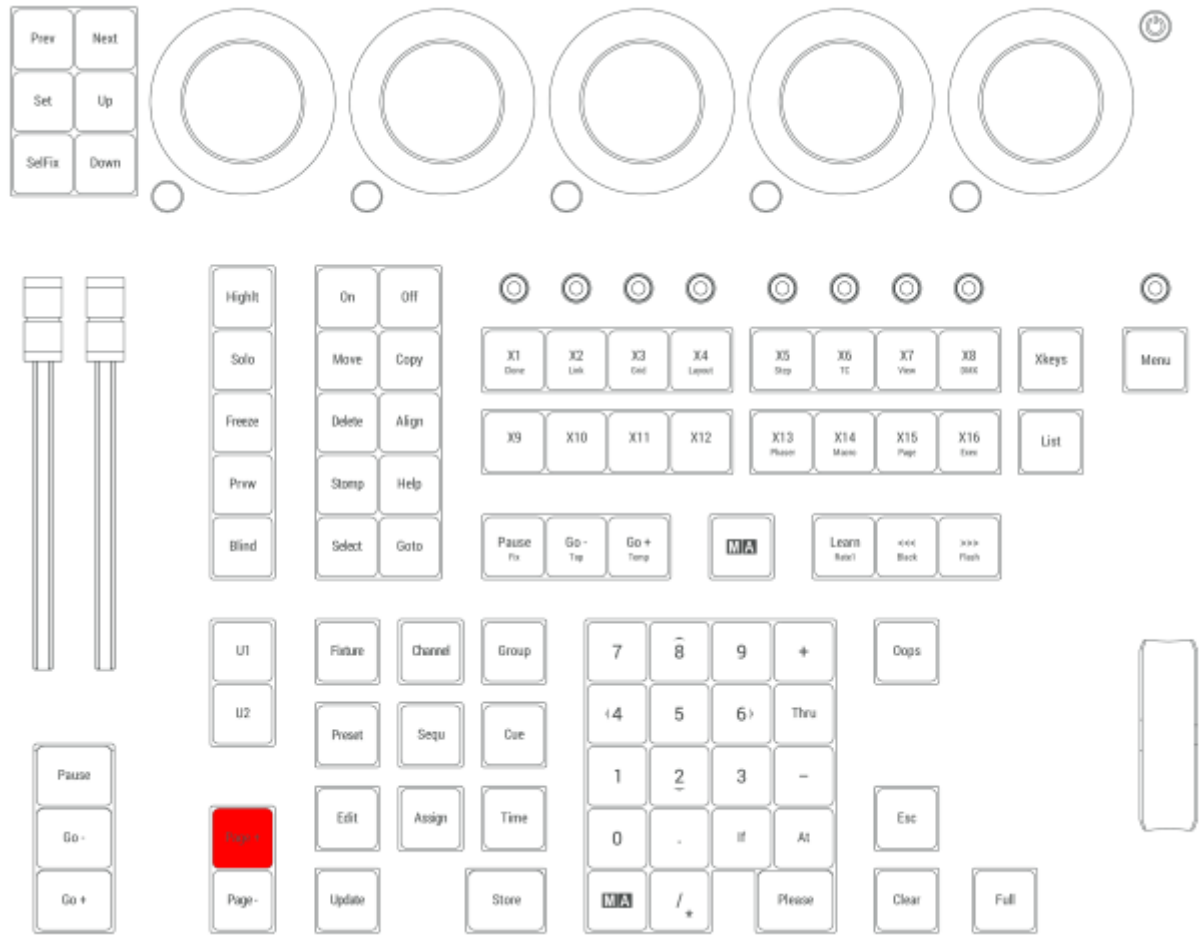
```
OK
:  Next Page
```

Location

Page+ is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

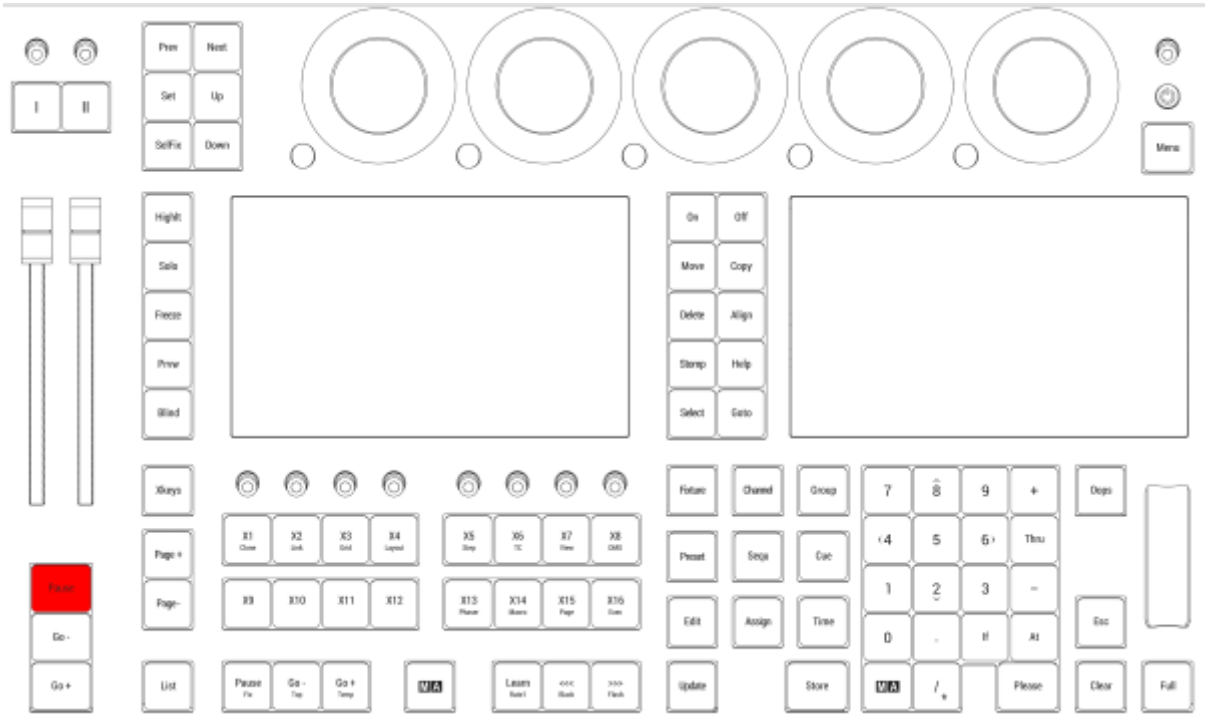
1.3.15.44. Pause [large]

Pressing Pause [large] executes the **Pause Executor** command in the command line

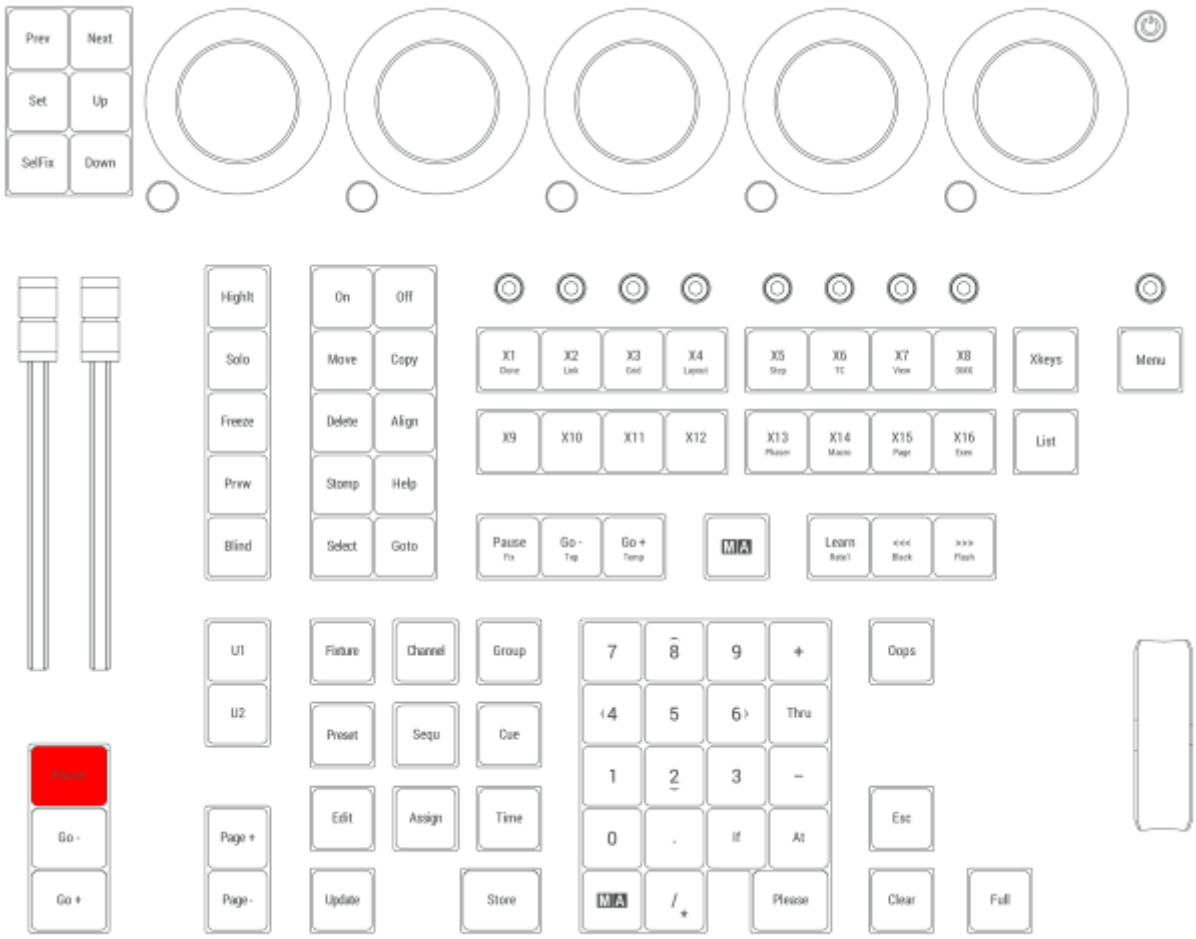
For more information about the Pause, see the **Pause keyword**.

Location

Pause [large] is located in the master section under the two master faders.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.45. Pause | Fix

Pressing Pause enters the Pause keyword into the command line.

For more information about Pause, see the **Pause keyword**.

Pause

Pressing Pause Pause enters the Park keyword into the command line.

For more information about Park, see the **Park keyword**.

Fix

Pressing and holding MA + Pause enters the Fix keyword into the command line.

For more information about Fix, see the **Fix keyword**.

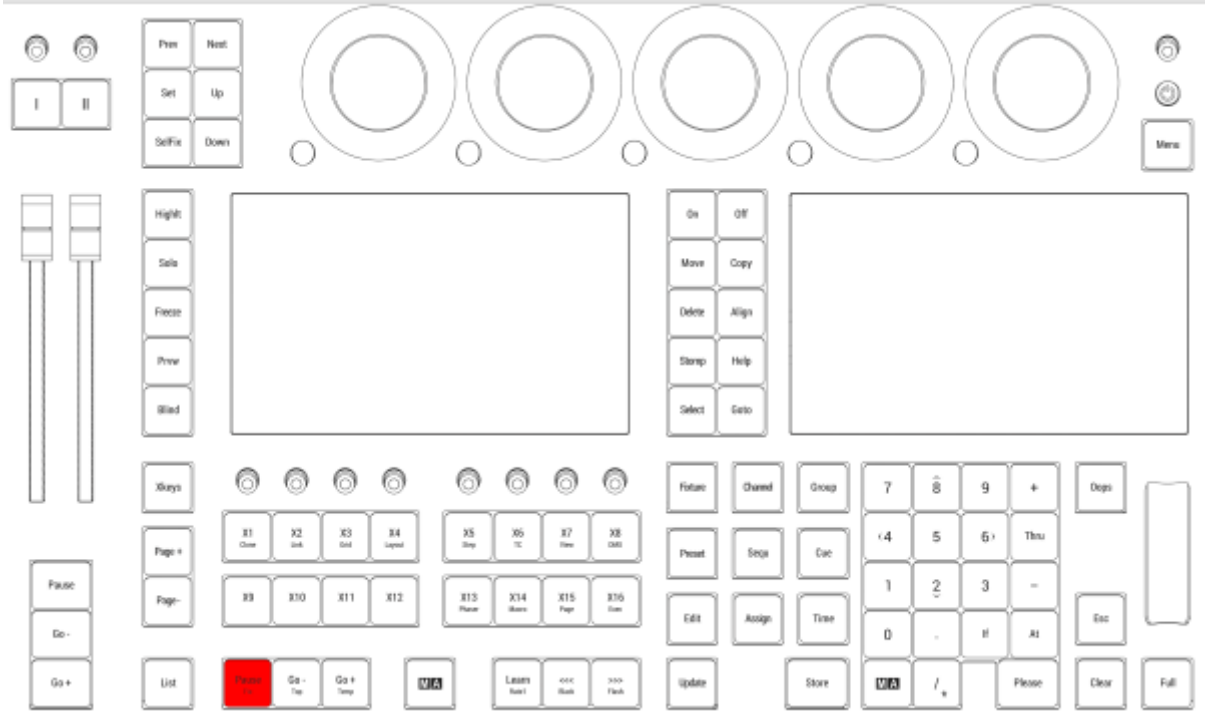
Desk Lock

Pressing and holding MA + MA + Pause toggles the desk lock.

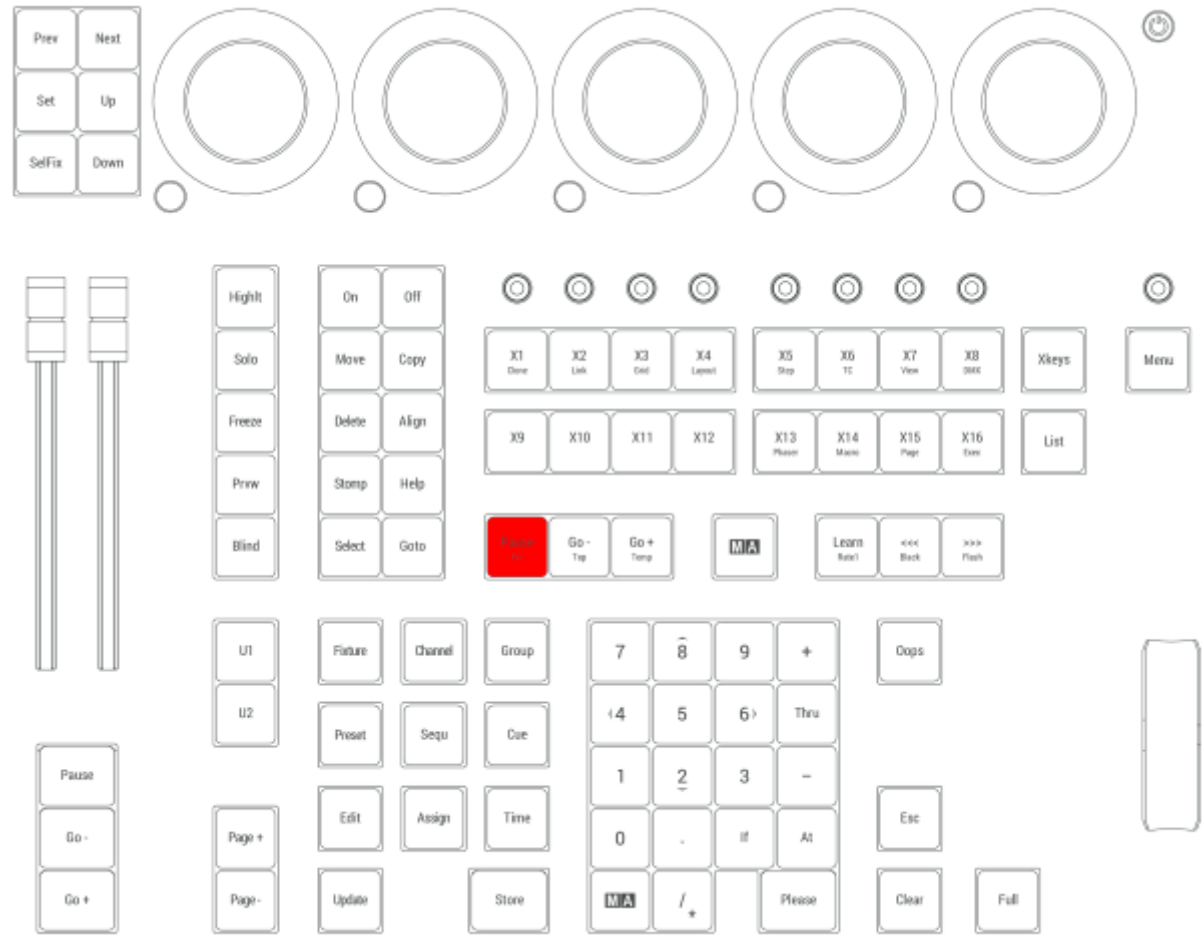
For more information about the desk lock, see **Desk Lock**.

Location

Pause is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.46. Please

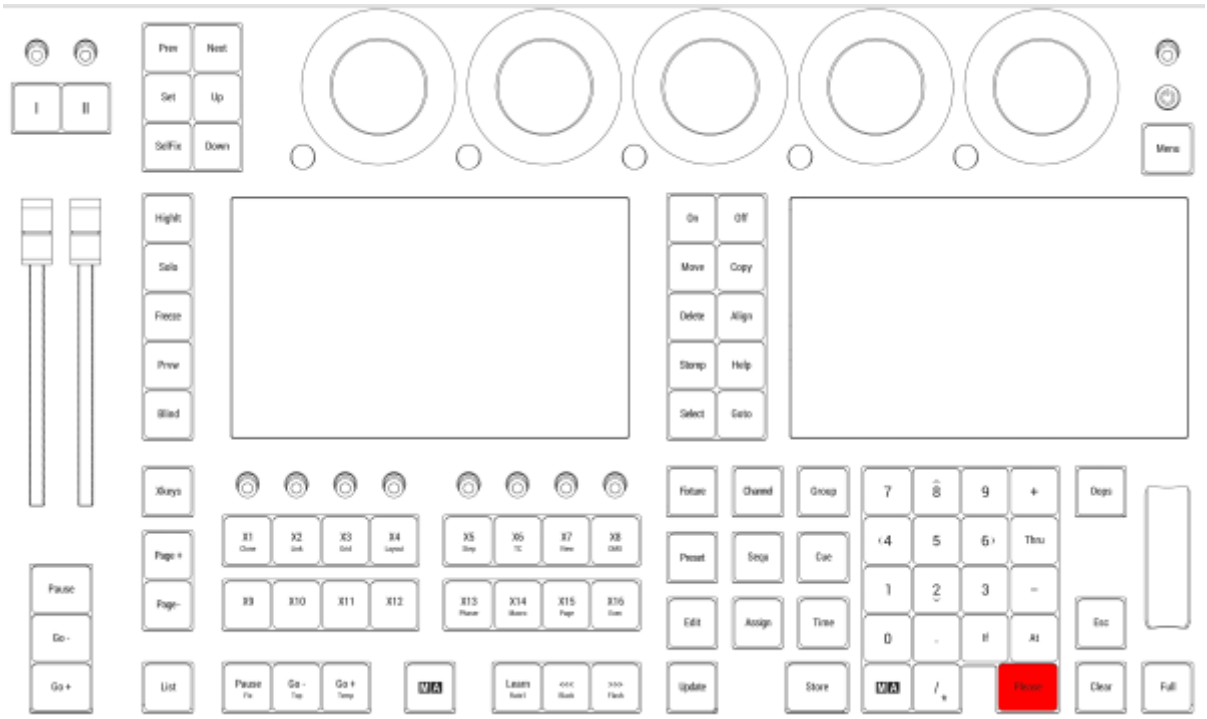
Pressing Please executes a command. It behaves the same as pressing Enter on an external keyboard.

Activate all Attributes

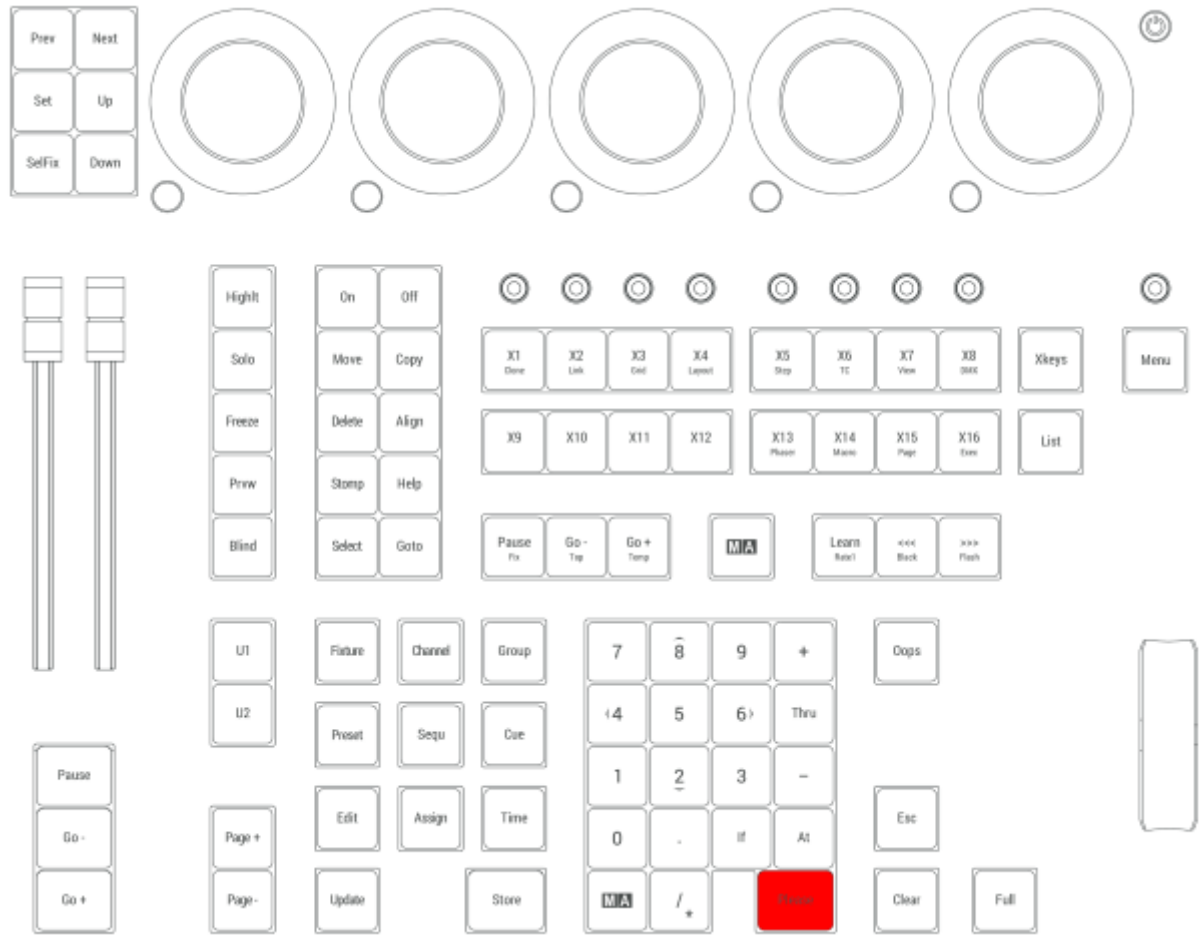
- Pressing Please Please activates all attributes of the selected fixtures.
- Pressing Please once more deactivates all attributes of the selected fixtures.

Location

Please is located in the numeric keys section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

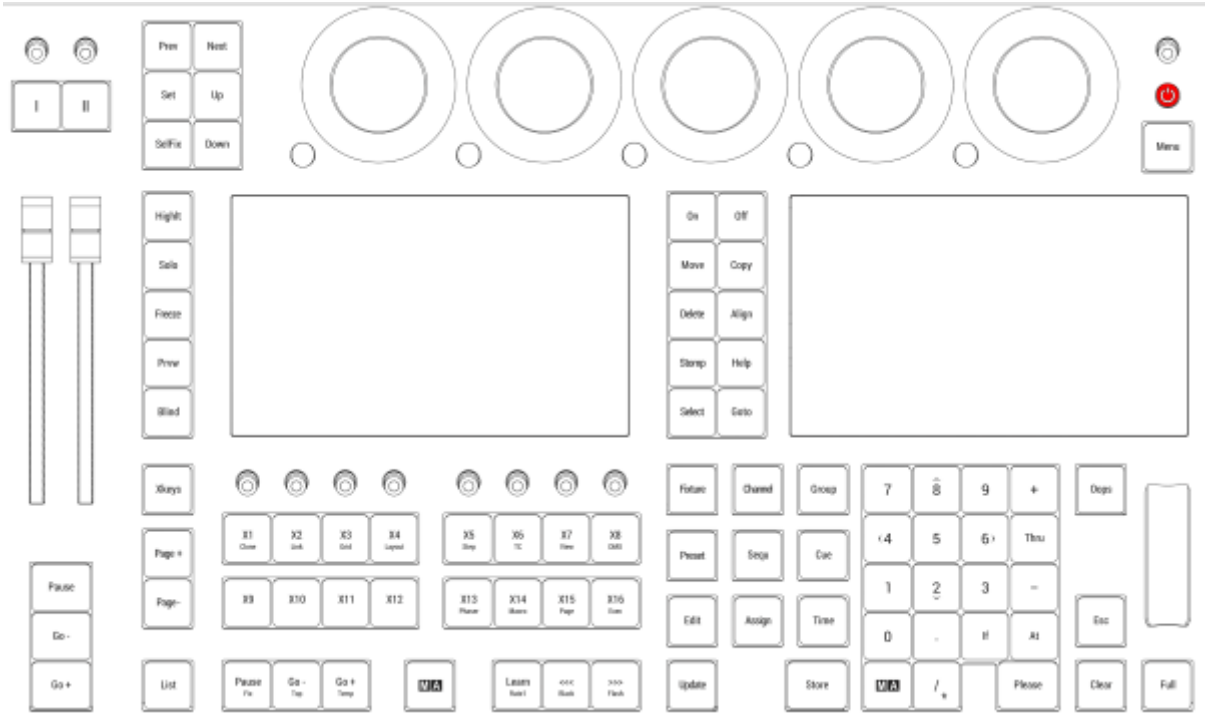
1.3.15.47. Power

Pressing Power boots up the console or shuts it down.

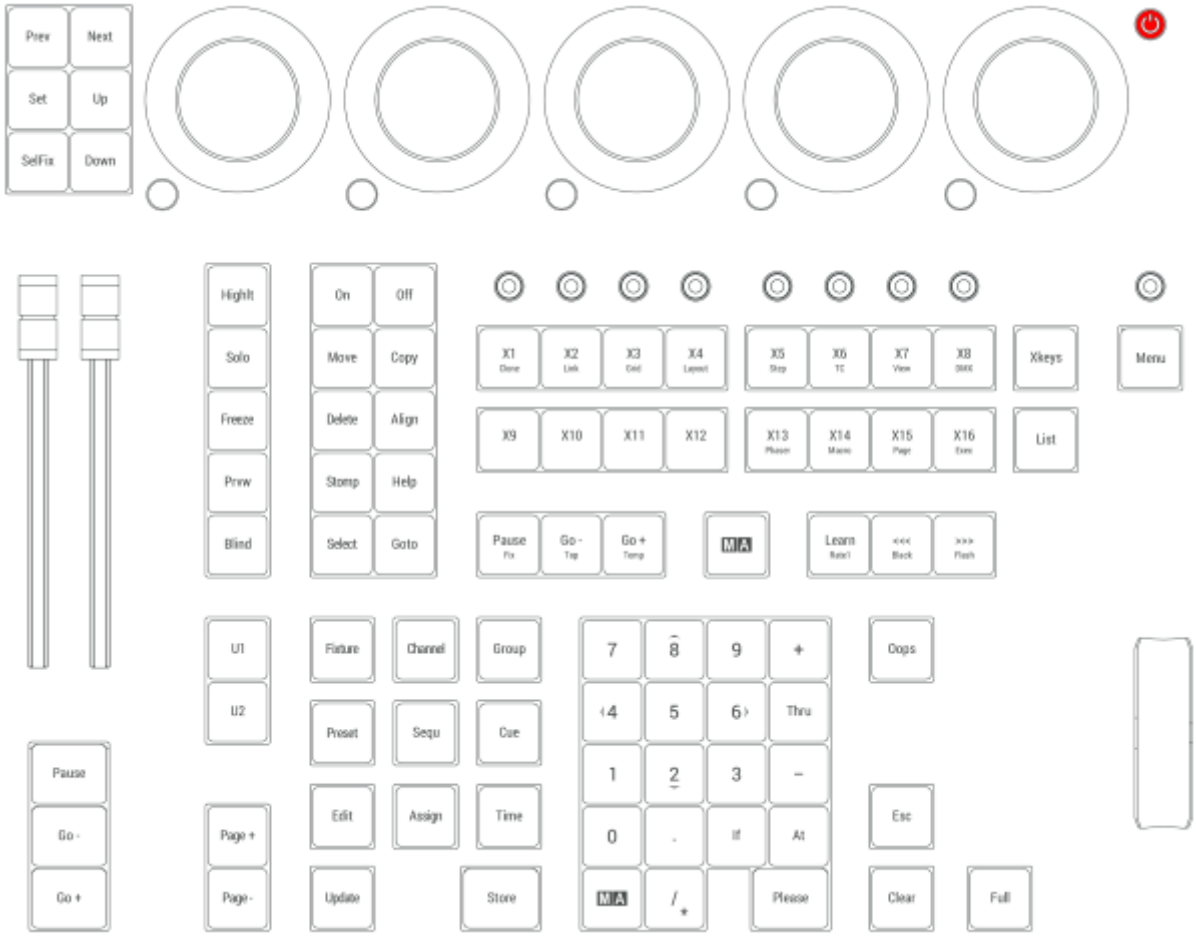
	Important:
	To switch the console on or off, use the power switch on the rear panel.

Location

Power is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.48. Preset

Pressing Preset enters the Preset keyword into the command line.

For more information about Preset, see the **Preset keyword**.

Attribute

Pressing Preset Preset enters the Attribute keyword into the command line.

For more information about Attribute, see the **Attribute keyword**.

Gel

Pressing Preset Preset Preset enters the Gel keyword into the command line.

For more information about Gel, see the **Gel keyword**.

FeatureGroup

Pressing MA + Preset enters the FeatureGroup keyword into the command line.

For more information about FeatureGroup, see the **FeatureGroup keyword**.

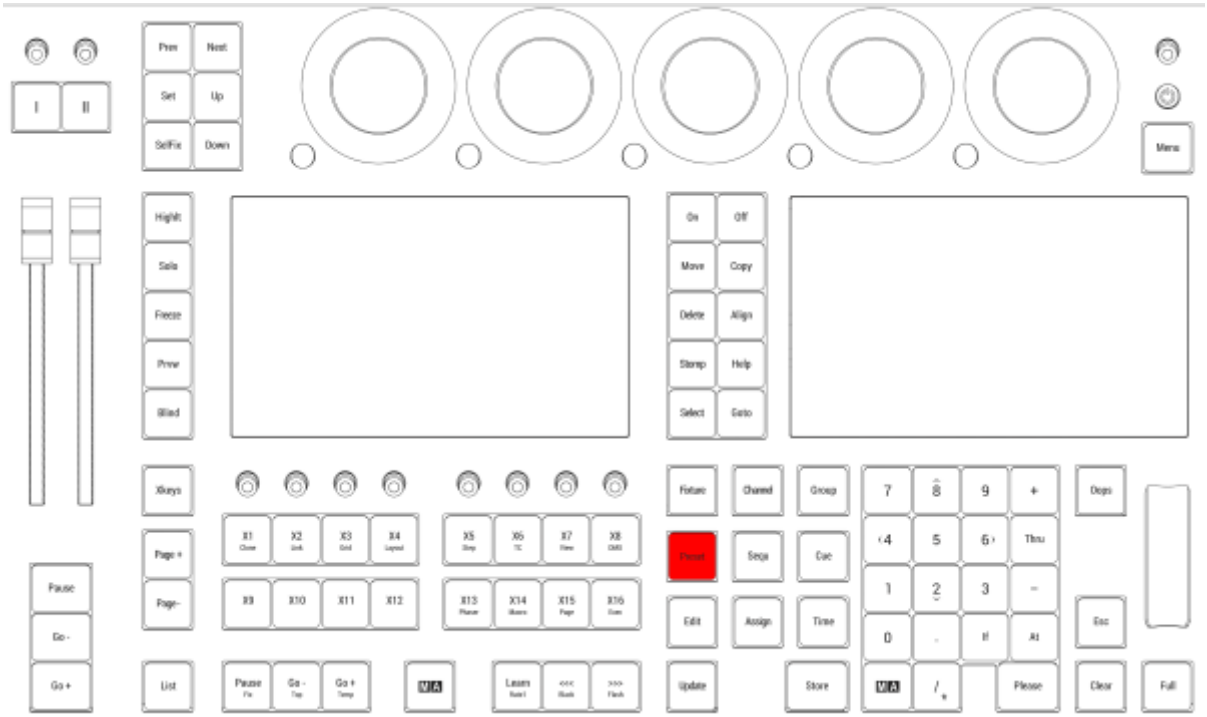
DataPool

Pressing MA + Preset + Preset enters the DataPool keyword into the command line.

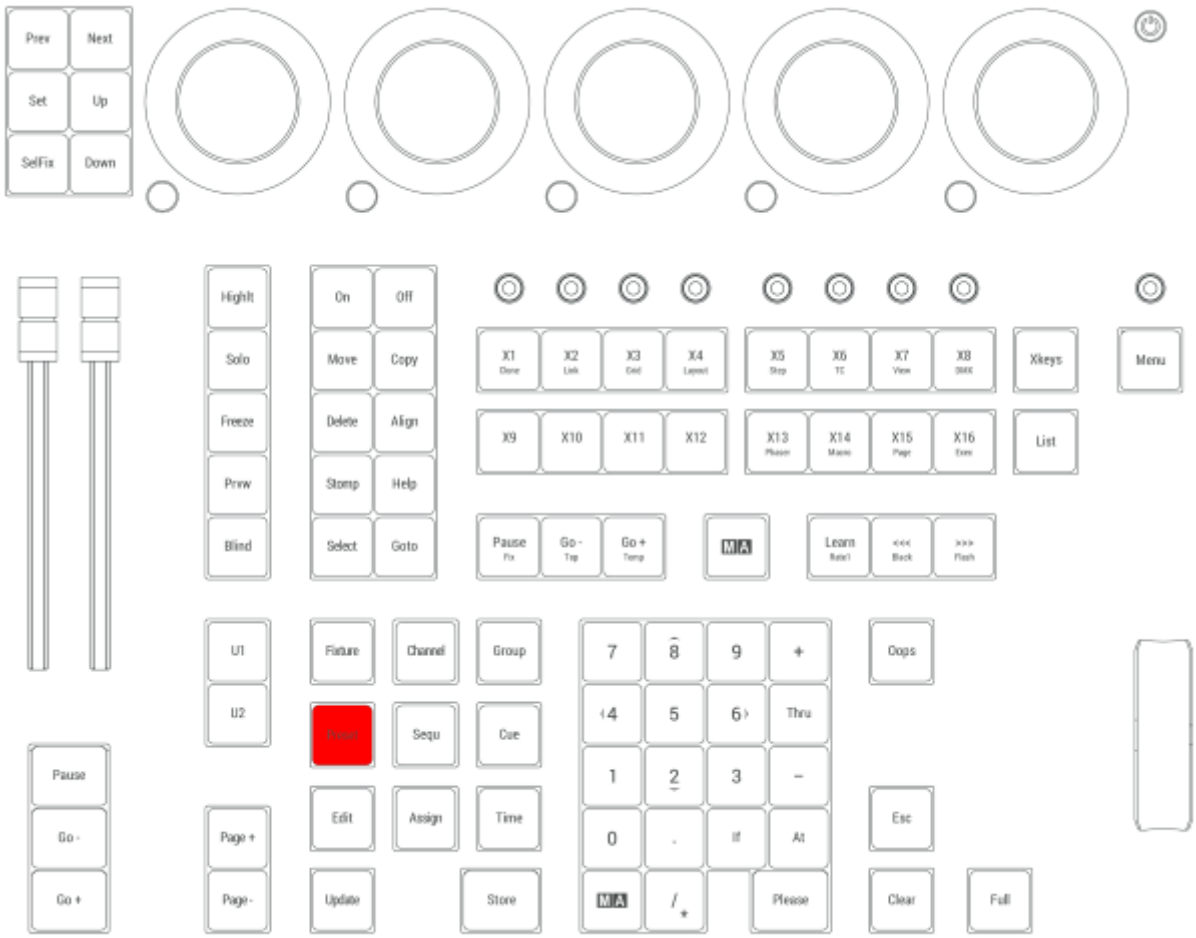
For more information about DataPool, see the **DataPool keyword**.

Location

Preset is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.49. Prev [Previous]

Pressing Prev executes the Previous keyword in the command line.

For more information about Previous, see the **Previous keyword**.

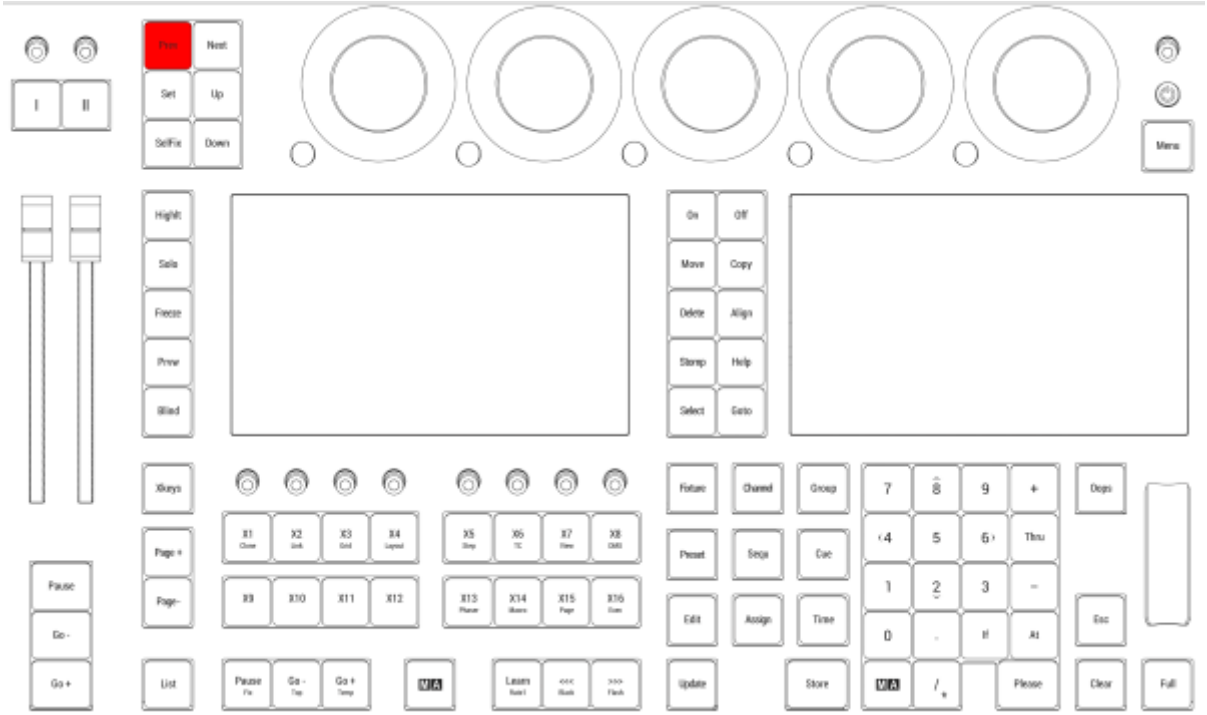
Previous Step

Pressing MA + Prev executes the **Previous Step** command in the command line.

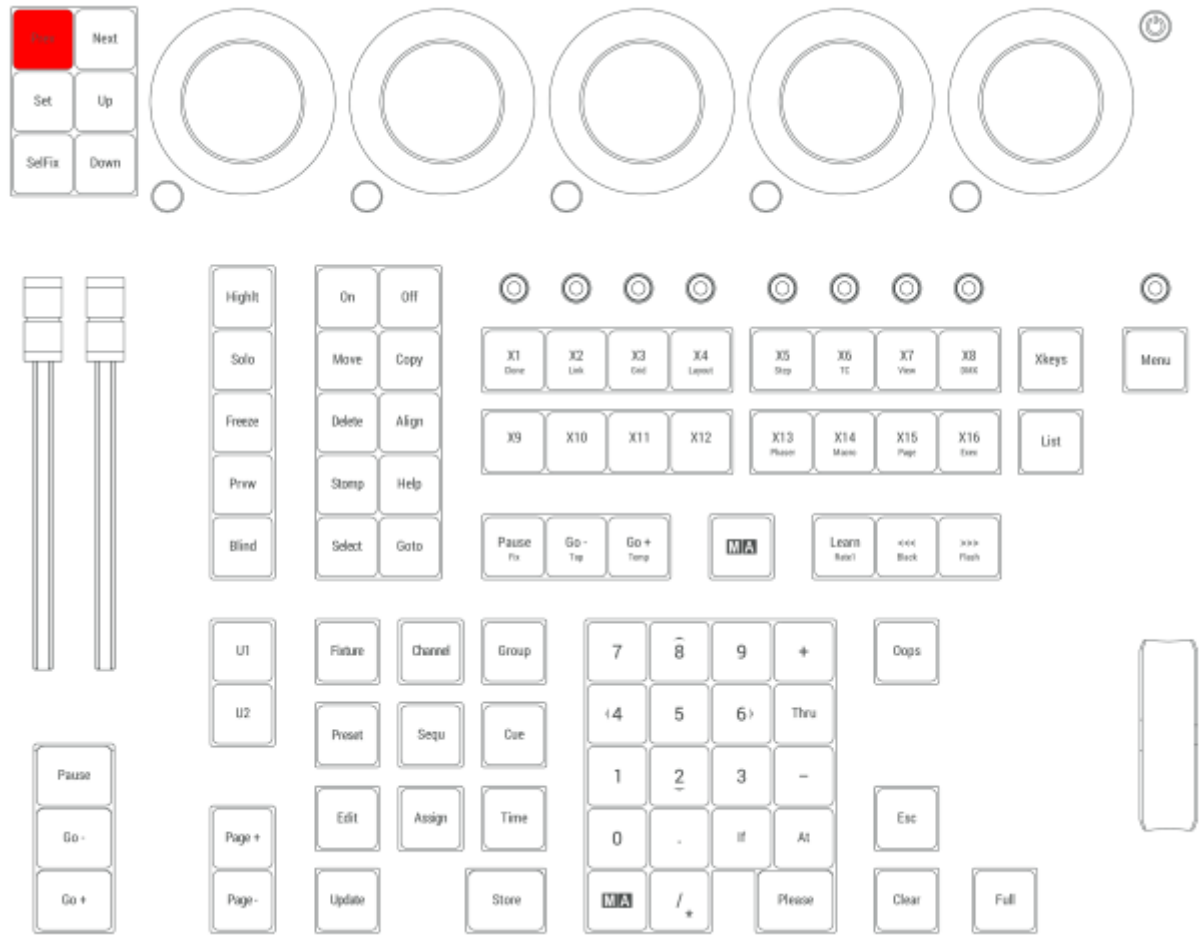
For more information about Previous Step, see the **Previous keyword** and the **Step keyword**. For mor information about steps see **Phaser**.

Location

Prev is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

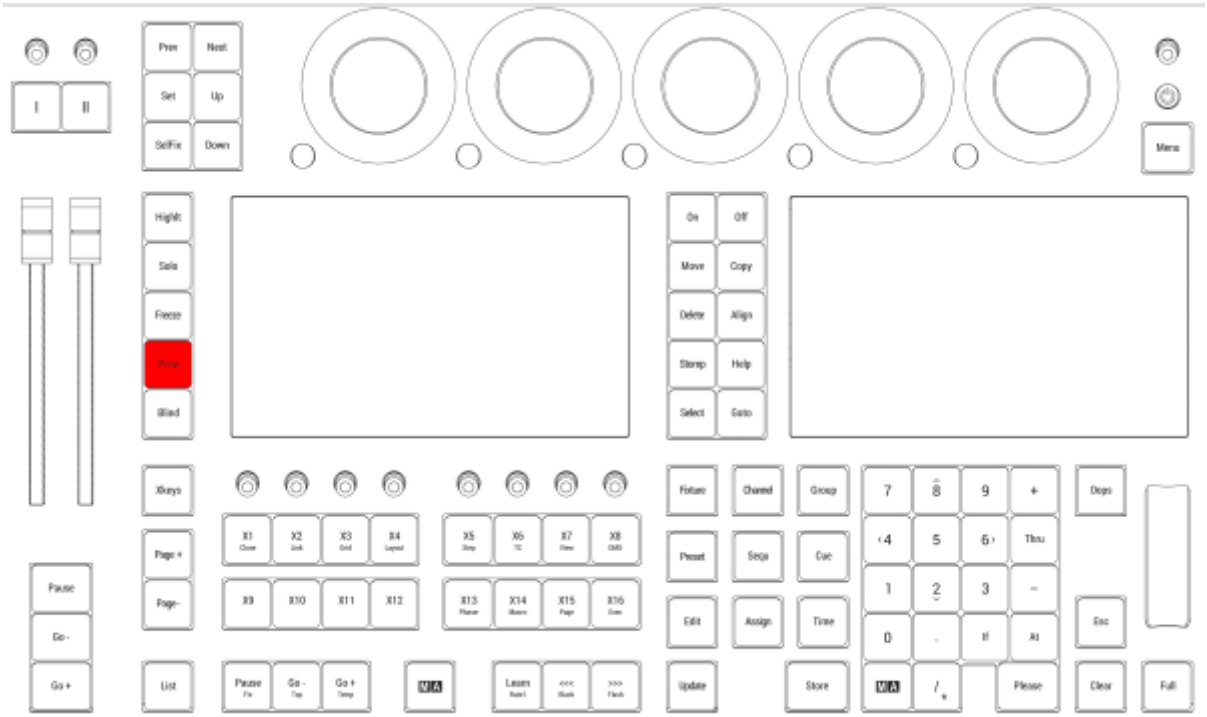
1.3.15.50. Prvw [Preview]

Pressing Prvw executes the the Preview keyword in the command line.

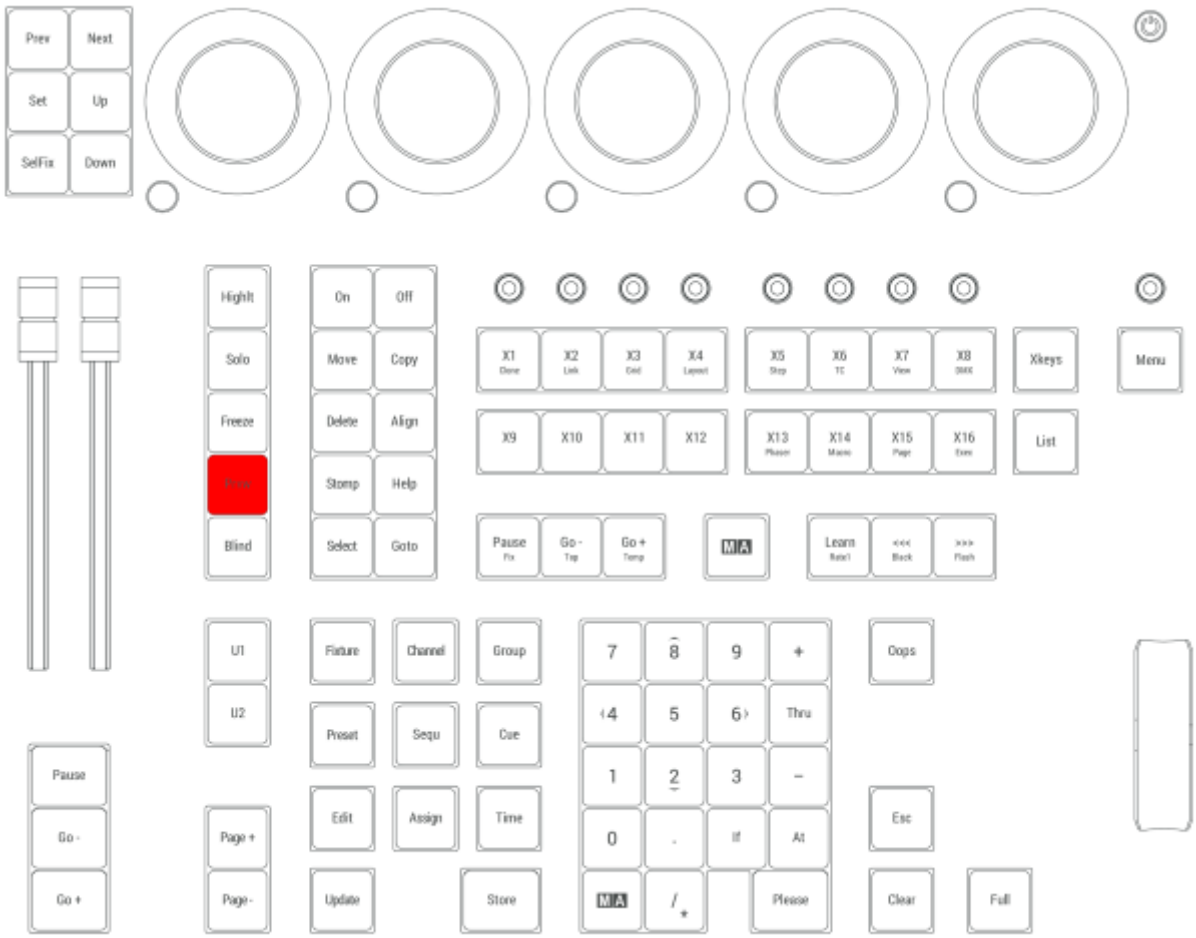
For more information about Prvw, see the **Preview keyword**.

Location

Prvw is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

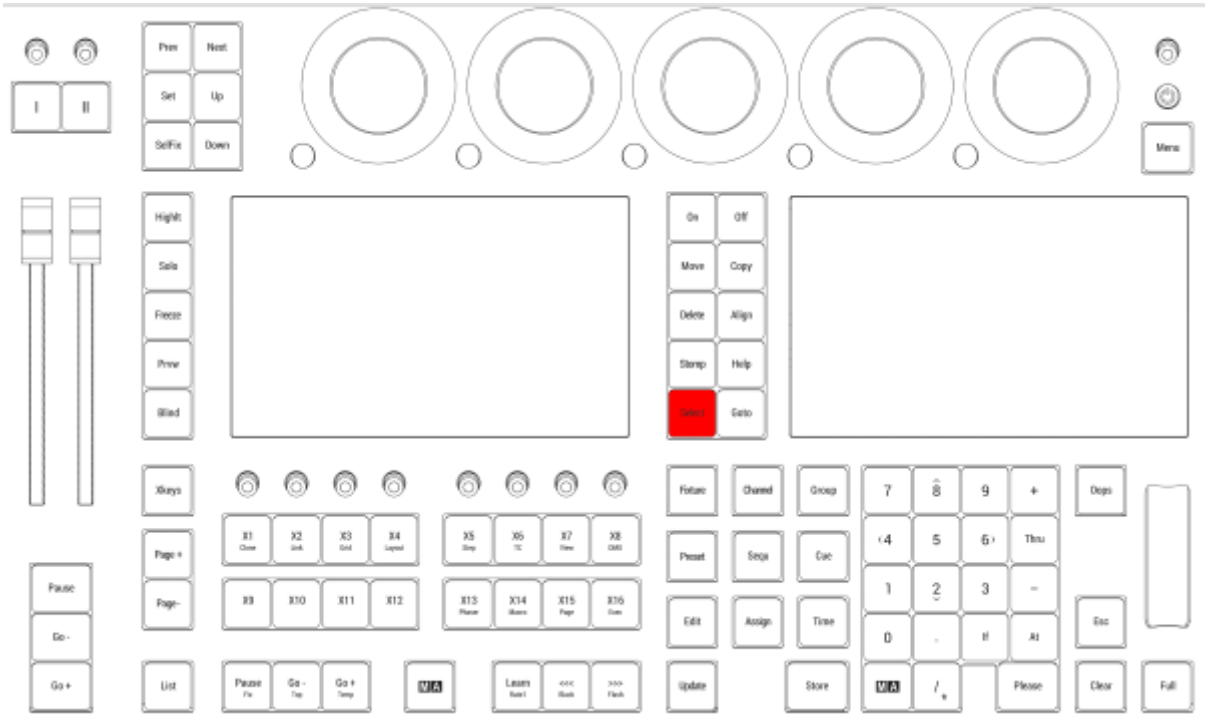
1.3.15.51. Select

Pressing Select enters the Select keyword into the command line.

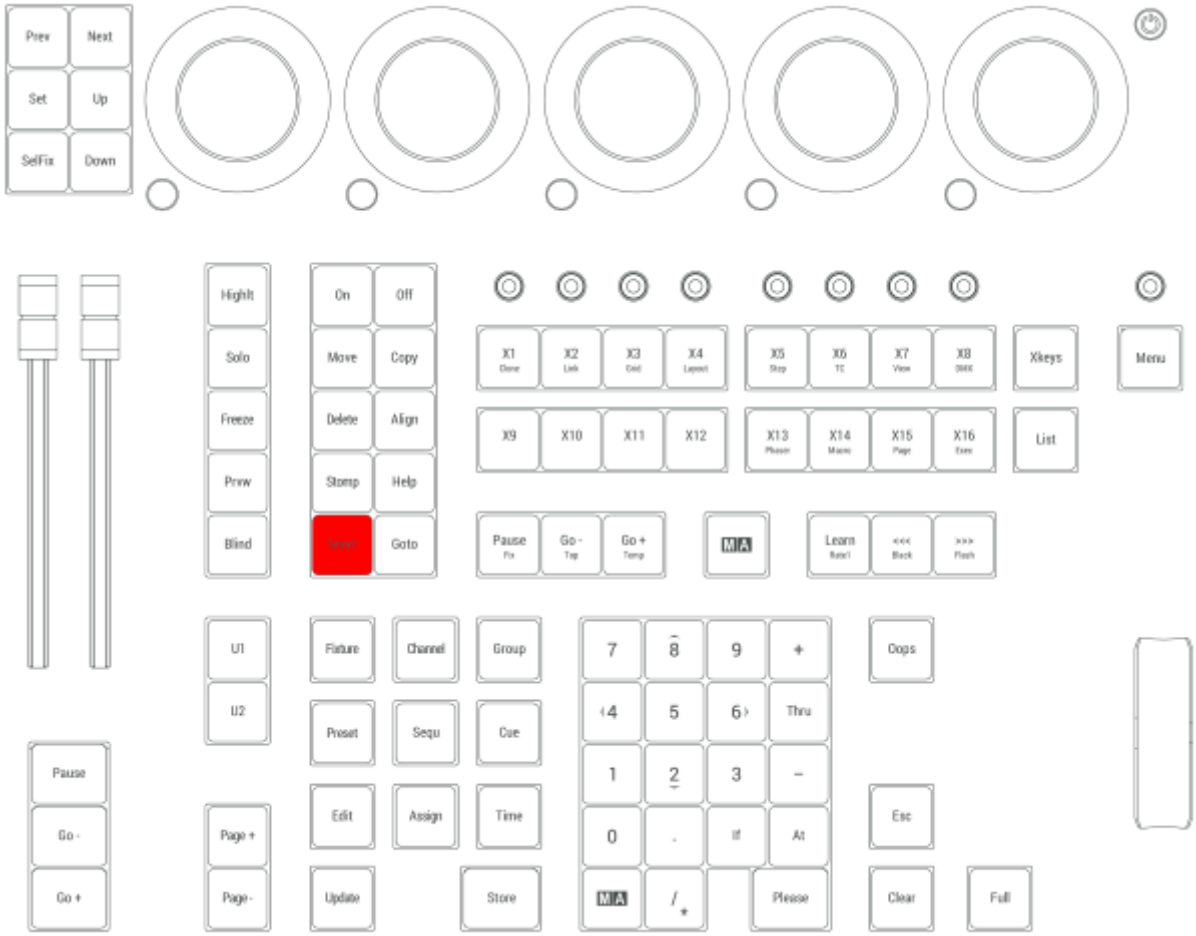
For more information about Select, see the **Select keyword**.

Location

Select is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

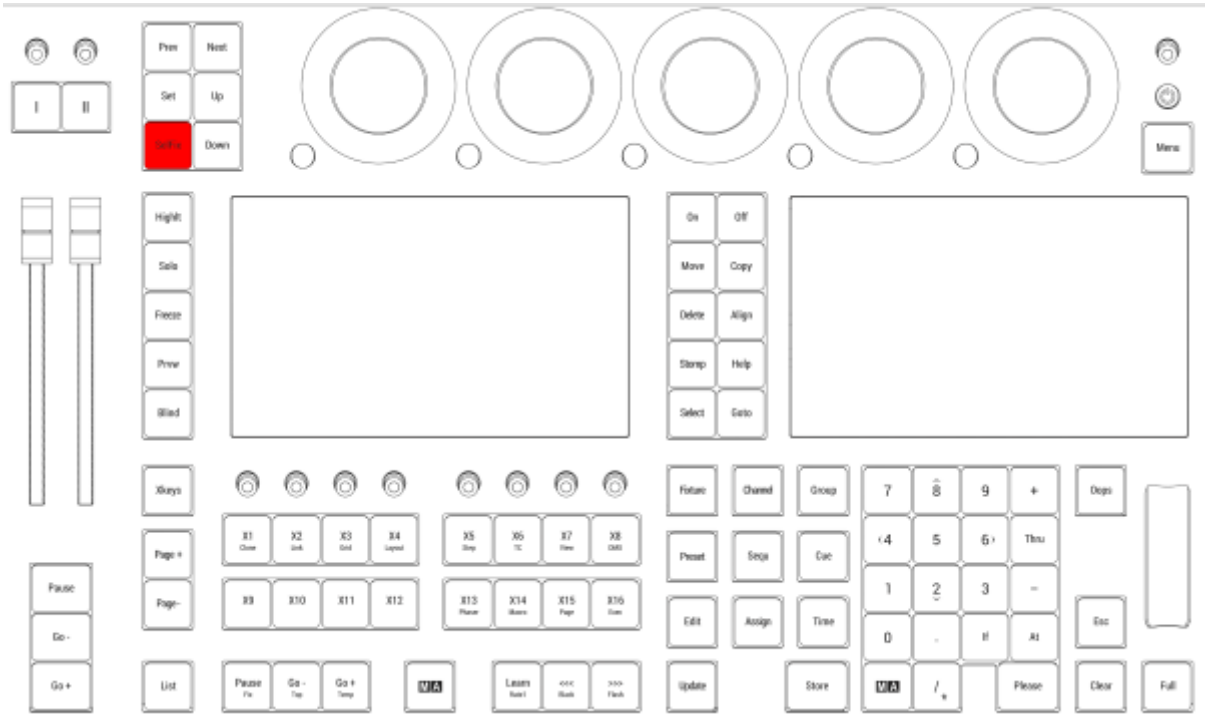
1.3.15.52. SelFix [SelectFixtures]

Pressing SelFix enters the SelectFixtures keyword into the command line.

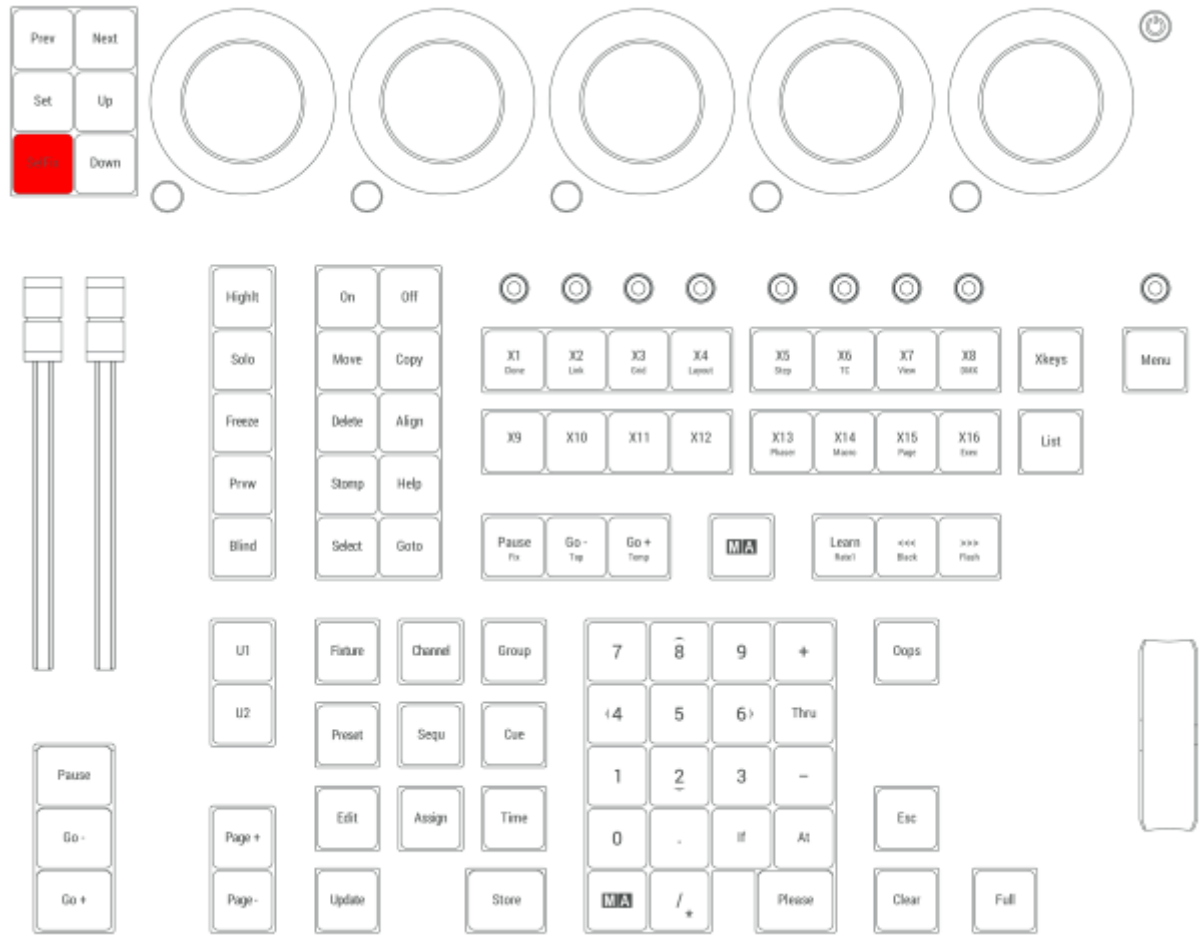
For more information about SelFix, see the **SelectFixtures keyword**.

Location

SelFix is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

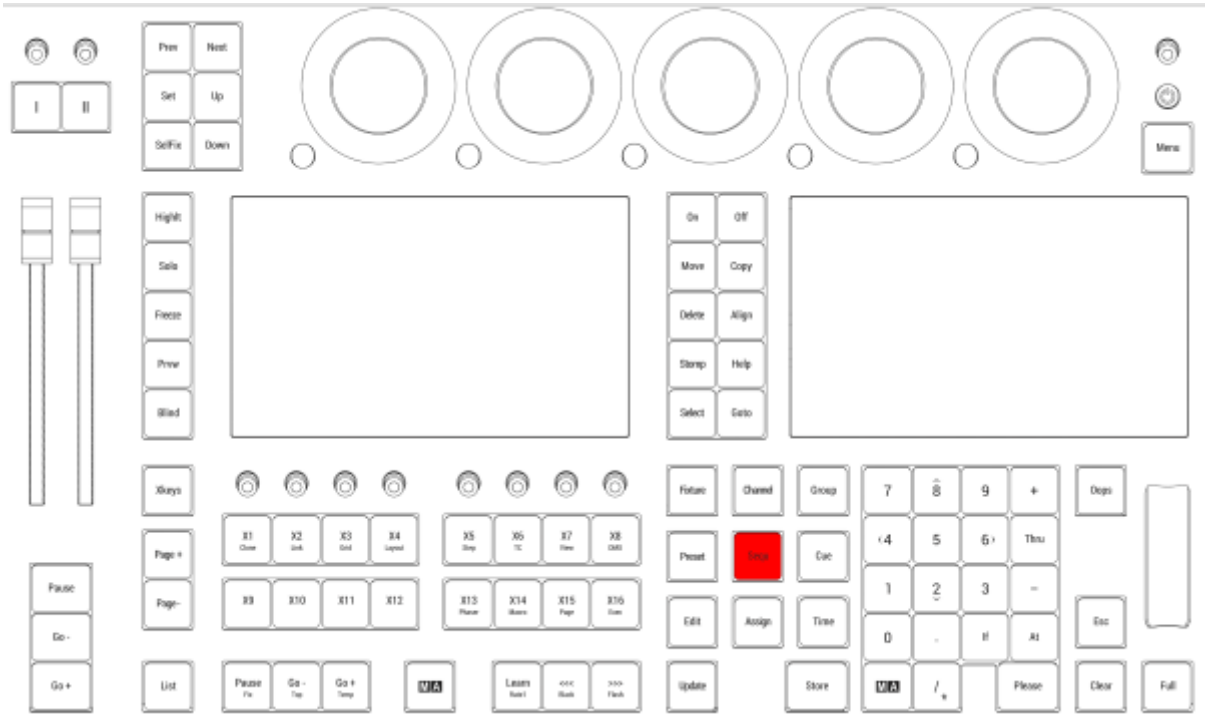
1.3.15.53. Sequ [Sequence]

Pressing Sequ enters the Sequence keyword into the command line.

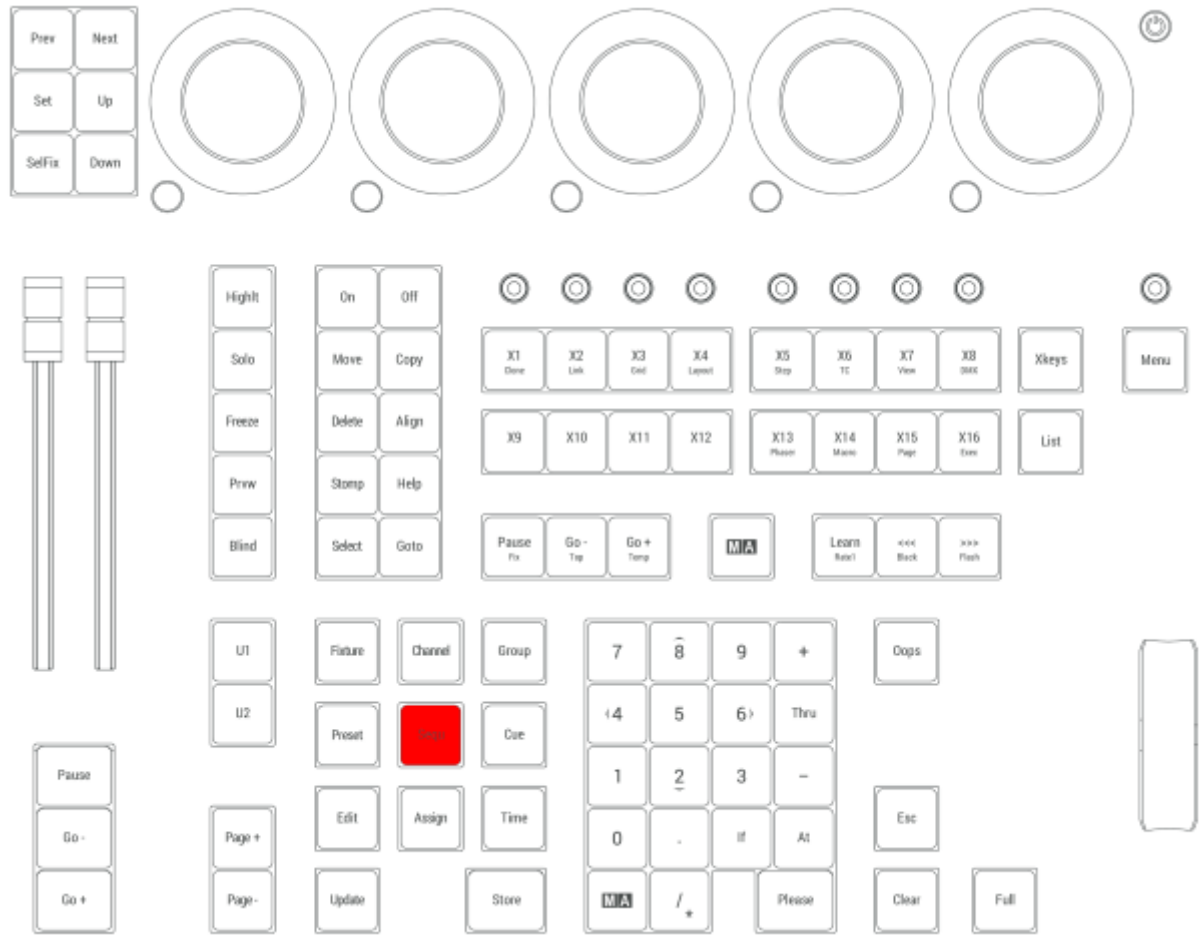
For more information about Sequence, see the **Sequence keyword**.

Location

Sequ is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.54. Set

Pressing Set toggles the activation status of MAtricks in the command line.

```
OK
:   Set Selection Property "Active"
```

For more information about MAtricks, see the **MAtricks keyword**.

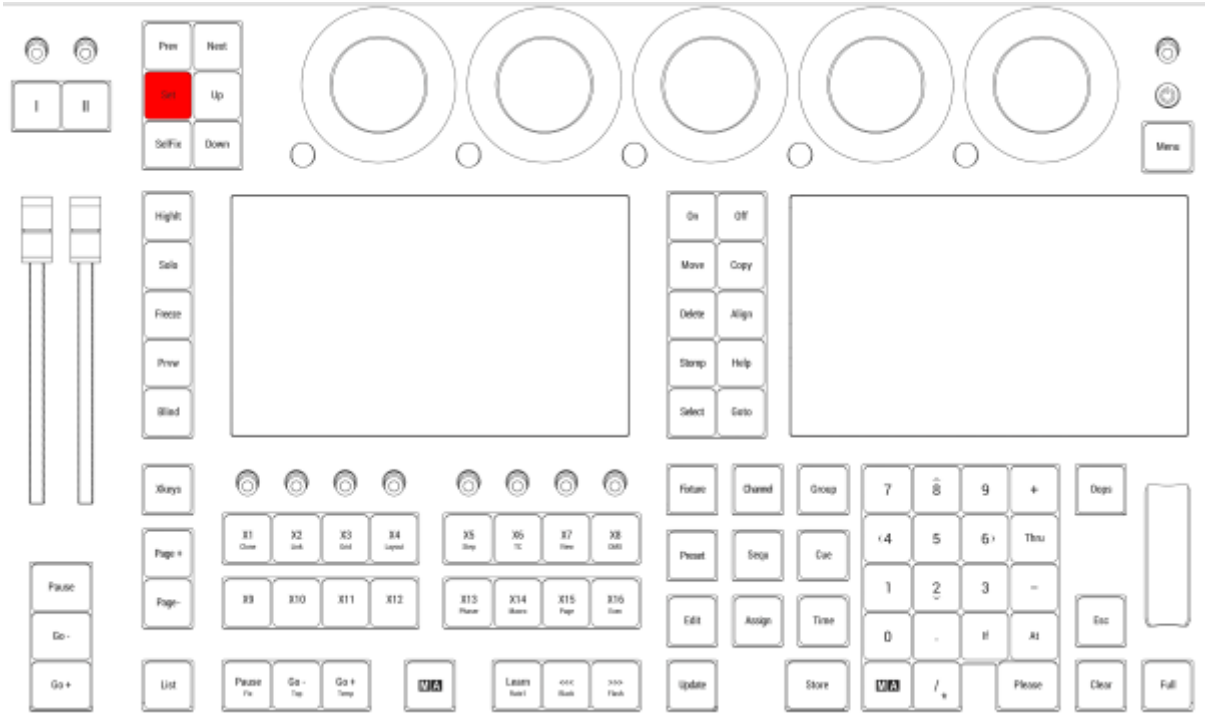
Step Toggle Executor

Pressing MA + Set executes the **Step Toggle Executor** command in the command line.

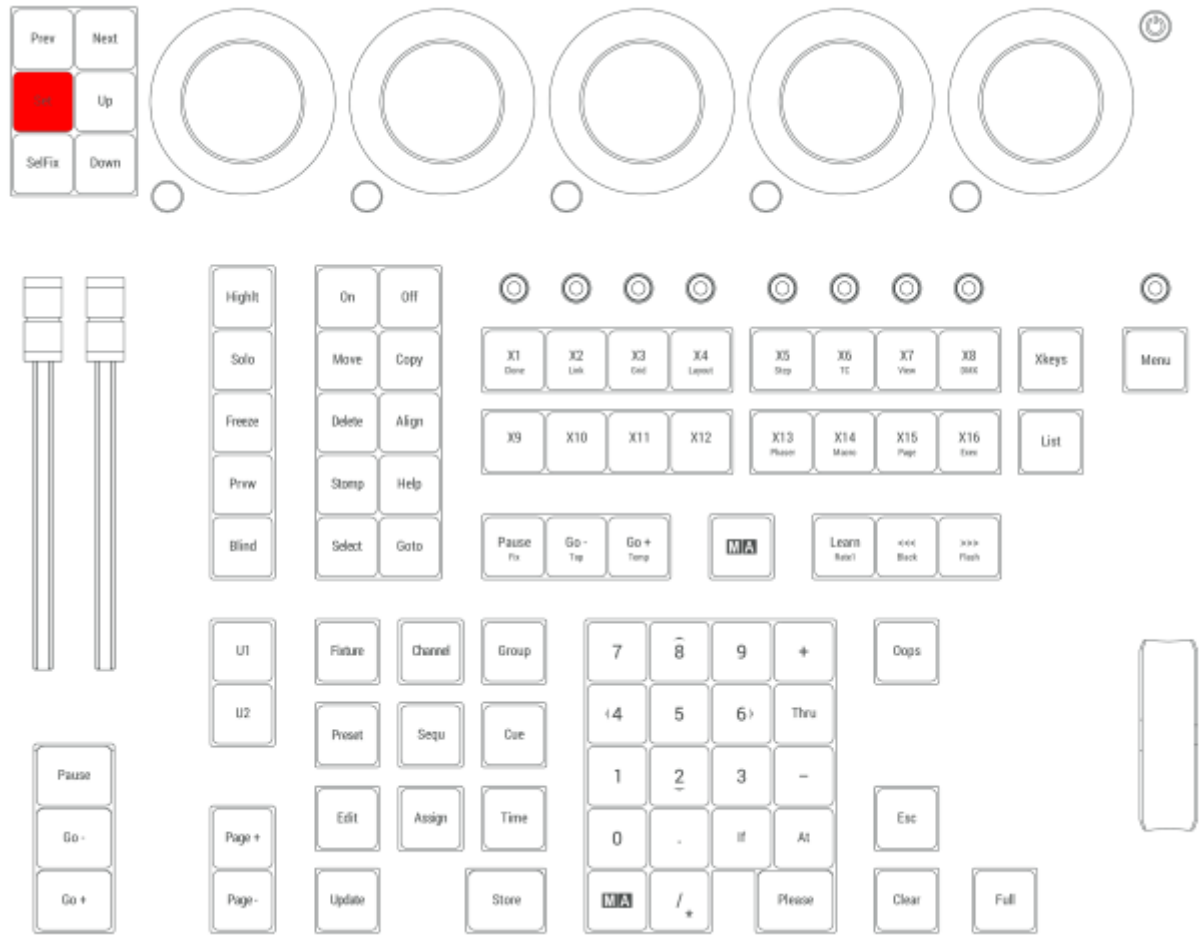
```
OK
:   Step Toggle Executor
```

Location

Set is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.55. Stomp

Pressing Stomp enters Stomp into the command line.

For more information on Stomp, see the **Stomp keyword**.

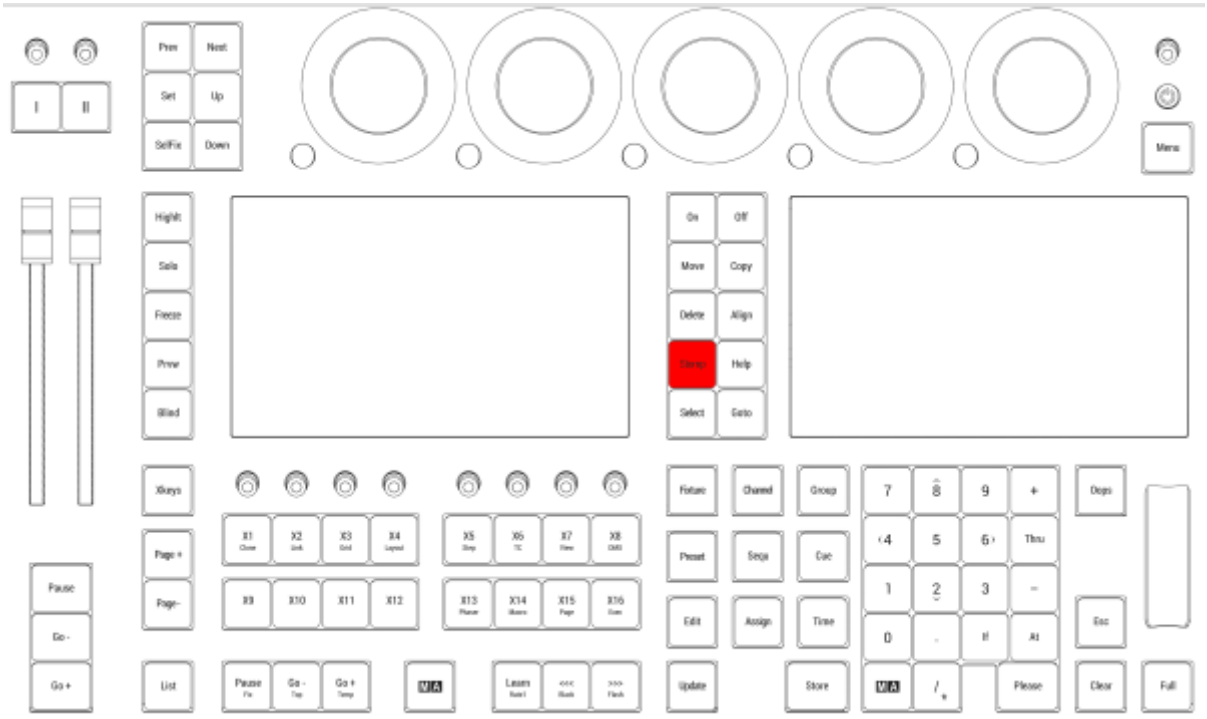
Capture

Pressing Stomp Stomp enters Capture into the command line.

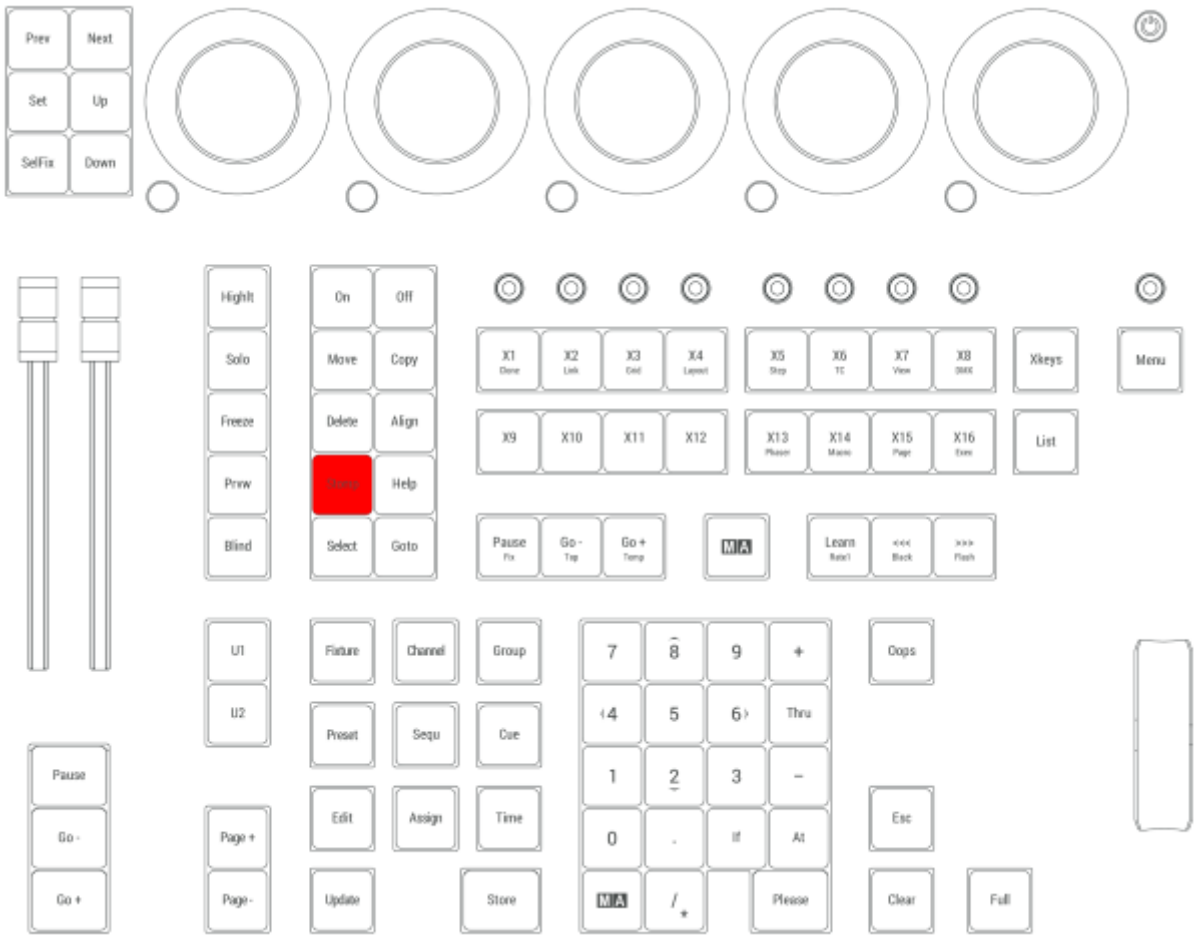
For more information on Capture, see the **Capture keyword**.

Location

Stomp is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

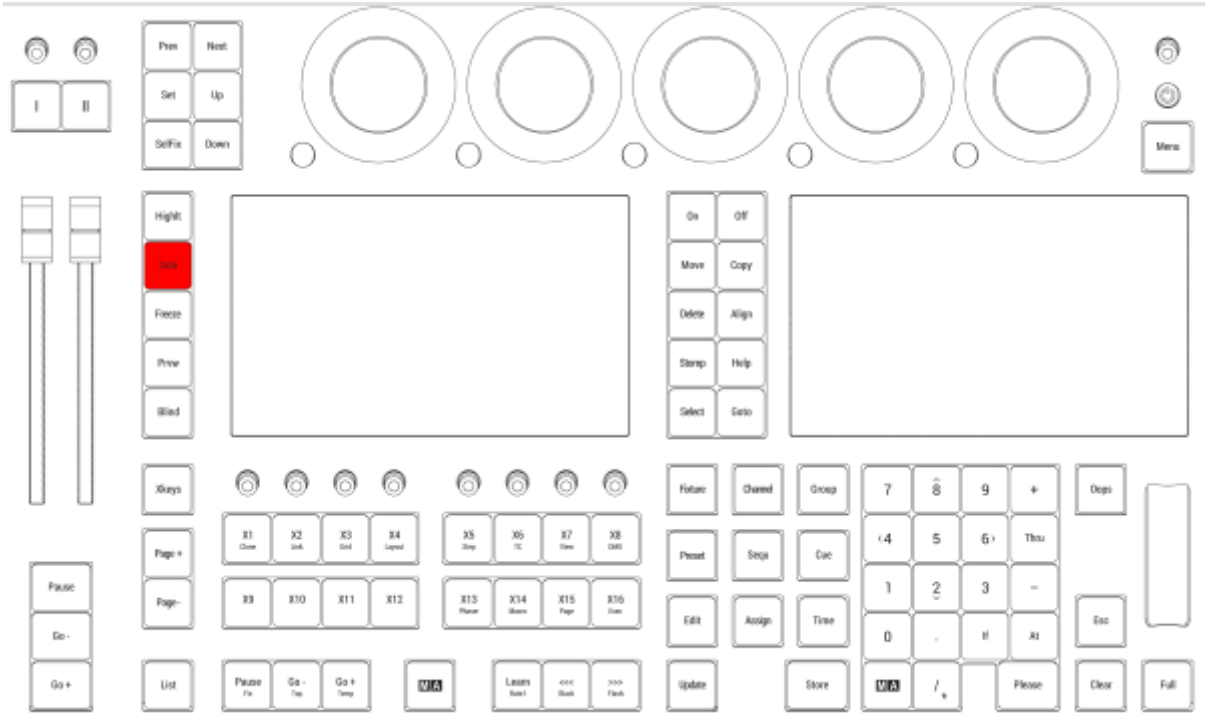
1.3.15.56. Solo

Pressing Solo directly executes solo selection and enables or disables Solo.

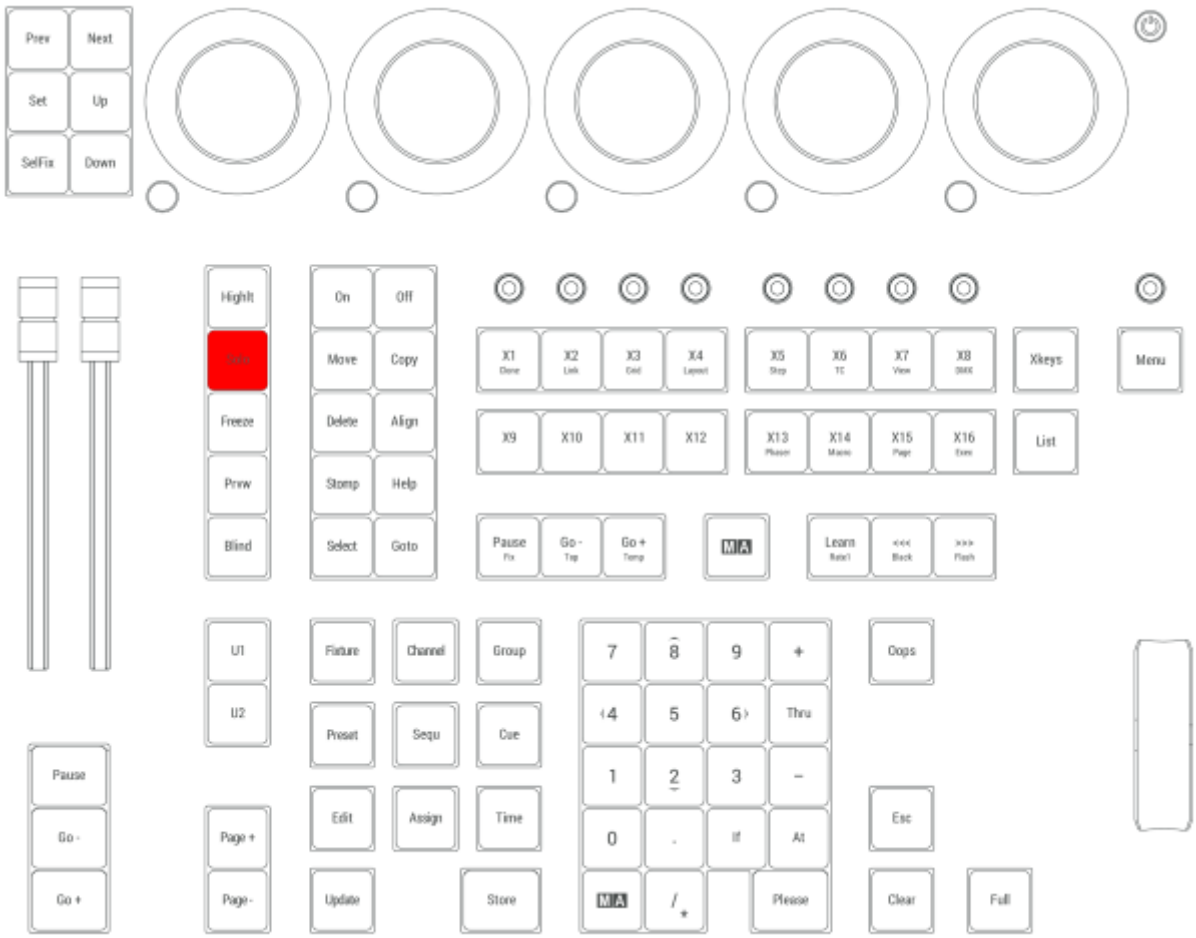
For more information about Solo, see the **Solo keyword**.

Location

Solo is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

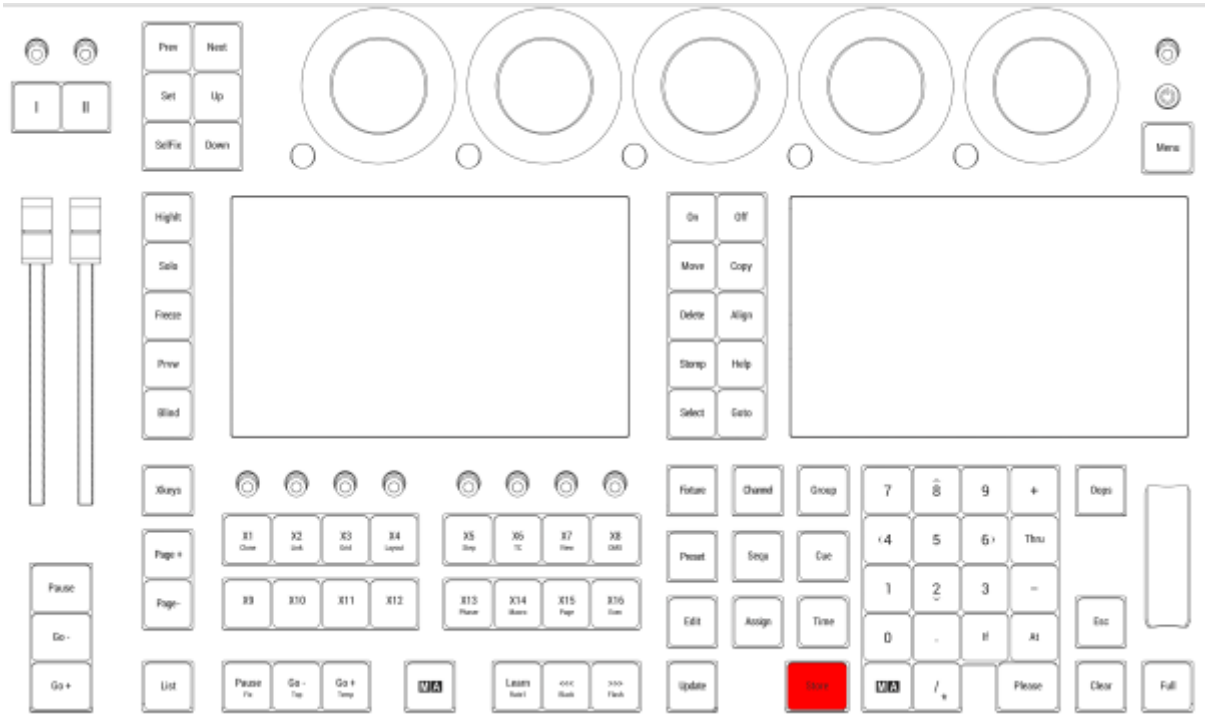
1.3.15.57. Store

Pressing Store enters the Store keyword into the command line.

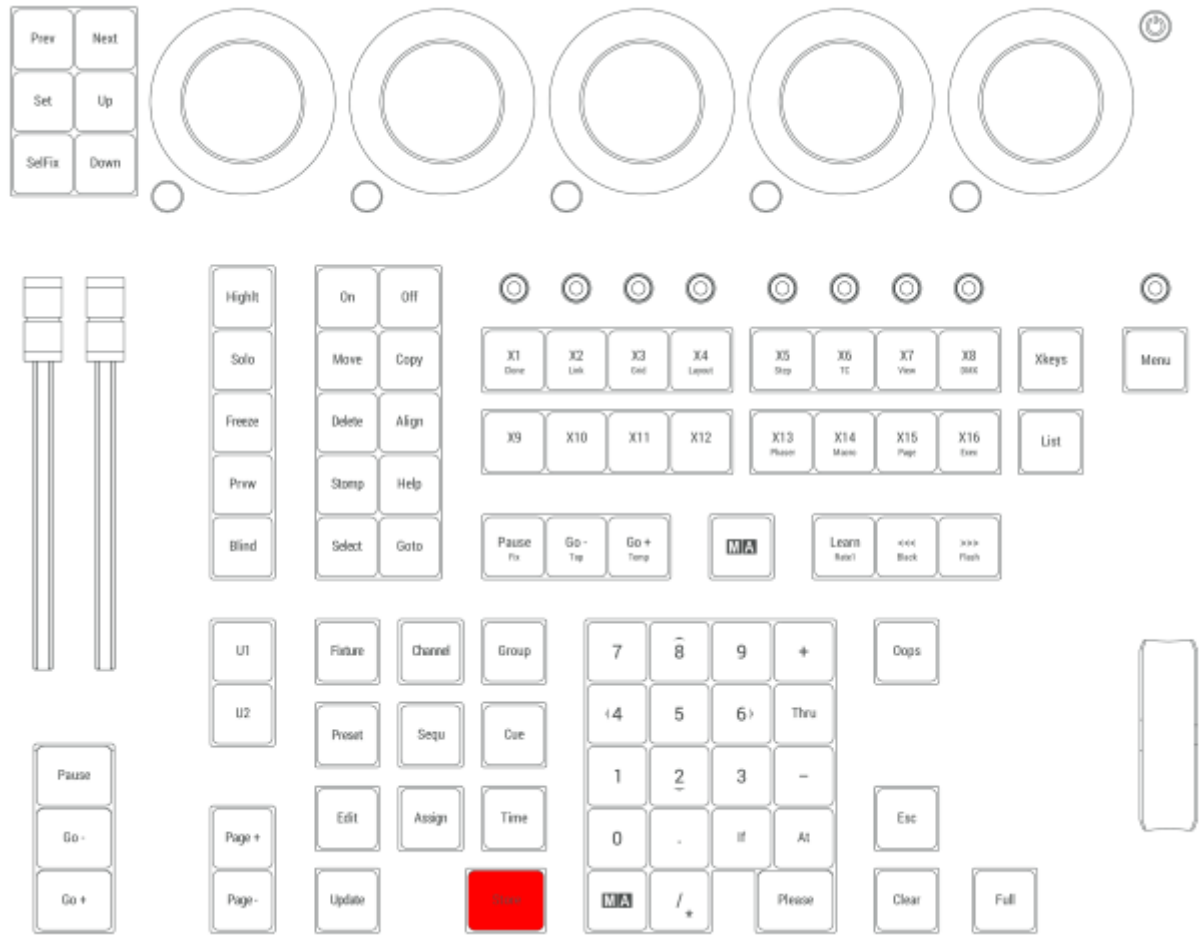
For more information about Store, see the **Store keyword**.

Location

Store is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

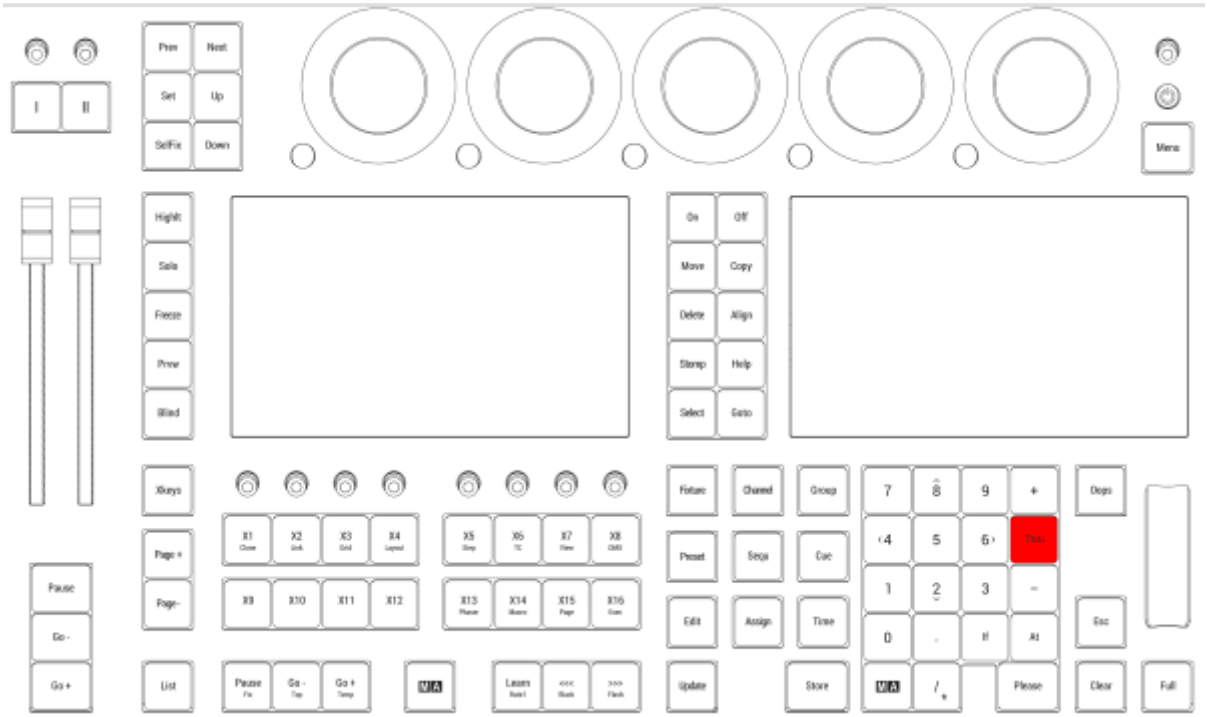
1.3.15.58. Thru

Pressing Thru enters the Thru keyword into the command line.

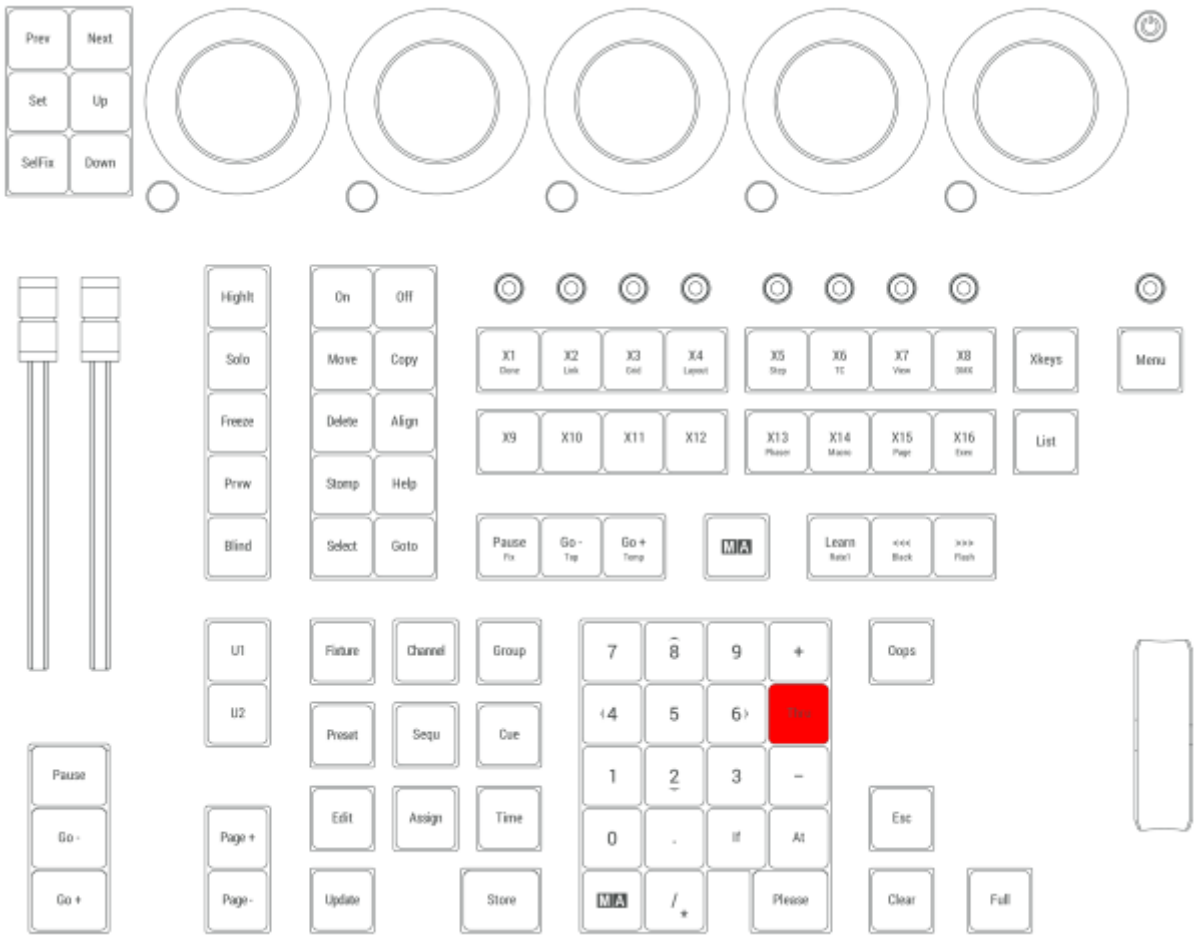
For more information about Thru, see the **Thru keyword**.

Location

Thru is located in the numeric keys section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.59. Time

The default functionality of Time depends on the "Time Key Target" setting in the current user profile.

If the "Time Key Target" is set to "Cue" and the command line is empty, pressing Time will toggle between CueFade and CueDelay in the command line.

For more information about Time, see the **CueFade keyword** or **CueDelay keyword**.

If the "Time Key Target" is set to "Cue" and the command line contains a fixture selection, pressing Time will toggle between Fade and Delay in the command line.

For more information about Time, see the **Fade keyword** or **Delay keyword**.

If the "Time Key Target" is set to "Fixture" and the command line is empty, pressing Time will toggle between Fade and Delay in the command line.

For more information about Time, see the **Fade keyword** or **Delay keyword**.

If the "Time Key Target" is set to "Fixture" and either Store or Cue is in the command line, pressing Time will toggle between CueFade and CueDelay in the command line.

For more information about Time, see the **CueFade keyword** or **CueDelay keyword**.

Regardless of the "Time Key Target" setting, pressing MA Time will toggle between Relative, Fade, Delay, and Absolute.

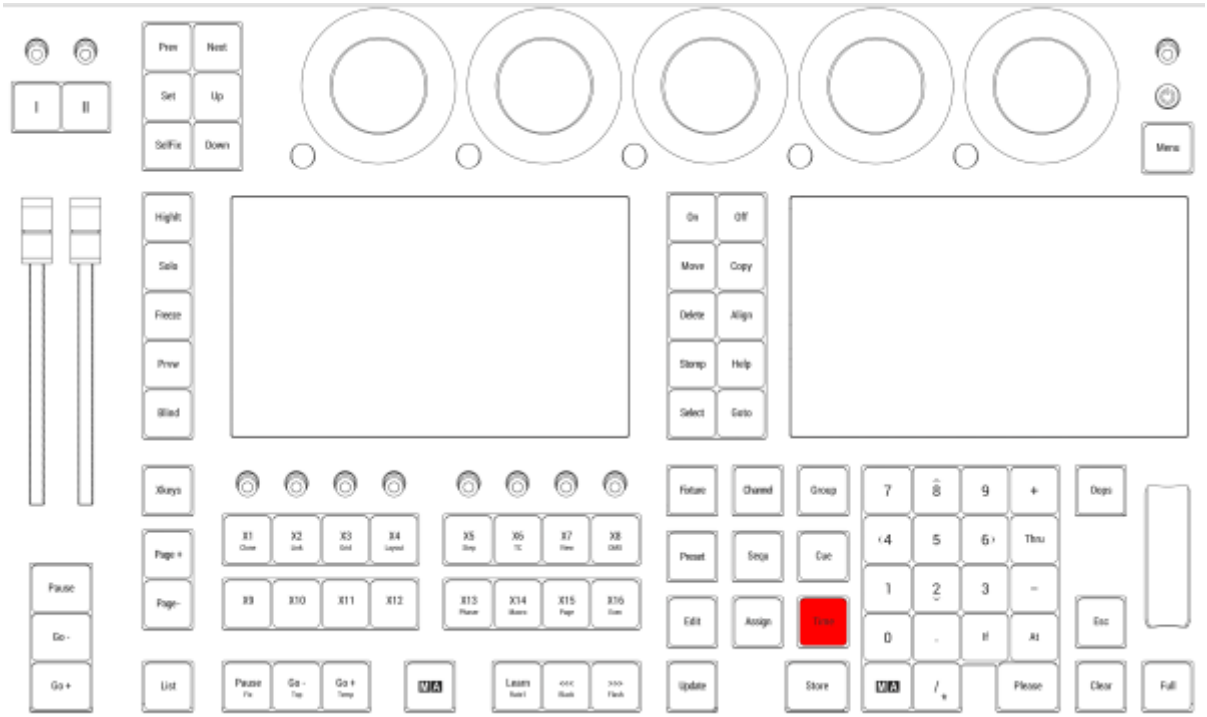
For more information about Time, see the **Relative keyword**, **Fade keyword**, **Delay keyword**, or **Absolute keyword**.

Regardless of the "Time Key Target" setting, additionally adding **At** to the command and pressing Time will cycle through the value layer keywords.

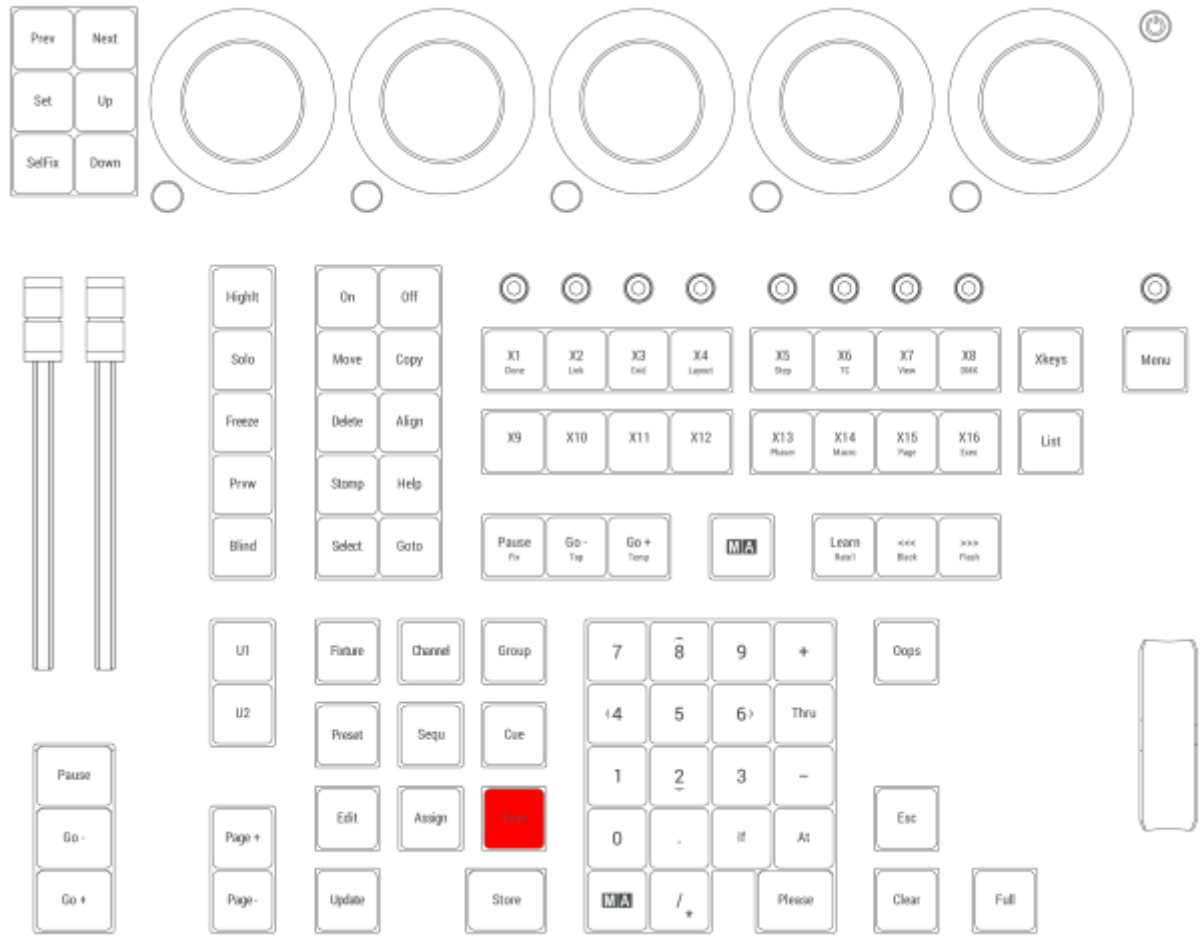
Example: At 5 0 Time

Location

Time is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



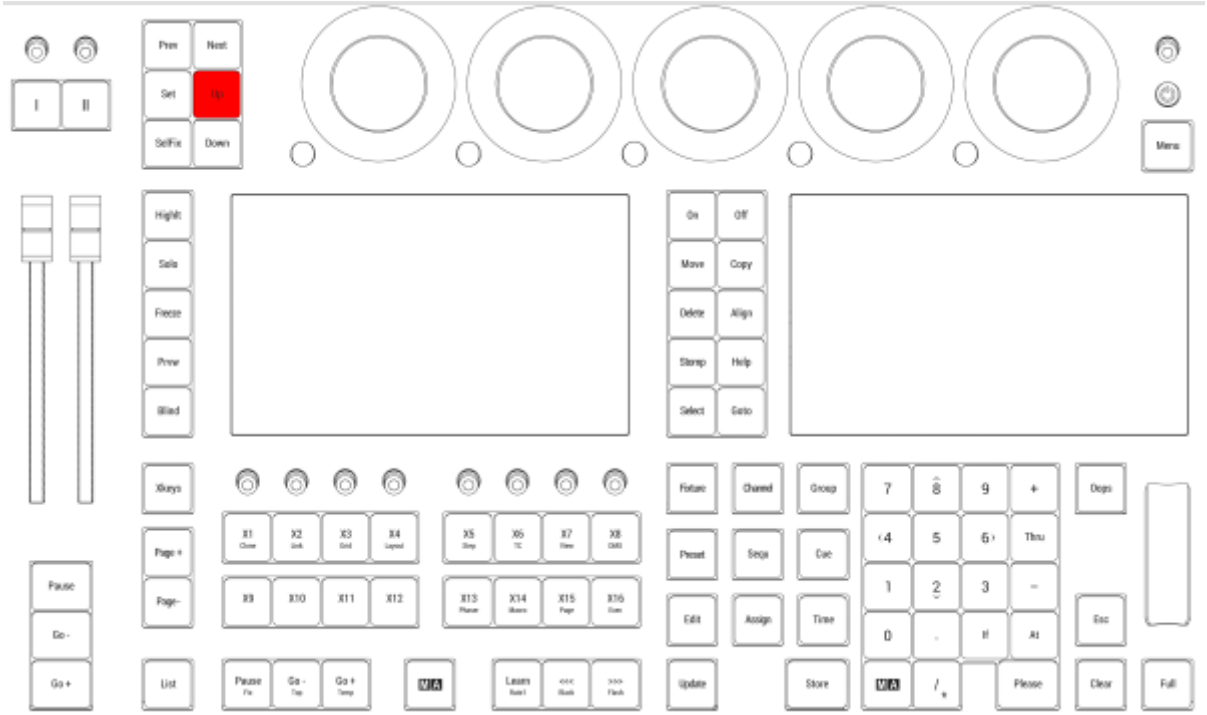
Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.60. Up

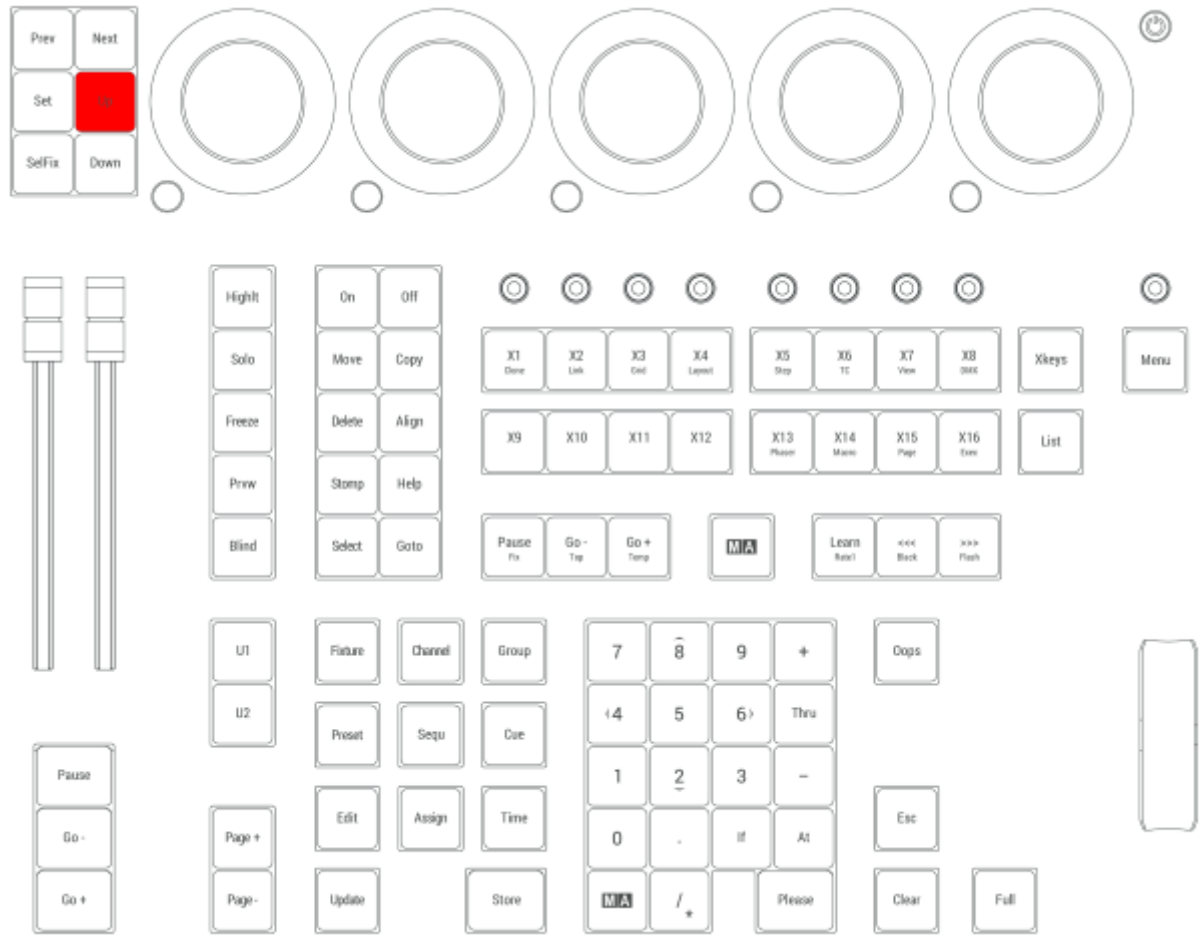
Pressing Up navigates up in the fixture/subfixture structure.

Location

Up is located in the command section on the left side of the five dual encoders.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.61. Update

Pressing Update enters the Update keyword into the command line.

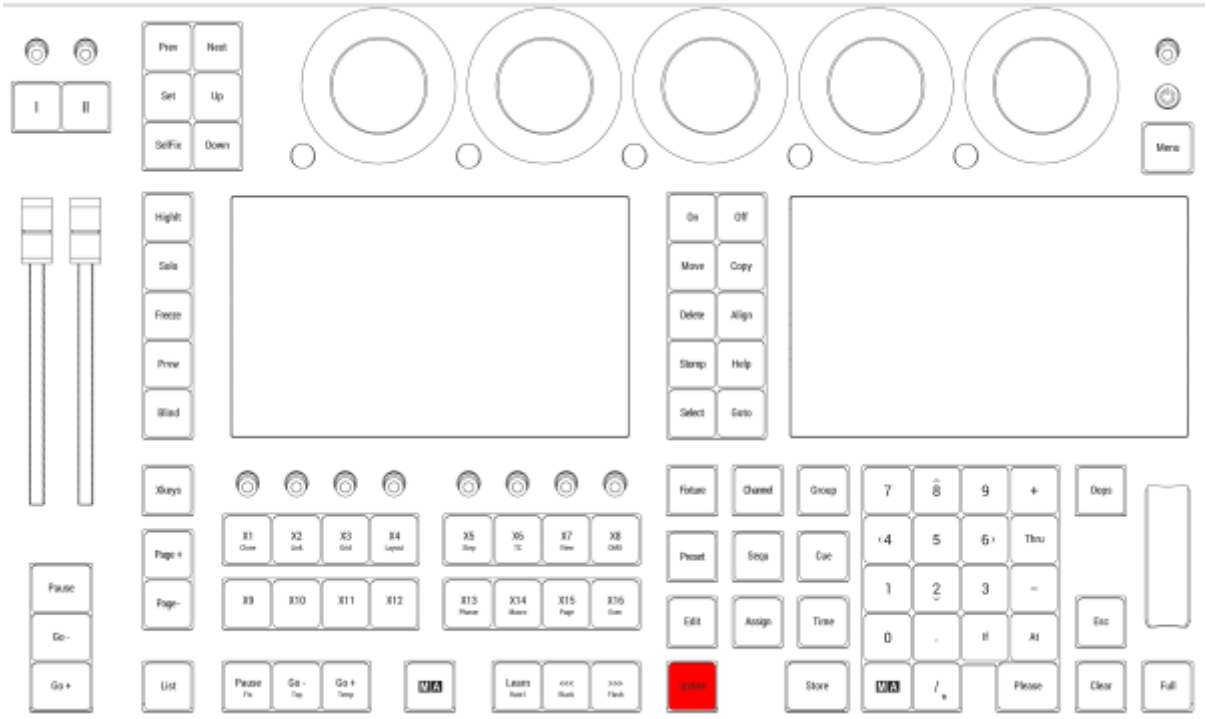
For more information about Update, see the **Update Keyword**.

Pressing MA + Update enters the Cook keyword into the command line.

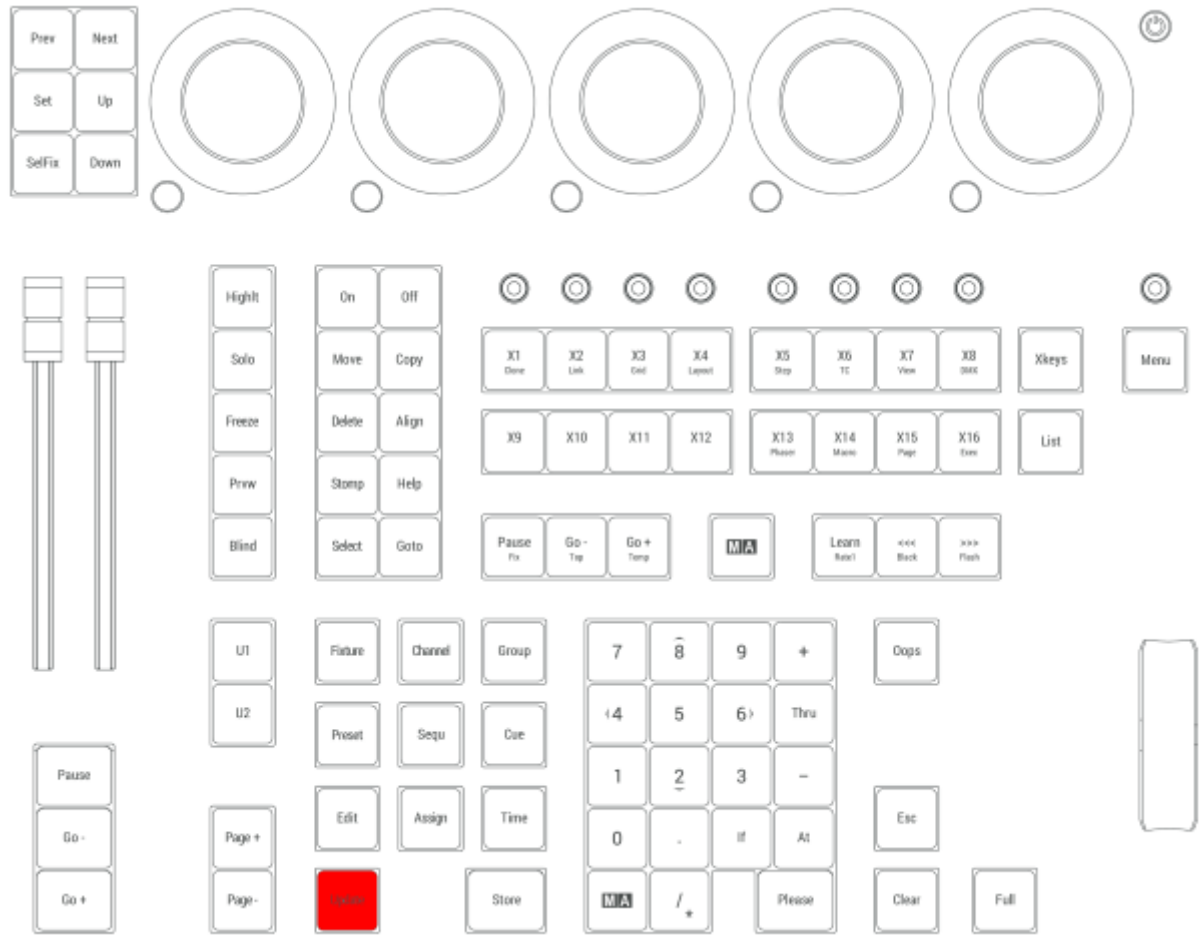
For more information about Cook, see the **Cook Keyword**.

Location

Update is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.62. U1

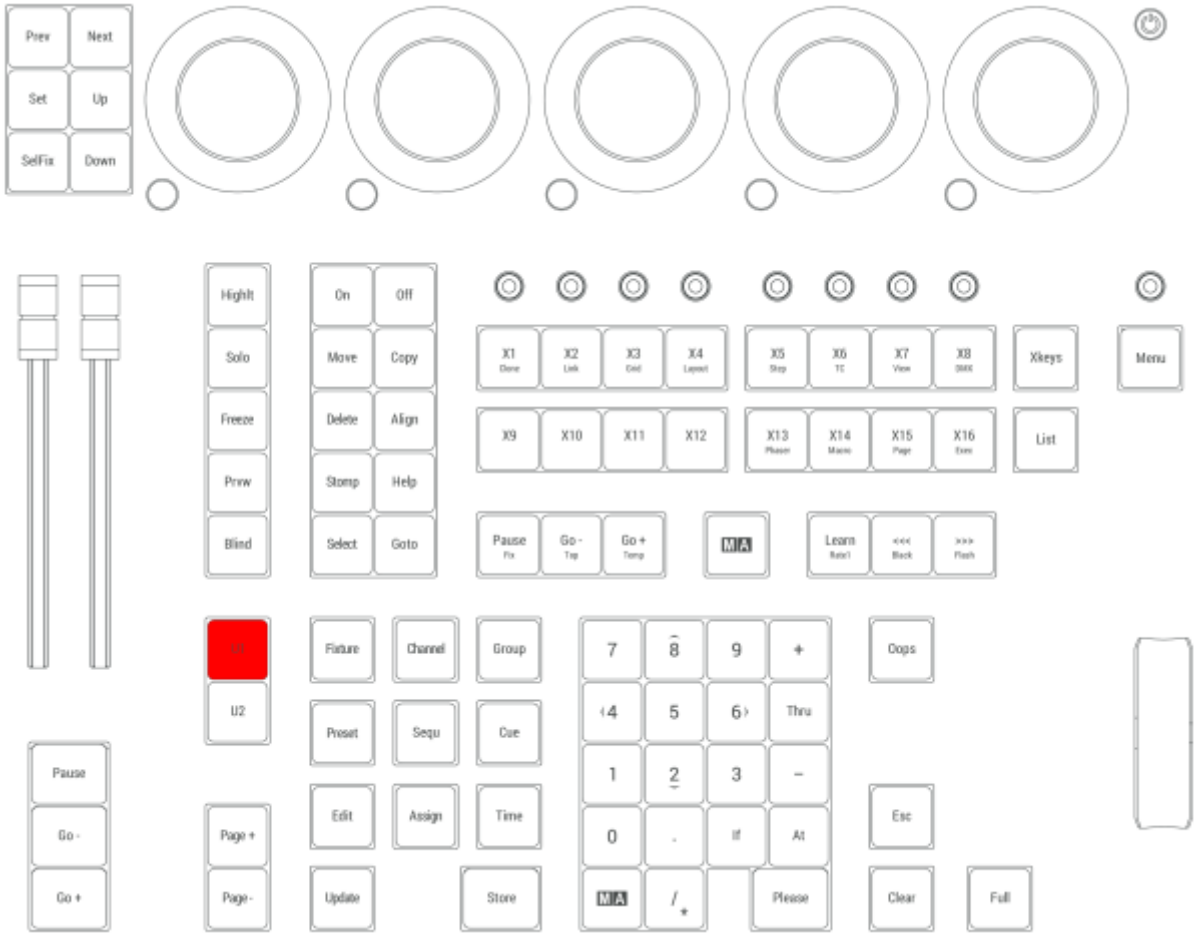
Pressing U1 executes the User1 command in the command line.

This key is only present on grandMA3 compact, grandMA3 compact XT, grandMA3 onPC command wing and grandMA3 onPC command wing XT.

For more information about U1, see **User1 keyword**.

Location

U1 is located in the command section on the left side of the numeric keys.



Location on grandMA3 compact consoles and grandMA3 onPC command wings

1.3.15.63. U2

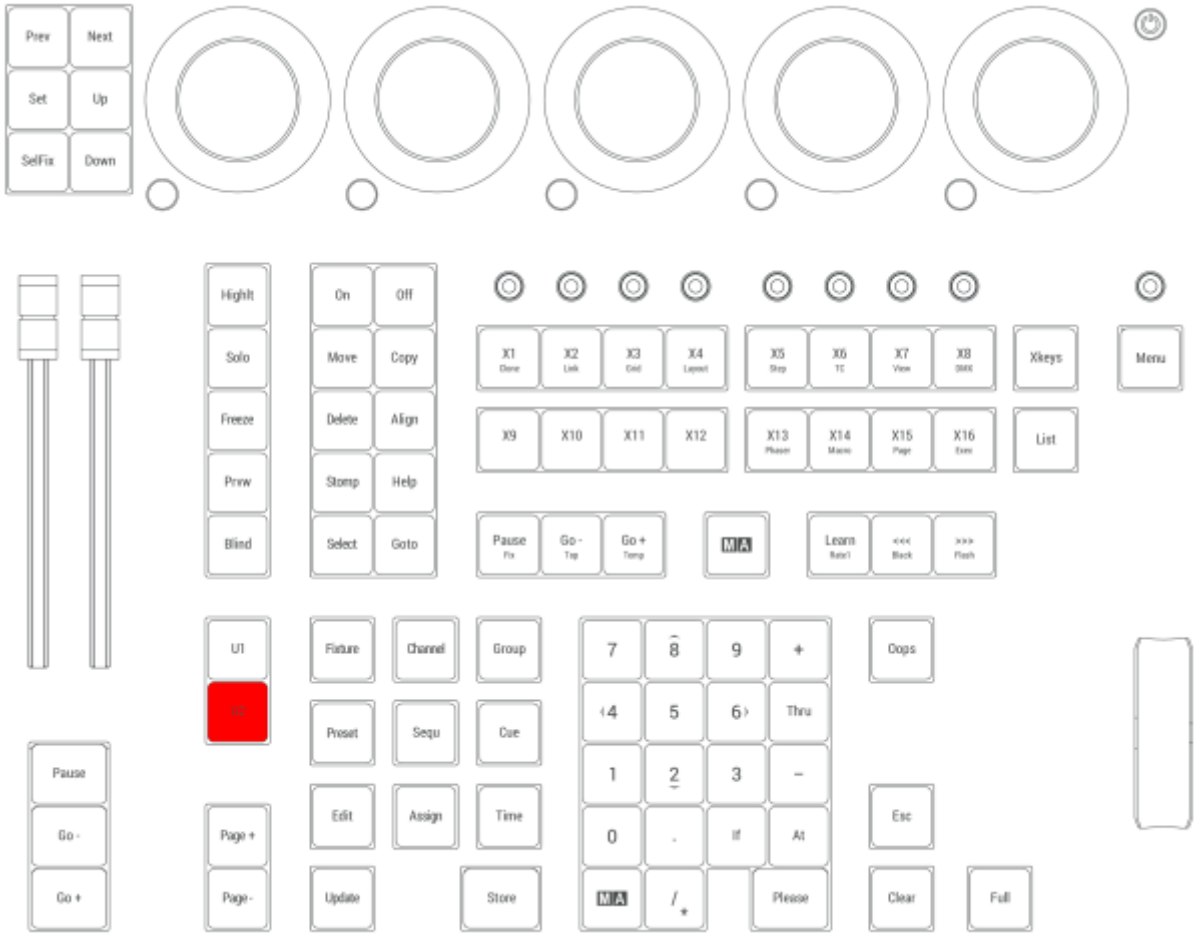
Pressing U2 executes the User2 command in the command line.

This key is only present on grandMA3 compact, grandMA3 compact XT, grandMA3 onPC command wing and grandMA3 onPC command wing XT.

For more information about U2, see the **User2 keyword**.

Location

U2 is located in the command section on the left side of the numeric keys.



Location on grandMA3 compact consoles and grandMA3 onPC command wings

1.3.15.64. X1 | Clone

X1 | Clone is executor 291.

	Hint:
	All Xkeys behave like executors.

Clone

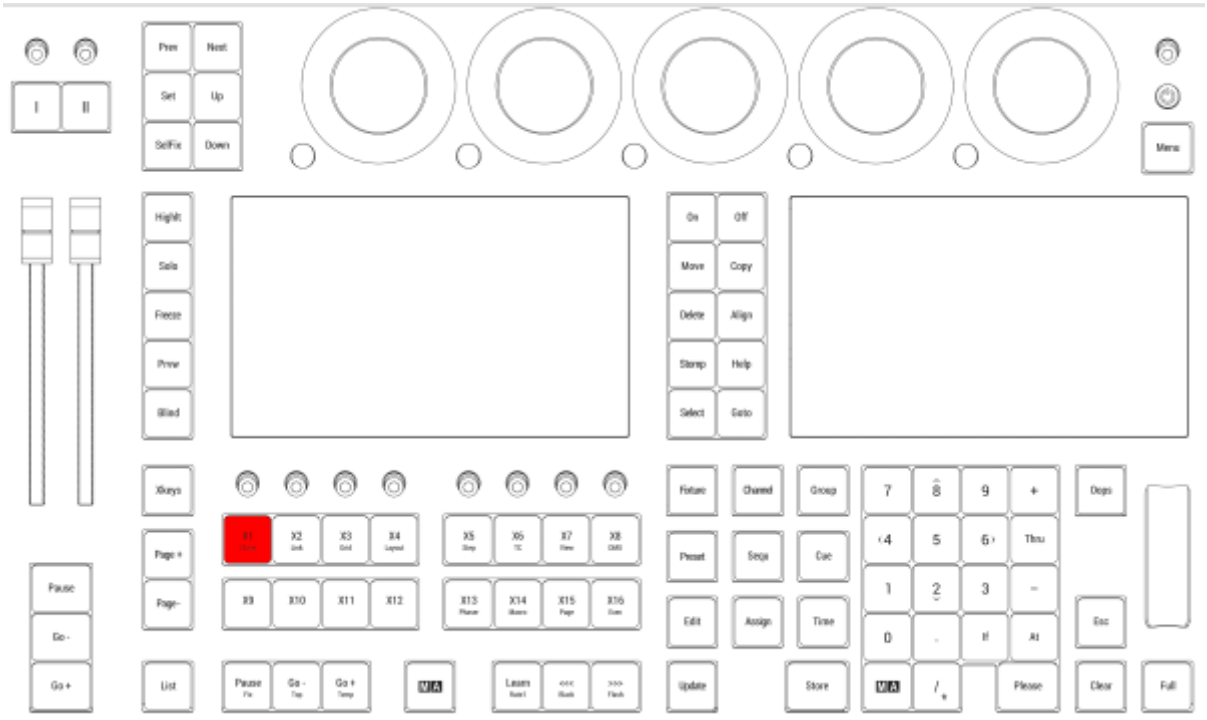
Pressing and holding MA + X1 | Clone enters the Clone keyword into the command line.

MA	User name[Fixture]>Clone

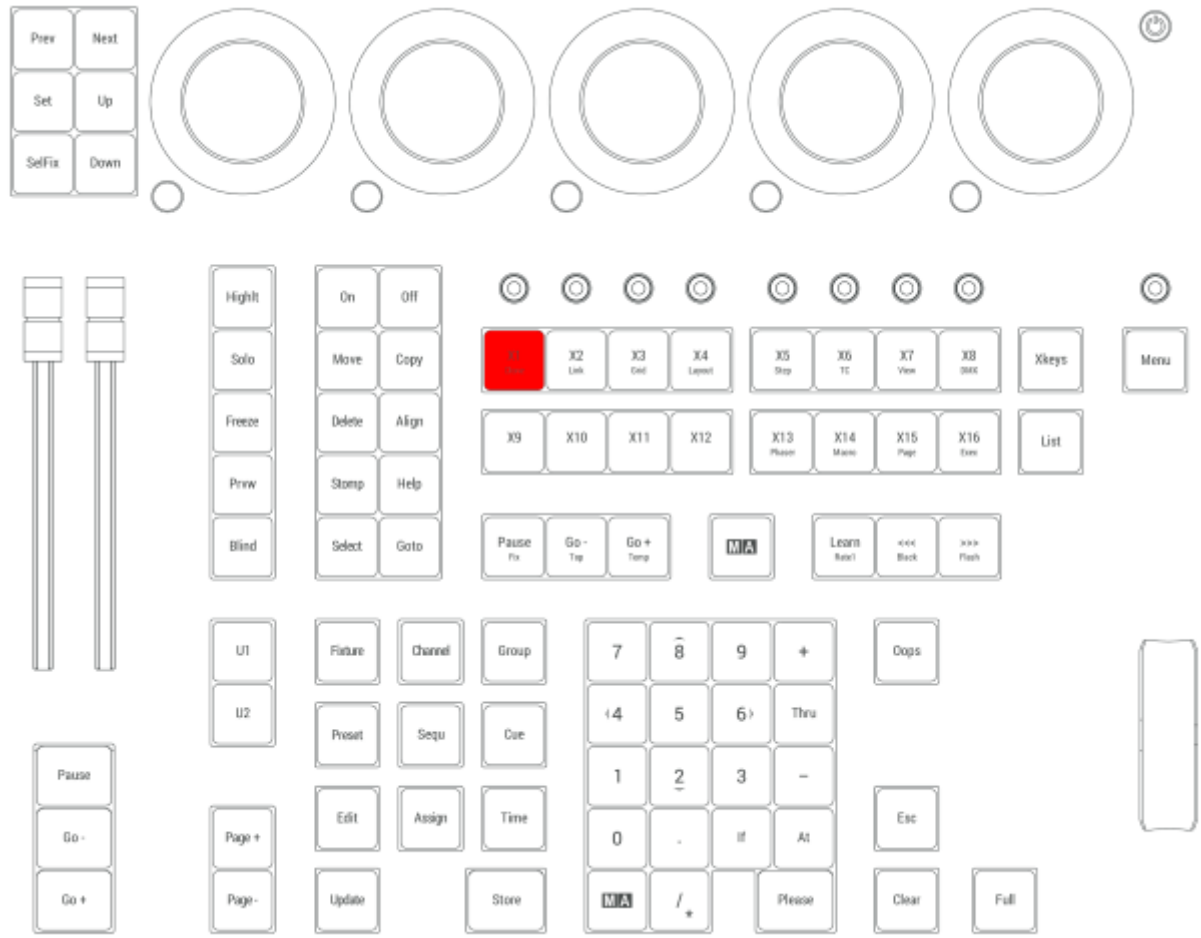
For more information about Clone, see the **Clone keyword**.

Location

X1 | Clone is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

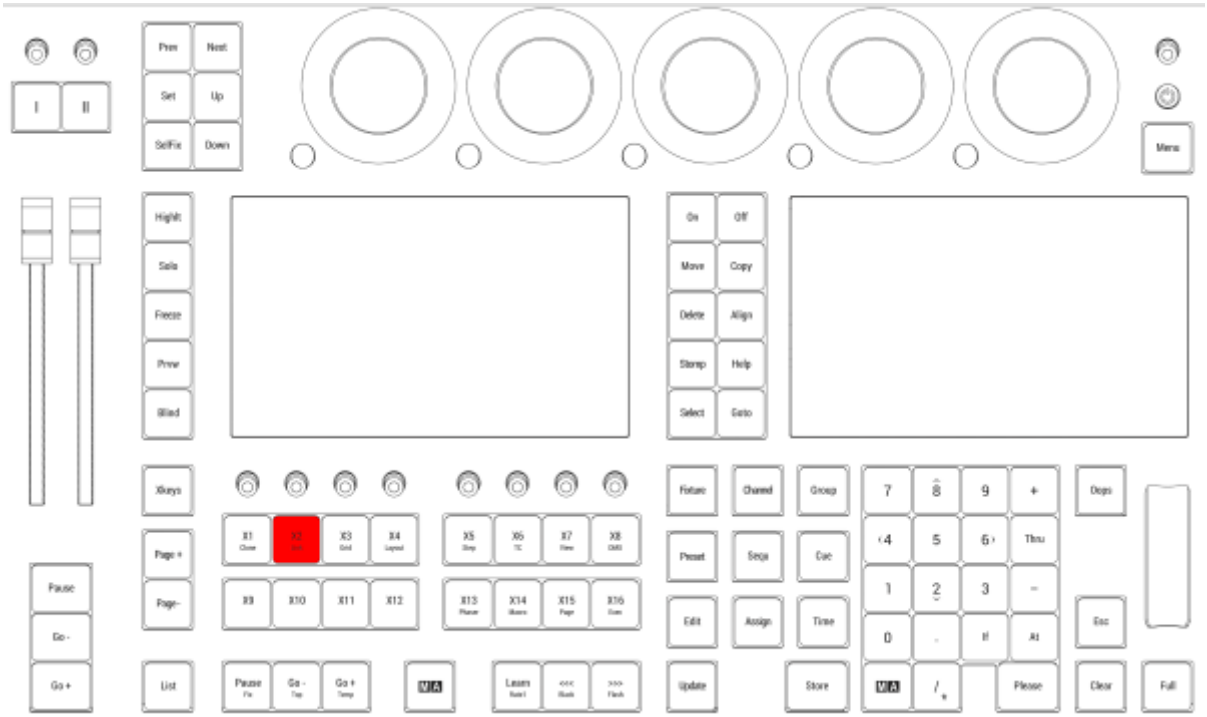
1.3.15.65. X2 | Link

X2 | Link is executor 292.

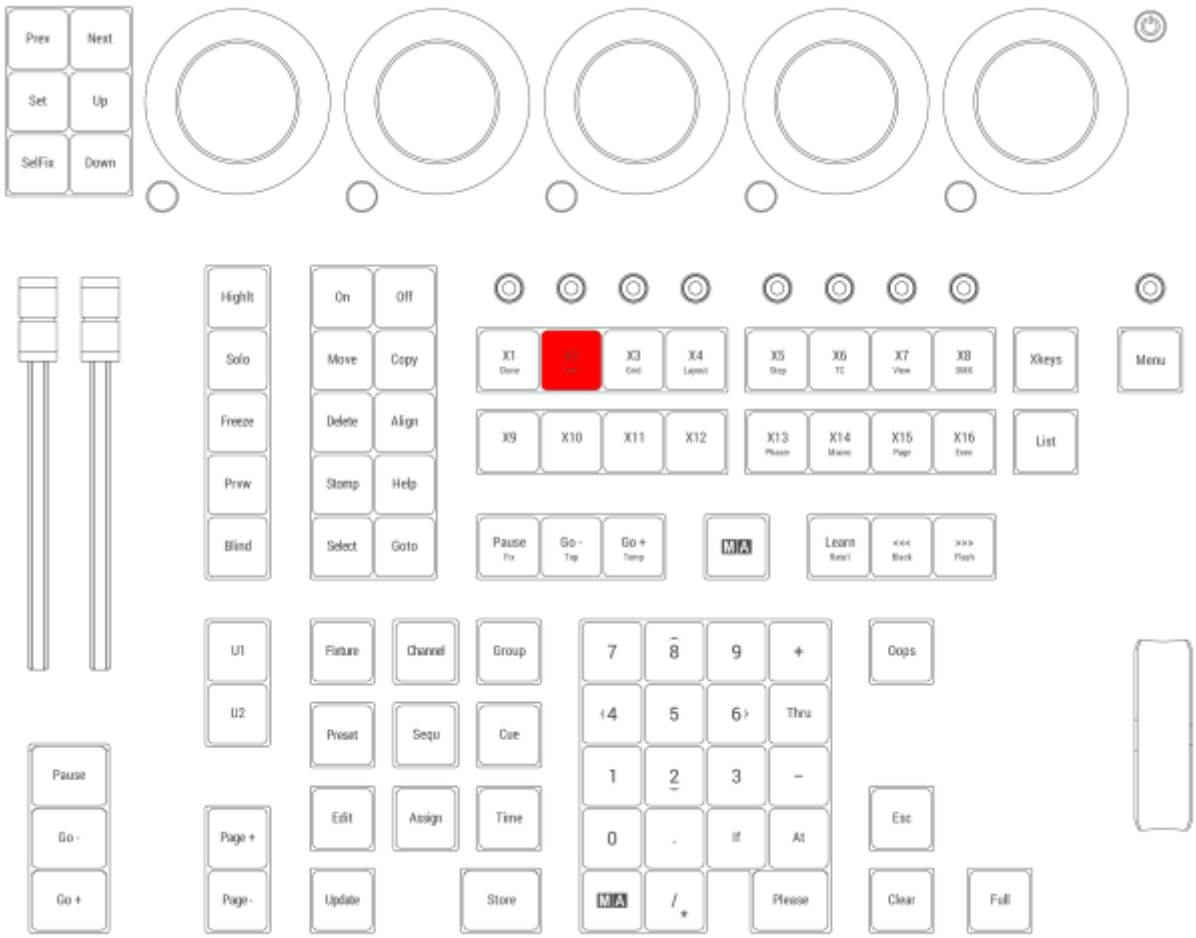
	Hint:
	All Xkeys behave like executors.

Location

X2 | Link is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.66. X3 | Grid

X3 | Grid is executor 293.

	Hint:
	All Xkeys behave like executors.

Grid

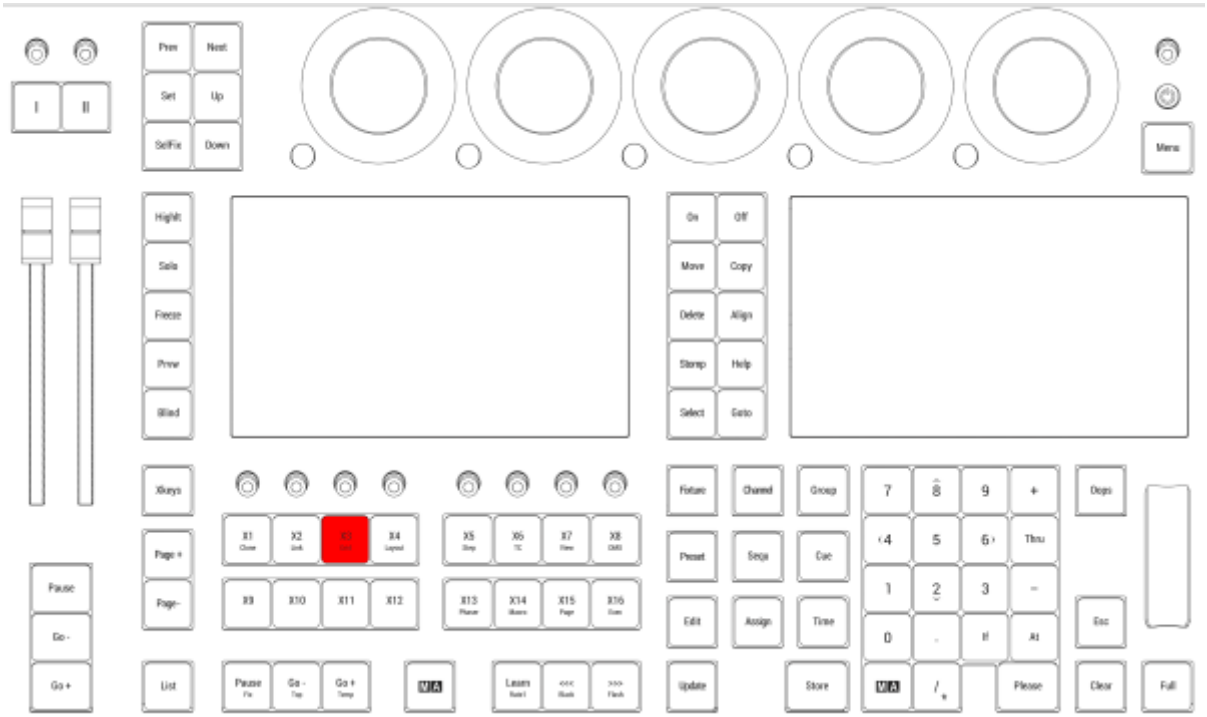
Pressing and holding MA + X3 | Grid enters the Grid keyword into the command line.

MA User name[Fixture]>Grid

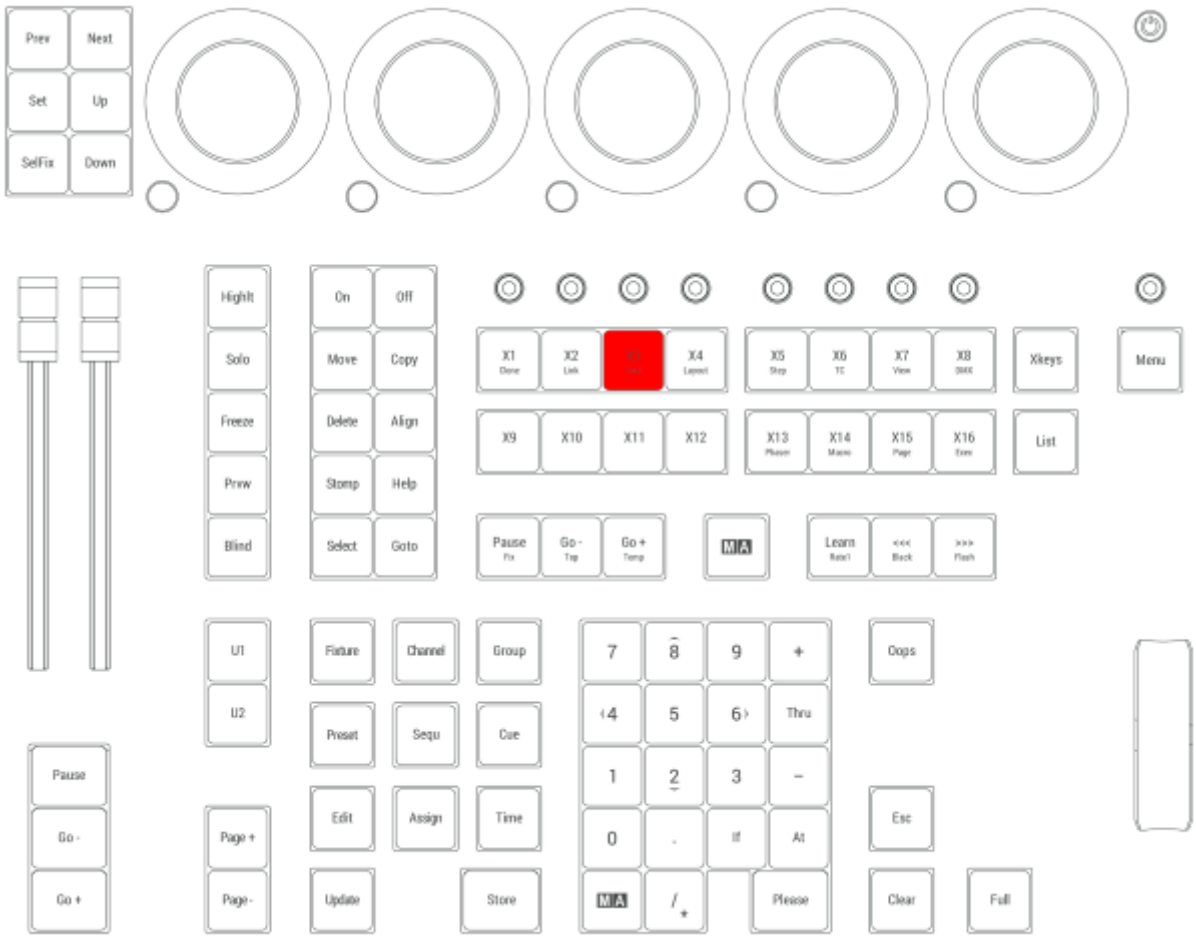
For more information about Grid, see the **Grid keyword**.

Location

X3 | Grid is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.67. X4 | Layout

X4 | Layout is executor 294.

	Hint:
	All Xkeys behave like executors.

Layout

Pressing and holding MA + X4 | Layout enters the Layout keyword into the command line.

For more information about Layout, see the **Layout keyword**.

Appearance

Pressing and holding MA + X4 | Layout + X4 | Layout enters the Appearance keyword into the command line.

For more information about Appearance, see the **Appearance keyword**.

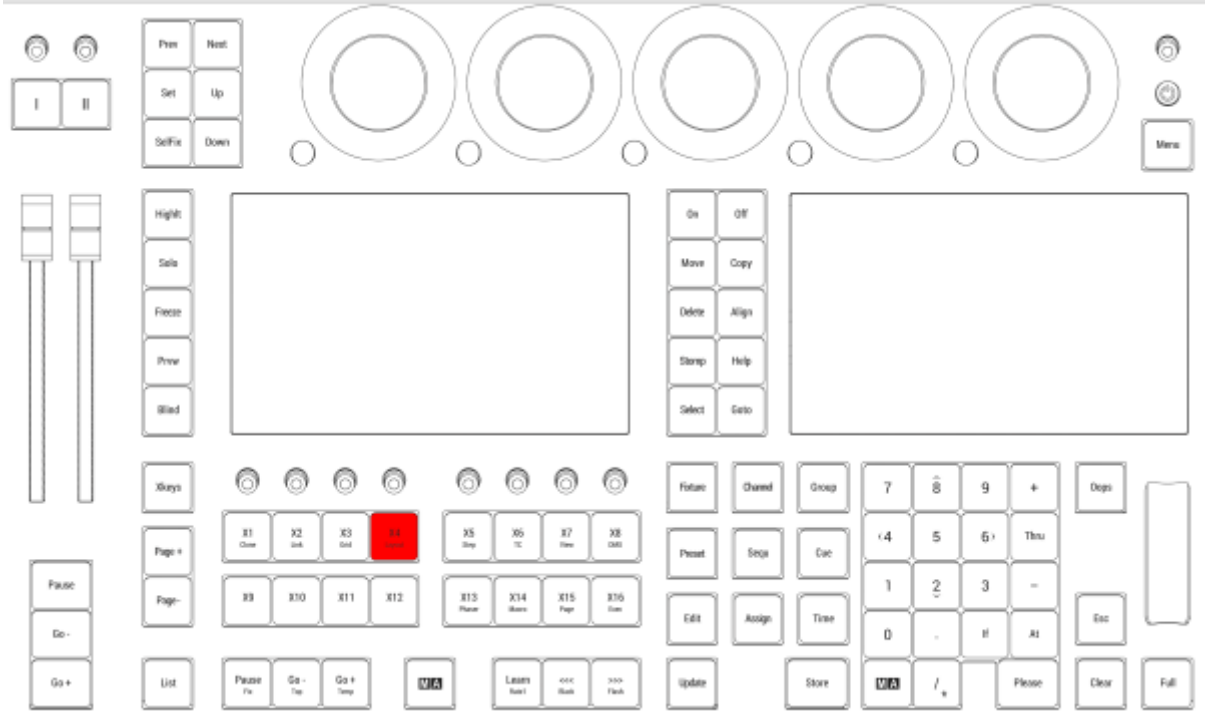
Scribble

Pressing and holding MA + X4 | Layout + X4 | Layout + X4 | Layout enters the Scribble keyword into the command line.

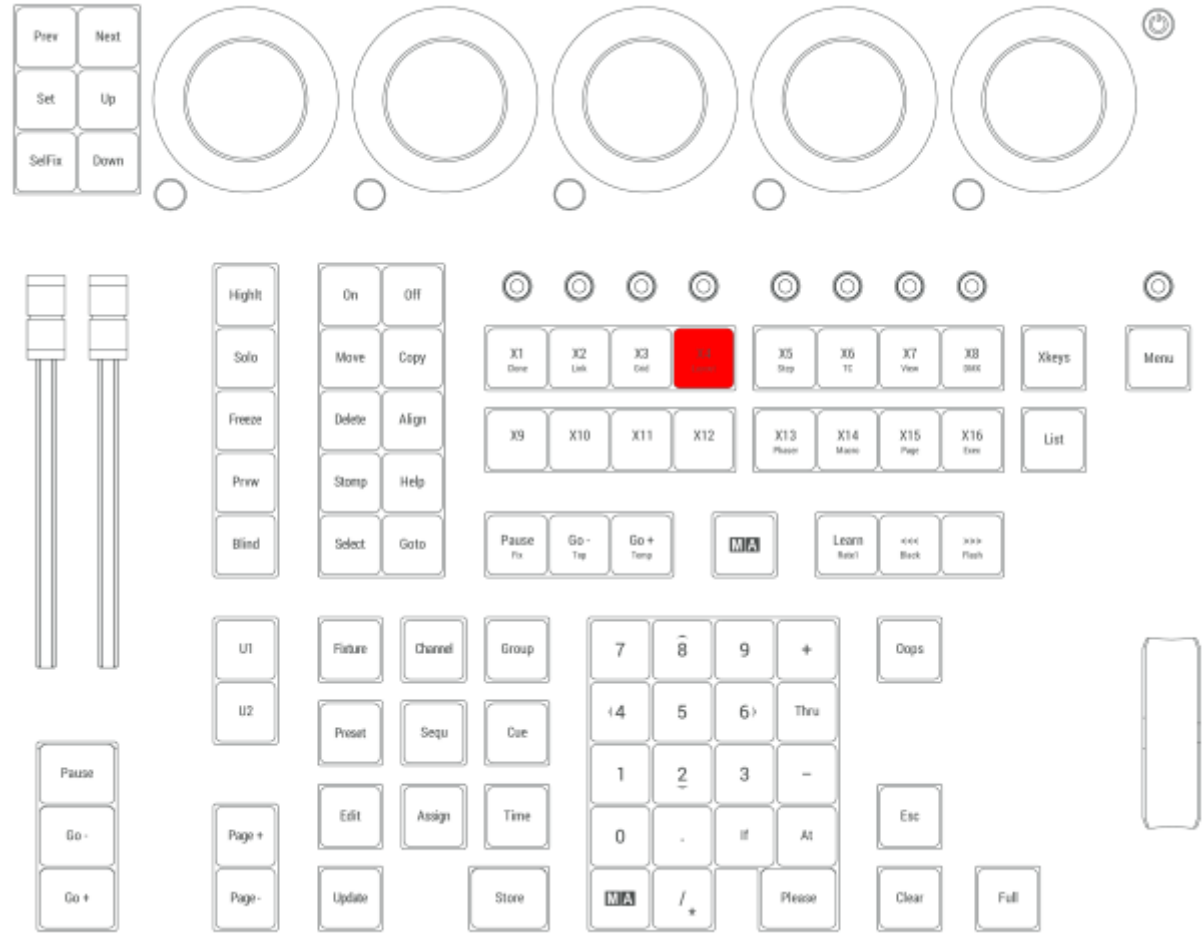
For more information about Scribble, see the **Scribble keyword**.

Location

X4 | Layout is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.68. X5 | Step

X5 | Step is executor 295.

	Hint:
	All Xkeys behave like executors.

Step

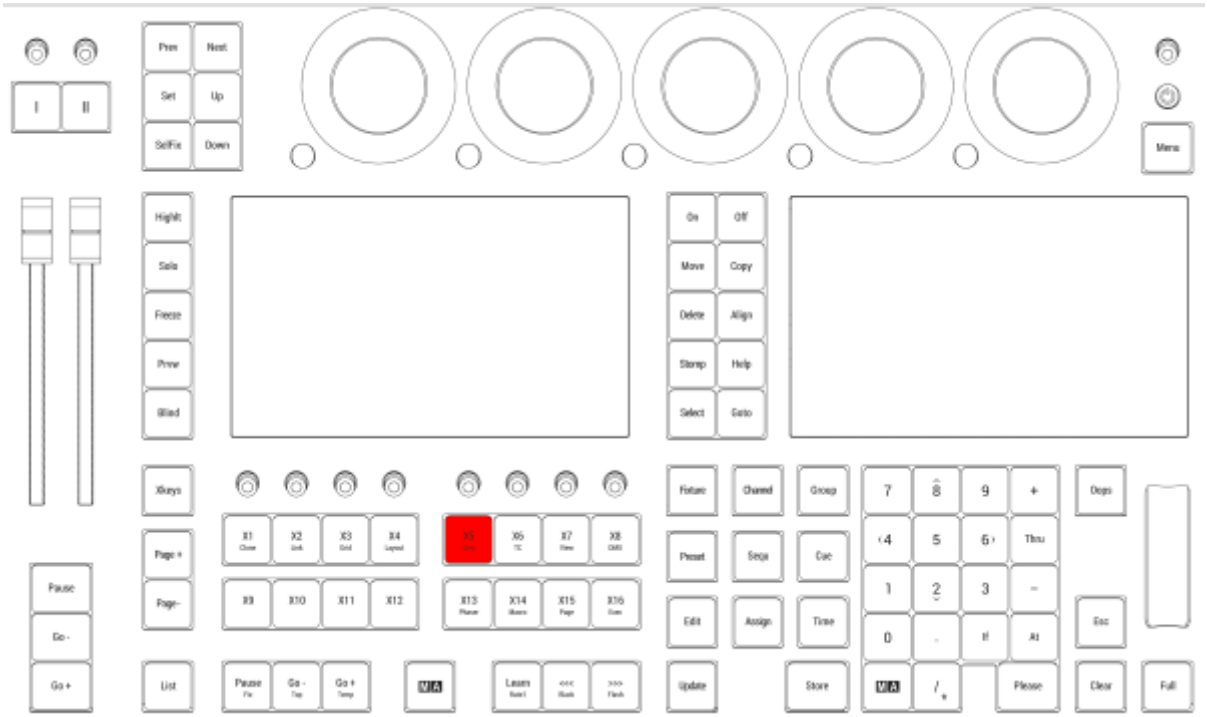
Pressing and holding MA + X5 | Step enters the Step keyword into the command line.

MA User name[Fixture]>Step

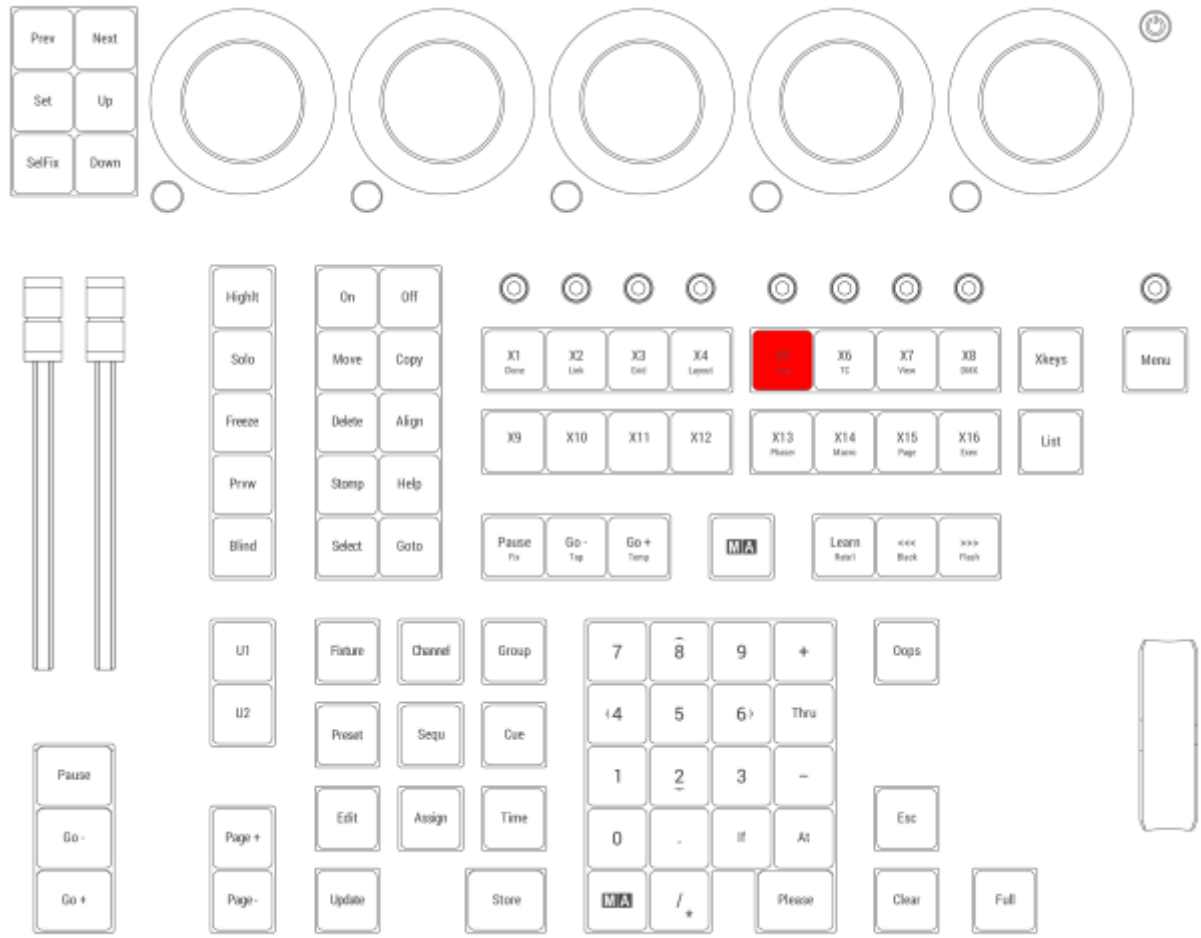
For more information about Step, see the **Step keyword**.

Location

X5 | Step is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.69. X6 | TC

X6 | TC is executor 296.

	Hint:
	All Xkeys behave like executors.

Timecode

Pressing and holding MA + X6 | TC enters the Timecode keyword into the command line.

For more information about Timecode, see the **Timecode keyword**.

TimecodeSlot

Pressing and holding MA + X6 | TC + X6 | TC enters the TimecodeSlot keyword into the command line.

For more information about TimecodeSlot, see the **TimecodeSlot keyword**.

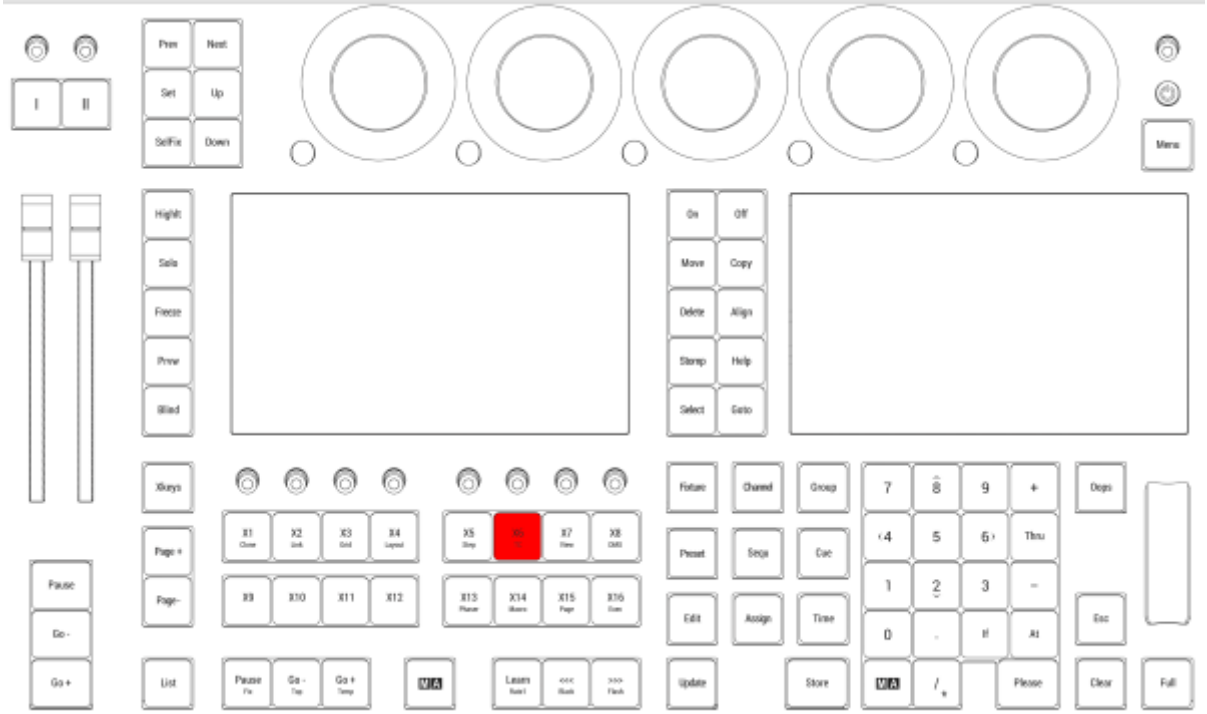
Timer

Pressing and holding MA + X6 | TC + X6 | TC + X6 | TC enters the Timer keyword into the command line.

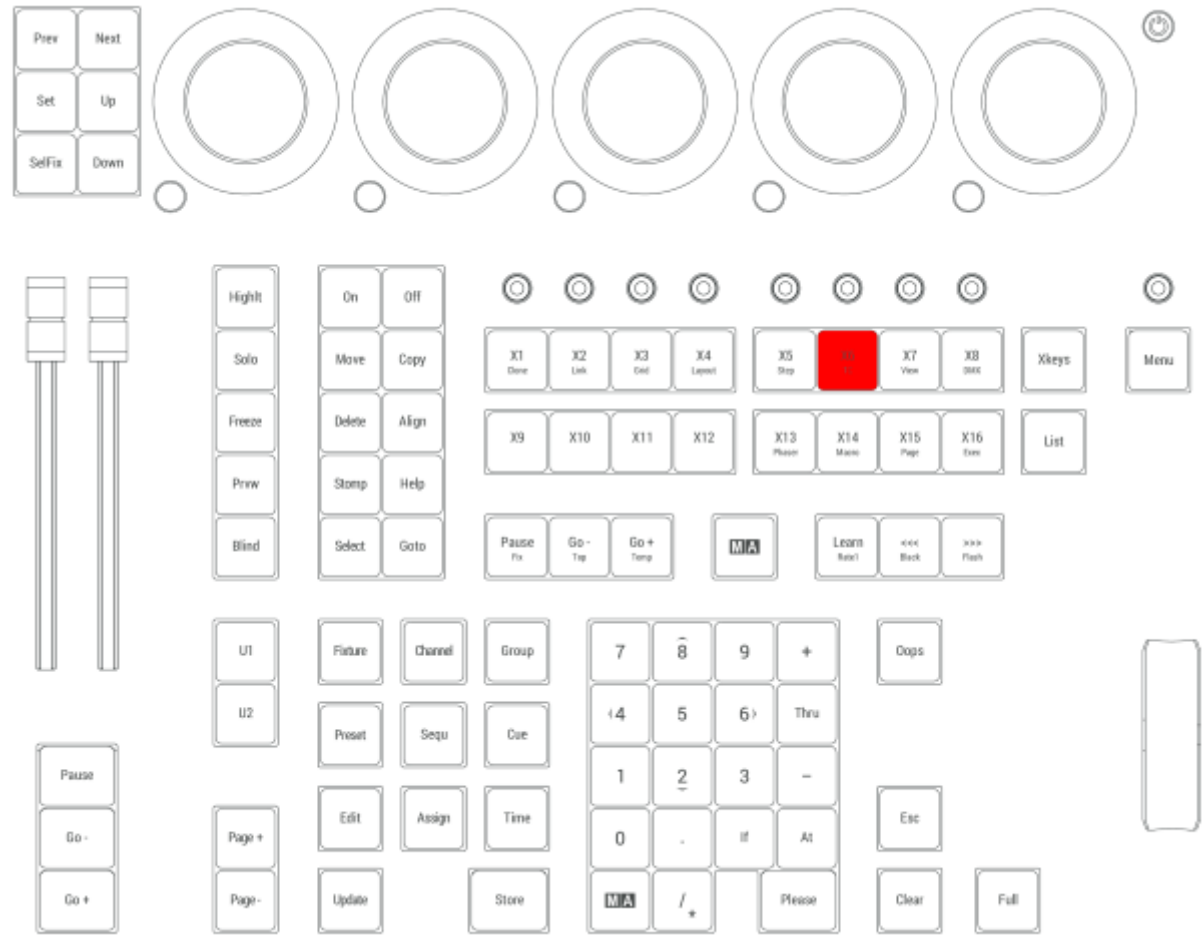
For more information about Timer, see the **Timer keyword**.

Location

X6 | TC is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.70. X7 | View

X7 | View is executor 297.

	**Hint:**
	All keys behave like executors.

View

Pressing and holding MA + X7 | View enters the View keyword into the command line.

For more information about View, see the **View keyword**.

ViewButton

Pressing and holding MA + X7 | View + X7 | View enters the ViewButton keyword into the command line.

For more information about ViewButton, see the **ViewButton keyword**.

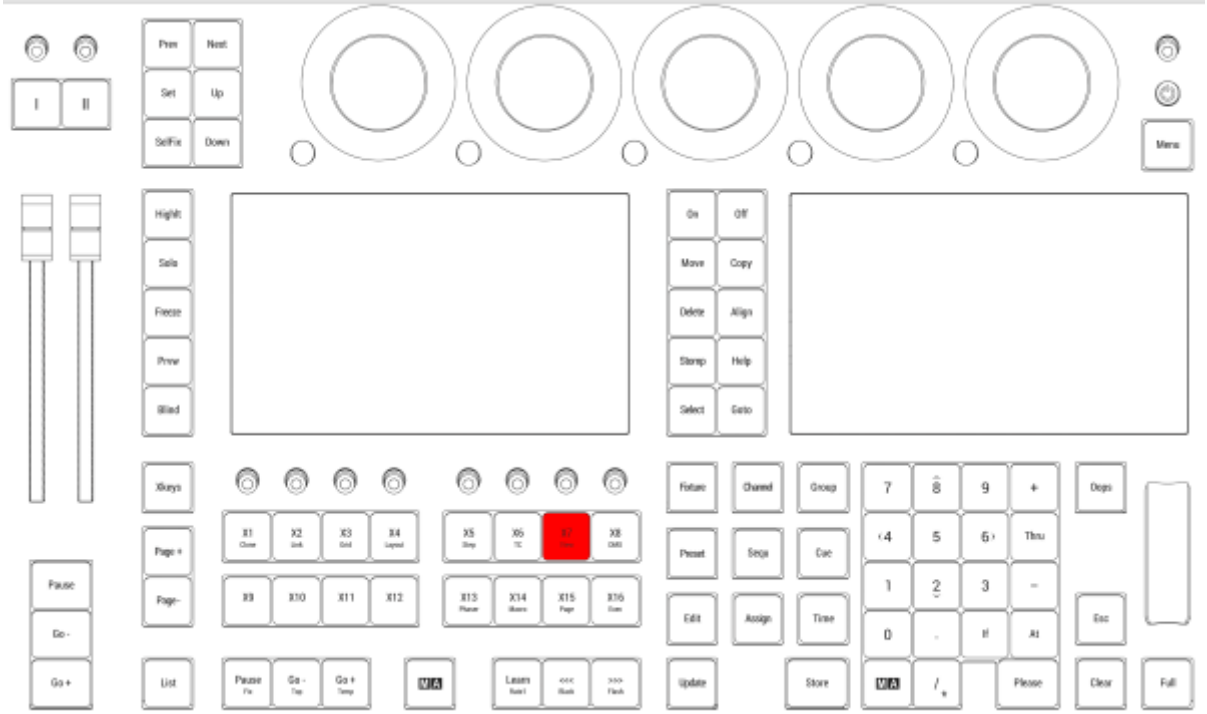
ScreenContent

Pressing and holding MA + X7 | View + X7 | View + X7 | View enters the ScreenContent keyword into the command line.

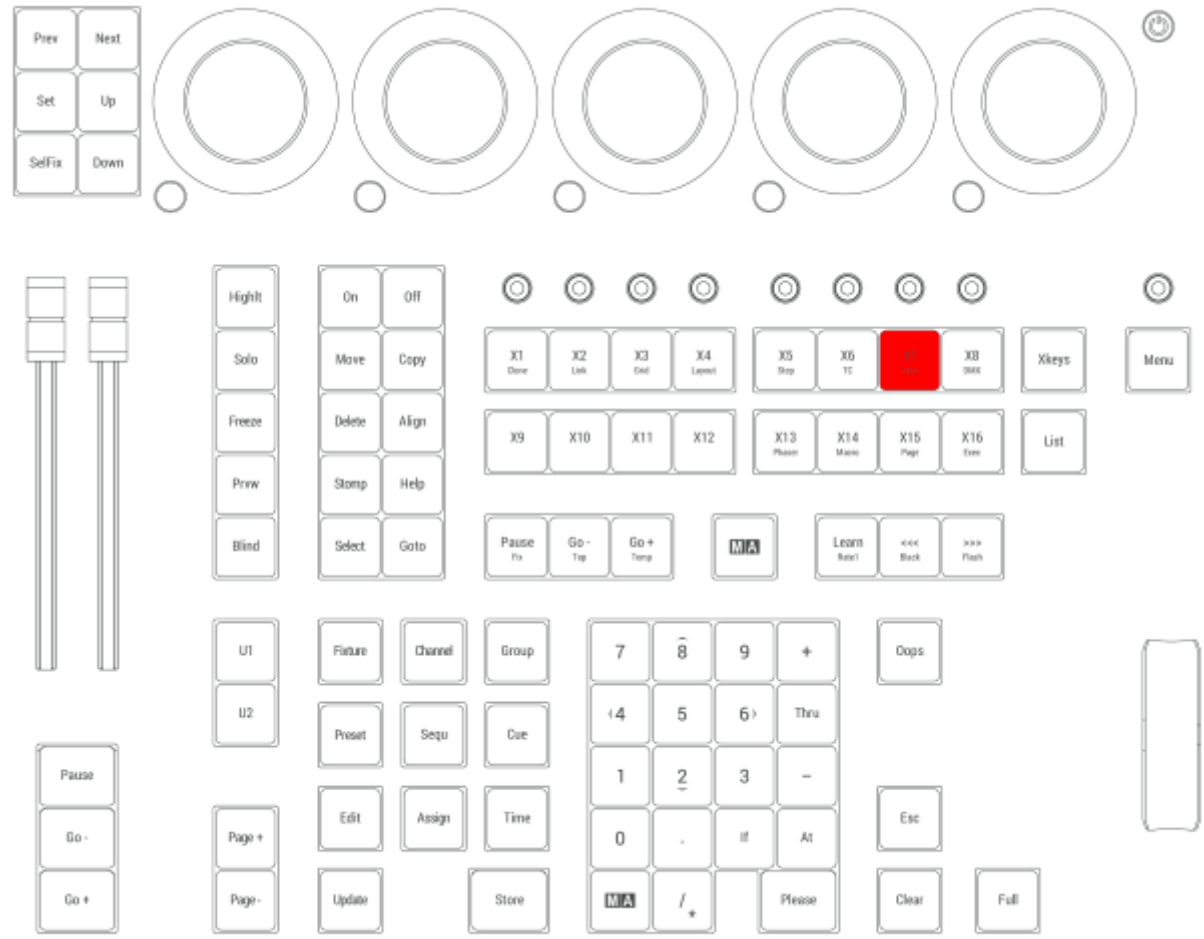
For more information about ScreenContent, see the **ScreenContent keyword**.

Location

X7 | View is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.71. X8 | DMX

X8 | DMX is executor 298.

	Hint:
	All Xkeys behave like executors.

DMXUniverse

Pressing and holding MA + X8 | DMX enters the DMXUniverse keyword into the command line.

For more information about DMXUniverse, see the **DMXUniverse keyword**.

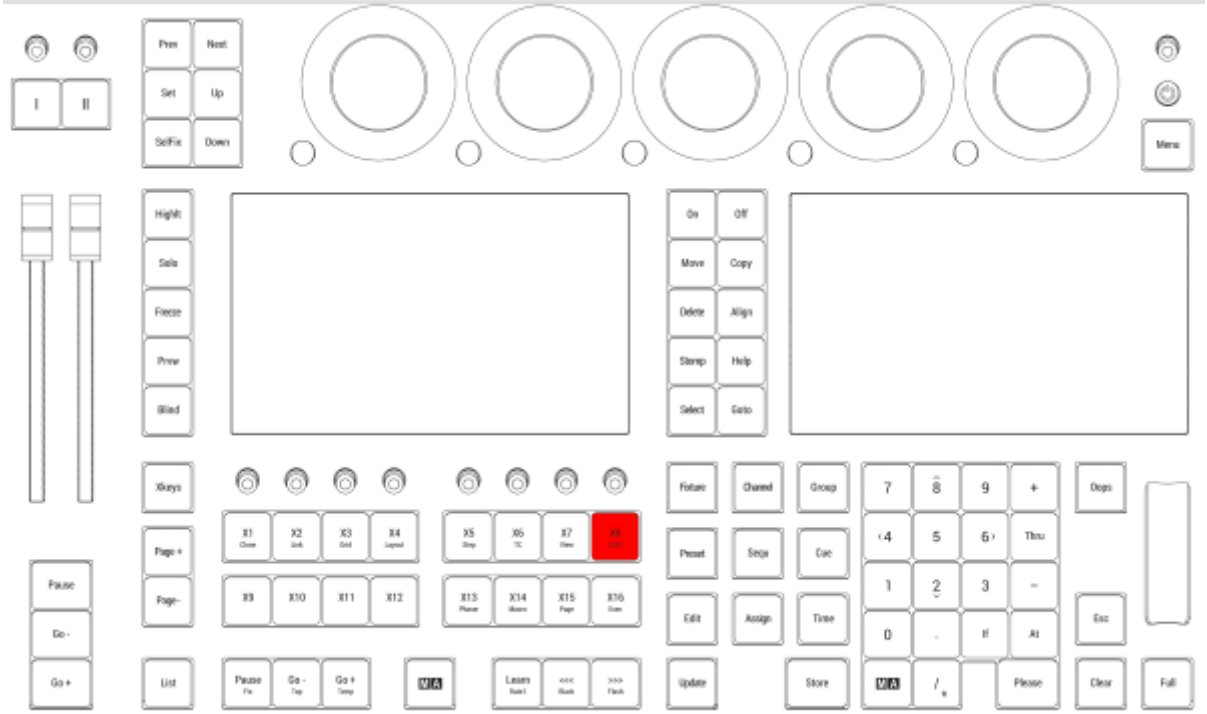
DMXAddress

Pressing and holding MA + X8 | DMX + X8 | DMX enters the DMXAddress keyword into the command line.

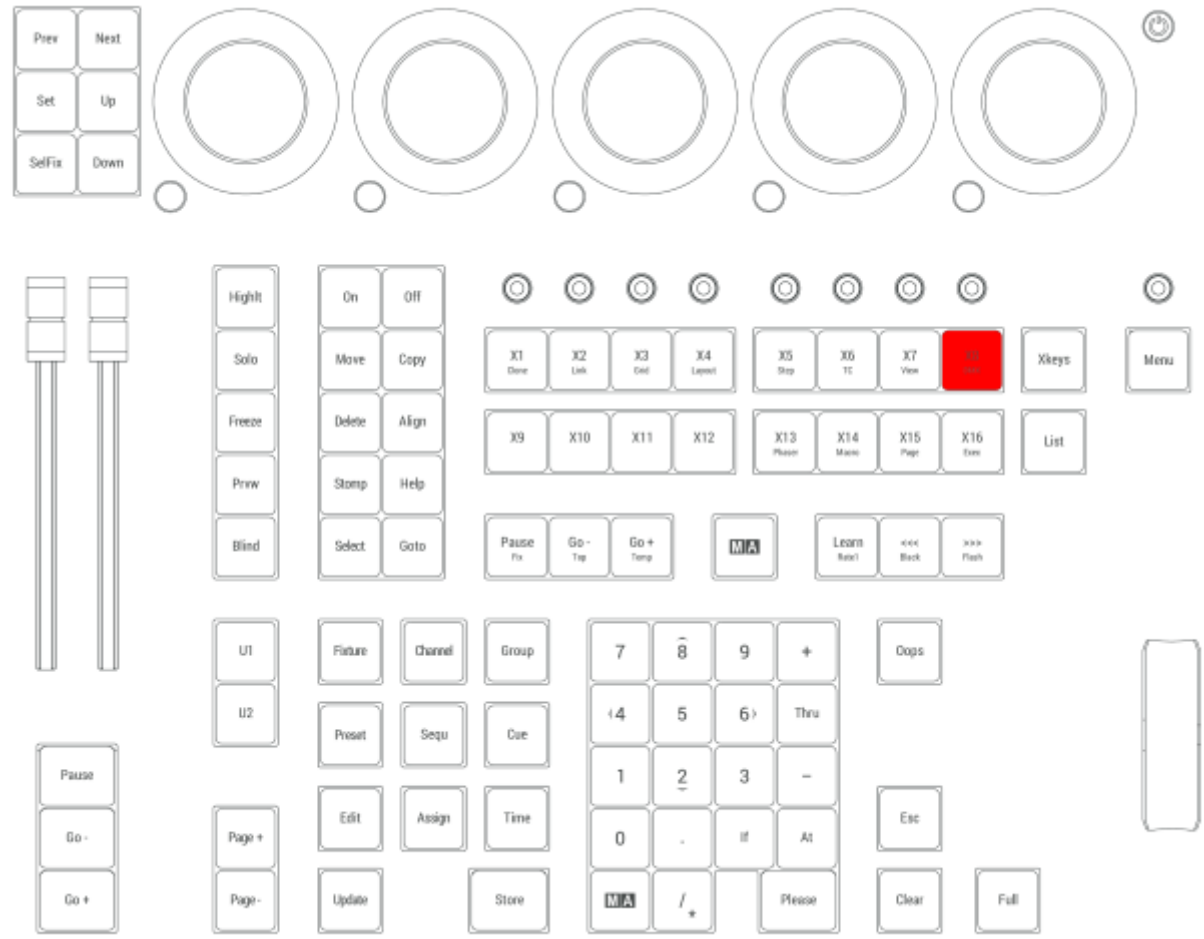
For more information about DMXAddress, see the **DMXAddress keyword**.

Location

X8 | DMX is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

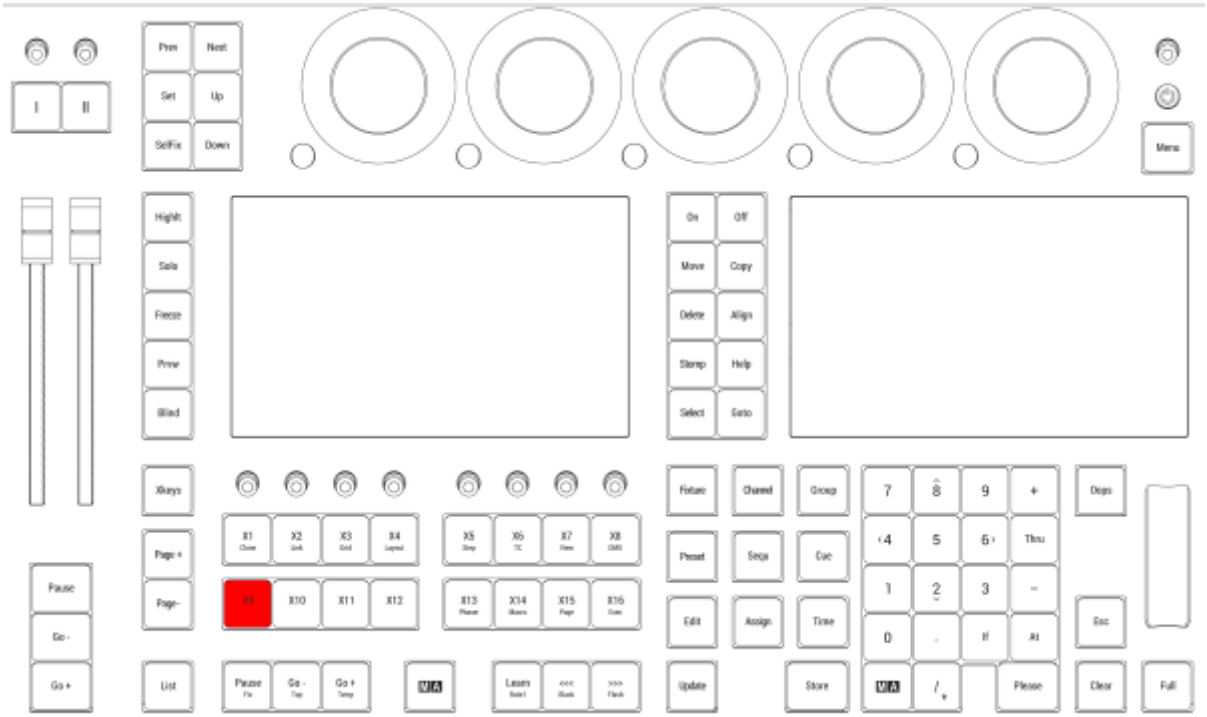
1.3.15.72. X9

X9 is executor 191.

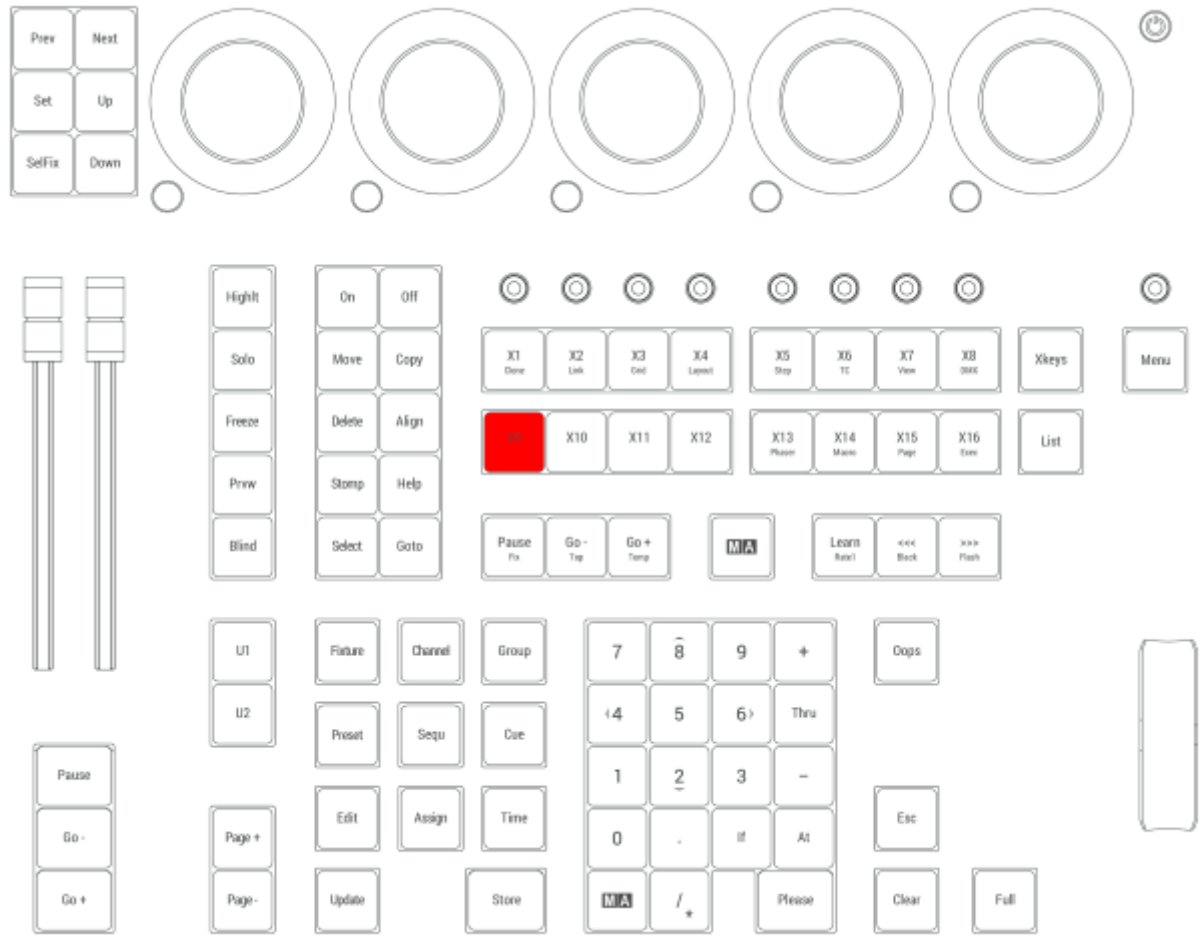
	Hint:
	All Xkeys behave like executors.

Location

X9 is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

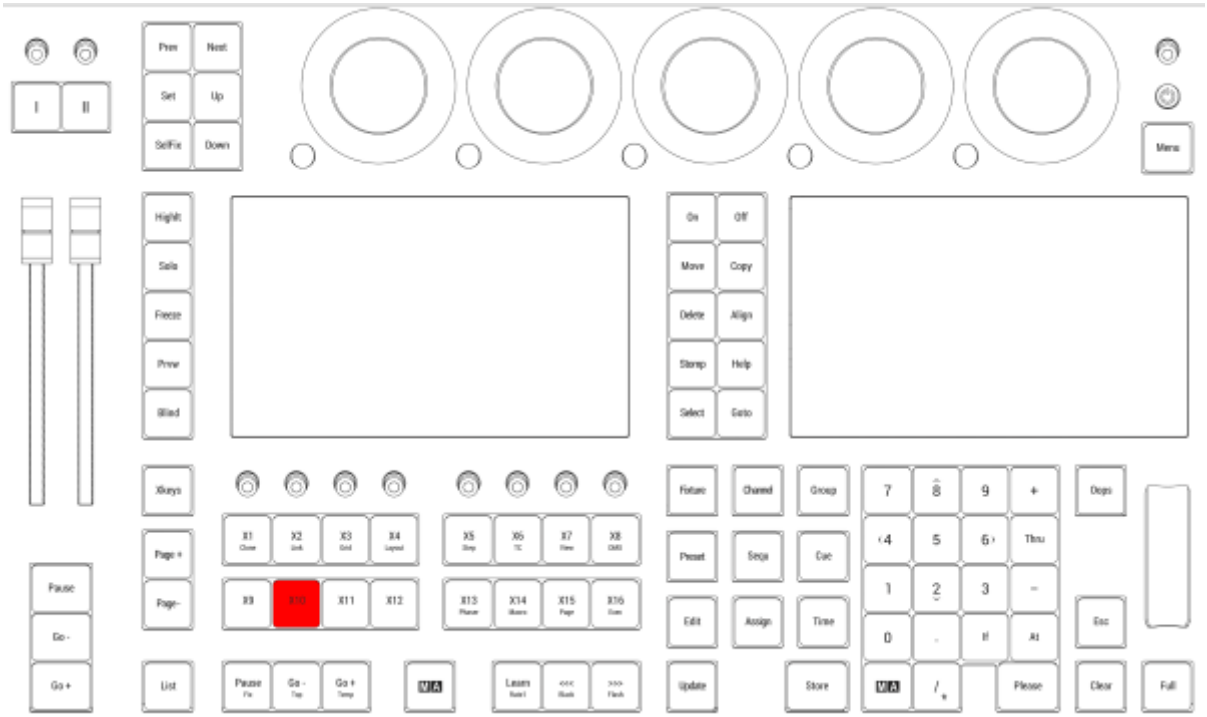
1.3.15.73. X10

X10 is executor 192.

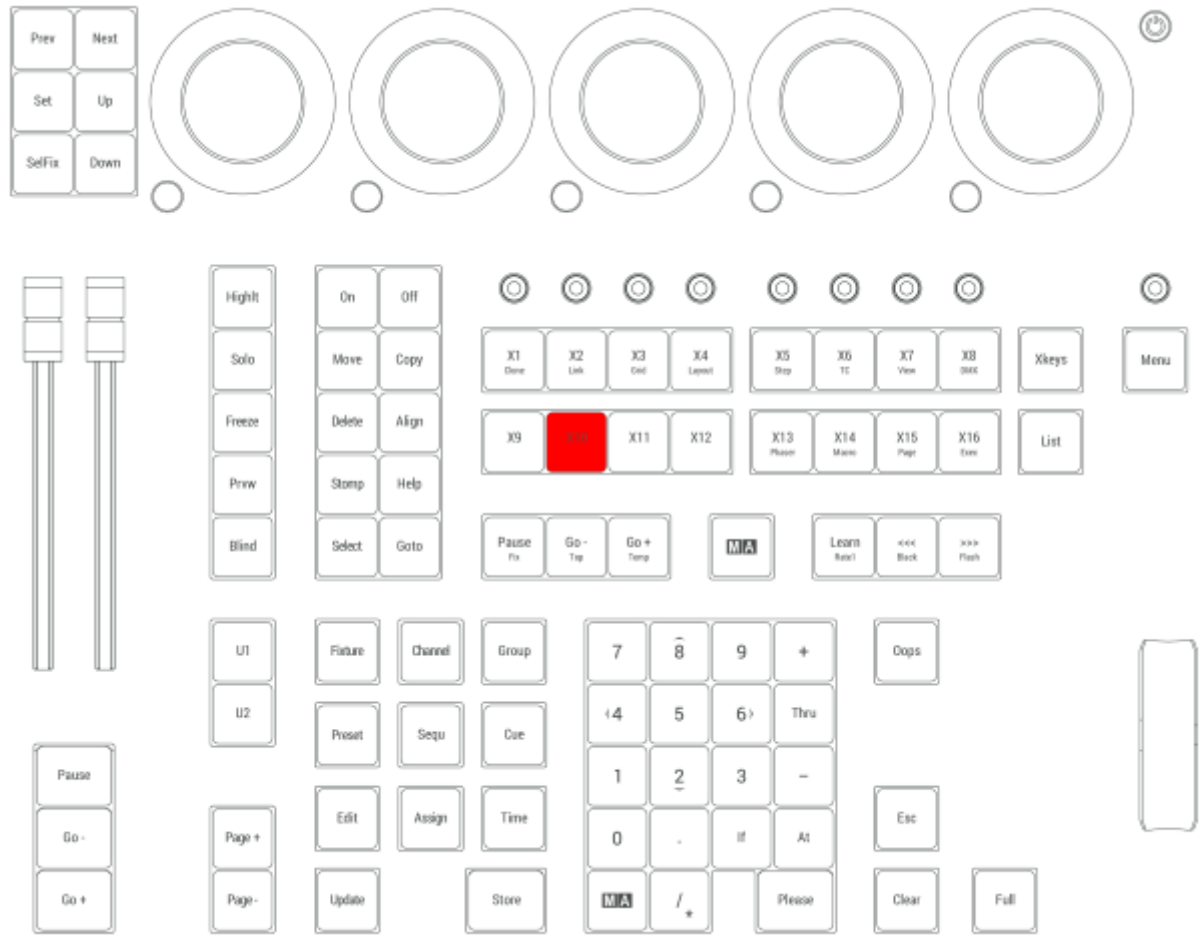
	Hint:
	All Xkeys behave like executors.

Location

X10 is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

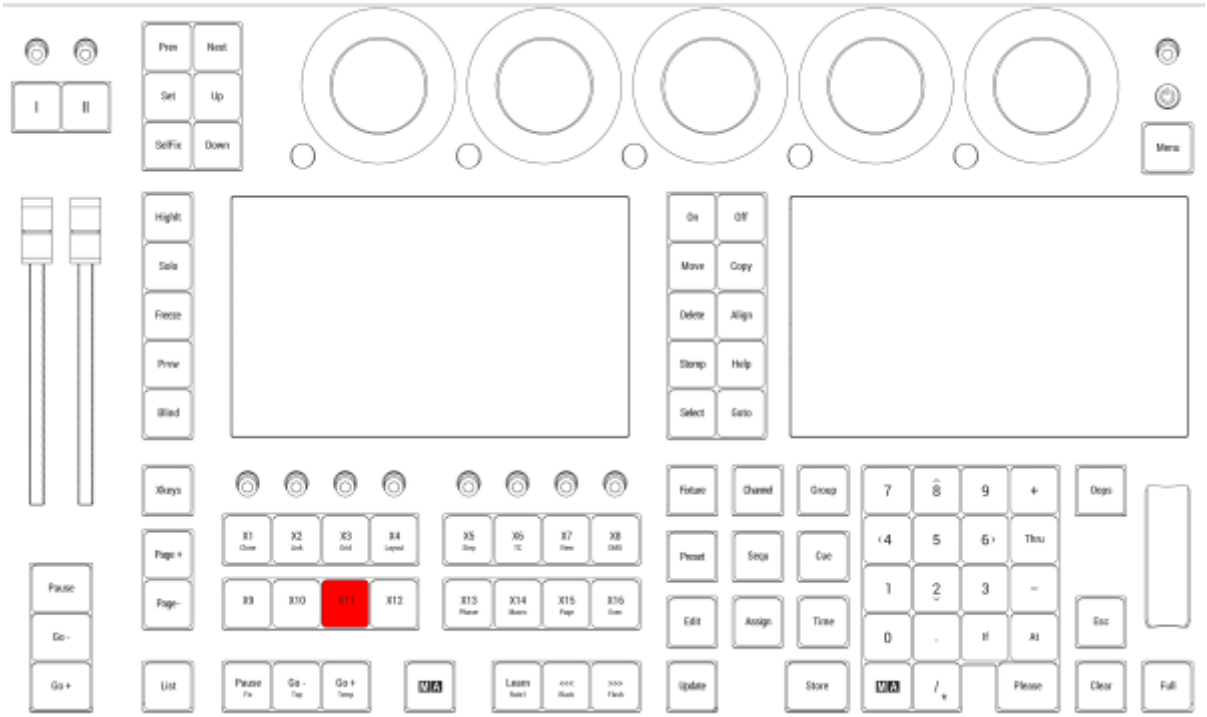
1.3.15.74. X11

X11 is executor 193.

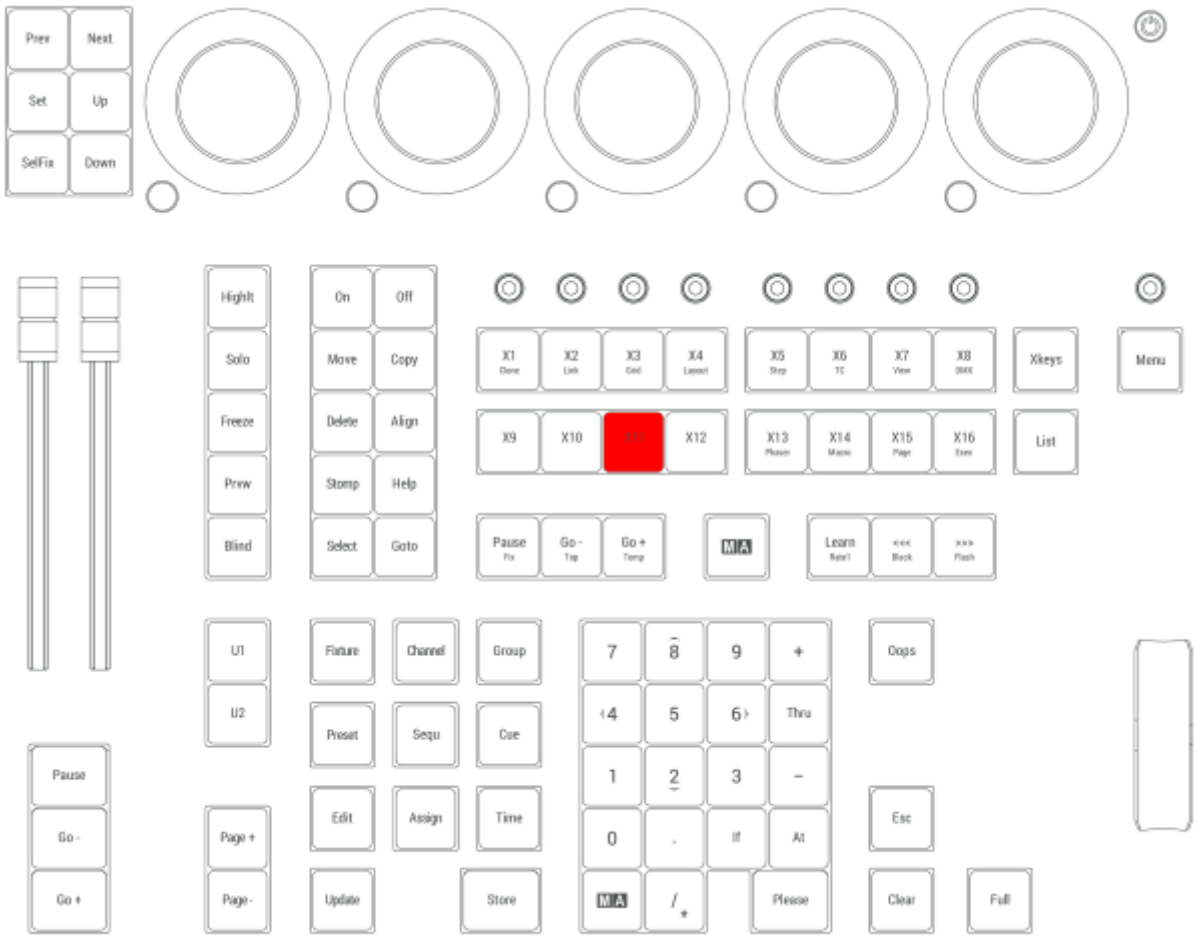
	Hint:
	All Xkeys behave like executors.

Location

X11 is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

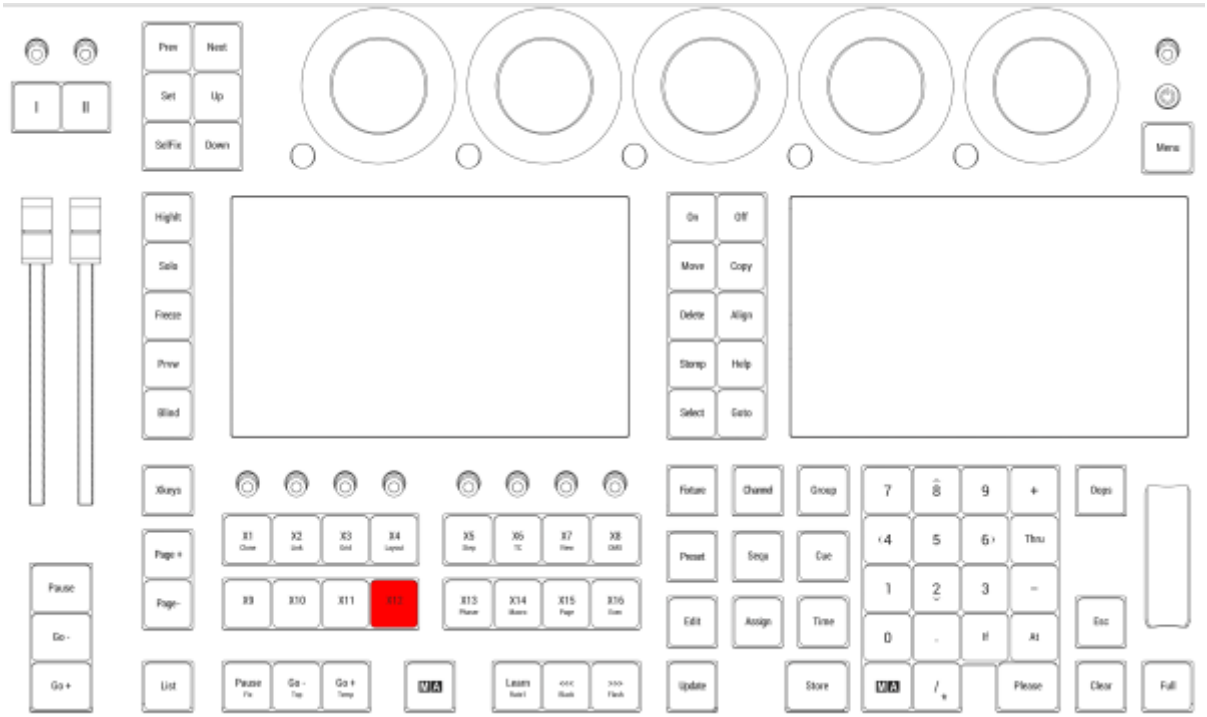
1.3.15.75. X12

X12 is executor 194.

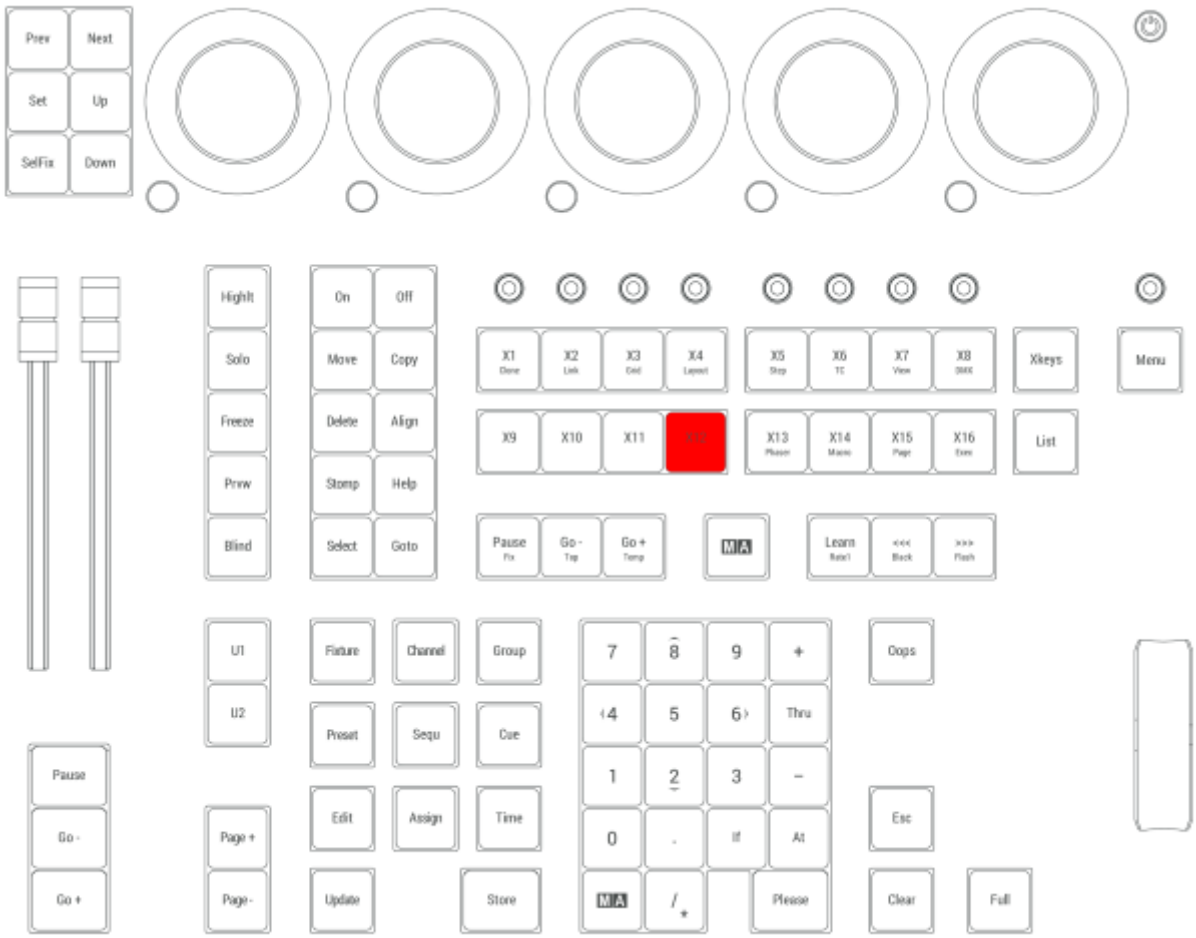
	Hint:
	All Xkeys behave like executors.

Location

X12 is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.76. X13 | Phaser

X13 | Phaser is executor 195.

	Hint:
	All Xkeys behave like executors.

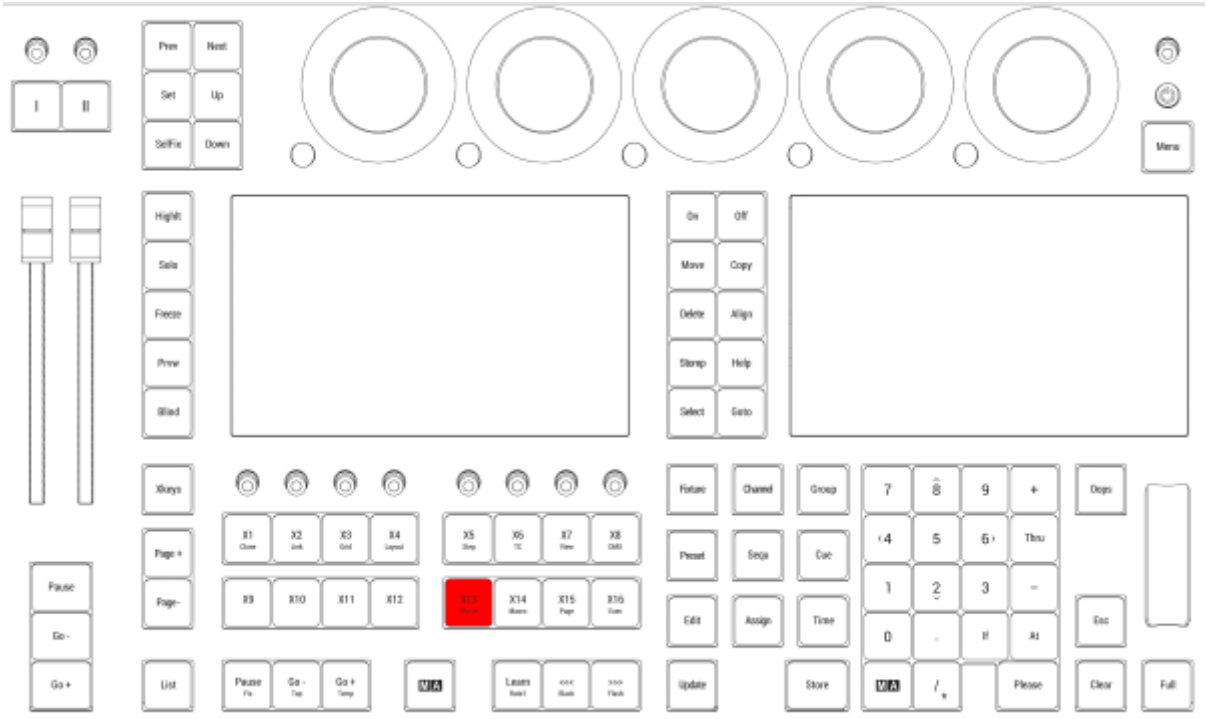
Phaser

Pressing and holding MA + X13 | Phaser opens the **Phaser Editor**.

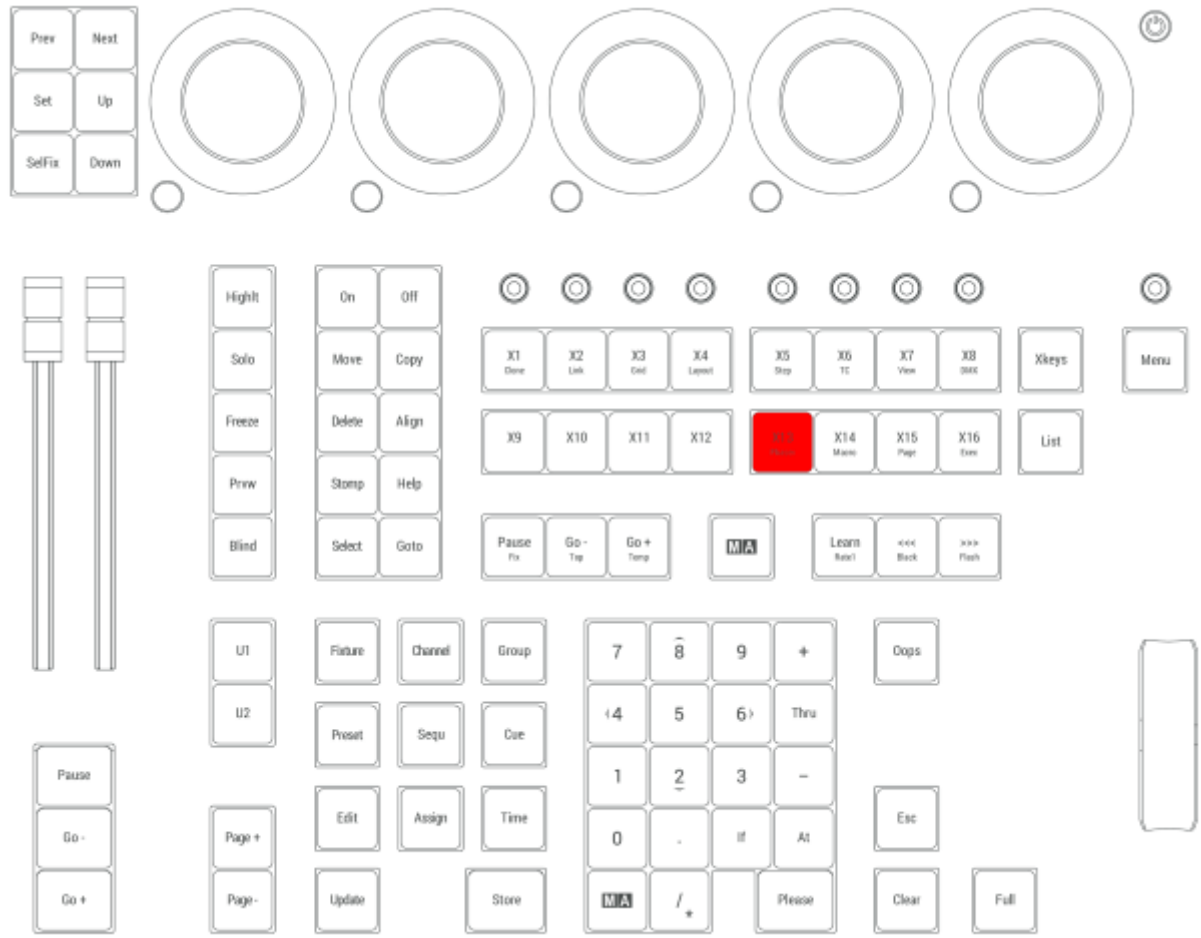
For more information about Phasers, see **Phasers** and **Phaser Editor**.

Location

X13 | Phaser is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.77. X14 | Macro

X14 | Macro is executor 196.

	Hint:
	All Xkeys behave like executors.

Macro

Pressing and holding MA + X14 | Macro enters the Macro keyword into the command line.

For more information about Macro, see the **Macro keyword**.

Plugin

Pressing and holding MA + X14 | Macro + X14 | Macro enters the Plugin keyword into the command line.

For more information about Plugin, see the **Plugin keyword**.

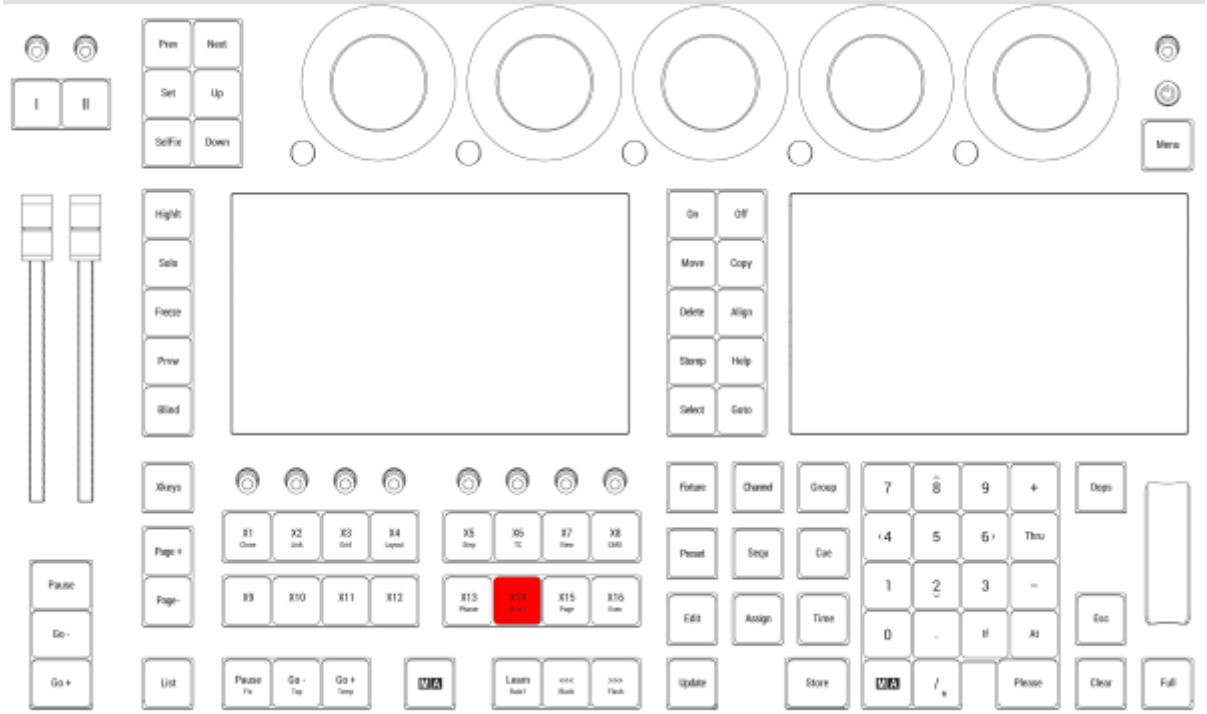
Quickey

Pressing and holding MA + X14 | Macro + X14 | Macro + X14 | Macro enters the Quickey keyword into the command line.

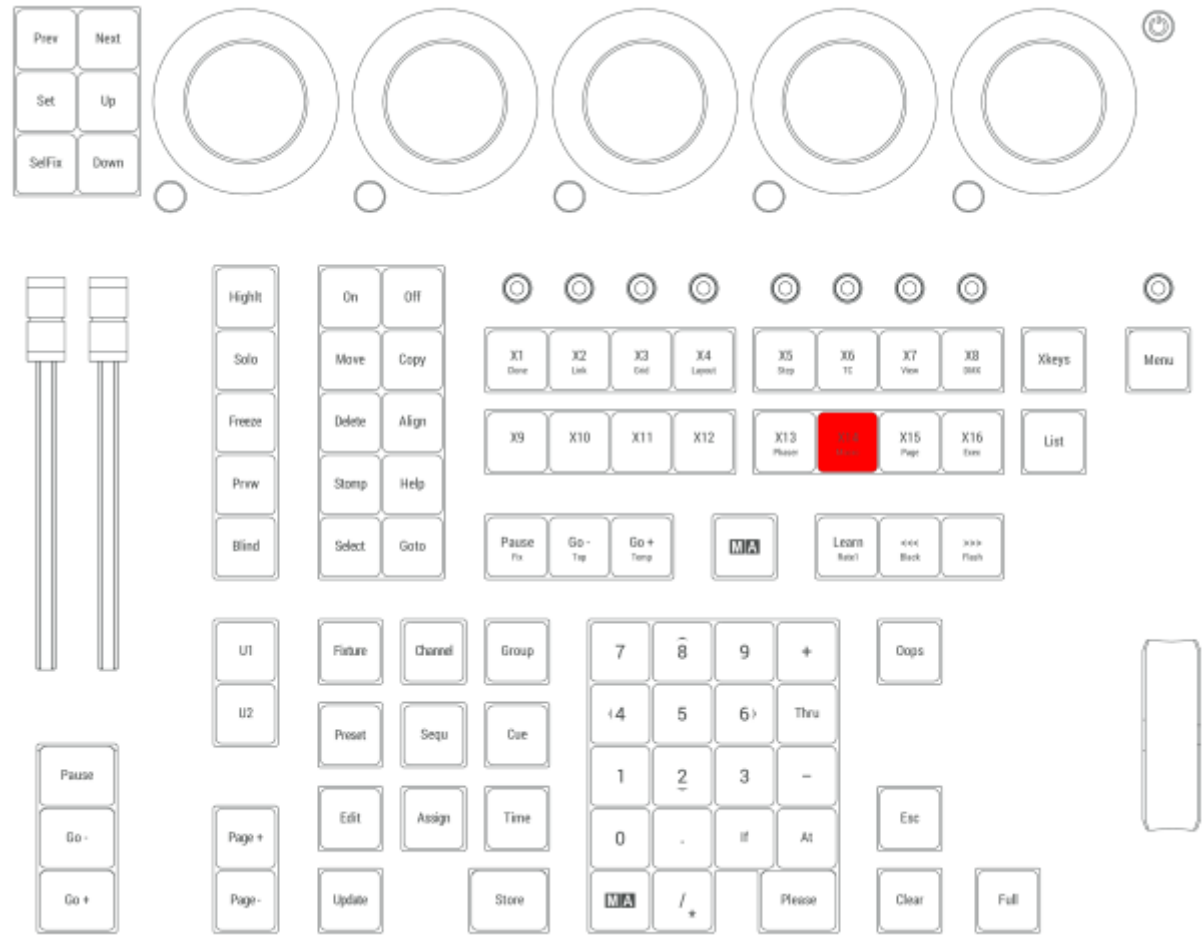
For more information about Quickey, see the **Quickey keyword**.

Location

X14 | Macro is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.78. X15 | Page

X15 | Page is executor 197.

	Hint:
	All Xkeys behave like executors.

Page

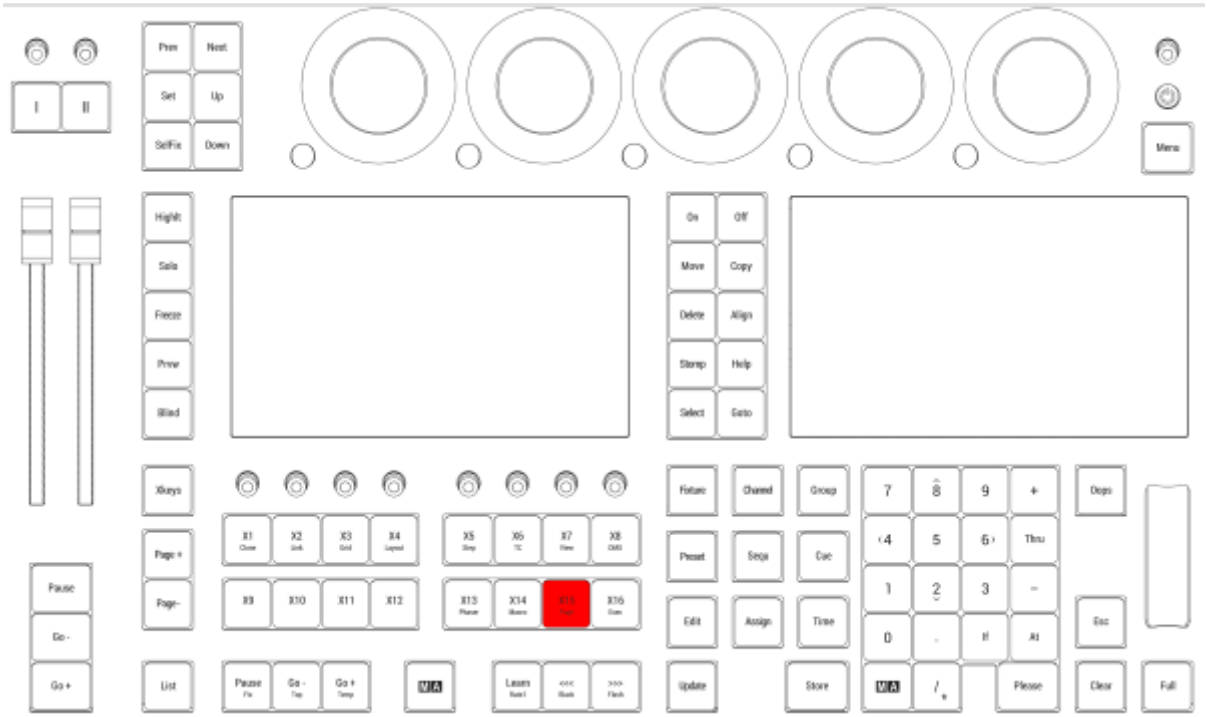
Pressing and holding MA + X15 | Page enters the Page keyword into the command line.

MA ⌨ User name[Fixture]>Page

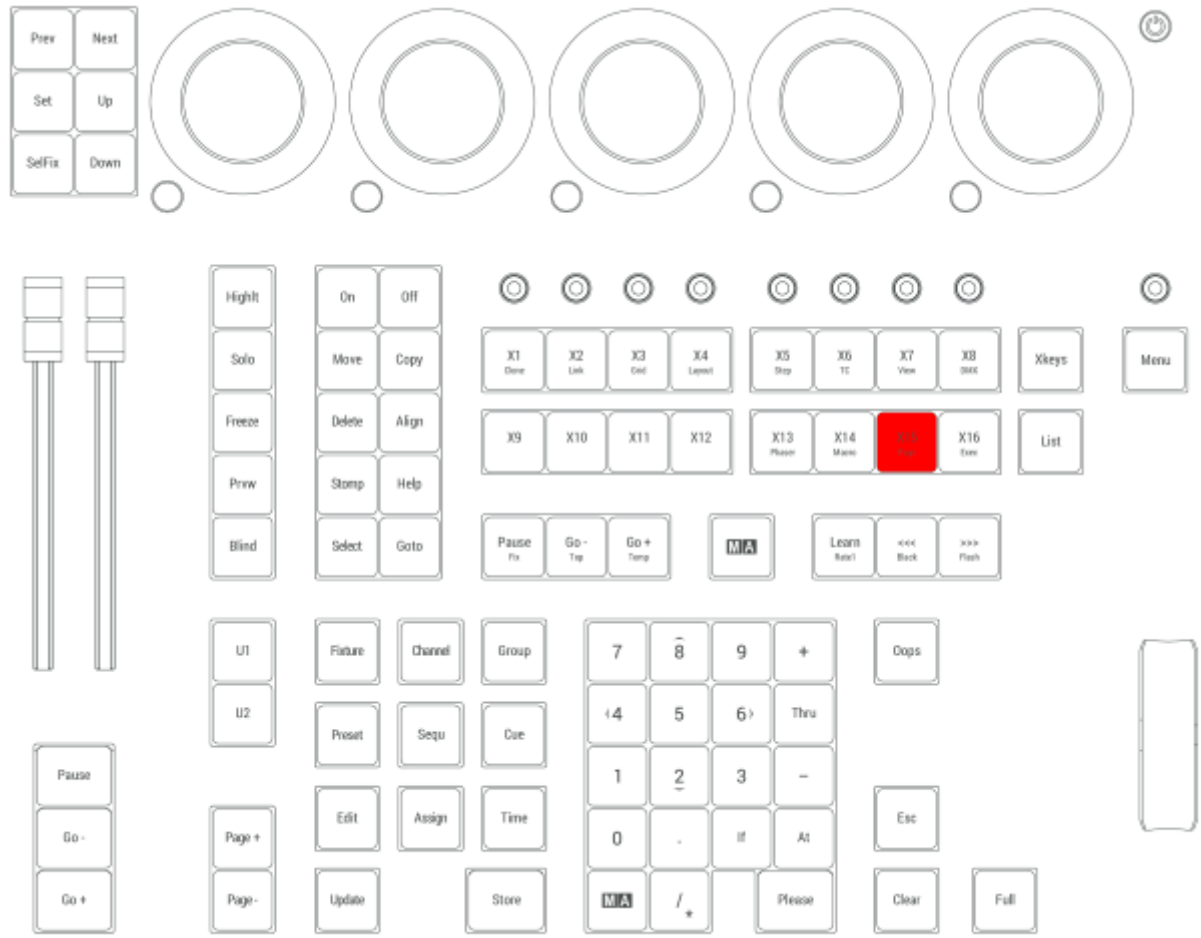
For more information about Page, see the **Page keyword**.

Location

X15 | Page is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.79. X16 | Exec

X16 | Exec is executor 198.

	**Hint:**
	All Xkeys behave like executors.

Executor

Pressing and holding MA + X16 | Exec enters the Executor keyword into the command line.

For more information about Executor, see the **Executor keyword**.

SpecialExecutor

Pressing and holding MA + X16 | Exec + X16 | Exec enters the SpecialExecutor keyword into the command line.

For more information about SpecialExecutor, see the **SpecialExecutor keyword**.

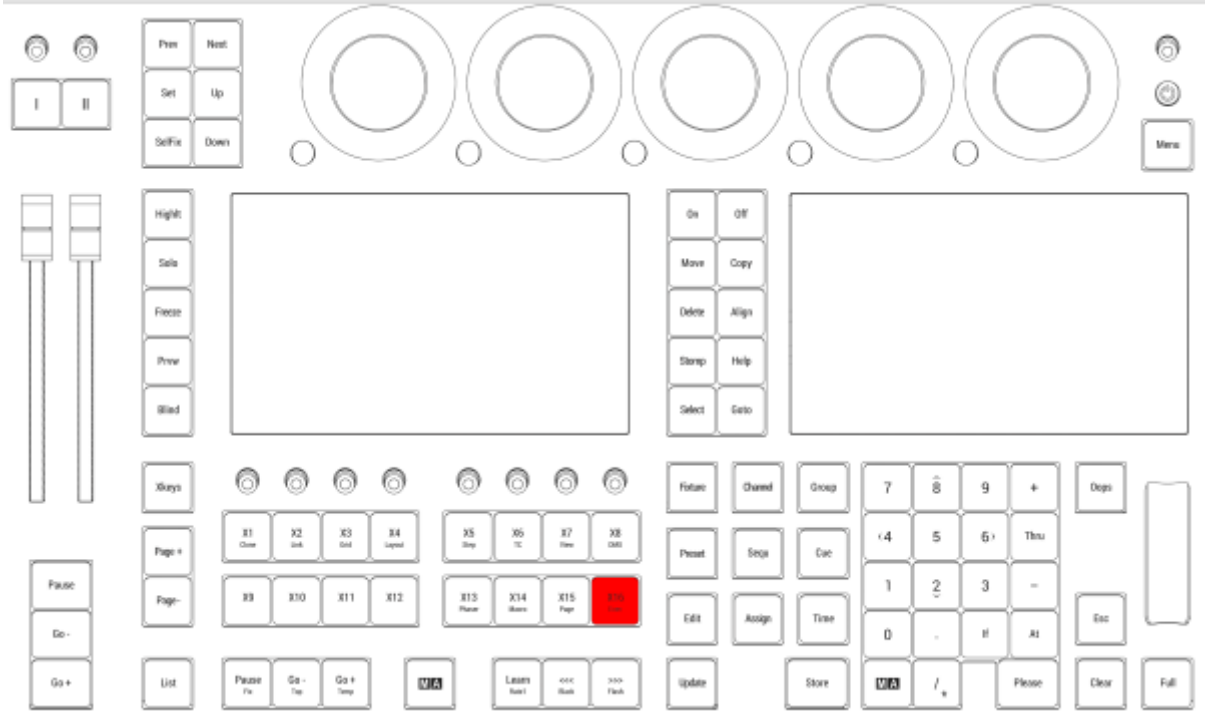
FaderMaster

Pressing and holding MA + X16 | Exec + X16 | Exec + X16 | Exec enters the FaderMaster keyword into the command line.

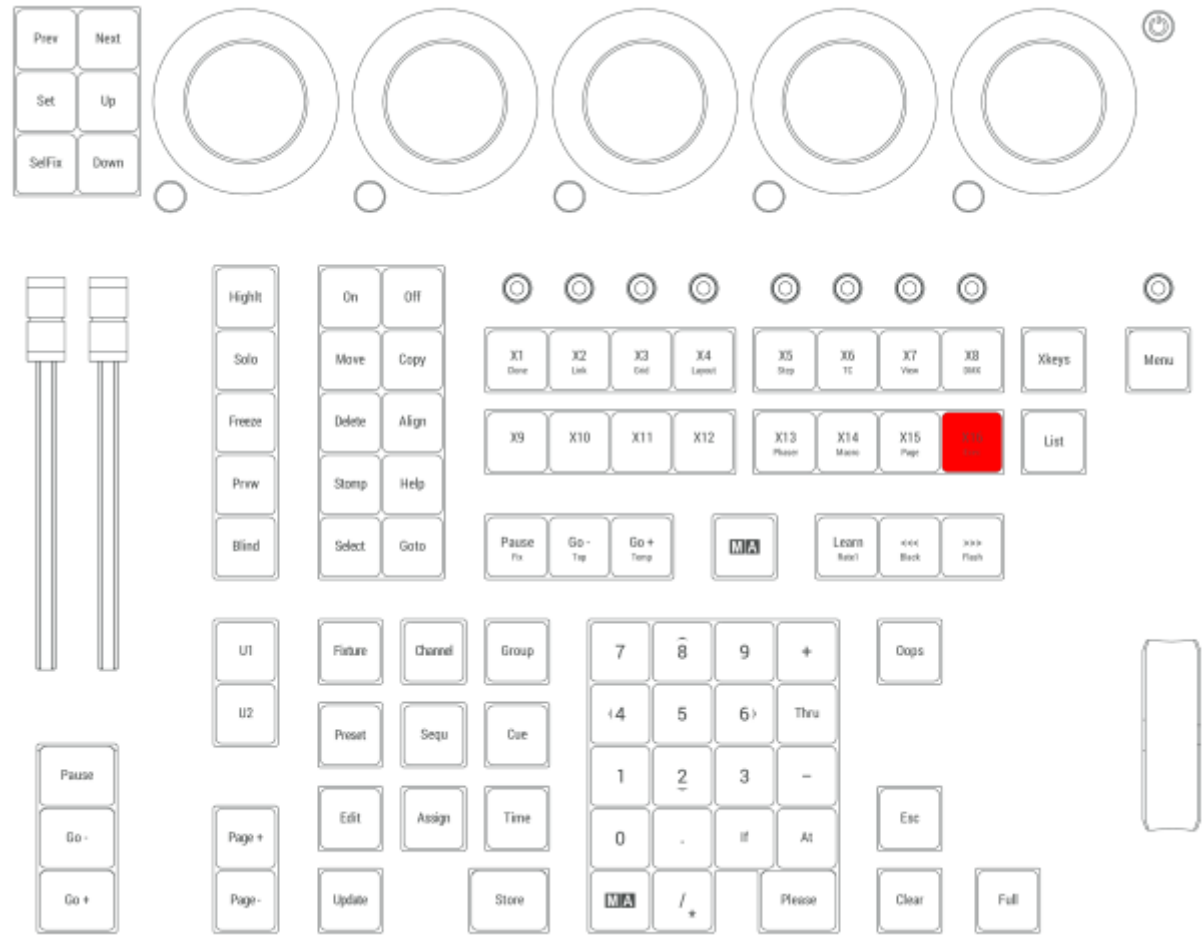
For more information about FaderMaster, see the **FaderMaster keyword**.

Location

X16 | Exec is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



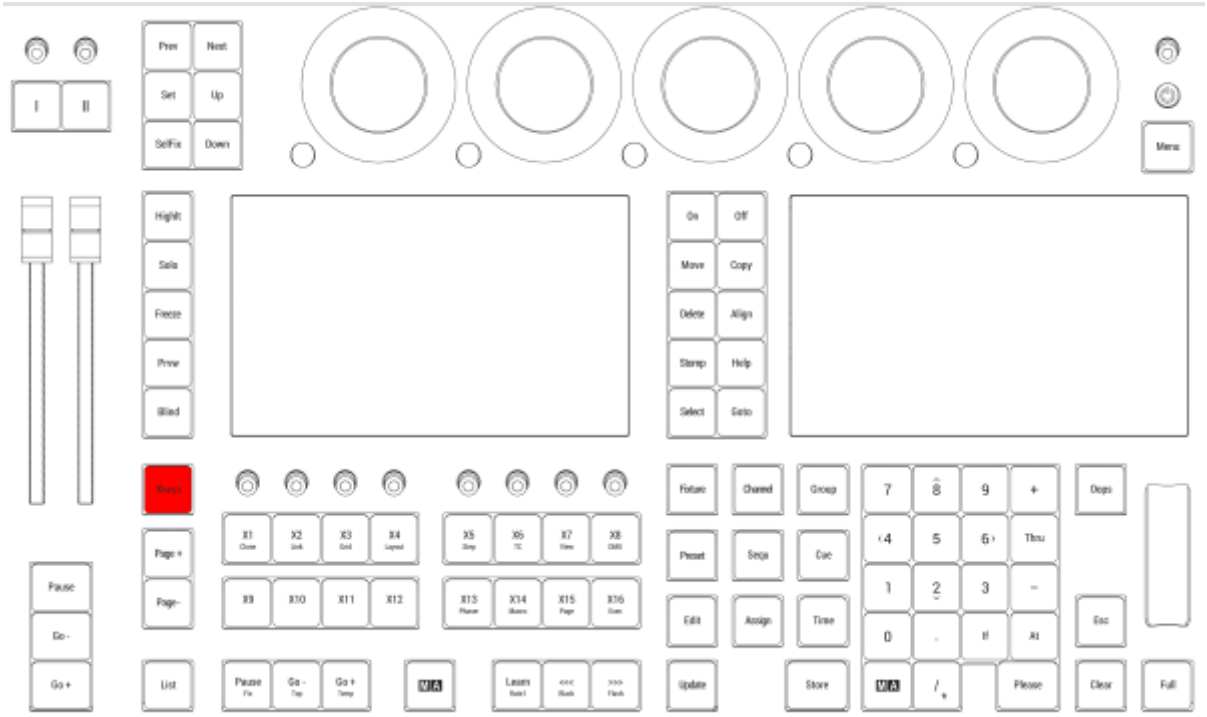
Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.15.80. Xkeys

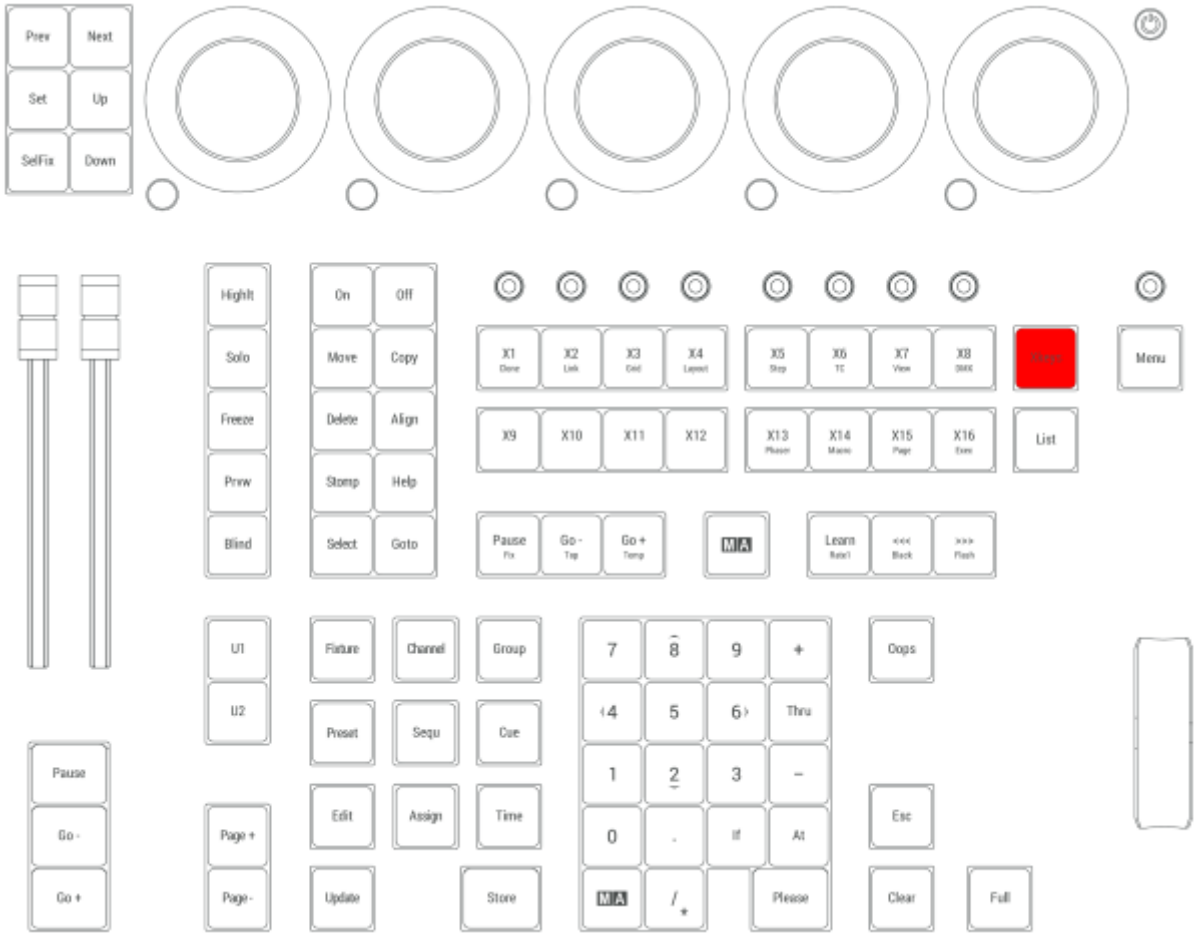
Pressing Xkeys enters the Xkeys keyword into the command line.

Location

Xkeys is located in the command section.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.16. Control Elements

Control Elements

The following chapters describe the different control elements of the grandMA3 devices.

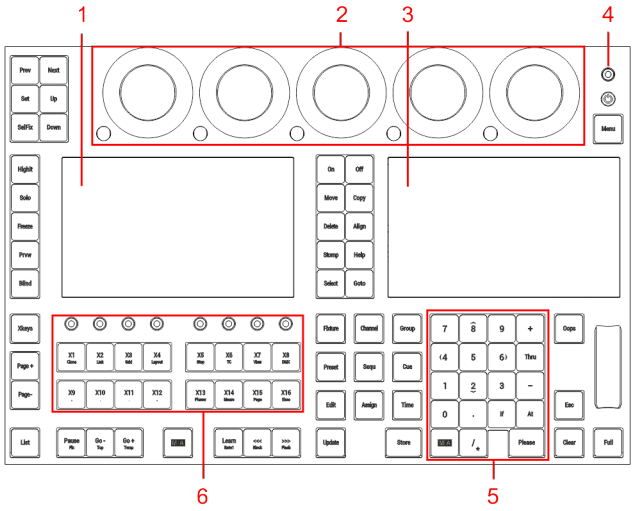
- Each chapter uses drawings of the grandMA3 product for visualization.
- Control elements described in the chapters are marked in red.

Subtopics

- **Command Area**
- **Master Area**
- **Custom Area**
- **Dual Encoders**
- **Level Wheel**
- **Grand Master**
- **Executor Elements**

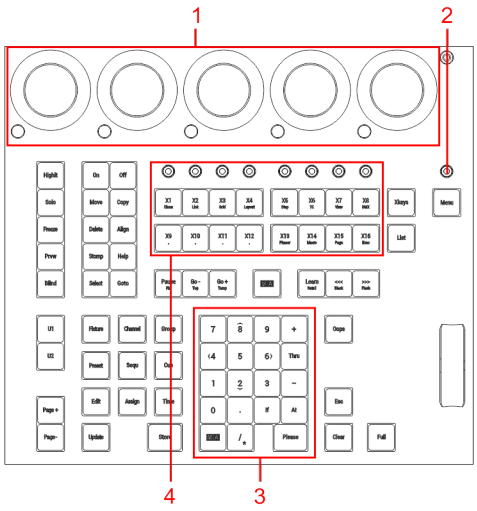
1.3.16.1. Command Area

Command Area



Command area of *grandMA3 full-size (CRV) and grandMA3 light (CRV)*

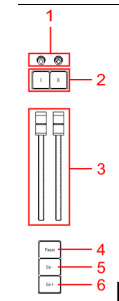
1. Left command screen
2. **Dual encoder section**
3. Right command screen
4. **Grand master**
5. Number pad
6. **Xkeys section**



Command area of *grandMA3 compact (XT) and grandMA3 onPC command wing (XT)*

1. **Dual encoder section**
2. **Grand master**
3. Number pad
4. **Xkeys section**

1.3.16.2. Master Area



Master area *grandMA3 full-size (CRV)*

1. Master knobs 1+2
2. Master keys 1+2
3. Master faders 1+2
4. Pause key
5. Go- key
6. Go+ key

Go+, Go- and Pause

The Go+ [Large] key, the Go- [Large] key, and the Pause [Large] key are located in the lower part of the master area. These keys execute the Go+, Go- or Pause command for the selected sequence.

For more information about Go+, Go- and Pause, please see the **Go+[Large] key**, **Go-[Large] key**, or the **Pause[Large] key** topic.

Master Fader 1 and Master Fader 2

The two faders in the master area are master fader 1 and master fader 2.

By default, master fader 1 is set to the master function for the selected sequence.

By default, master fader 2 is set to the crossfade function for the selected sequence.

- To change the function of the faders, please read below.

Master Key 1 and Master Key 2, Master Knob 1 and Master Knob 2

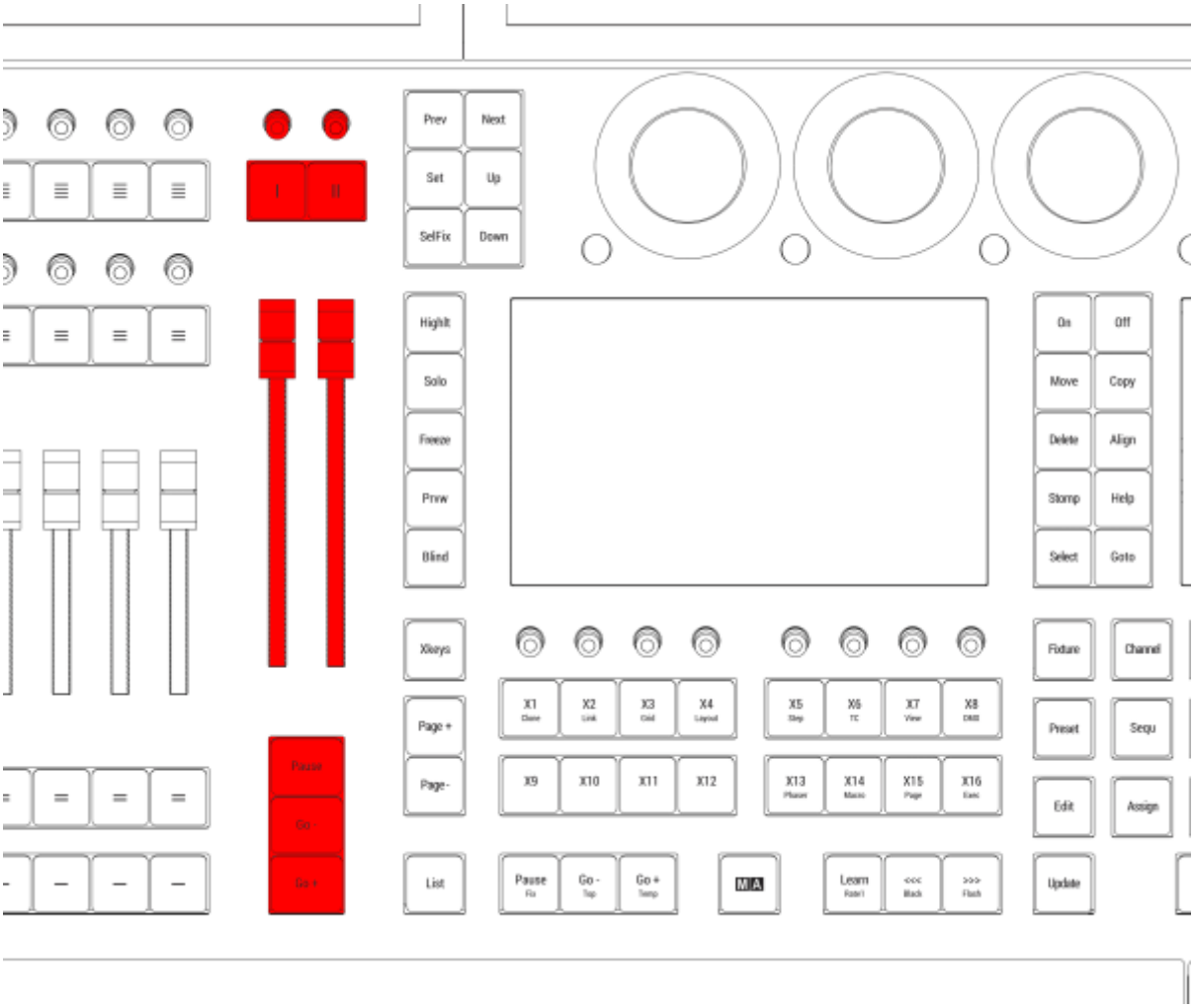
	Restriction:
	Master keys and master knobs are only available on grandMA3 full-size (CRV) and grandMA3 light (CRV) consoles.

The master keys 1+2 and the master knobs 1+2 are located in the upper part of the master area.

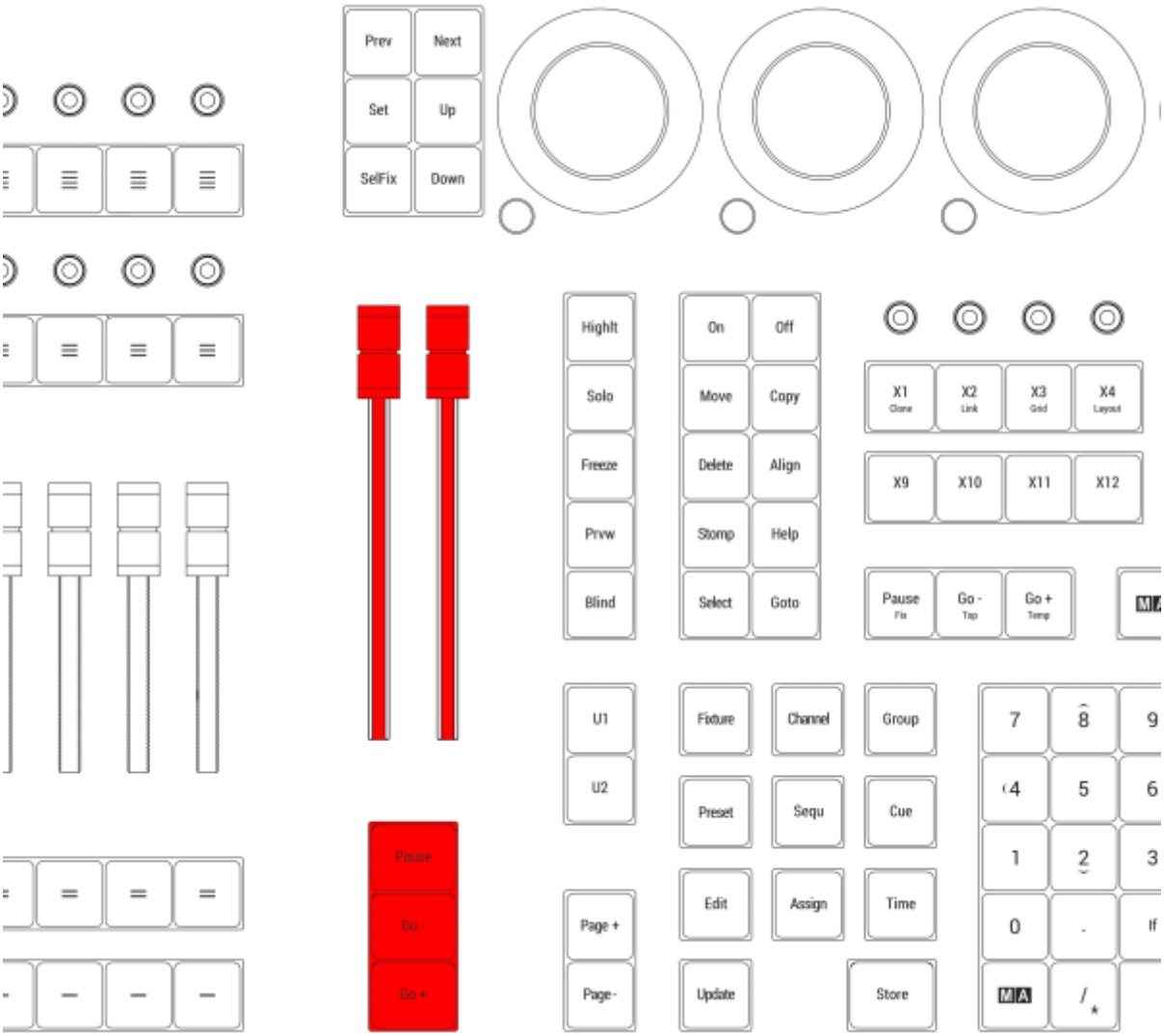
By default, these are set to toggle **Highlight** and **Solo** on and off and set the intensity of these.

- To change the function of these buttons and knobs, please read the topic **Special Executors**.

The master area is located between the **command area** and the **executor area**.



Location on grandMA3 full-size (CRV) and grandMA3 light (CRV) consoles

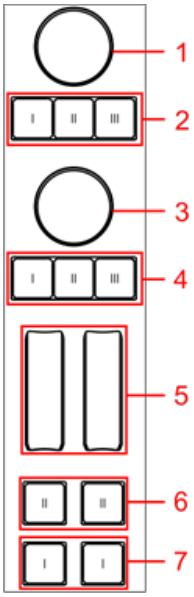


Location on grandMA3 compact consoles and grandMA3 onPC command wings

1.3.16.3. Custom Area

	Restriction:
	The custom area is only available on grandMA3 full-size, grandMA3 full-size CRV, and grandMA3 extension consoles.

Custom Area



Custom area

1. Upper encoder
2. Upper encoder keys 1-3
3. Lower encoder
4. Lower encoder keys 1-3
5. Wheels 1+2
6. Wheels 1+2 upper keys
7. Wheels 1+2 lower keys

To change the function of these control elements, please read the topic **Special Executors**.

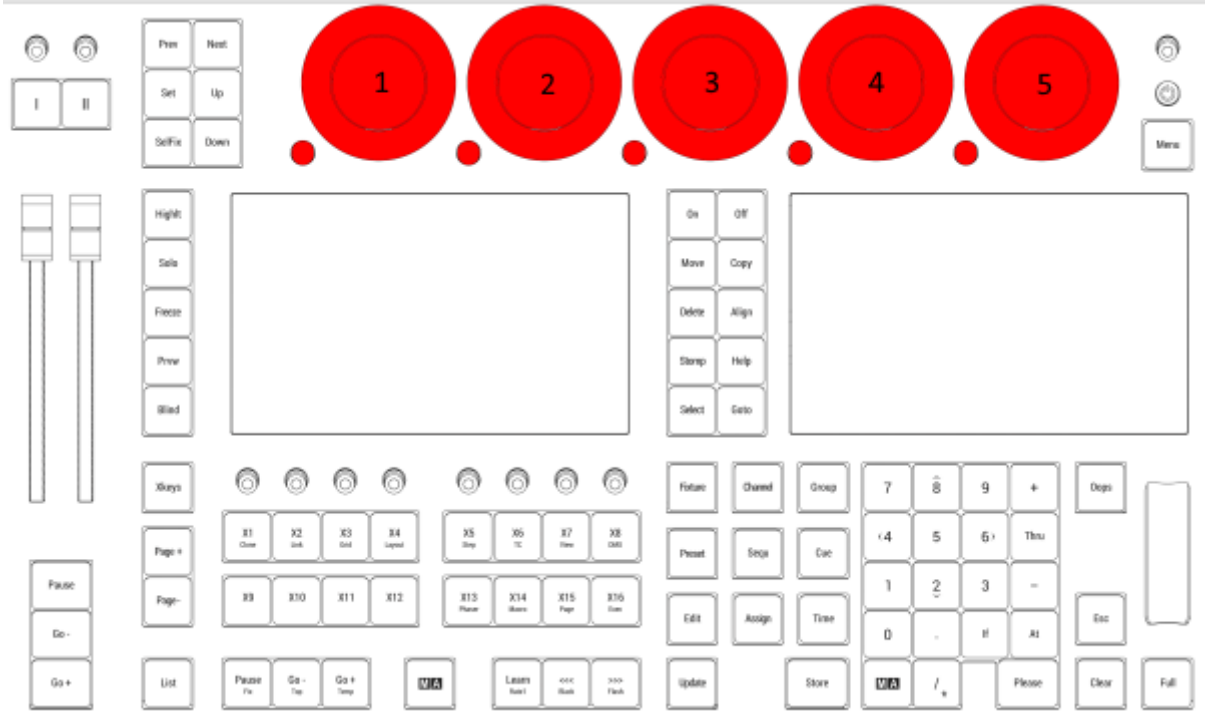
1.3.16.4. Dual Encoders

The five dual encoders are used to adjust the different attributes of the fixtures. There are four different actions on the encoders:

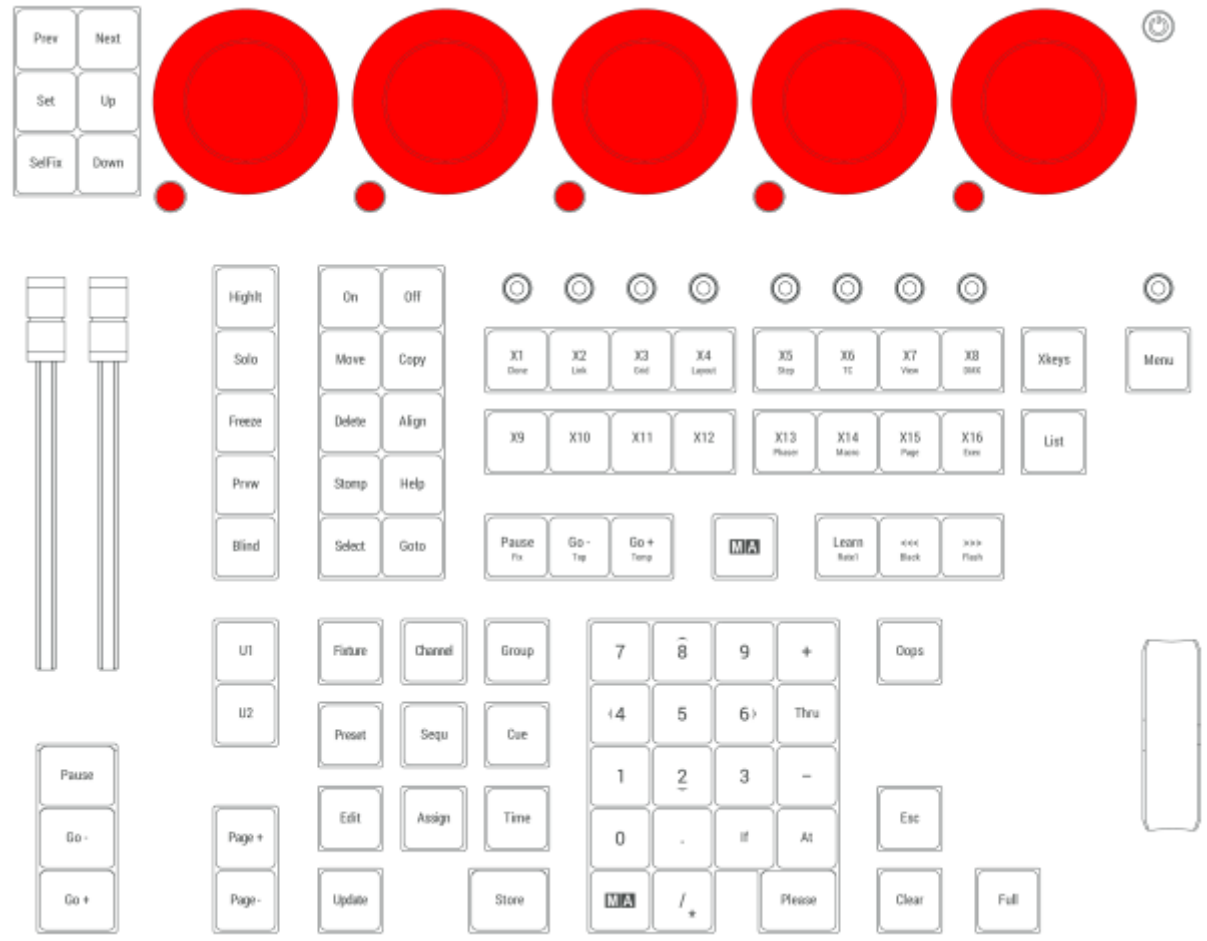
- Turn the inner ring to adjust the attribute "Coarse".
- Turn the outer ring to adjust the attribute "Fine".
- Press the inner ring to open a calculator to type in the value.
- Press and turn the inner ring to adjust the attribute "Coarse" with a higher speed.
- Press the dual encoder key to open a calculator to type in the value. The dual encoder key replaces the press function of the outer ring.

	**Hint:**
	For more information on the Readout and Resolution of the encoders, see **Readout Keyword** and **Encoder Resolution** topics.

These encoders are placed above the command section.



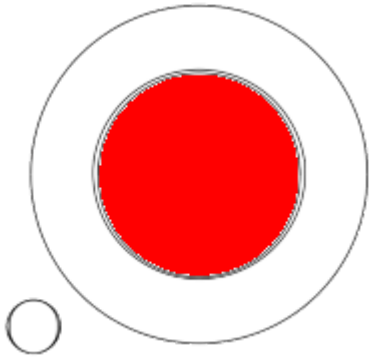
Location on grandMA3 full-size and grandMA3 light consoles



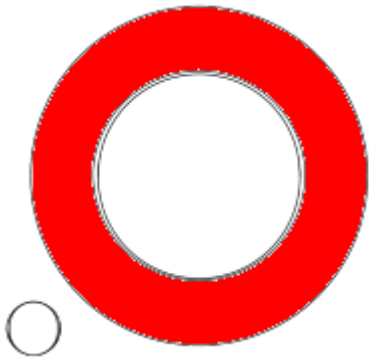
Location on grandMA3 compact consoles and grandMA3 onPC command wing and command wing XT

Each of the dual encoders is numbered 1 to 5 from left to right. There are three elements in each encoder:

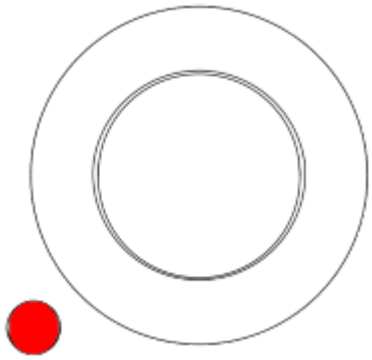
The inner ring



The outer ring



The dual encoder key

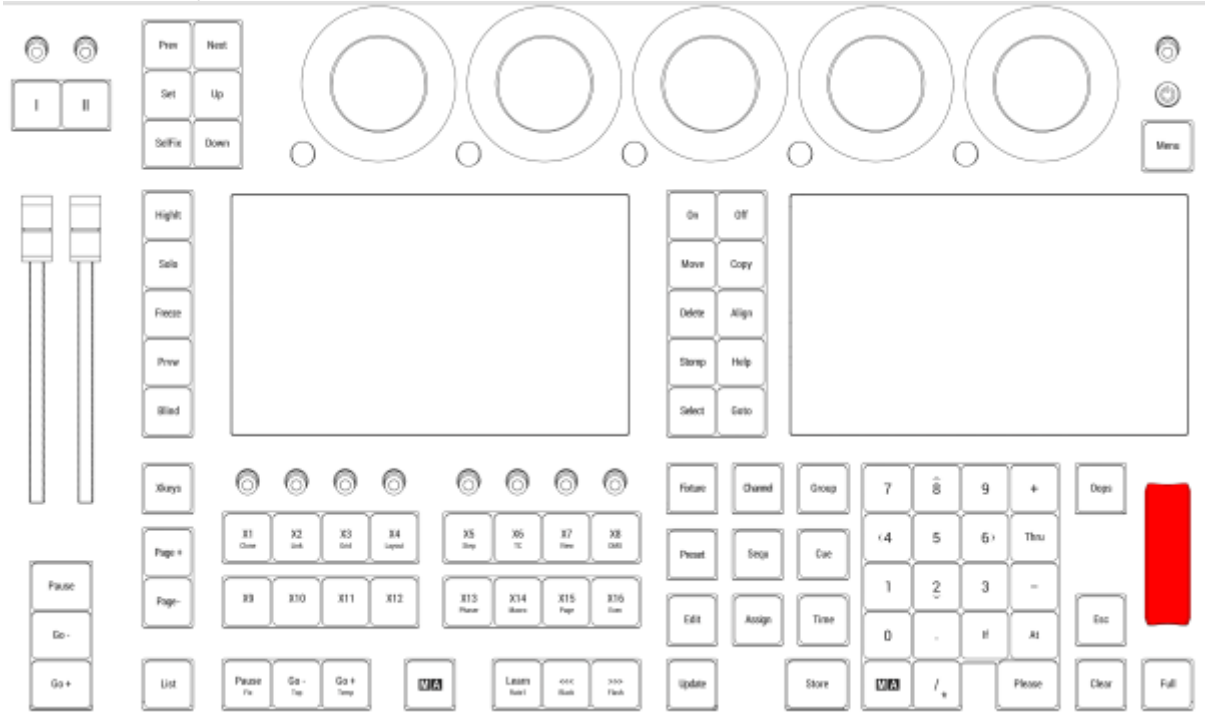


1.3.16.5. Level Wheel

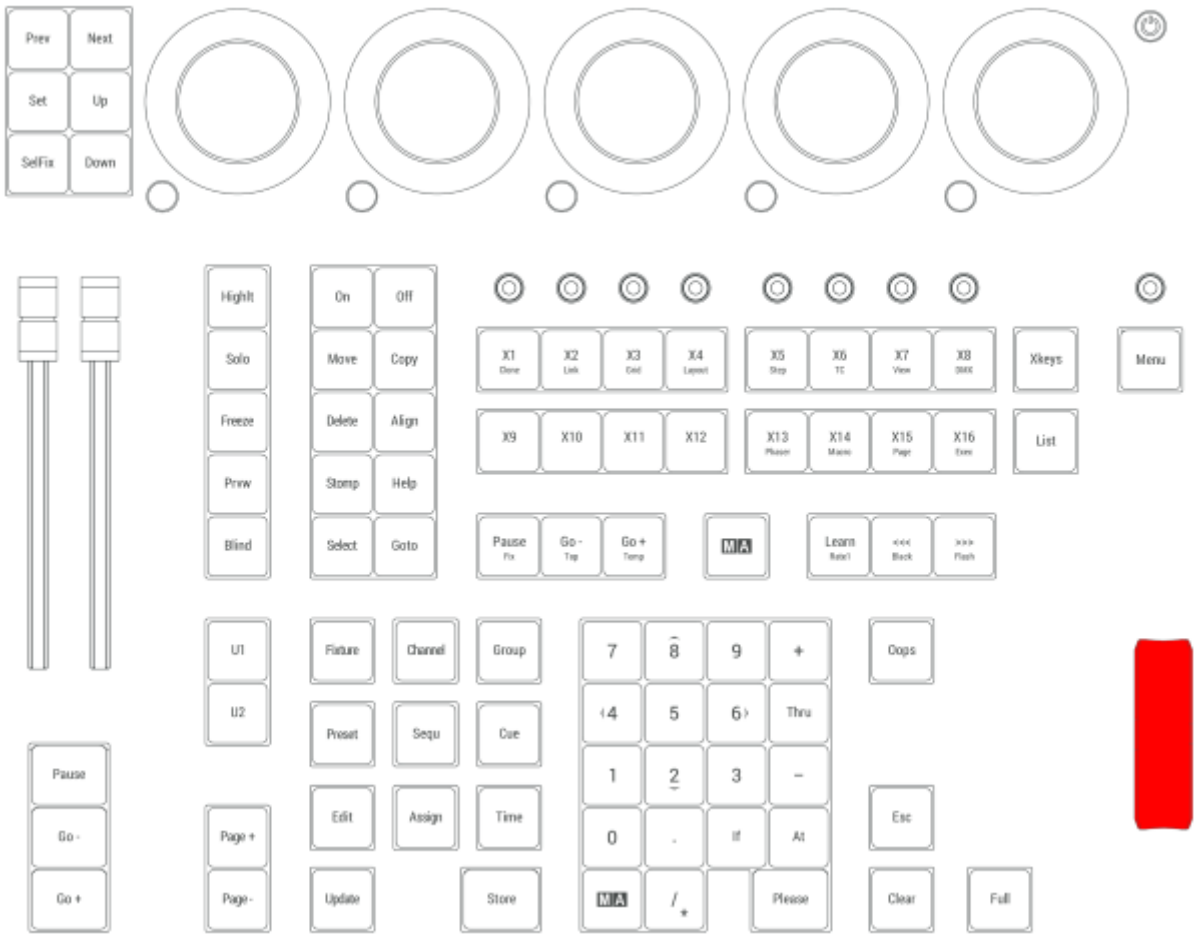
The level wheel is used to adjust the intensity of the selected fixtures.

The level wheel is placed on the right side of the numeric keys.

- To adjust the wheel, see Wheel Mode and Wheel Resolution in the **User settings**.



Location on grandMA3 full-size and grandMA3 light consoles



Location on grandMA3 compact consoles and grandMA3 onPC command wing/command wing XT

1.3.16.6. Grand Master

The Grand Master is used to limit the output of the intensity of all the fixtures patched in the show.

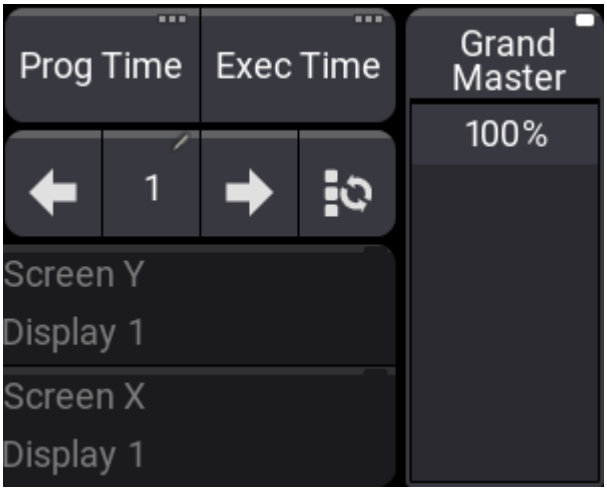
By default, turning the knob adjusts intensity. This can be changed in the Special Executor Configuration menu. To access the Special Executor Configuration menu:

1. Press Assign.
2. Press Grand Master knob.

This menu can also be opened using the SpecialExecutor keyword in the command line. For more information about SpecialExecutor, see the **SpecialExecutor keyword** topic.

To read more about Executor Configuration, see the **Executor Configuration** topic.

The Grand Master is also displayed on the right side of the encoder bar on screen 1:



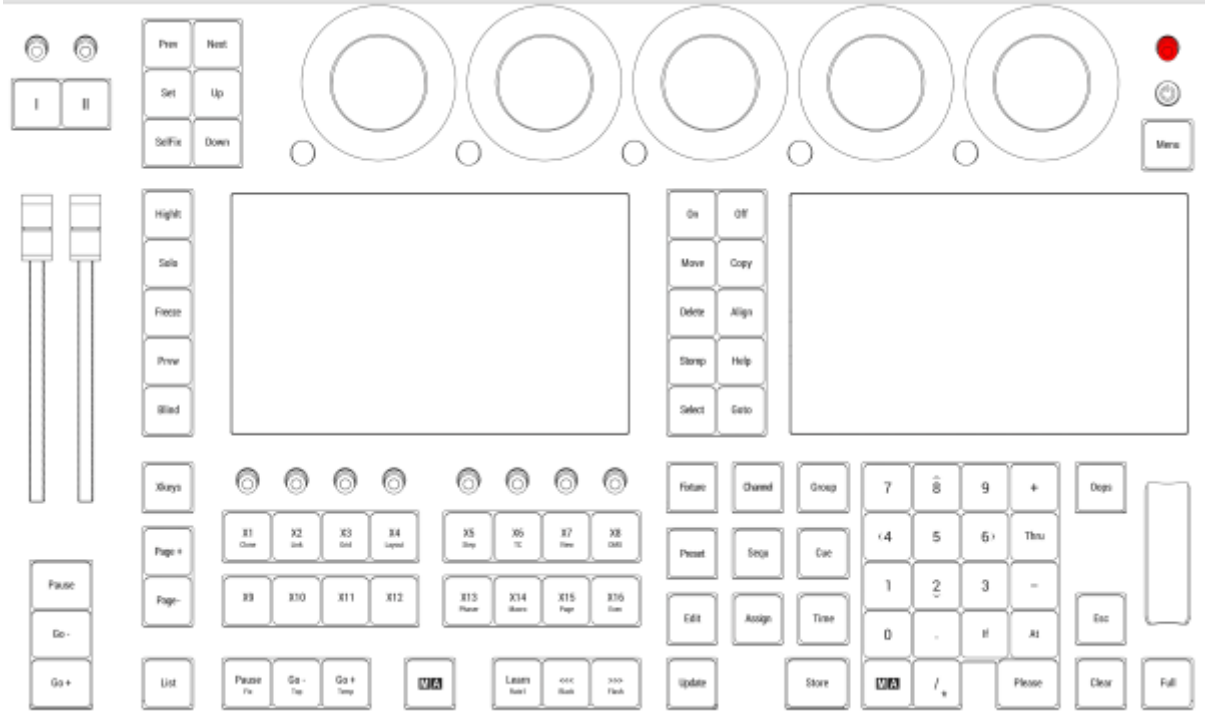
Grand Master in the encoder bar

- The Grand Master level can be set by the fader.
- To enable or disable the Grand Master function, see **Master Modes**.

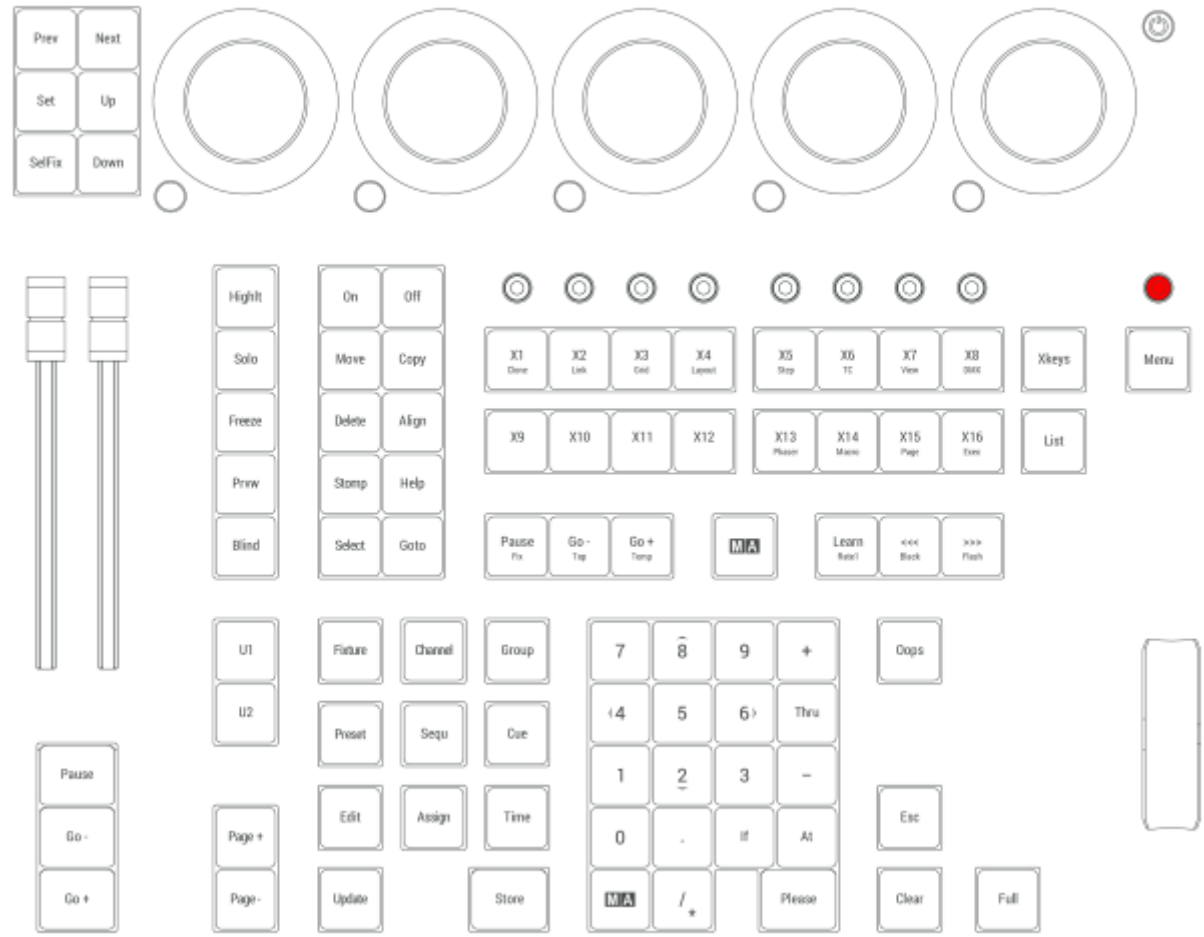
The Grand Master can also be assigned to any executor. See the **Assign object to an Executor** topic.

	Hint:
	The Output layer and the DMX layer in both the **Fixture Sheet** and the **Layout view** as well as the **DMX Sheet** and the **3D window** all display values as adjusted by the Grand Master

The Grand Master is located on the right side of the console.



Location on grandMA3 full-size and grandMA3 light consoles



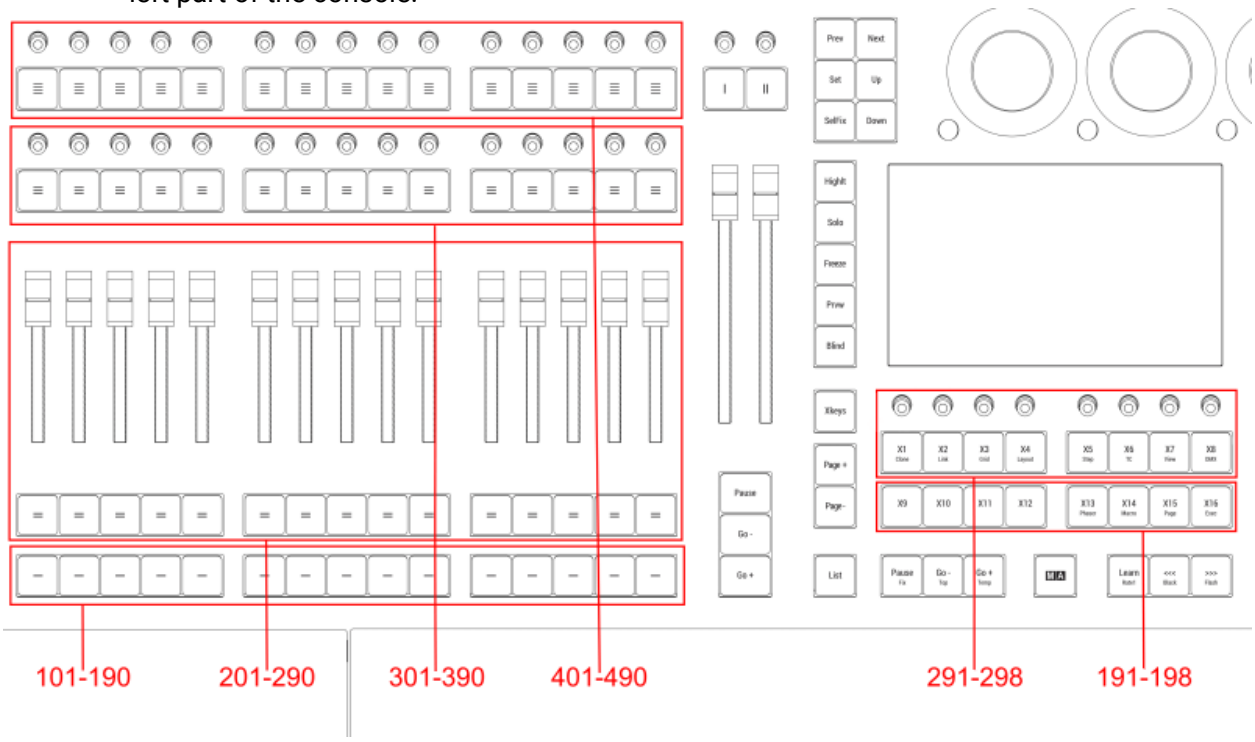
Location on grandMA3 compact consoles and grandMA3 onPC command wing

1.3.16.7. Executor Elements

Executors are used to playback and control elements of a show-file in an accessible way. For example they can be used to store a sequence.

For more informations about executors, see the **Executors topic**.

- To open the Playback Control window, in the **Add Window pop-up** tap Common and Playback.
- To open the Playback Control overlay, press F5 on the keyboard or tap in the control bar.
- The number of physical executors depends on the console type.
- The executors are handles to control objects in the show file. The executors are located on the left part of the console.



Executor location on large consoles

Executor 101 thru 190

The executors 101 thru 190 are located as the lower buttons in the executor section. On the hardware keys, these have labeled one horizontal line on top of them. In the Playback Control overlay or the Playback window, they are named by their number.

Executor 191 thru 198

The executors 191 thru 198 are located as the lower row of Xkeys (X9 to X16 | Exec). For more information see **Keys**.

Executor 201 thru 290

The executors 201 thru 290 are located as the second lower buttons and the faders in the executor section. One button and the above fader is one executor. On the hardware keys, these have labeled two horizontal lines on top of them. In the Playback Control overlay or the Playback window, they are named by their number.

Executor 291 thru 298

The executors 291 thru 298 are located as the upper row of Xkeys (X1 | Clone to X8 | DMX). For more information see **Keys**.

Executor 301 thru 390

The executors 301 thru 390 are located as the second upper buttons and knobs in the executor section. One button and one knob is one executor. On the hardware keys, these have labeled three horizontal line on top of them. In the Playback Control overlay or the Playback window, they are named by their number.

Executor 401 thru 490

The executor 401 thru 490 are located as the upper buttons and knobs in the executor section. One button and one knob is one executor. On the hardware keys, these have labeled four horizontal line on top of them. In the Playback Control overlay or the Playback window, they are named by their number.

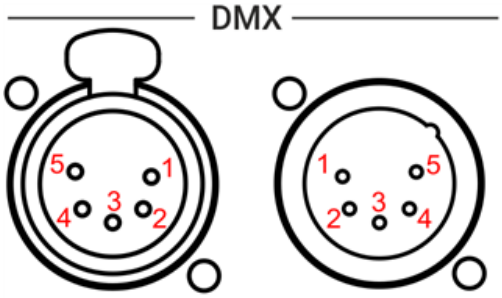
1.3.17. Connector Pin Assignment

This page gives an overview of all connectors and their respective pinouts.

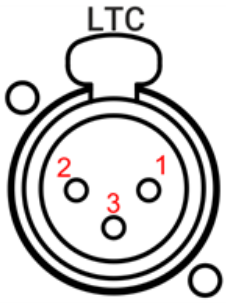
- To receive more information, tap images below.

XLR Connectors

1 Shield
2 nc
3 12V, max. 300mA
4 GND (PWM)



1 Shield
2 DMX -
3 DMX +
4 nc
5 nc



1 Shield
2 Signal +
3 Signal -

1 Shield
2 Signal +
3 Signal -

D-Sub Connectors

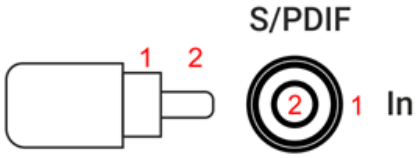
1 Remote 1
2 Remote 2
3 Remote 3
4 Remote 4
5 +10V DC, max. 100mA
6 Remote 5
7 Remote 6
8 Remote 7
9 GND

Thread UNC 4-40

GPI

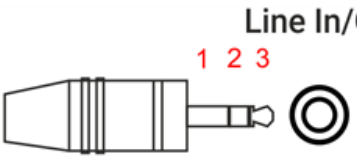
for grandMA3 consoles, grandMA3 onPC command wing, command wing XT

RCA Connectors



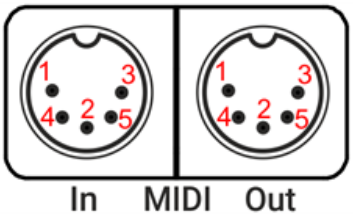
1 Shield
2 Signal

1/8" (3.5mm) Jack Sockets



1 Shield
2 Signal R
3 Signal L

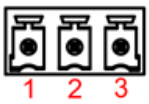
DIN Connectors



1 nc
2 Shield
3 nc
4 Signal+ (Voltage Reference Line)
5 Signal- (Data Line)

Phoenix Terminals

Power



1 PE
2 N
3 L

Power connector: MC 1.5/ 3-ST1-5.08

LTC

1 Shield
2 Signal -
3 Signal +

LTC connector: FK-MC 0.5/ 3-ST-2.5

DMX

1 Shield
2 Signal -
3 Signal +

DMX connector: FK-MC 0.5/ 3-ST-2.5

MIDI

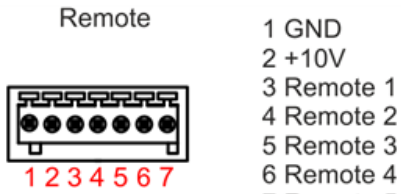
1 Shield
2 Signal- (Data Line)
3 Signal+ (Voltage Reference Line)

MIDI connector: FK-MC 0.5/ 3-ST-2.5

Remote

1 Remote 6
2 Remote 7
3 GND

Remote connector: FK-MC 0.5/ 3-ST-2.5



Remote connector: FK-MC 0.5/ 7-ST-2.5

To show the rear panel connectors on the letterbox screens on grandMA3 light and full-size:

1. Press Menu and then tap Output Configuration.
2. To show the rear panel connector, enable Show Connectors button.

or use the command line:

1.3.18. UPS Battery

The grandMA3 full-size, grandMA3 light, and their CRV versions have a rechargeable lithium-ion battery for an uninterrupted power supply (UPS).

The purpose of the battery is to enable to safe the show file and shut down the console in case of a power loss.

	**Hint:**
	To receive general information on the power status and the battery status, see the **Command Line** topic.
	**Restriction:**
	Do not use the battery to bypass the console for a longer period! Do not move the console around while in UPS mode!

In case of a power loss:

- A Power Lost pop-up appears on every screen and command screen, depending on the console type.
- The power status icon in the command line turns red.

Power Lost Pop-up

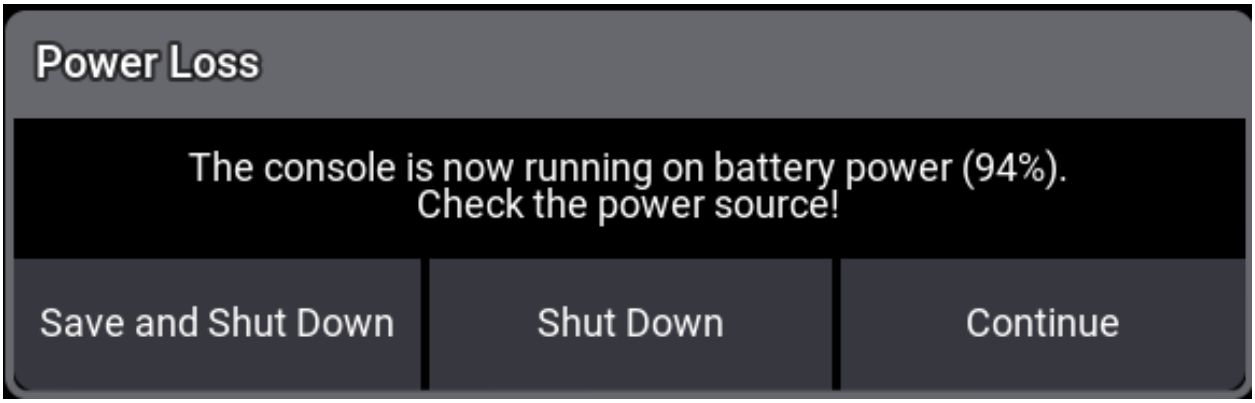
While this pop-up is open, the percentage in the parentheses (X%) continuously decreases. The percentage in the parentheses is linked to the power status of the battery.

In the pop-up window, three commands are available to tap:

Save and Shut Down: Saves the current show file and shuts down the console properly.

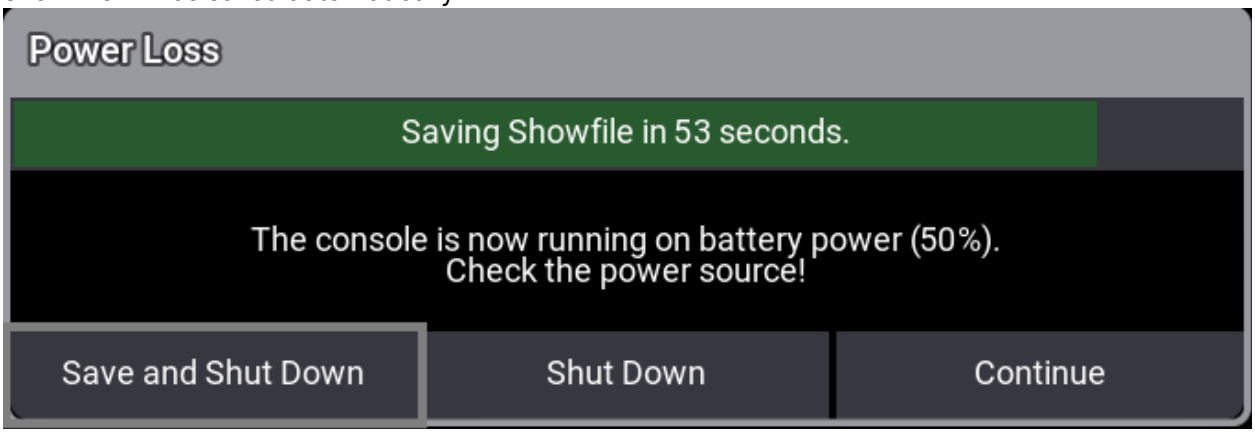
Shut Down: Properly shuts down the console without saving the show file.

Continue: The console continues running on battery power. Further Power Lost pop-ups appear during battery discharge process. When continued, the console will automatically shut down reaching (0%). The console reboots automatically when power supply is back.



Power Loss - Pop-up.

A power loss of 50% indicates another pop-up with a progress bar. When the progress bar runs out, the show file will be saved automatically.



Power Loss - Progress bar

	Hint:
	To reboot the console after a proper shutdown, press Power when the power supply is back.

Battery Status Information

To avoid deep discharge, recharge the battery to 80% capacity once a year.

- To check the battery status, switch on the console, boot it up, and mind the status bar of the UPS battery.

Battery Transportation

- Follow the transportation regulations for each country. If it is necessary to discharge the battery at a level of 30% for air cargo, run the console in UPS mode until the battery voltage reaches 17V. For more transportation information, see **Transport** in the grandMA3 Quick Manual.

Software Check

The battery function of the grandMA3 console should be checked before every major show programming or at least every two months.

To check the battery:

1. Power up the grandMA3 console and wait until the battery is fully charged.
2. Load a new show file.
3. To disconnect the console from the power, flick the power switch on the back of the console to off (0). The console now runs on battery.
4. The Power Loss pop-up appears. To proceed with the battery check, tap Continue.
5. Let the console run on battery for four minutes. After four minutes the battery status should not be lower than approximately 50%.
6. To connect the console back to power, flick the power switch to on (I). Let the battery charge to 100%.

	**Hint:**
	It may take several minutes for the battery to indicate a charge after the power switch is turned on.
	**Important:**
	The discharge speed depends on several conditions. For example, the type of console, show size, or 3D rendering.
	**Important:**
	Battery degredation can vary depending on time and charge-discharge cycles.

1.4. System Overview

This section examines the different possibilities with a standalone console and the options when expanding the system.

Subtopics

- **Standalone Device**
- **Locally Networked Devices**
- **World Server**
- **Parameters**

1.4.1. Standalone Device

If a console is not connected to anything, then it is a **Standalone** system. It is also a standalone system using a grandMA3 onPC with a connected grandMA3 onPC command wing.

Any grandMA3 device that can create and run a **session** is called a **Station**.

Stations with network disabled are in **Standalone** mode.

A station is in **IdleMaster** mode when the network is On, and it is ready to be in a session with other stations, but currently, it is alone. So, a console that is connected to a network but not in a session with other stations is considered as a master that is ready to connect to other stations.

The current status can be seen in the Network menu. The title bar has an area that displays the status.

If the station is in a session and it needs to be set to Standalone mode, then turn off the network. This can be done using the **GUI** or the command line (**LeaveSession keyword**).

In the **Standalone** or **IdleMaster** mode, the station is limited to only controlling the number of **parameters** the console/onPC unlocks.

You can only use the DMX ports on the console/wings.

If DMX is output via an Ethernet connection, then the console needs to be in a session. Even though it is not connected to other stations, it still needs to run an active session as IdleMaster.

1.4.2. Locally Networked Devices

When more stations are connected in a **session**, then it is a networked system. Local networked systems are sessions running in the same network. This might be a big network spanning a big area, but if it is a closed system without contact with the rest of the world (for instance, through the internet), then it is considered a local networked system.

The smallest networked system is two stations running in a session together where one is the master of the session and the other is connected running as a backup. A single station that outputs DMX locally is considered a **standalone system**.

Many systems have one console with one backup console or onPC and some networked devices to output the DMX.

If the devices that translate Ethernet data to DMX are MA nodes, then the DMX output is in sync, no matter where the DMX is output. So, a blackout cue on LED fixtures is in sync when some LEDs get a signal from the console and some LEDs from a node somewhere in the system.

Besides the stations, there are a lot of different devices that can be added to the system. Read more about the different devices in the **Device Overview topics**.

In all networked systems, there is one station, the session master, called **GlobalMaster**. This station is in charge of communication and executes commands triggered by remote inputs or from a command in a sequence.

Other stations in the system are in a **Connected** status. They are connected to the master station.

Traditionally, we recommend creating local networks with good-quality switches, cables, and equipment. Most systems are local systems without any connection to the outside world.

1.4.3. World Server

Each station can be connected to the internet and a world server.

The station automatically tries connecting to a world server if an internet connection is detected.

The connection status of the world server can be seen in the **Command Line**. A globe icon indicates the connection status.

If the globe is green, then the station is connected to the server.

Learn more about networking in the **Networking topic**.

The address of the official world server, provided by MA Lighting, is **worldserver.malighting.de**

The server address can be changed in the Network Menu. Learn more about this in the **Session topic**.

The server offers two functions: Fixture type files and Crash Log upload.

Fixture Type Files

Fixture type files from the GDTF-share and grandMA3 Share are provided as a direct import to the show file. Learn more about manually importing GDTF in the **Import GDTF topic** and patching fixtures in the **Add Fixtures to the Show topic**.

Crash Logs

A station creates a crash log if it crashes. These log files are automatically sent to the server when there is a connection. This means that the file is sent the next time the station is connected. This is also true if the station returns from a job and is connected to the world server when it returns to the workshop.

If Tech Support is needed, please make sure the station has been online and provide the Tech Support date and time the crash happened and the station's serial number (grandMA3 onPC stations also have a generated serial number). The serial number can be found using the **Version Keyword**.

	Important:
	If these functions are not desired, then change the server address in the **Network menu** to "0".

1.4.4. Parameters

Most people are used to thinking in DMX channels when considering the number of fixtures a system can control.

MA Lighting cares more about **Parameters**.

What are Parameters

Parameters are also called **Attributes** in the software. A dimmer function is a parameter, and a pan function is also a parameter.

The software calculates the different attributes with a higher precision than what is output via DMX. So, the software calculates the parameter or attribute once, and it is then scaled to the number of DMX channels a fixture uses - typically one or two per attribute.

Why Counting in Parameters and Not DMX

It is to your advantage. In the MA world, you do not pay extra for fixtures that are running 16-bit or 24-bit instead of the 8-bit used by one DMX channel.

More networked nodes or devices might be needed to output all the parameters that can be controlled.

Example:

A simple moving head with a dimmer might use 5 DMX channels. The channels can be defined like this:

Definition:	DMX Channel:
Dimmer	1
Pan	2
Pan Fine	3
Tilt	4
Tilt Fine	5

The pan and tilt are one attribute each. Even though both pan and tilt each use two DMX channels, they are only counted as one each in the parameter count. This means that the fixture only costs 3 parameters.

Definition:	DMX Channel:	Parameter cost:
Dimmer	1	1
Pan	2	2
Pan Fine	3	free
Tilt	4	3
Tilt Fine	5	free

This can be a big advantage when many fixtures have 16-bit (or fine) channels.

What about Preprogramming and Parameters?

The show can be preprogrammed and visualized in the **3D window** without any parameter unlocking hardware. The lights are still visualized.

If third-party visualizers are used, then grandMA3 hardware is needed to give access to the parameters. The grandMA3 viz-key can be added to unlock visualization on a third-party visualizer. Learn more about the grandMA3 viz-key in the **Connect grandMA3 viz-key topic**.

Subtopics

- **Calculate Parameters**
- **Expand the Amount of Parameters**

1.4.4.1. Calculate Parameters

Calculating the needed amount of parameters can be a big task, depending on the size of the setup.

It is generally true that most modern fixtures use fewer parameters than DMX channels.

The best way to see how many parameters a show needs is to patch all the fixtures and then open the **Details** in the **System Info Window**.

Here, the amount of needed (used) parameters can be seen, as well as the amount of currently available parameters.

Read the **Expand the amount of parameters** topic to learn how to get more.

Parameter Count

Some devices provide parameters:

- **grandMA3 full-size and grandMA3 full-size CRV:**
  20 480 parameters.
- **grandMA3 light and grandMA3 light CRV:**
  16 384 parameters.
- **grandMA3 extension:**
  none parameters.
- **grandMA3 replay unit:**
  8 192 parameters.
- **grandMA3 compact XT:**
  8 192 parameters.
- **grandMA3 compact:**
  8 192 parameters.
- **grandMA3 processing unit XL:**
  16 384 parameters.
- **grandMA3 processing unit L:**
  8 192 parameters.
- **grandMA3 processing unit M:**
  4 096 parameters.
- **grandMA3 8Port Node**:
  none parameters.
- **grandMA3 4Port Node:**
  none parameters.
- **grandMA3 2Port Node:**
  none parameters.
- **grandMA3 8Port Node DIN-Rail**:
  none parameters.
- **grandMA3 4Port Node DIN-Rail**:
  none parameters.
- **grandMA3 2Port Node DIN-Rail**:
  none parameters.

- **grandMA3 onPC rack-unit**:
  4 096 parameters.
- **grandMA3 onPC command wing XT:**
  4 096 parameters.
- **grandMA3 onPC command wing:**
  4 096 parameters.
- **grandMA3 onPC fader wing**:
  4 096 parameters.
- **grandMA3 onPC 8Port Node 4k:**
  4 096 parameters (when connected to grandMA2 onPC this node provides only 2 048 parameters).
- **grandMA3 onPC 4Port Node 4k:**
  4 096 parameters (when connected to grandMA2 onPC this node provides only 2 048 parameters).
- **grandMA3 onPC 2Port Node 2k:**
  4 096 parameters (when connected to grandMA2 onPC this node provides only 1 024 parameters).
- **grandMA3 onPC 8Port Node DIN-Rail 4k**:
  4 096 parameters (when connected to grandMA2 onPC this node provides only 2 048 parameters).
- **grandMA3 onPC 4Port Node DIN-Rail 4k**:
  4 096 parameters (when connected to grandMA2 onPC this node provides only 2 048 parameters).
- **grandMA3 onPC 2Port Node DIN-Rail 2k**:
  4 096 parameters (when connected to grandMA2 onPC this node provides only 1 024 parameters).
- **grandMA3 onPC Software**:
  none parameters.

1.4.4.2. Expand the Amount of Parameters

The grandMA3 processing units are the **only units** that expand the parameter count when using grandMA3 consoles.

Every grandMA3 processing unit added to the network also adds a number of parameters depending on the model!

There is a **maximum limit of 262 144 parameters** in a grandMA3 session.

The grandMA3 processing units help with parameter calculations.

Examples:

1 grandMA3 full-size (20 480) + 1 grandMA3 processing unit XL (16 384) = 36 864 parameters

1 grandMA3 full-size (20 480) + 15 grandMA3 processing unit XL (16 384) = 262 144 parameters (the calculation is 266 240, but the limit is 262 144)

1 grandMA3 light (16 384) + 1 grandMA3 processing unit M (4 096) = 20 480 parameters

1 grandMA3 light (16 384) + 15 grandMA3 processing unit XL (16 384) = 262 144 parameters

1 grandMA3 full-size (20 480) + 1 grandMA3 light (16 384) = 20 480 parameters (consoles cannot expand the parameter count, so the parameters from the console with the highest number are unlocked)

1 grandMA3 light (16 384) + 15 grandMA3 onPC 2Port Node 2k (4 096) = 16 384 parameters (nodes cannot expand the parameter count with consoles, so the parameters from the console are used)

1 grandMA3 light (16 384) + 1 onPC computer with a grandMA3 onPC command wing (4 096) = 16 384 parameters (onPC command wings cannot expand the parameter count with consoles, so the parameters from the console are used)

Using a grandMA3 onPC

When a grandMA3 onPC is used as the primary station (the system does not include any grandMA3 consoles), some grandMA3 **onPC** hardware is needed to unlock parameters.

The grandMA3 processing units also unlock parameters in an onPC system.

	Important:
	The maximum number of parameters allowed in a grandMA3 onPC system is 4 096.

Any amount of onPC hardware and processing units can be used with a computer or the grandMA3 onPC rack-unit. Every piece of hardware will add its parameters until the limit of 4 096 parameters is reached.

grandMA3 xPort Nodes need to be onPC versions to unlock parameters.

These are the only two rules for parameters with grandMA3 onPC.

Remember, more units can be added to get more DMX ports even after reaching the parameter limit.

Examples:

grandMA3 onPC rack-unit (4 096) + grandMA3 onPC 2Port Node 2k (4 096) = 4 096 parameters (the limit)

grandMA3 onPC rack-unit (4 096) + grandMA3 onPC 4Port Node 4k (4 096) = 4 096 parameters (the limit)

grandMA3 onPC + grandMA3 onPC command wing (4 096) = 4 096 parameters (the limit)

grandMA3 onPC rack-unit (4 096) + grandMA3 onPC command wing (4 096) = 4 096 parameters (the limit)

grandMA3 onPC command wing XT (4 096) = 4 096 parameters (the limit)

grandMA3 onPC + grandMA3 onPC command wing (4 096) + grandMA3 onPC 2Port Node 2k (4 096) = 4 096 parameters (the limit)

grandMA3 onPC + grandMA3 onPC command wing (4 096) + grandMA3 onPC fader wing (4 096) = 4 096 parameters (the limit)

grandMA3 onPC + grandMA3 onPC 2Port Node DIN-Rail 2k (4 096) = 4 096 parameters

grandMA3 onPC + grandMA3 onPC 2Port Node 2k (4 096) + grandMA3 8Port Node (none) = 4 096 parameters (the 8Port Node is not an onPC node)

grandMA3 onPC + grandMA3 onPC command wing (4 096) + grandMA3 onPC 8Port node 4k (4 096) = 4 096 parameters (the limit is reached and cannot be exceeded)

grandMA3 onPC + 1 grandMA3 processing unit M (4 096) = 4 096 parameters (the limit)

grandMA3 onPC + 1 grandMA3 processing unit XL (16 384) = 4 096 parameters (the limit is reached and cannot be exceeded)

1.5. First Steps

This chapter describes how to prepare the grandMA3 consoles and the grandMA3 onPC for operation, from unpacking the device to turning it on for the first time.

Subtopics

- **Unpack the Device**
- **Check Scope of Delivery**
- **Position the Device**
- **Connect Power**
- **Connect Desk Light**
- **Connect External Screens**
- **Connect USB Devices**
- **Connect DMX**
- **Connect Audio In**
- **Connect MIDI**
- **Connect Sound Out**
- **Connect LTC**
- **Connect Ethernet**
- **Connect DC Remote In**
- **Connect grandMA3 extension**
- **Connect grandMA3 fader wing**
- **Connect grandMA3 viz-key**
- **Turn on the device for the first time**
- **Shut Down the System**

1.5.1. Unpack the Device

- Unpack the device. Remove all packing material, strips and protection films.
- Keep the packing material for transport.

Hint:
The device was tested for proper function at the factory.

1.5.2. Check Scope of Delivery

The list below shows the scope of delivery. If anything is missing, contact your local distributor.

grandMA3 full-size

- 2 x LED desk light grandMA3
- 1 x Dust cover grandMA3 full-size
- 2 x Magnetic plate for easy labeling of 15 faders
- 1 x grandMA3 Quick Manual consoles

grandMA3 full-size CRV

- 2 x LED desk light grandMA3
- 1 x Dust cover grandMA3 full-size CRV
- 2 x Magnetic plate for easy labeling of 15 faders
- 1 x grandMA3 Quick Manual consoles

grandMA3 light

- 2 x LED desk light grandMA3
- 1 x Dust cover grandMA3 light
- 1 x Magnetic plate for easy labeling of 15 faders
- 1 x grandMA3 Quick Manual consoles

grandMA3 light CRV

- 2 x LED desk light grandMA3
- 1 x Dust cover grandMA3 light CRV
- 1 x Magnetic plate for easy labeling of 15 faders
- 1 x grandMA3 Quick Manual consoles

grandMA3 compact XT

- 1 x LED desk light grandMA3
- 1 x Dust cover grandMA3 compact XT
- 1 x Magnetic plate for easy labeling of 15 faders
- 1 x grandMA3 Quick Manual consoles

grandMA3 compact

- 1 x LED desk light grandMA3
- 1 x Dust cover grandMA3 compact
- 1 x Magnetic plate for easy labeling of 10 faders
- 1 x grandMA3 Quick Manual consoles

grandMA3 extension

- 1 x LED desk light grandMA3
- 1 x Dust cover
- 1x Magnetic plate for easy labeling of 15 faders
- 1 x grandMA3 Quick Manual consoles

grandMA3 replay unit

- 1x grandMA3 Quick Manual consoles

grandMA3 processing units

- 1x grandMA3 Quick Manual processing unit

grandMA3 xPort Nodes

- 1x grandMA3 Quick Manual Nodes

grandMA3 Nodes DIN-Rail

- 1x grandMA3 Quick Manual Nodes DIN-Rail

grandMA3 I/O Nodes

- 1x grandMA3 Quick Manual I/O Nodes

grandMA3 I/O Nodes DIN-Rail

- 1x grandMA3 Quick Manual I/O Nodes

grandMA3 onPC command wing

- 1 x Dust cover
- 1 x Magnetic plate for easy labeling of 10 faders
- 1 x USB cable
- 1 x grandMA3 Quick Manual onPC command wing

grandMA3 onPC command wing XT

- 1 x Dust cover
- 1 x Magnetic plate for easy labeling of 10 faders
- 1 x grandMA3 Quick Manual onPC command wing XT

grandMA3 onPC fader wing

- 1x Dust cover
- 1x USB cable
- 1x Magnetic plate for easy labeling of 10 faders
- 1x grandMA3 Quick Manual onPC fader wing

grandMA3 rack-unit

- 1x grandMA3 Quick Manual onPC rack-unit

grandMA3 viz-key

- 1x USB cable (C/C)
- 1x USB cable (C/A)
- 1x grandMA3 Quick Manual viz-key

1.5.3. Position the Device

Follow the instructions below:

- Do not place the device in a humid area.
- Place the device on a stable, flat, dry surface.
- Do not cover the ventilation holes.
- Do not expose the device to direct sunlight.
- Maintain a minimum distance of 15 cm (5.91 inches) between the multi-touch screen surface and radio intercom systems. Going below the minimum distance may cause unexpected behavior, such as unwanted multi-touch actions or mouse movements.

1.5.4. Connect Power

Devices with powerCON Connector or powerCON TRUE1 TOP Connector

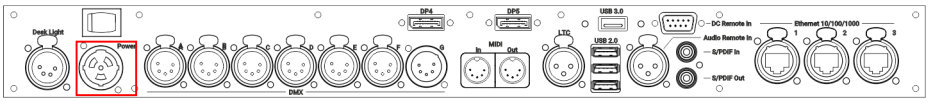
Warning:
If only the powerCON TRUE1 connector or powerCON TRUE1 TOP connector is provided, connect a suitable cable to the connector. To connect a suitable cable to the powerCON TRUE 1 connector or powerCON TRUE1 TOP connector, contact the local distributor.

1. Insert the powerCON TRUE1 connector or powerCON TRUE1 TOP connector in the powerCON TRUE1 plug or powerCON TRUE1 TOP plug and twist it to lock clockwise.
   An audible click is heard.
2. Connect the other end of the cable to a suitable power source.

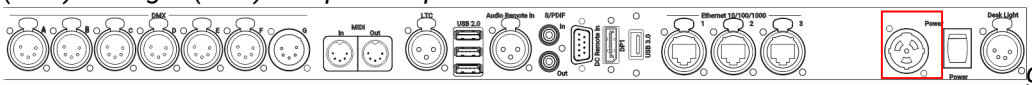
Devices with IEC Connector

1. Insert the IEC connector into the corresponding jack.
2. Connect the power plug.

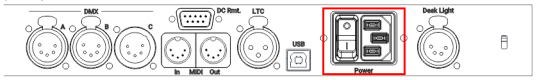
The device is connected to power.



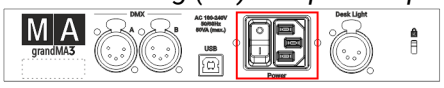
grandMA3 full-size (CRV) and light (CRV) rear panel – power



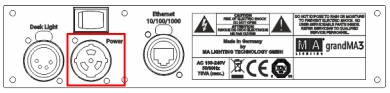
grandMA3 compact (XT) rear panel – power



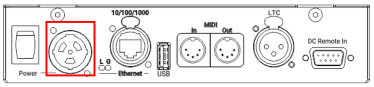
grandMA3 onPC command wing (XT) rear panel – power



grandMA3 fader wing rear panel – power



grandMA3 extension *rear panel – power*



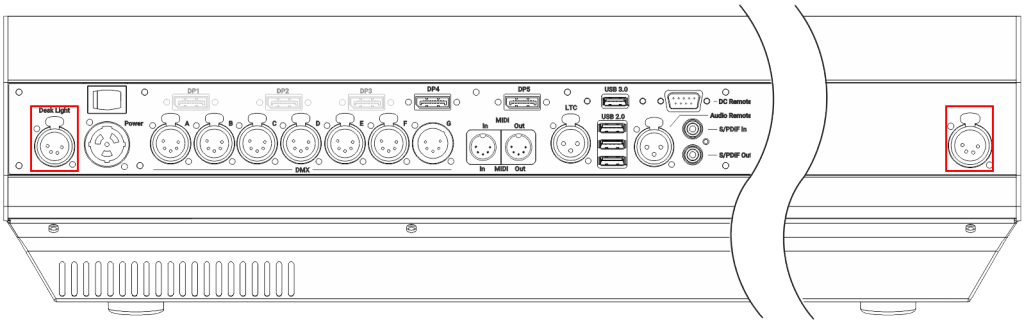
grandMA3 I/O Node *rear panel* – power

1.5.5. Connect Desk Light

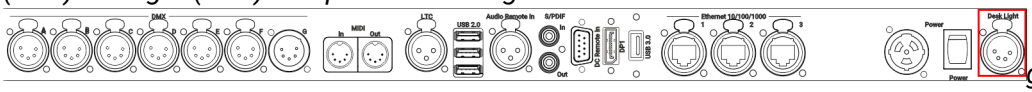
The grandMA3 comes with one or two desk lights, depending on the model. For more information see, **delivery contents**.

- Connect the 4 pin XLR connector to the desk light connectors on the rear panel.

The desk lights are connected.

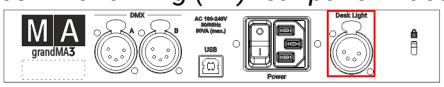


grandMA3 full-size (CRV) and light (CRV) rear panel – desk light

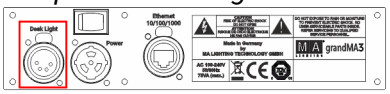


grandMA3 compact (XT) rear panel – desk light

grandMA3 onPC command wing (XT) rear panel – desk light



grandMA3 fader wing rear panel – desk light



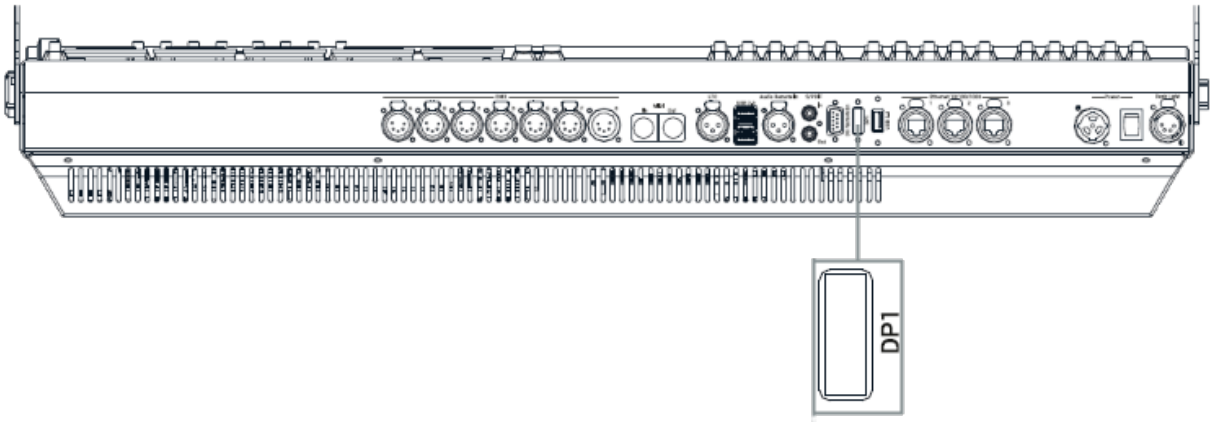
grandMA3 extension rear panel – desk light

To view the pinout of the XLR connector, refer to the topic **Connector Pin Assignment**.

1.5.6. Connect External Screens

You can connect up to five external touch screens with a grandMA3 full-size CRV (= Control Room Version). For more information about the amount of DisplayPort connectors of each model, see **Quick Manual Consoles - Technical Data**.

	Important:
	The grandMA3 series supports only native DisplayPort connectors on external screens.
	Important:
	In grandMA3 CRV versions, use at least one Full HD display and a touch display or a mouse.
	Hint:
	The grandMA3 compact and the compact XT models only have one DisplayPort called DP1.



Example of a DisplayPort connector on a grandMA3 compact XT

Screen Arrangement

Place the screens in the following order:

grandMA3 full-size screen order

- Screens 5 and 4 are always external screens.

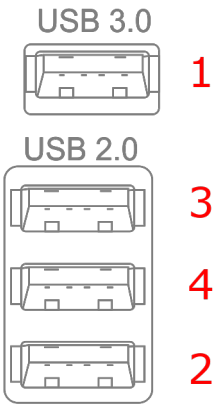
- Screens 3, 2, and 1 can be internal or external screens, depending on the product.
- The screen order is important for the mouse behavior from screen to screen.

Requirements for external monitors:

- Native DisplayPort connection
- Resolution of 1920 x 1080 (full HD)
- Microsoft Windows® 10 multi-touch compliance
- Separated USB connection for touch functionality

Connect external touch screens:

1. Make sure the console is turned off.
2. Place the first external screen right beside or above screen 1 and the second external screen left beside or above screen 3.
   The screen numbers of the external screens will be numbered consecutively.
3. Connect the DisplayPort cable of screen 4 with the DisplayPort connector DP4.
4. Connect the USB cable with the corresponding USB port.
5. Connect the DisplayPort cable of screen 5 with the DisplayPort connector DP5.
6. Connect the touch screen USB cable to the appropriate USB port in the correct order as shown below:

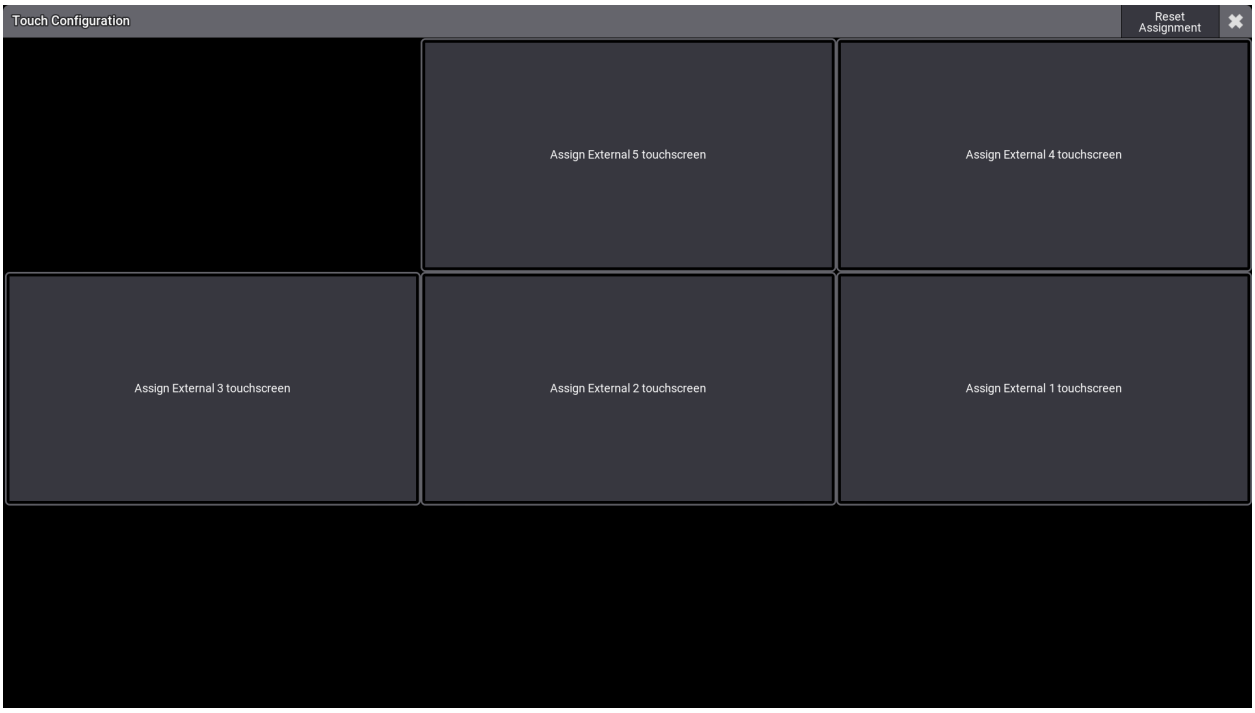


Screen allocation for a grandMA3 light CRV.

7. Press Power to boot up the console.

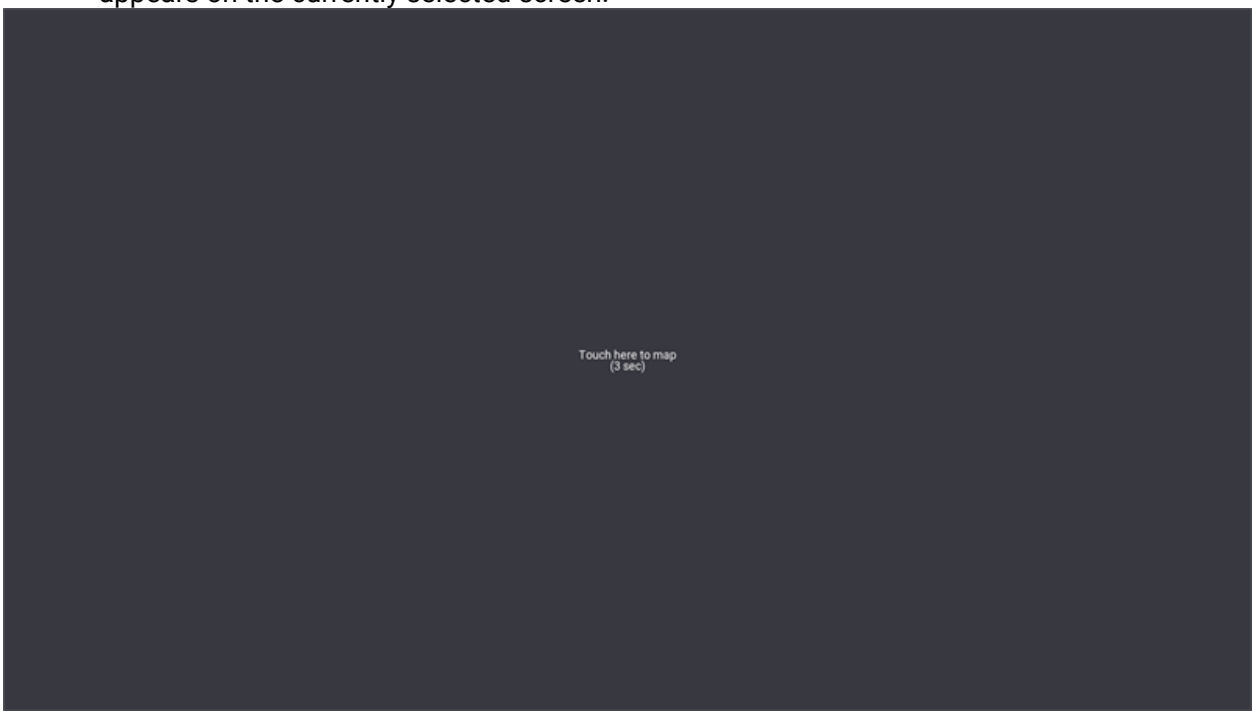
Touch Configuration

1. To adjust the touch configuration, tap , Settings and then tap Touch Configuration. The Touch Configuration overlay opens:



grandMA3 full-size CRV touch configuration

2. To assign a touch input to a monitor, tap a monitor button in the touch configuration overlay. For example, Assign External 1 touchscreen.
3. Touch the appropriate monitor screen to assign the touch input to that device. An overlay appears on the currently selected screen:



Touch mapping overlay

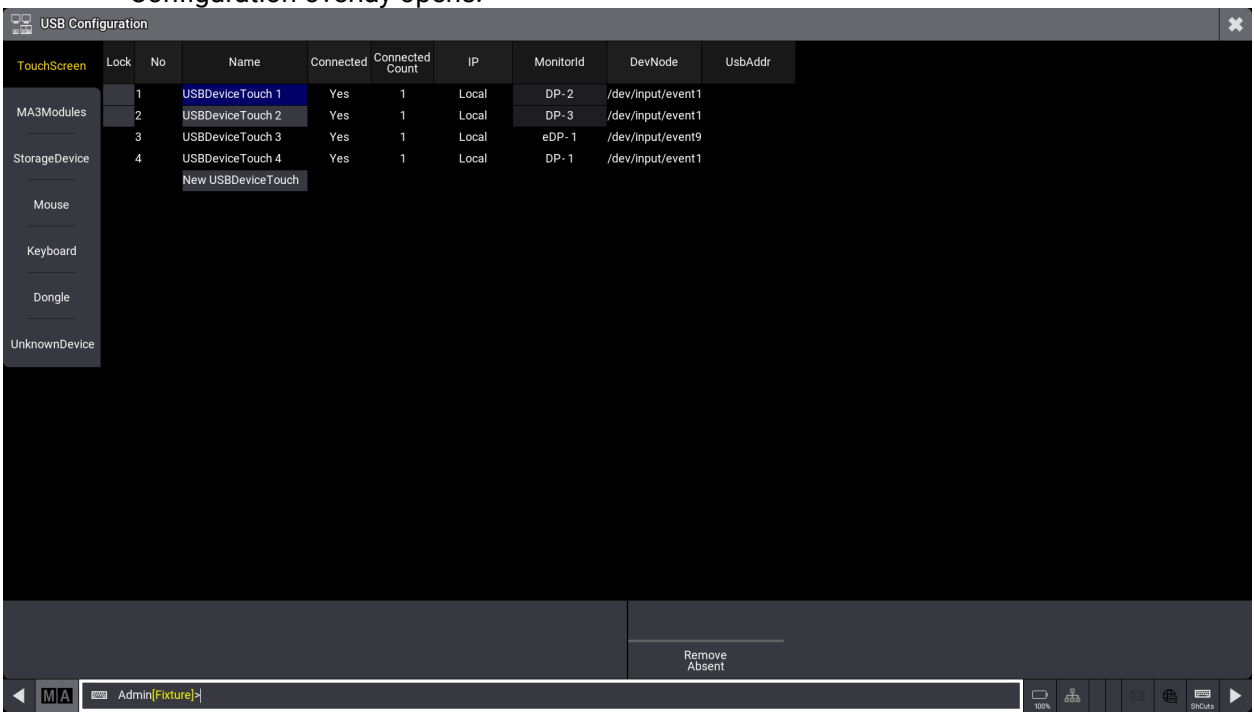
4. The touch function is assigned to the monitor.
   - Reset Assignment: Resets the screen allocation.

	Hint:
	grandMA3 onPC settings do not have a Touch Configuration button, as they

	are handled by the operating system.

USB Configuration

- To check the USB settings, tap , Settings and then tap USB configuration. The USB Configuration overlay opens:



USB configuration window with display information

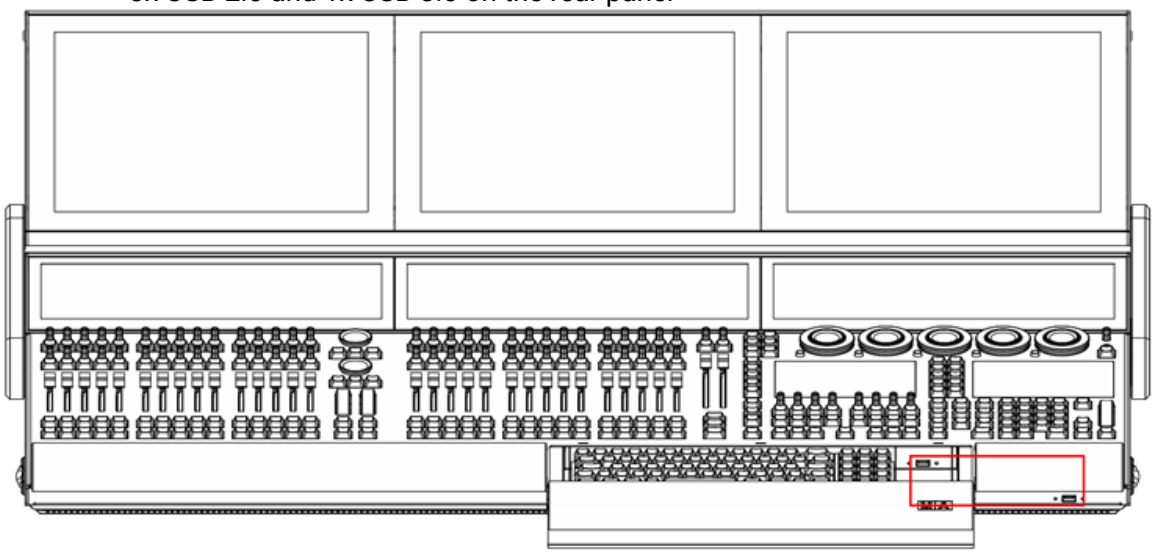
- Remove Absent: Removes disconnected monitors from the list above.

1.5.7. Connect USB Devices

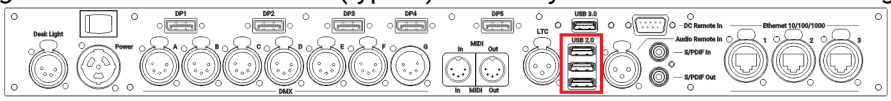
You can connect an external USB mouse, an external USB keyboard, or an external touch screen to the grandMA3 device using the USB ports.

Depending on the type of the grandMA3 device there are up to 6 USB connectors:

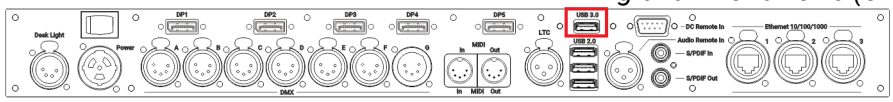
- 1x USB 3.0 at the front on the right of the console
- 1x USB 3.0 inside the keyboard drawer
- 3x USB 2.0 and 1x USB 3.0 on the rear panel



grandMA3 full-size USB 3.0 (type A) *inside keyboard drawer and on the right*



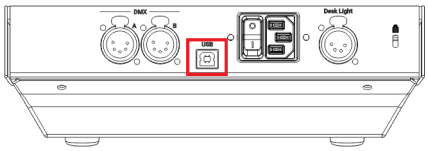
grandMA3 full-size (CRV) rear panel USB 2.0 (type A)



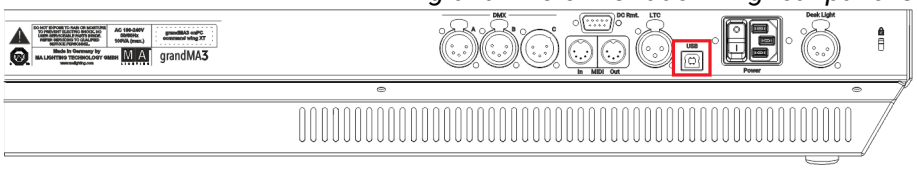
grandMA3 full-size rear panel USB 3.0 (type A)

grandMA3 compact (XT) rear panel USB 3.0 (type A)

To connect a grandMA3 onPC device with the computer or laptop, use the USB connector (type B):



grandMA3 onPC fader wing rear panel USB 2.0 (type B)



grandMA3 onPC command wing rear panel USB 2.0 (type B)

1.5.8. Connect DMX

You can connect DMX devices to the grandMA3 devices.

Depending on the type of the grandMA3 device there are up to six DMX Out ports (e.g. A-F) and one DMX In port (e.g. G).

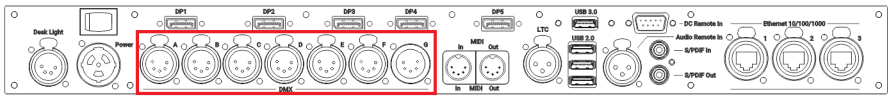
No matter which physical direction the connector has, it can be configured in the software and is therefore capable for both directions: in and out.

To adjust the DMX Remote settings, read the **In and Out topic**.

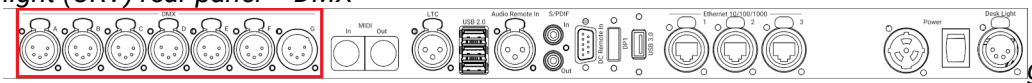
To configure the DMX ports (Off, Out, RDM, In), read the **DMX Port Configuration topic**.

- Connect the 5pin XLR DMX cable to a DMX connector on the console and a DMX device.

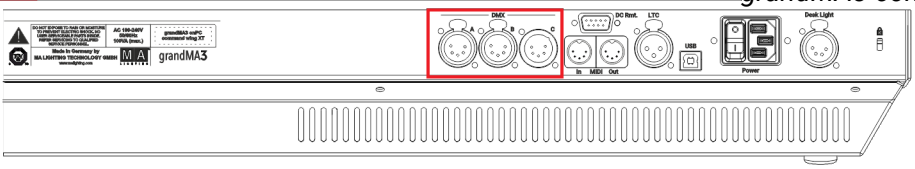
The DMX device is connected to the **XLR connector**.



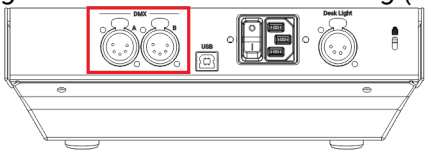
grandMA3 full-size (CRV) and light (CRV) rear panel – DMX



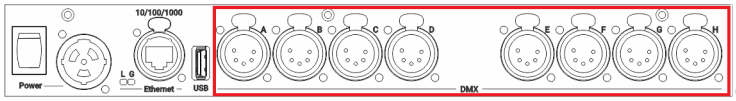
grandMA3 compact (XT) rear panel – DMX



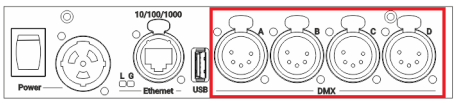
grandMA3 onPC command wing (XT) rear panel – DMX



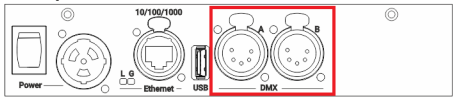
grandMA3 onPC fader wing *rear panel – DMX*



grandMA3 8Port Node *rear panel – DMX*



grandMA3 4Port Node *rear panel – DMX*



grandMA3 2Port Node *rear panel – DMX*

To view the pinout of the XLR connector, refer to the topic **Connector Pin Assignment**.

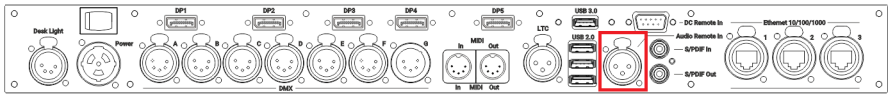
1.5.9. Connect Audio In

To use a sound trigger or BPM, connect a sound source to the console.

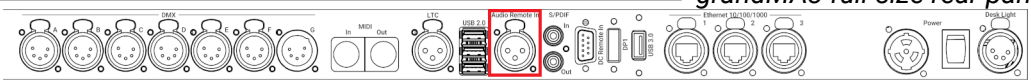
	Hint:
	The signal strength should be a minimum of 50 mV.

- Connect the 3pin XLR sound source cable to the **Audio Remote In connector** on the rear panel.

The sound source is connected to the **Audio Remote In connector**.



grandMA3 full-size rear panel – Audio Remote In



grandMA3 compact (XT) rear panel – Audio Remote In

To view the pinout of the DIN connector, refer to the topic **Connector Pin Assignment**.

1.5.10. Connect MIDI

MIDI connectors can be used for MIDI input or output. For example to send the MIDI timecode signal to the sound engineer.

- To adjust the MIDI Remotes settings, read the **Remote In and Out topic**.
- To configure the MIDI Port settings per grandMA3 device, read **Output Configuration menu**.

1. To access the Output configuration window, press Menu and tap Connector Configuration.

-OR-

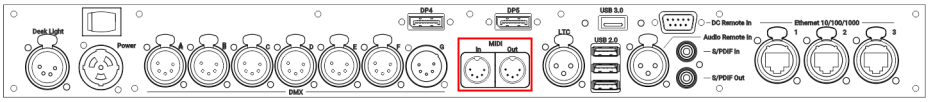
1. Use the command line to open the menu:

```
User name[Fixture]>Menu "ConnectorConfig"
```

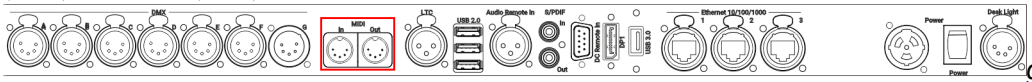
2. To select the MIDI Mode, tap and hold or right-click MIDI Mode. A pop-up window opens.
3. To set the MIDI direction, select In, Out, or Through.
4. Connect the MIDI source to the **DIN connector** on the rear panel of the console.

	Hint:
	MIDI Mode 'Through' receives and forwards MIDI Notes, MIDI Control Changes, and MIDI Program Changes.

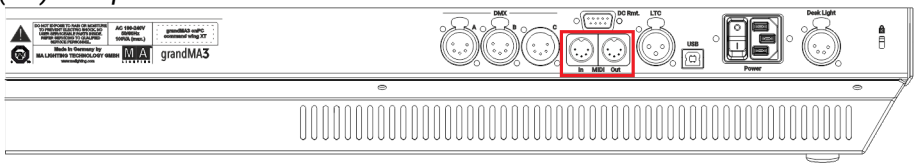
Location of MIDI Connectors on Different Products



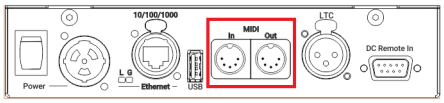
grandMA3 full-size (CRV) and light (CRV) rear panel – MIDI



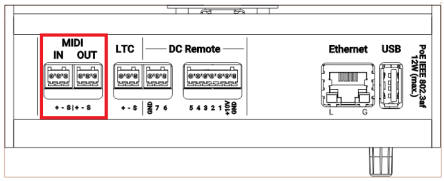
grandMA3 compact (XT) rear panel – MIDI



grandMA3 onPC command wing (XT) rear panel – MIDI



grandMA3 I/O Node *rear panel – MIDI*

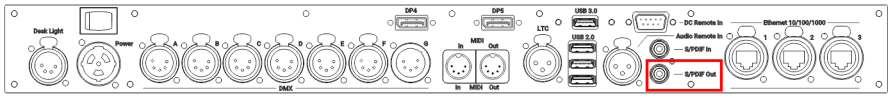


grandMA3 I/O Node DIN-Rail *top panel – MIDI*

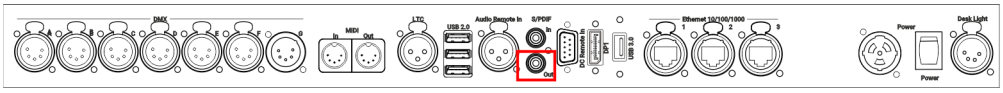
	Hint:
	For more information on DIN connector pinout, see **Connector Pin Assignment**.

1.5.11. Connect Sound Out

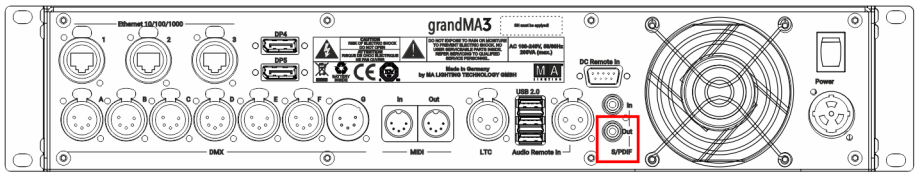
The following grandMA3 products have a S/PDIF (Sony/Philips Digital Interface) connectors on the rear panel to output sound.



grandMA3 full-size (CRV) and light (CRV) rear panel - S/PDIF Out



grandMA3 compact XT rear panel - S/PDIF Out



grandMA3 replay unit rear panel - S/PDIF Out

Sound Cards

To output sounds, third-party USB sound cards can be used as well. To connect sound via USB, see **Local Settings** and **Sound**.

The following sound cards have been successfully tested:

- Focusrite Saffire USB
- M-Audio Air Hub
- Palmer PLI 04 USB
- Radial USB-Pro

	Hint:
	Other sound cards that have not been tested may work as well.

1.5.12. Connect LTC

The LTC connector can be used for timecode input or for timecode output, e.g. to send the timecode signal to the sound engineer.

You can configure the direction of the LTC port in the **Output Configuration**.

To synchronize the console with an SMPTE timecode source, connect an SMPTE source to the LTC port.

	Information: The supported time formats are: - 24 fps - 25 fps - 30 fps If you send 29.97 or 30 drop frame, it will be interpreted as 30 fps.

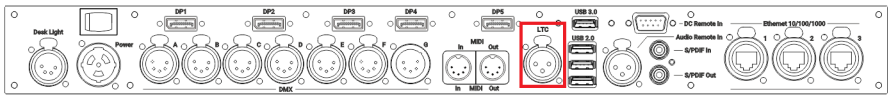
Sound and timecode signal levels

Min. level	Max. level	Recom. level
-11 dBu	+15 dBu	0 dBu
0.2 Veff	4.4 Veff	0.8 Veff

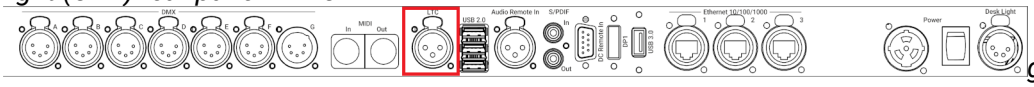
	Information: The signal strength should be a minimum of 200 mV. Pin 1: Ground Pin 2: - (minus) Pin 3: + (plus)

- Connect the SMPTE source to the LTC connector on the rear panel of the console.

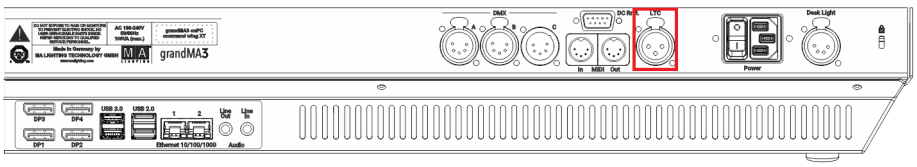
The SMPTE source is connected to the LTC connector.

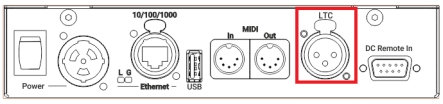


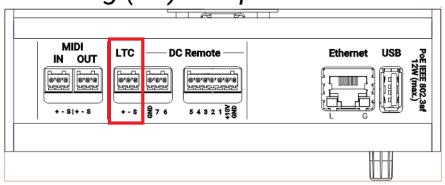
grandMA3 full-size (CRV) and light (CRV) rear panel – LTC



grandMA3 compact (XT) rear panel – LTC



grandMA3 onPC command wing (XT) rear panel – LTC 

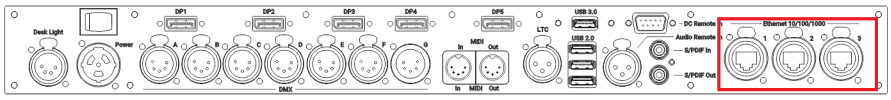
grandMA3 I/O Node *rear panel – LTC* 

grandMA3 I/O Node DIN-Rail *rear panel – LTC*

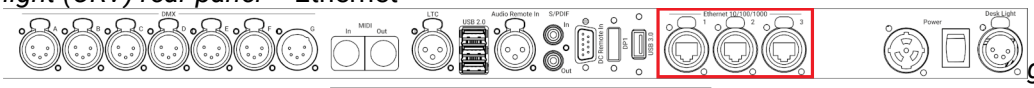
To view the pinout of the DIN connector, refer to the topic **Connector Pin Assignment**.

1.5.13. Connect Ethernet

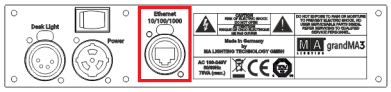
Depending on the type of grandMA3 device, there are up to three Ethernet ports on the rear panel.



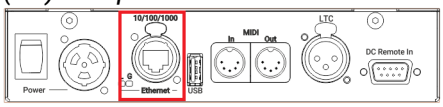
grandMA3 full-size (CRV) and light (CRV) rear panel – Ethernet



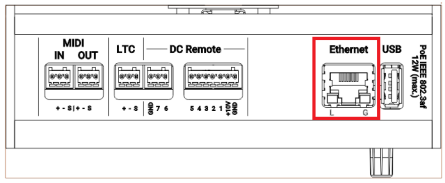
grandMA3 compact (XT) rear panel – Ethernet



grandMA3 extension *rear panel* – Ethernet



grandMA3 I/O Node *rear panel* – Ethernet



grandMA3 I/O Node DIN-Rail *rear panel* – Ethernet

	Hint:
	The naming of the Ethernet ports is individual, for example, Con1, Con2, and Con3 in the console. In the onPC software, the Ethernet ports are named by the operating system, for example Ethernet, Ethernet 2, etc.
	Hint:
	All Ethernet ports can use Art-Net and sACN as DMX protocols.
	Important:
	To avoid damage, the Ethernet ports must not be connected with Power over Ethernet (PoE) except for the grandMA3 devices qualified for PoE (for example, xPort Nodes PoE, onPC xPort Nodes PoE, or I/O Node PoE).

Ethernet Ports 1-3

Requirement: Use STP (shielded twisted pair) cable with an RJ45 connector, at least CAT-5e.

To connect one of the three Ethernet ports:

1. Connect one end of an Ethernet cable to the **Ethernet 1, 2, or 3**.
2. Connect the other end to a suitable switch.
3. Connect other grandMA3 equipment to the switch and the switch to the power supply.

Ethernet is connected to a port.

Ethernet LEDs

The L (link) LED flashes when the Ethernet connection is active.

The G (gigabit) LED flashes when a gigabit connection is available.

1.5.14. Connect DC Remote In

To use the DC Remote In with grandMA3 consoles, onPC command wings, and I/O Nodes connect a contact closure switch, for example, a light barrier or a push button.

For further information see the topics **Remote keyword**, **Remote In and Out**, and **Output configuration**.

	Hint:
	You can use up to 64 input channels within a session.
	Hint:
	It is possible to analogously move the Master Fader using a grandMA3 console or I/O Node connected with a potentiometer (0 to +10V DC). The onPC products can switch on and off, but do not fade.

- Generate a switch or connect an external source that sends up to +10V DC to pin 1 for the console to react to analog input number 1.
- The recommended resistance is 5 kohms to 10 kohms.
- To use the DC Remote, feed a voltage signal (max. +10V DC into the corresponding input pin. For more information see the pinout image below.

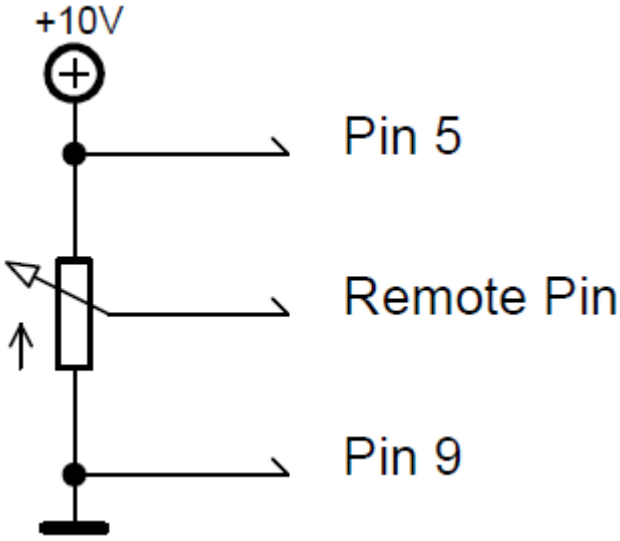
To connect a switch:

1. Take +10V DC voltage in pin 5 for the grandMA3 console or I/O Node
   -or-
   +5V DC voltage for grandMA3 onPC command wing and command wing XT
2. Take an external voltage source (+10V DC in grandMA3 consoles or I/O Nodes and +5V DC in grandMA3 onPC command wing and command wing XT), connect its ground to the common ground pin of the device.

Connect the +10V DC voltage source to one input pin 1-4 or 6-8 with a potential-free contact (switch, buzzer, motion detector, or any other switching device) in between.

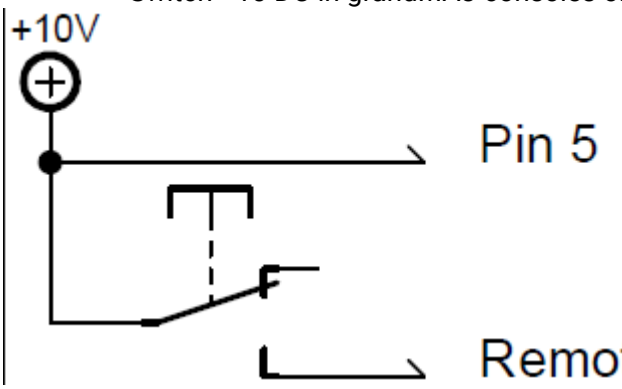
Circuit examples:

- Potentiometer +10V DC in grandMA3 consoles or I/O Nodes



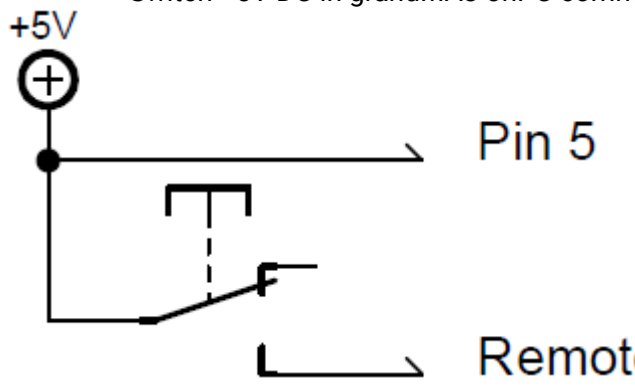
Potentiometer +10V DC

- Switch +10 DC in grandMA3 consoles or I/O Nodes



Switch +10V DC

- Switch +5V DC in grandMA3 onPC command wing and command wing XT

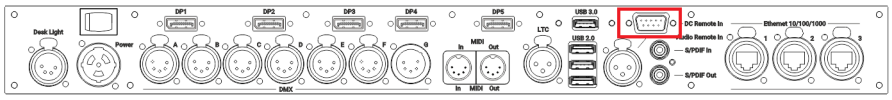


Switch +5V DC

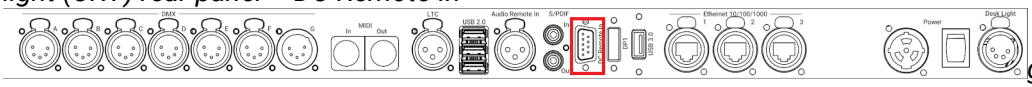
	Important:
	Pin layout in grandMA3 consoles or I/O Nodes: The grandMA3 consoles or I/O Nodes have a 9-pin D-sub, enabling 7 remote inputs: Pin 1-4 = input channels 1, 2, 3, 4 Pin 5 = +10V DC grandMA3 consoles or I/O Nodes/+5V DC grandMA3 onPC command wing and command wing XT Pin 6-8 = input channels 5, 6, 7 Pin 9 = common ground

- Connect a D-sub plug to the DC Remote In connector on the rear panel.

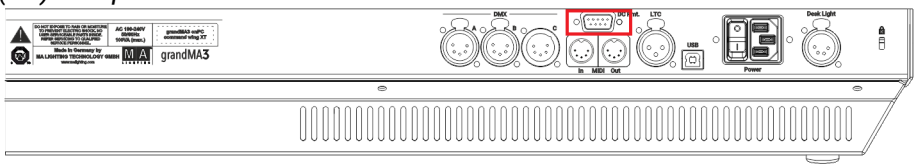
DC Remote In control is connected.



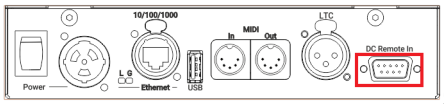
grandMA3 full-size (CRV) and light (CRV) rear panel – DC Remote In



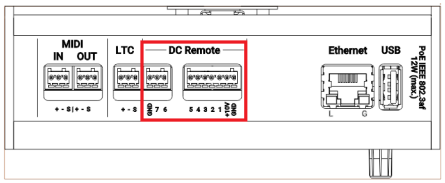
grandMA3 compact (XT) rear panel – DC Remote In



grandMA3 onPC command wing rear panel – DC Remote In



grandMA3 I/O Node *rear panel – DC Remote In*



grandMA3 I/O Node DIN-Rail *rear panel – DC Remote In*

To view the Pinout of the Sub-D connector, refer to the topic **Connector Pin Assignment**.

1.5.15. Connect grandMA3 extension

The grandMA3 extension allows to extend the amount of physical executor handles for a grandMA3 full-size, grandMA3 light or grandMA3 replay unit.

For more information about this device, please refer to **grandMA3 extension** in the Device Overview section.

The number of grandMA3 extensions connected to a grandMA3 console depends on the console type:

Device	Number of grandMA3 extensions
grandMA3 full-size / full-size CRV	1
grandMA3 light / light CRV	2
grandMA3 replay unit	3

Requirements:

- The grandMA3 extension is similar to network / network interface as MA-Net communication. For more information about the network, see **Protocol Details**.
- The grandMA3 extension software version must match the console software version.

To display the software version on an extension:

- Boot the extension. The software version is displayed in the lower-left corner.

To display the extension software version on a console:

1. Tap Menu and Network. The Network window opens.
2. Tap on the arrow ▶ next to Extension. The flip-menu opens.
3. The software version is shown under Version Big in the grandMA3 extension column.

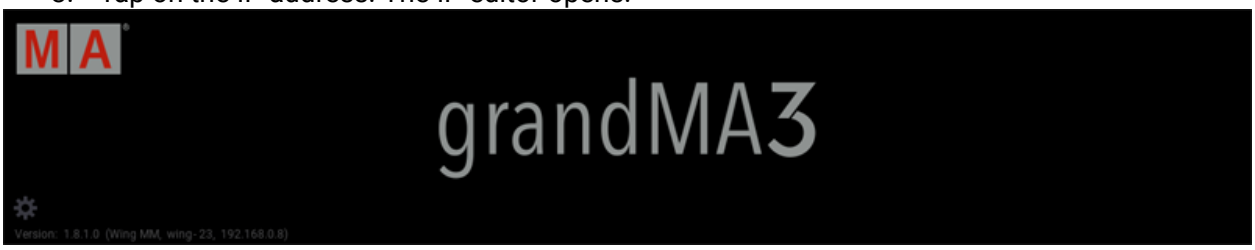
	**Hint:**
	An extension only connects to a specific console. It is not a standalone device.
	**Important:**
	To establish a connection with an extension, the IP address of the extension must be in the same IP range as the console.

To change the extensions IP address remotely from a console or onPC station:

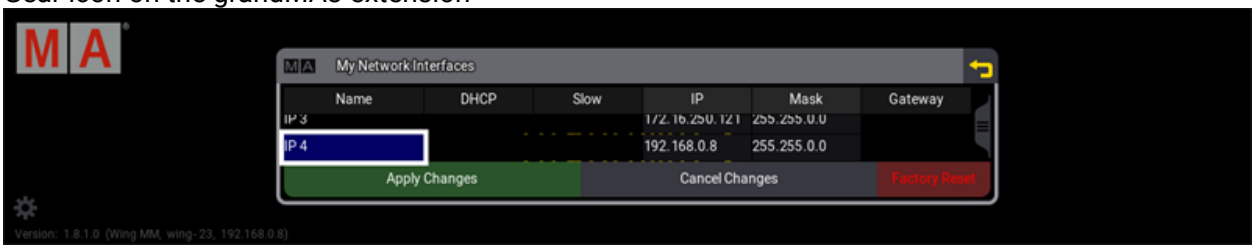
1. Tap Menu and Network. The network window opens.
2. Select the grandMA3 extension entry and click IP address / tap and hold IP address.
3. Use the pop-up window to change the IP settings.

To change the IP address on the extension:

1. Turn on the extension.
2. Tap the gear icon to open the Network Interface. A network interface pop-up opens.
3. Tap on the IP address. The IP editor opens.



Gear icon on the grandMA3 extension



Network Interfaces on the grandMA3 extension

To establish a connection between a console and an extension:

1. Select the extension in the network menu on the console,
2. Tap Invite Station. The connection will be initiated.

	Important:
	It is not possible to invite the extension to a second console, when it is already connected to a different console.

To disconnect the extension from a console, cancel the connection on the console the extension is connected to at the moment.

- Select the console in the network menu and tap Dismiss Station.

It can be connected to a different console now.

The column Remote IP in the network menu displays to which console the extensions are connected at the moment.

When an extension is successfully connected with a console, it will display the first wing of executors by default.

To display a different executor wing:

- Tap Menu - Settings - Extension Configuration. Within the Extension Configuration menu, it is only possible to change the WingID for the connected extension.

There are 2 column modes: Condensed (default) and Full.

1. The condensed mode only displays information about the connection state, IP, and WingID.
2. The full mode also displays the columns Connected Count and Device Type.

- Connection State: Can be Yes or No. This cell cannot be edited by the user and displays if the extension is connected with the console or not.
- IP: Displays the IP address of the extension. This cell also cannot be edited by the user.
- Wing ID: Displays the ID of the executor wing the extension controls. This property can be changed by the user.
- Connected Count: The higher the number the more often the extension tried to connect to the console. In a faulty network environment, the number can increase fast due to reconnection. This value cannot be edited by the user.
- Device Type: This column displays the device type of the extension. Typically it is grandMA3 Fader Module Encoder (MFE). This cell also cannot be edited.

The desk light of an extension is controlled together with the desk lights of the console.

The custom section of an extension can mirror the **custom section of the console** or can be independent of the console, depending on the user settings.

To learn more about user settings, read the **User settings topic**.

1.5.16. Connect grandMA3 fader wing

The grandMA3 onPC fader wing expands the onPC system with additional playback capabilities.

For more information about this device, please refer to **grandMA3 onPC fader wing** in the Device Overview section.

	Important:
	It is only possible to connect fader wings to grandMA3 onPC systems. A maximum of two fader wings and a command wing can be connected.
	Important:
	Regardless of the number of devices connected to the grandMA3 onPC software, the maximum number of parameters is limited to 4 096.

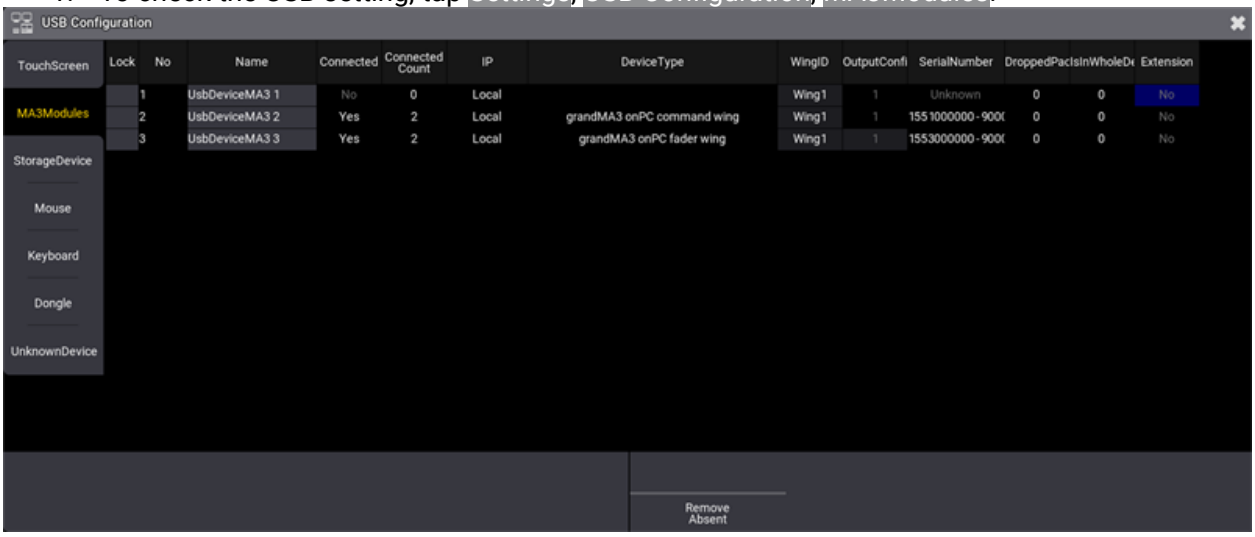
For more information, see **Expand the Amount of Parameters topic**.

Procedure:

1. Connect the grandMA3 onPC fader wing with:
    - command wing XT
    - PC
    - Laptop
2. Start the grandMA3 command wing XT, PC or laptop.
3. Switch on the grandMA3 onPC fader wing.

To connect a grandMA3 onPC fader wing in combination with a grandMA3 onPC command wing to expand the number of executors, the wing configuration has to be changed:

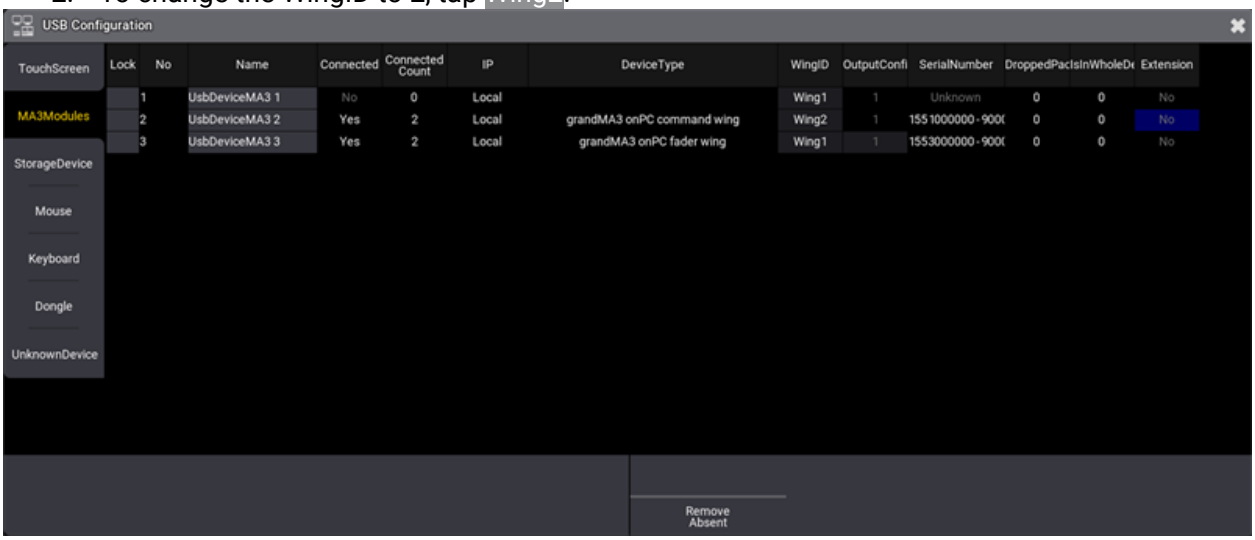
1. To check the USB setting, tap Settings, USB Configuration, MA3Modules.



USB Configuration setting

1. To adjust the wing ID of the command wing, tap or right-click WingID. A pop-up window opens.

2. To change the WingID to 2, tap Wing2.



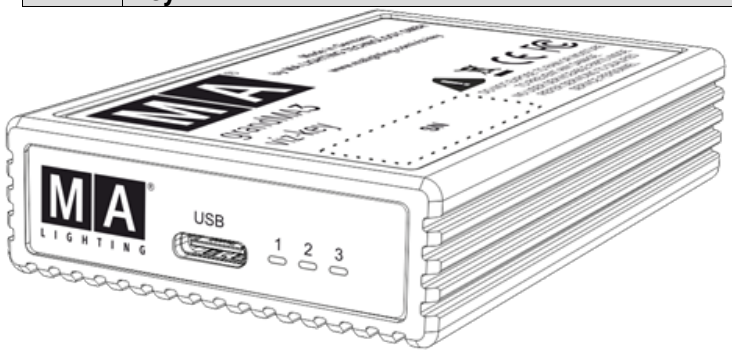
USB configuration set to Wing2

1.5.17. Connect grandMA3 viz-key

The grandMA3 viz-key is needed to:

- Establish a connection between a grandMA3 software and a third party visualizer with an unlimited number of virtual parameters.
- Output 512 parameters via network.

	Hint:
	If the correct version of grandMA3 onPC is installed on the visualizer workstation, no further download or installation is required.
	Hint:
	When sufficient parameters are already available in a session the grandMA3 viz-key hardware is not required. Only grandMA3 viz-key software has to be installed on the visualizer. For more information, see **Update grandMA3 viz-key**.



grandMA3 viz-key

For general information about grandMA3 viz-key and the supporting companies, see the following website: **https://www.malighting.com/grandma3/products/**.

To see the quick manual for the grandMA3 viz-key, see **Quick Manual viz-key**.

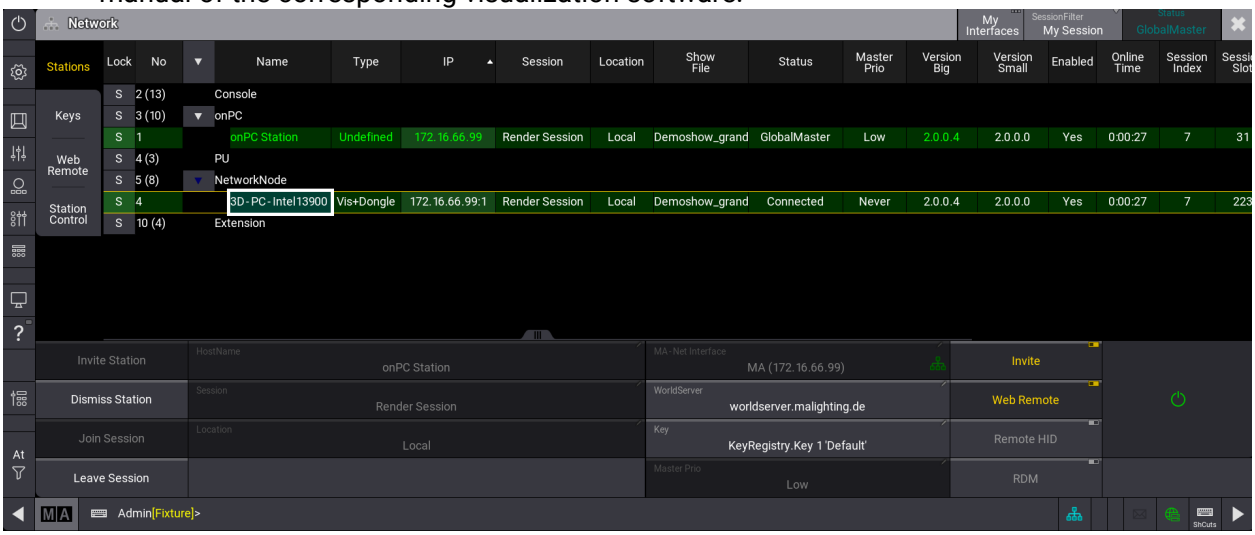
For a direct link to the information about the LEDs, see **Quick Start**.

Use Cases for a Viz-Key Connection

Connect a grandMA3 onPC and a third party visualizer on the same workstation.

1. Connect the grandMA3 viz-key to the workstation via USB.
2. Open the grandMA3 onPC software.
3. Click . The menu opens.
4. Click Settings and then click onPC Local Settings in the dropdown menu.
5. Disable Connect to viz-key. For more information, see **onPC Local Settings**.

6. Start the third-party visualization software and set up a network connection. Make sure to use the correct grandMA3 software version and the correct IP address, please refer to the user manual of the corresponding visualization software.

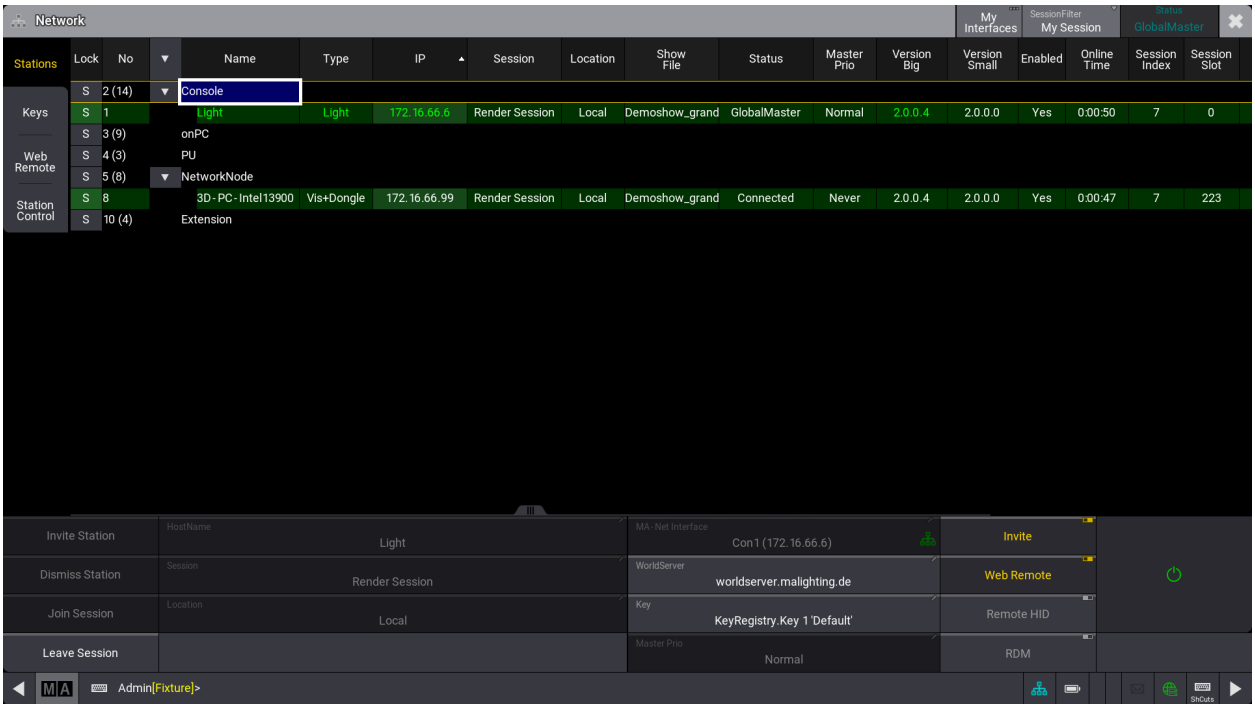


onPC station with visualizer connected

	Hint:
	Disable the connection between the viz-key and grandMA3 onPC to have unlimited parameters instead of the 512 parameters available for visualization.

Connect a grandMA3 onPC station / grandMA3 Console to a separate visualization workstation via network.

1. Connect the grandMA3 viz-key to the Visualizer workstation via USB.
2. Start the third-party visualization software and set up a network connection.
3. Invite the visualizer workstation to a session on the grandMA3 onPC station / grandMA3 console. The visualization workstation is connected to a session.



Console with connected visualizer workstation

	Hint:
	The configured session above gives access to more than the granted parameters of the corresponding grandMA3 devices for visualization.

Output DMX on grandMA3 onPC via a network protocol.

1. Connect the grandMA3 viz-key via USB to the workstation.
2. Open the grandMA3 onPC software.
3. Click . The menu opens.
4. Click Settings and then click onPC Local Settings in the dropdown menu.
5. Enable Connect to viz-key.
6. Create a session or use network protocol to output DMX. For more information, see **Create a Session** and **Ethernet DMX**.

Viz-key Licence Update

To validate the license of the grandMA3 viz-key an internet connection to the world server is required. Once validated, the license is valid for 10 days.

The grandMA3 viz-key has to be validated on the same workstation where it is used.

To indicate an invalid license, the **LicenceInvalid** status is displayed in the network menu.

To monitor the correct connection of the viz-key hardware, the hosttype **Vis+Dongle** is displayed on the corresponding instance.

1.5.18. Turn on the device for the first time

	Hint:
	The safety instructions and technical specifications can be found in the Quick Manual of the respective product, e.g. consoles: For information on safety instructions, see **grandMA3 Quick Manual Consoles - Safety**. For technical specifications, see **Technical Data** in the **grandMA3 Quick Manual consoles**.

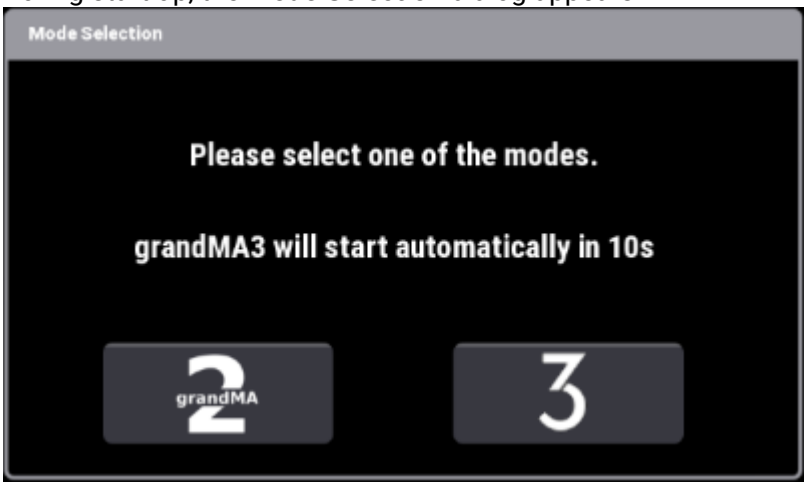
Turn on the grandMA3 Device

1. Unpack the grandMA3 device.
2. Remove packaging and cushioning material.
3. Place the device indoors on a stable surface.
4. Connect external monitors using the native DisplayPort connectors (optional).
5. Connect an external mouse or keyboard using the USB ports (optional).
6. Insert the power connector into the corresponding jack.
7. Connect the power plug with the mains.
8. Turn on the power switch on the rear panel.
9. Press the power key on the front panel.

The device starts booting.

Select Mode

During start-up, the **Mode Selection** dialog appears:

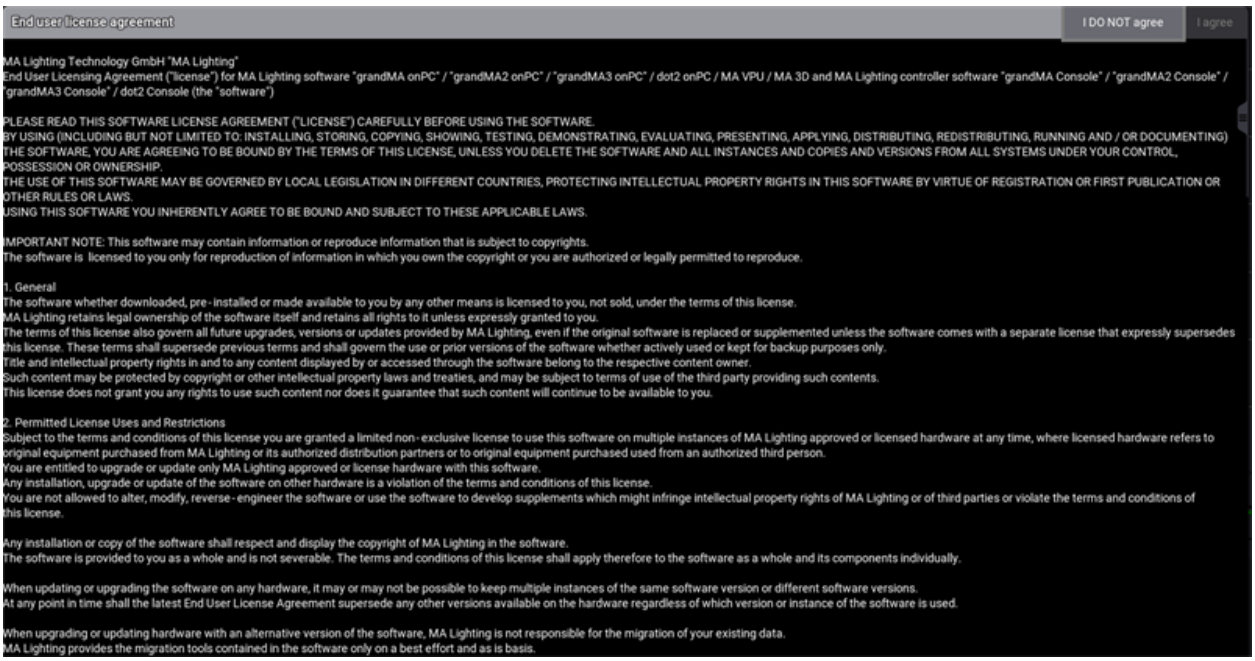


- Tap to select a mode.

For more information, read the **Mode2 topic** in the section grandMA3 Mode2 of the **grandMA2 User Manual**.

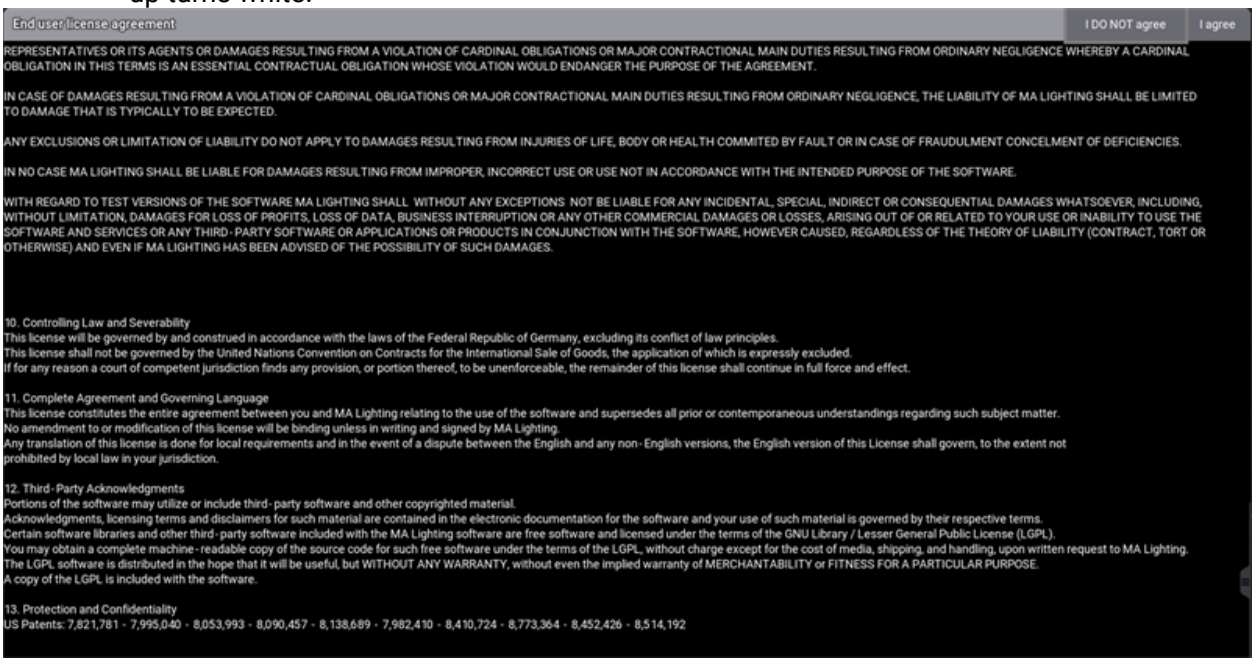
End-User License Agreement (EULA)

After selecting grandMA3 mode, the console starts to boot and the pop-up **End-User License Agreement (EULA)** opens.



End-user License Agreement (EULA)

- Scroll down to read the complete EULA. The button I agree in the upper right corner of the pop-up turns white.



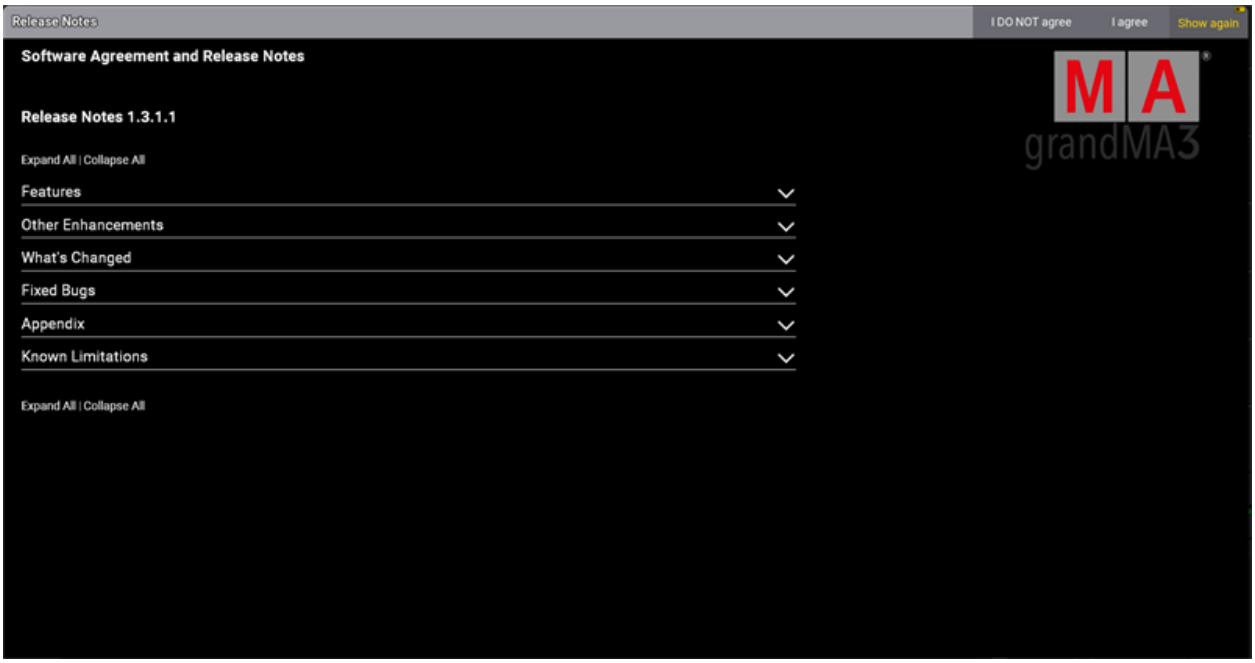
End-user License Agreement (EULA) I agree button

- To confirm the EULA, tap I agree.

	Hint:
	The EULA is only displayed the first time the console is booted and each time it is booted after a software update.

Release Notes

The Release Notes of the current version appear on the screen.



Release Notes

- To confirm the Release Notes, tap I agree.
- If you do not want the software to show the Release Notes again after booting, tap Show again to disable.

You can now use your grandMA3 device.

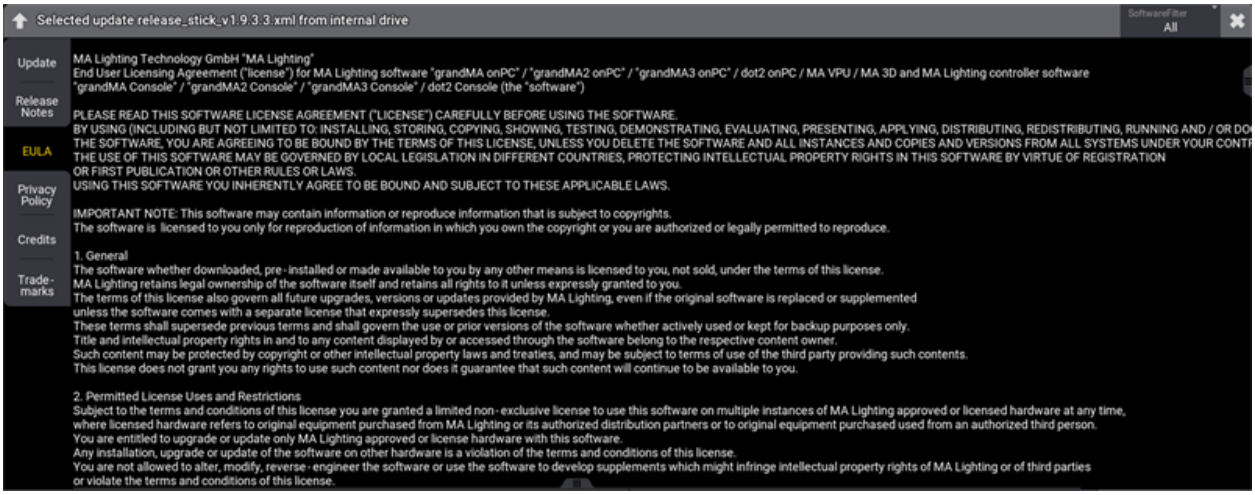
Third-party Software

The software comes with standard codecs to play videos and display images.

When you install third-party software, make sure that you have the required licenses.

Third-party software can be activated by accepting the additional third-party software agreement.

- To accept the additional third-party software agreement, **open the Software Update menu**.



- Tap EULA, then tap Third-Party Software.

- Tap I AGREE to accept the additional third-party software agreement

1.5.19. Shut Down the System

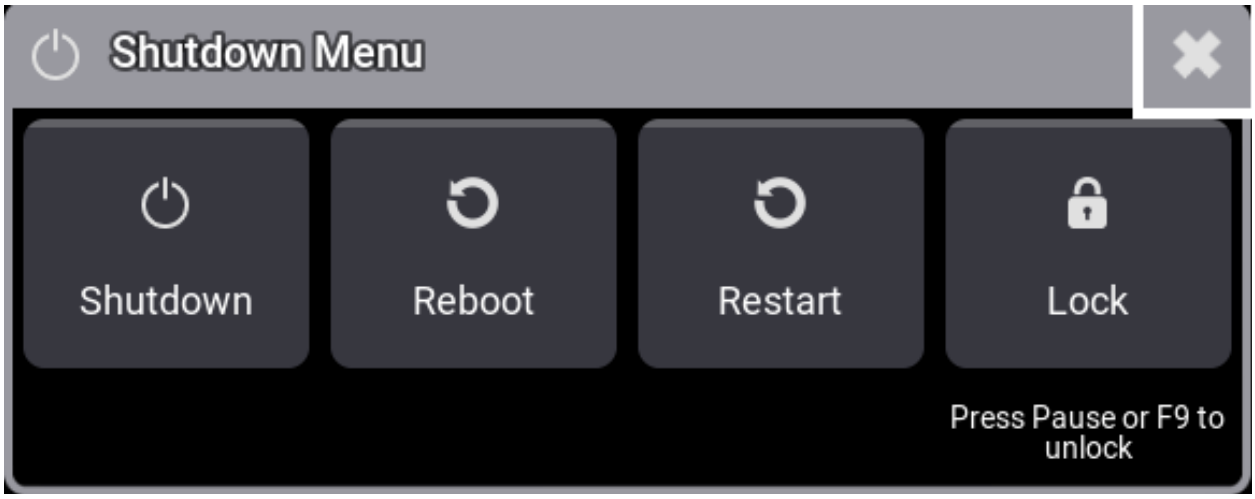
Before shutting down the system, save the show file via the **Backup menu**.

To start the shutdown procedure, use one of the following options:

Use the Software UI

- Tap the icon at the top of the control bar.

The shutdown menu opens:



Shutdown Menu - Buttons

- To shut down, tap the icon.
- To reboot, tap the Reboot icon. This button is only available on consoles.
- To restart, tap the Restart icon.
- To lock the desk, tap the icon. For more information, see **desk lock**.

Use the Power Key

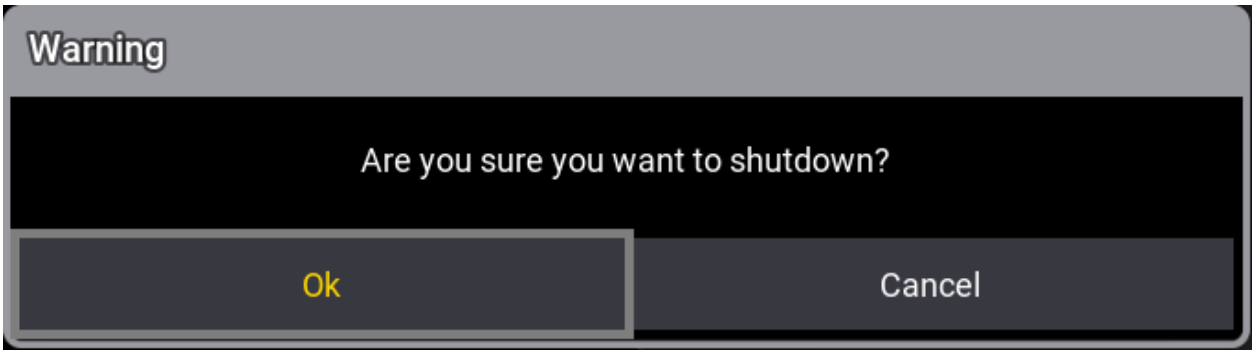
- Press the **power key** on the front panel.

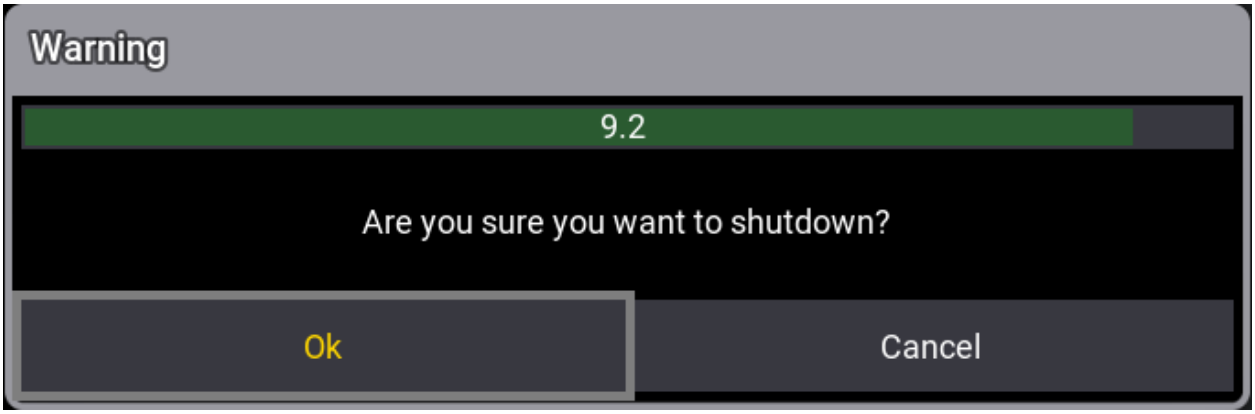
Use a Keyword

- **Reboot keyword**
- **Restart keyword**
- **ShutDown keyword**

Shutdown Warning Pop-Up

One of the following warning pop-ups appears:





- Tap OK.

The grandMA3 device shuts down.

	Hint:
	When using the **ShutDown keyword**, you can choose an option without a shutdown pop-up or without timer.
	Hint:
	If an attempt is made to shut down before saving the show file, a save show file pop-up will appear. For more information, see **Save Show File**.

- To shut down the current station without confirmation, type:

MA	User name[Fixture]>ShutDown /nc

- To shut down the current station without timer, type:

MA	User name[Fixture]>ShutDown / noautoclose

1.6. grandMA3 onPC

MA Lighting offers the perfect solutions for getting started with. Simply download the grandMA3 onPC software and connect to any of the grandMA3 onPC products to unlock parameters.

The amount of DMX outputs varies with each grandMA3 onPC product but all of them unlock parameters for the full grandMA3 software functionality. There is no compromise in feature set and the range of products allow you to make the right choice for your budget or individual needs.

There are three choices for onPC solutions: the grandMA3 onPC command wing, which offers the ultimate grandMA3 mobile solution, the grandMA3 onPC command wing XT, featuring a built-in MA mother board, and the grandMA3 onPC xPort Nodes, which offer the most cost-effective DMX output solution for software only show control.

Every grandMA3 onPC xPort Node is capable of handling MA-Net3, MA-Net2, sACN and Art-Net data and is fully RDM compliant. A 3.9" color display on the front face allows for simple configuration and provides a quick overview of the status of each Node.

All grandMA3 onPC xPort Nodes are equipped with a powerful processor and a 1,000Mbit/s Ethernet connection to ensure the most reliable and stable Ethernet to DMX and vice versa conversion.

The grandMA3 onPC xPort Nodes are high quality solutions that can handle busy Ethernet connections without ditching DMX or RDM messages.

Subtopics

1.6.1. System Requirements

If you would like to run grandMA3 on your PC, here is what it takes.

Type	Minimum	Recommended
Operating system	Windows® 10 64bit version 1803 or later, Windows® 11 64bit version 21H2 or later, Apple® macOS® Catalina 10.15 or later, Admin rights are required for all operating systems.	
Processor	6th Generation Intel® Core CPU™ processor-based platform or later with 4 or more cores and SSE 4.2 or comparable AMD® CPU, AVX2, Apple® M1® SoC CPU with 8 or more cores or later.	
RAM	8 GB	16 GB
Hard disk	32 GB of available space	type SSD
Graphics card	Any graphic card with hardware acceleration, OpenGL 4.1 support, and 1024 MB VRAM, or Apple® M1® SoC GPU with 8 or more cores or later.	4 096 MB VRAM 32 GB unified memory
Resolution	1 920 x 1 080 or higher.	
Network card	1000BASE-T	

Additional requirements:

- To use the grandMA3 onPC in session with grandMA3 consoles, a minimum of 16 GB of RAM (24GB Unified Memory when using Apple® M1® SoC or later) is required.
- To use the online help and download the latest version of grandMA3 onPC, make sure that you have internet access.
- To save the software on a USB flash drive, use a USB 2.0 or 3.0 port.

	**Hint:**
	The grandMA3 onPC software currently does not support Windows® 11 on Arm.
	**Hint:**
	In general, more cores and more cache memory are recommended instead of a higher processor clock speed.
	**Hint:**
	For Apple® macOS® High Sierra 10.13 or Apple® mac OS® Mojave 10.14, grandMA3 onPC version 1.8.8.2 is required.
	**Important:**
	We recommend always using the latest driver version.

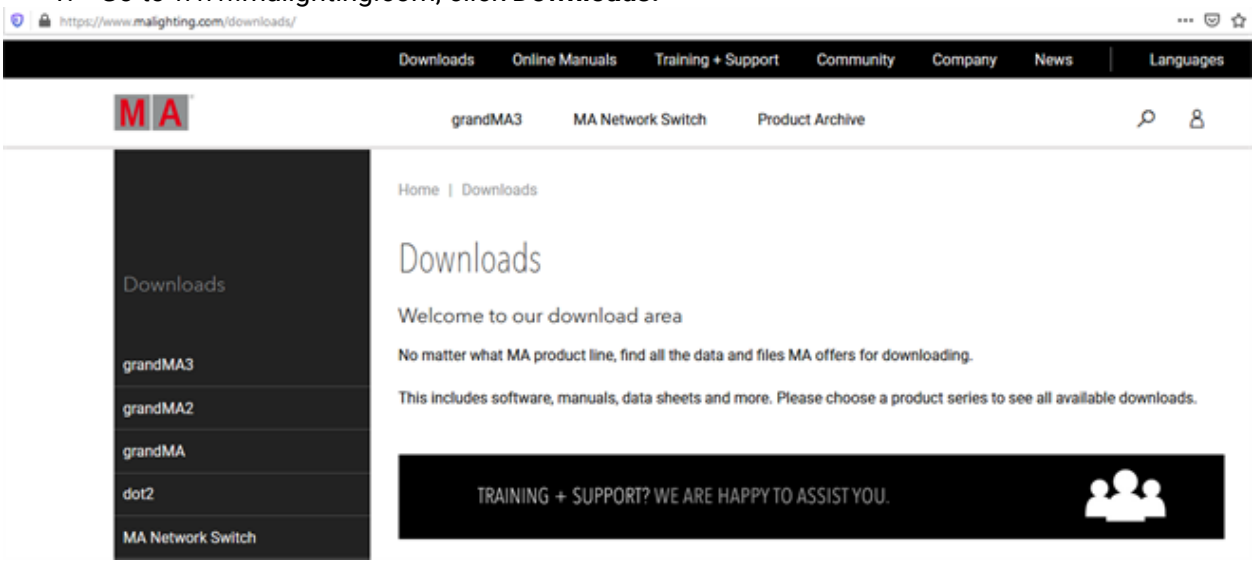
1.6.2. Windows™® Installation

To run the grandMA3 onPC software, copy and install the program files on your PC.

The installation is possible in every root directory or in the **standard directory** "C:\Program Files\MALightingTechnology".

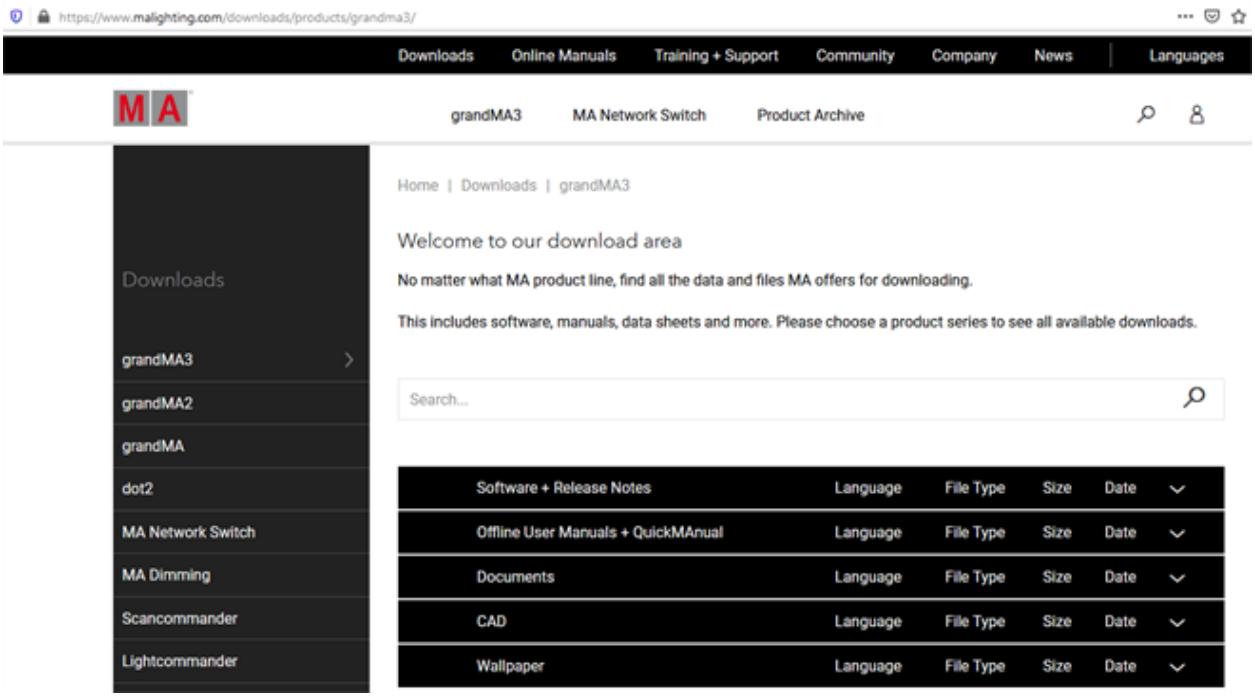
To download the grandMA3 software:

1. Go to www.malighting.com, click **Downloads.**



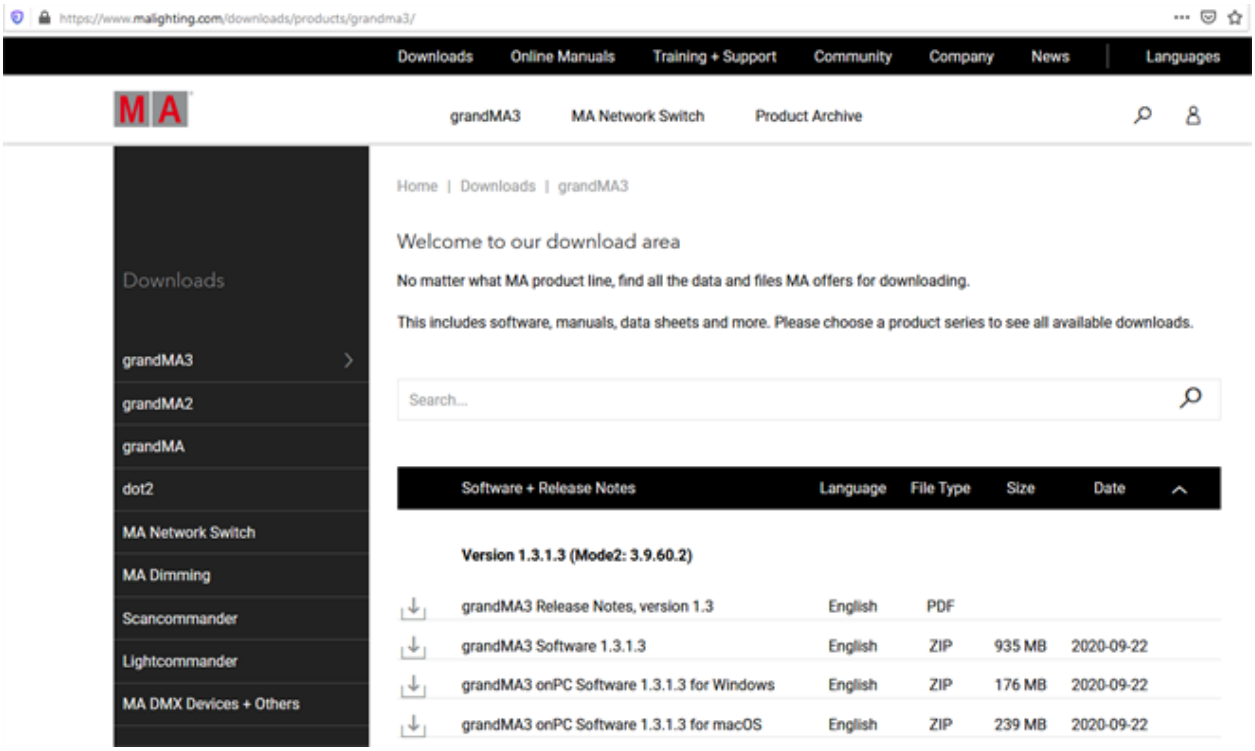
Website malighting.com Downloads

2. Click **grandMA3** in the bar on the left.



grandMA3 section

3. Click **Software + Release Notes** to find the latest software version.



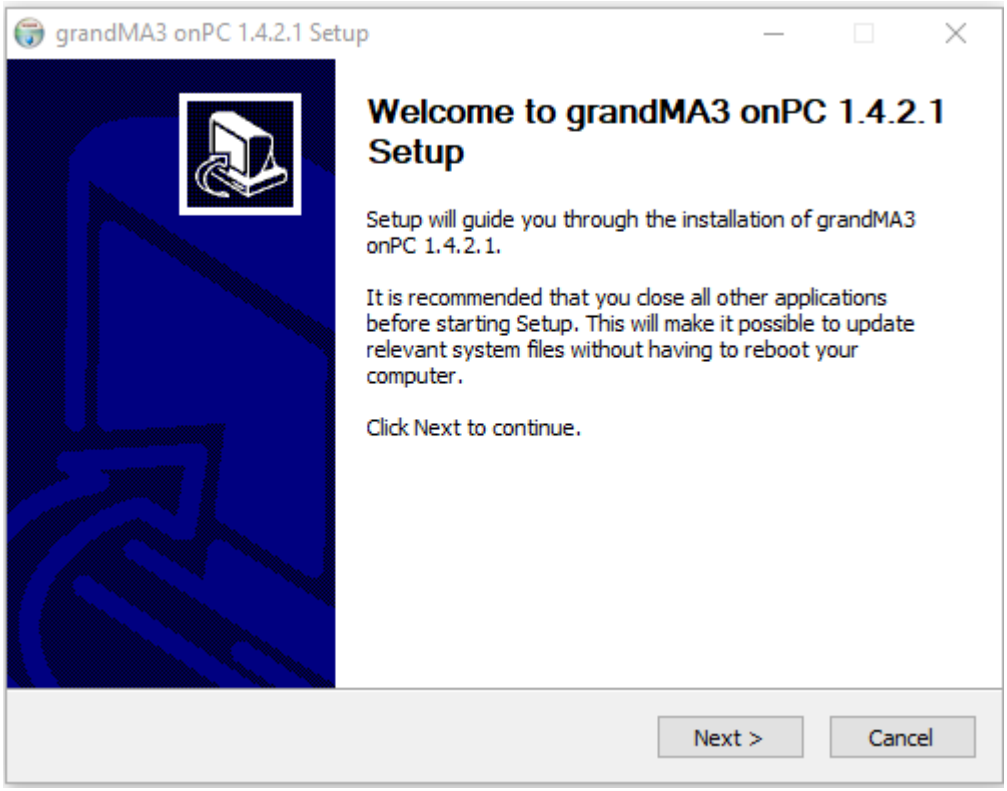
grandMA3 Software section

4. Click grandMA3 onPC Software x.x.x.x for Windows to download the desired installation package.

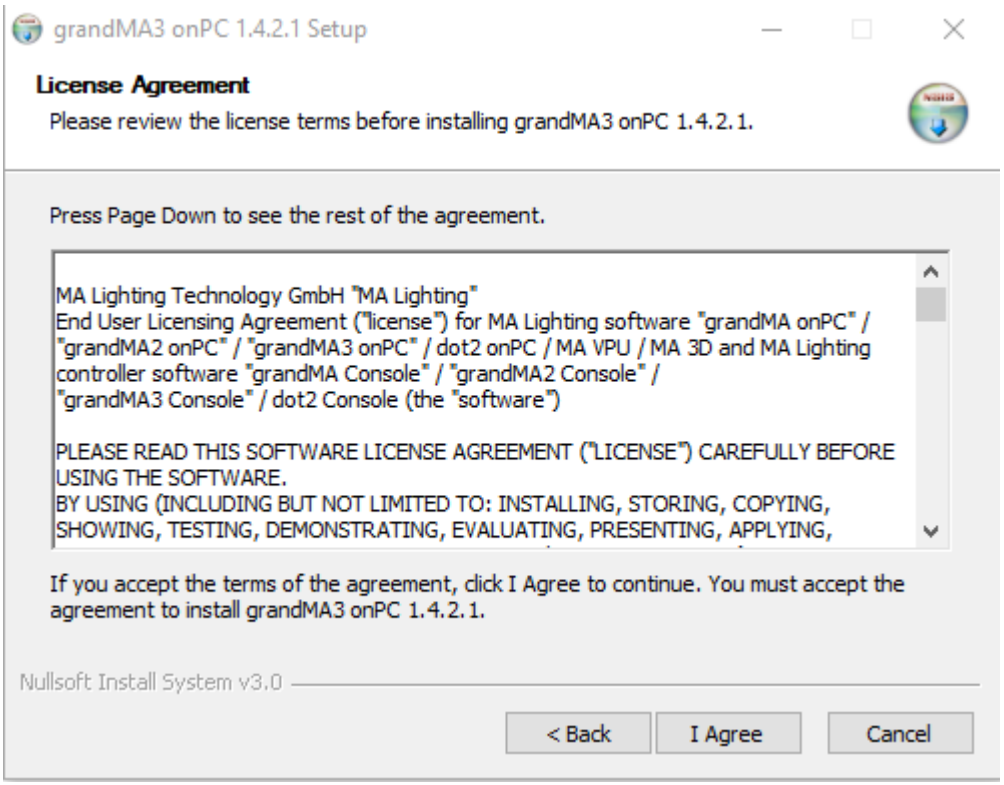
The download process starts.

- Extract the zip file.

- Go to the subfolder "ma".
- Double-click the installation program grandMA3_onPC-vx.x.x.x.exe.
- Confirm with Execute.
  The installation program opens:

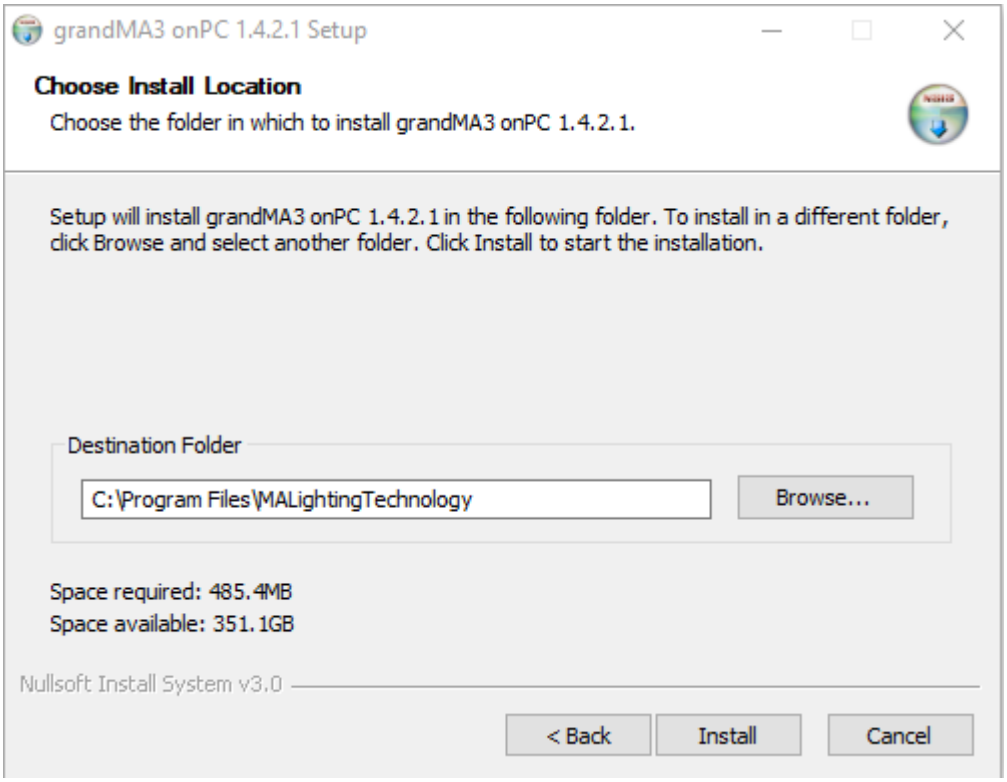


- Click Next. The End User License Agreement (EULA) appears:



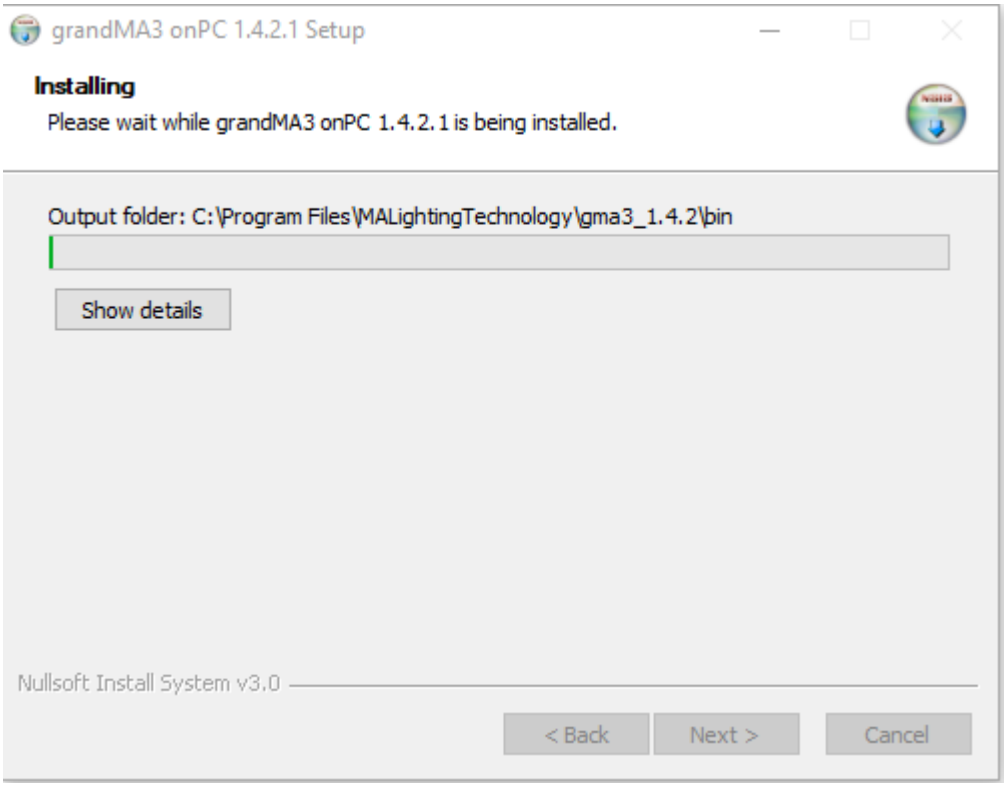
- Click I Agree to accept the agreement.

The Install Location window appears:

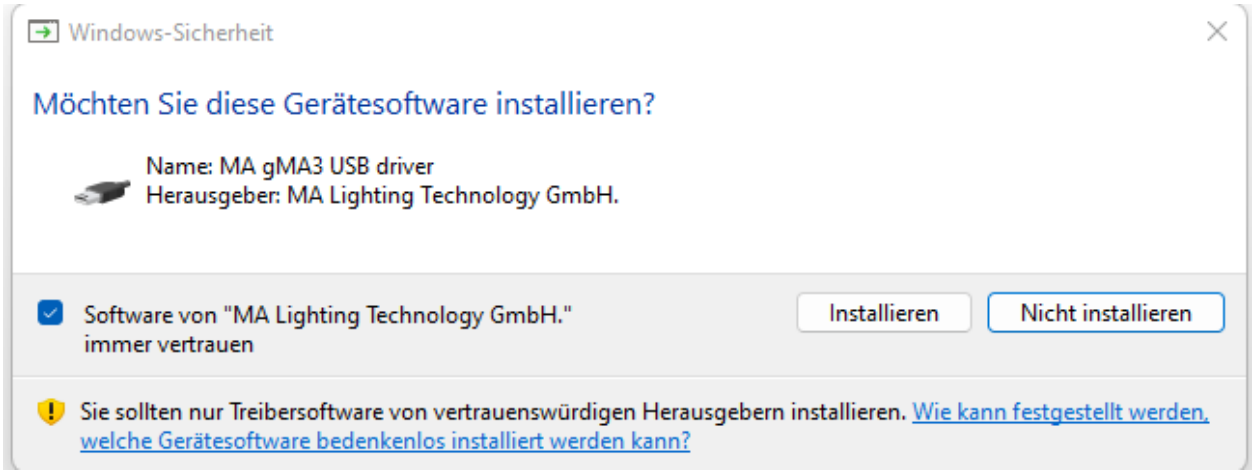


- Watch out for the suggested directory and change it if you would like to do so.

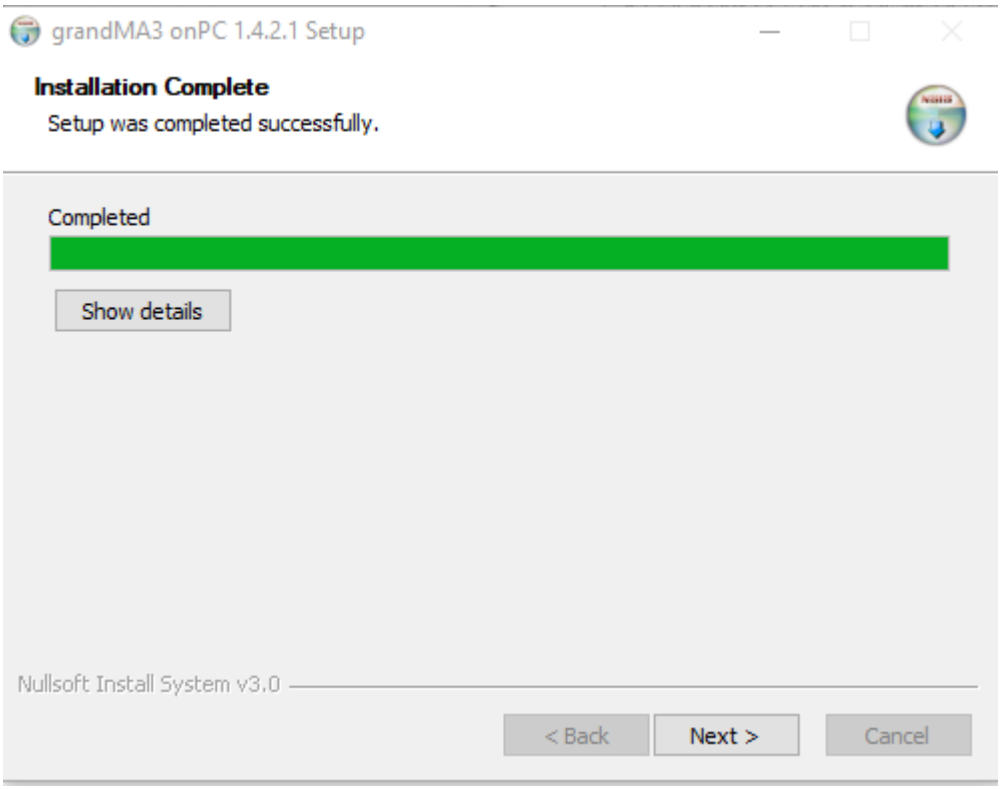
- The standard directory for the resources and user files is "C:\Program Files\MALightingTechnology".
- To learn more about the folder structure, read the **Folder Structure topic** in **File Management**.
- To confirm the installation directory, click Install.
- The program files are copied into the selected directory.



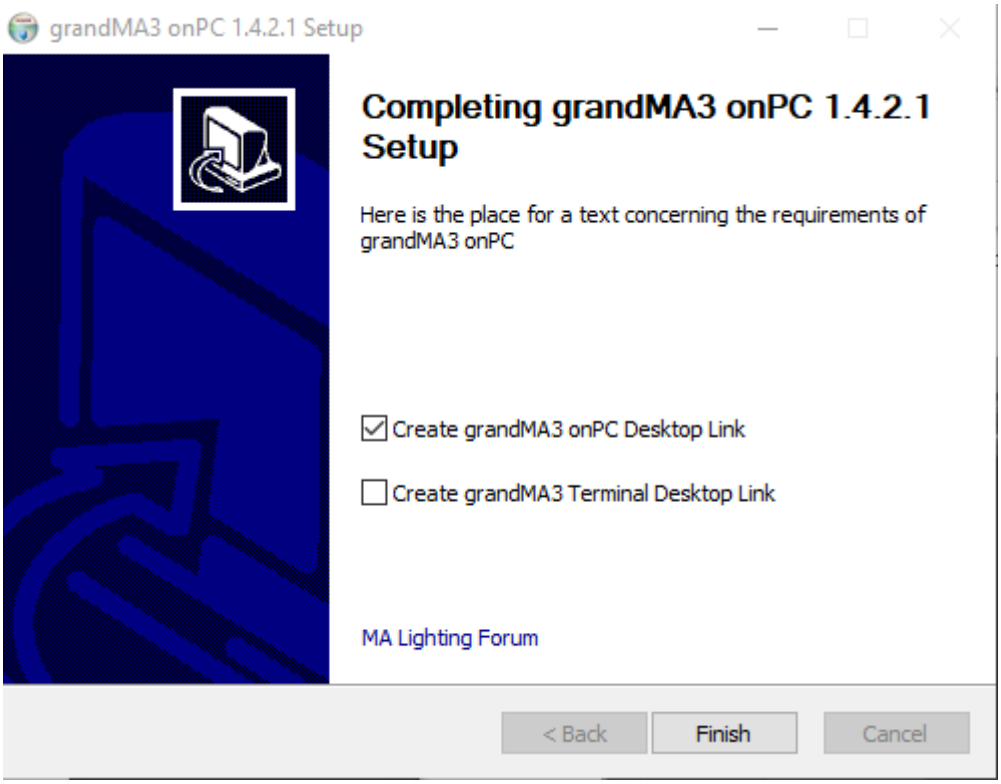
1. To install a USB driver, connect a usb stick before the installation.
2. During the installation, a pop-up window appears. Tap Install.



Click Next to finish the installation.

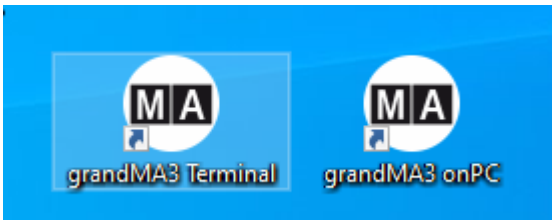


At the end of the installation process, you can choose to set a link to the terminal app or not.



It is disabled by default.

Click the checkbox if you want to create a desktop link to the terminal app.



grandMA3 onPC for Windows

As soon as the installation is finished, the MA icon appears on the desktop.

If several versions of the software have been installed, several subfolders are created in the main folder "C:\ProgramData\MALightingTechnology".

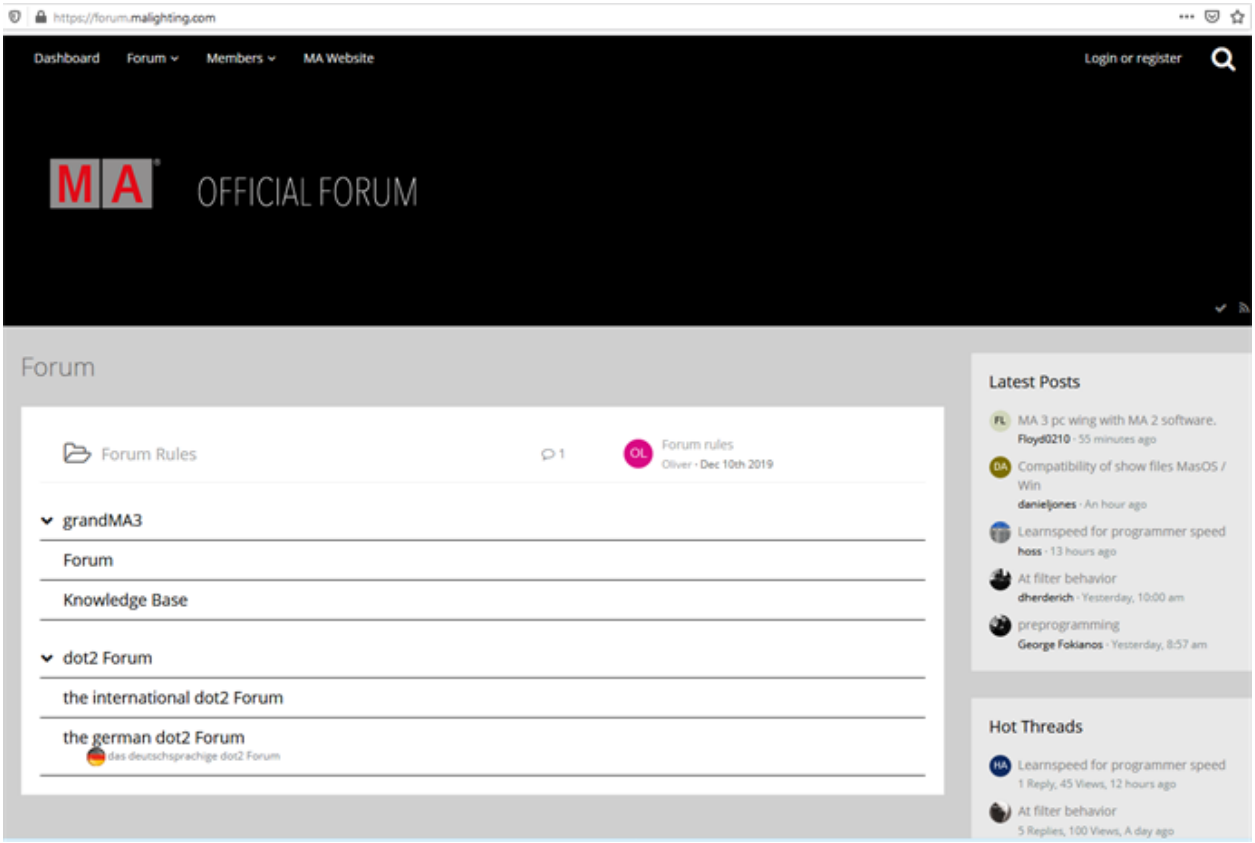
The MA icon on the desktop will always link to the latest version:

This is the main app.

MA Lighting Forum

- To visit the MA Lighting Forum on the internet, click MA Lighting Forum.

The MA Lighting Forum opens in the browser:



1.6.3. Optimize Windows™ ®

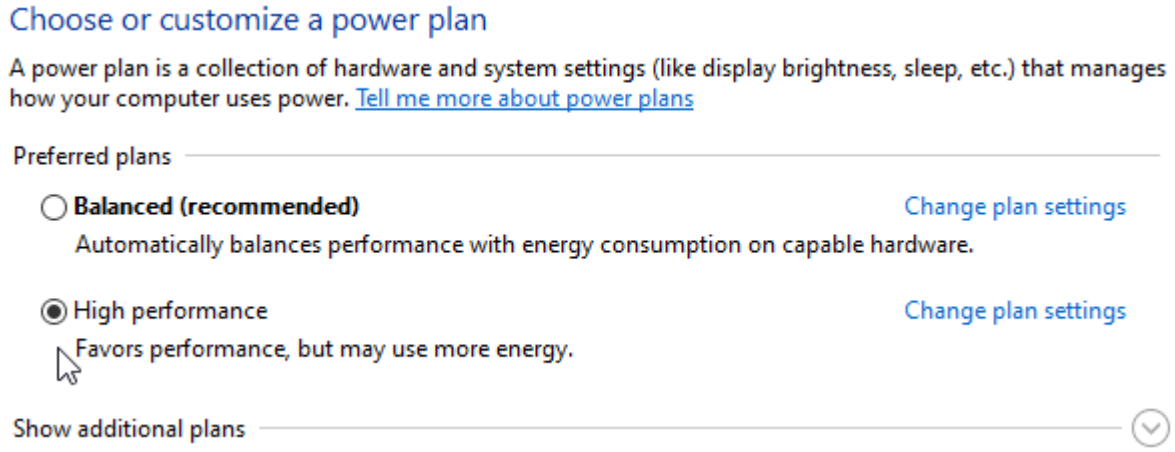
	Hint:
	The grandMA3 onPC software is available for Windows™® and **macOS®**.

- Please refer to the **system requirements topic** to learn more about the computer specifications.
- For more information about the grandMA3 onPC app, the terminal app, and the different version icons, read the **Installation of grandMA3 onPC topic**.
- To run the grandMA3 onPC software even more efficiently, we recommend adjusting the following settings on your computer.

Turn on High-Performance Power Plan

To turn on the high-performance power plan:

1. Click the Start button.
2. Type Control Panel in the search bar.
3. In the search bar of the Control Panel, type power options.
4. Click Power Options.
5. Under Preferred plans, click High performance.



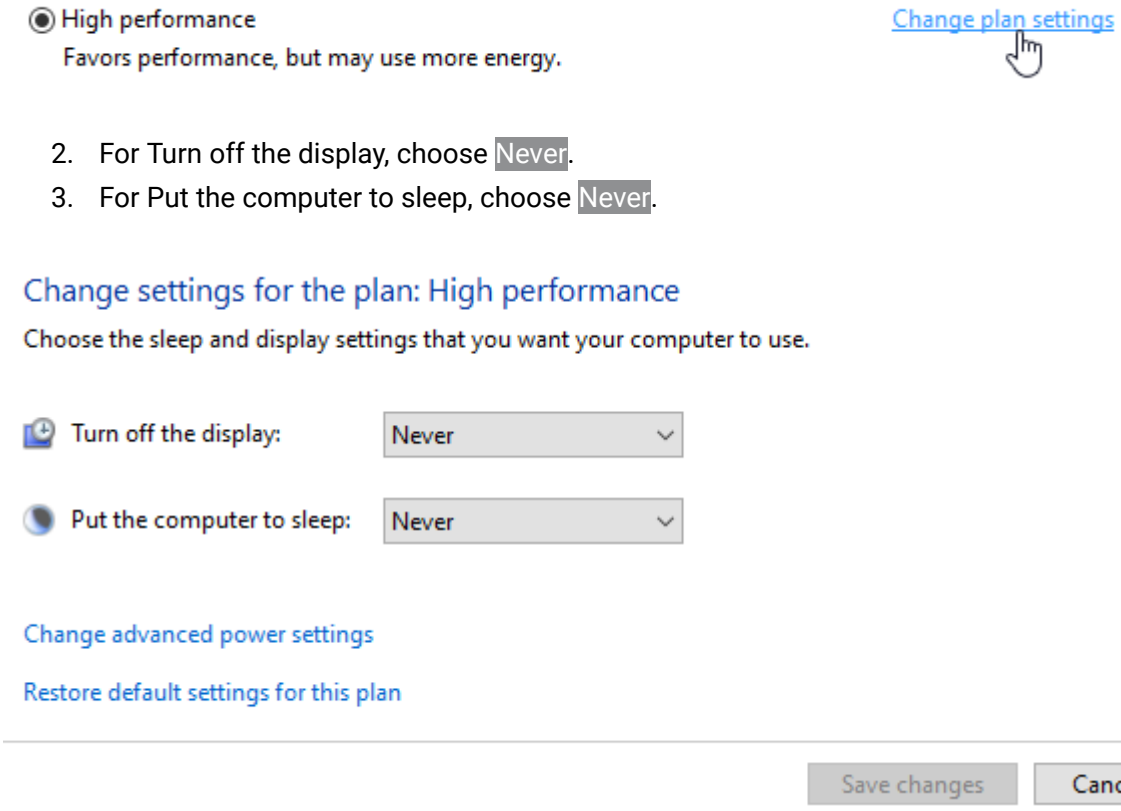
The high-performance power plan is turned on.

Optimize High-Performance Power Options

- Turn off everything that can interrupt the grandMA3 onPC.

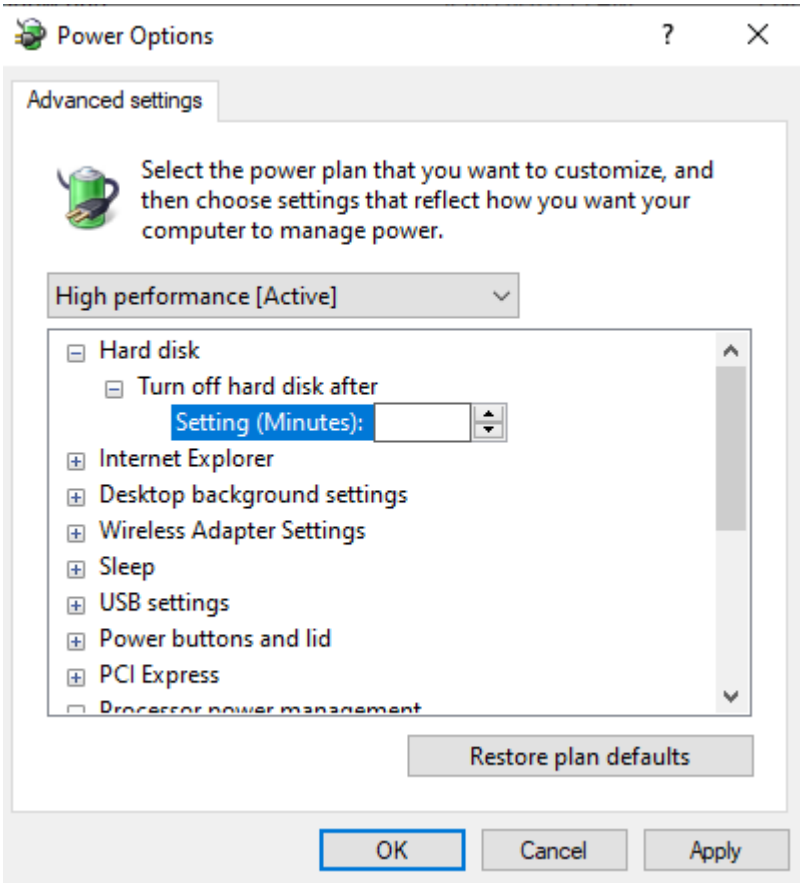
To optimize the power options:

1. Click Change plan settings next to High performance



2. For Turn off the display, choose Never.
3. For Put the computer to sleep, choose Never.

4. Click Save changes.
5. Click Change advance power settings.



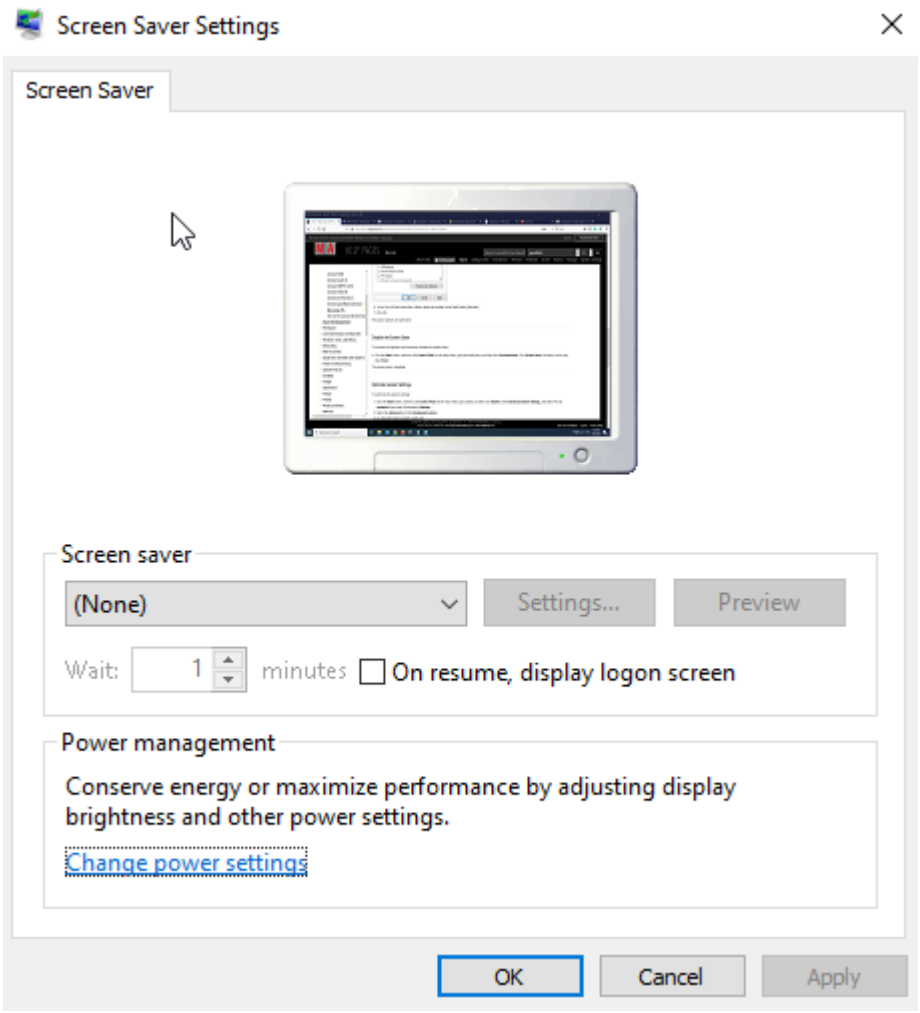
6. To set Turn off hard disk after to Never, delete the number in the field Setting (Minutes).
7. Click OK.

The power options are optimized.

Disable the Screen Saver

To conserve the graphic card resources, disable the screen saver.

1. Open the Control Panel.
2. In the search box, type Screen Saver.
3. Click Change Screen Saver to open the menu.
4. In the Screen saver select (None).
5. Click OK.



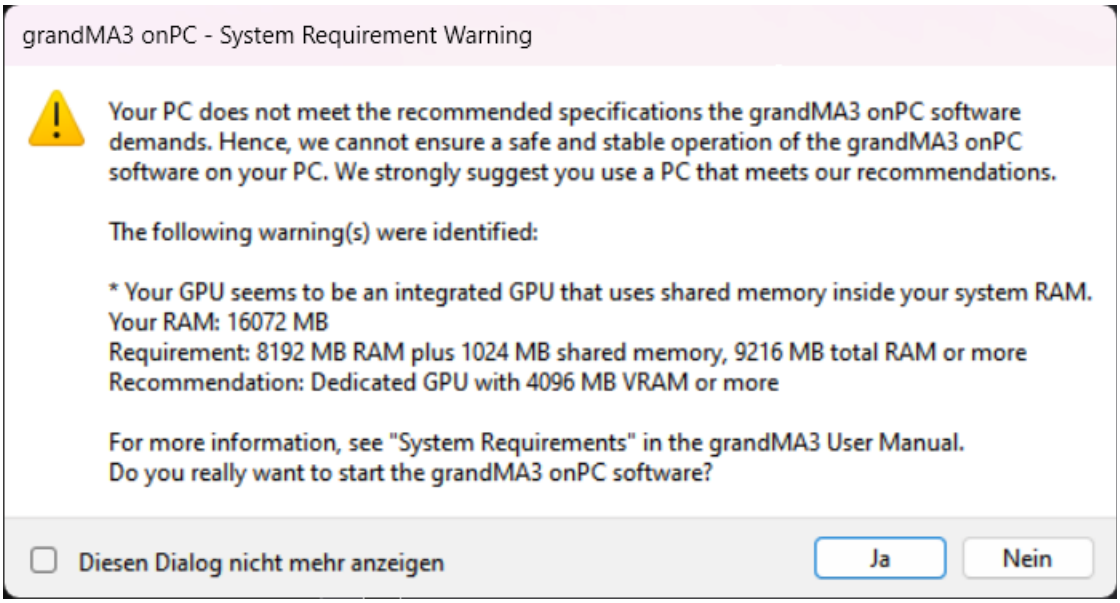
The screen saver is disabled.

System Requirement Warning

When starting the grandMA3 onPC software, a warning pop-up may appear due to hardware performance.

To suppress the system requirement pop-up, enable Do not show this message again.

	Hint:
	Note that each time you update grandMA3 onPC for Windows™®, the system requirements pop-up will appear again.

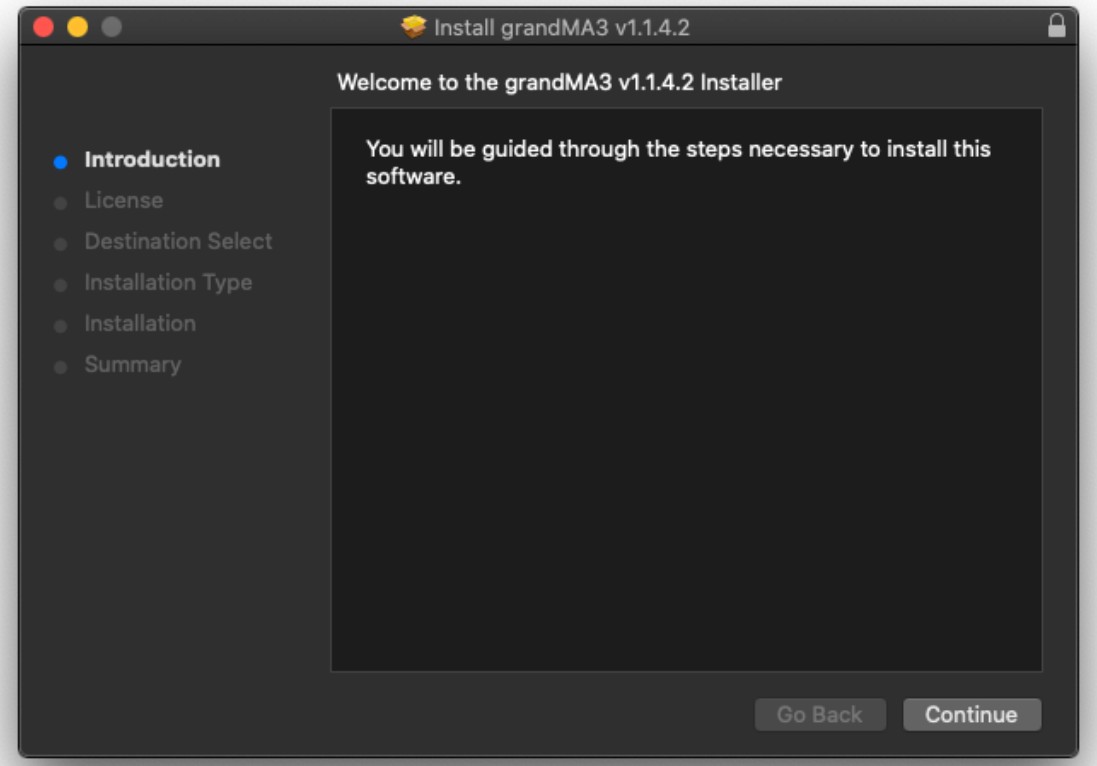


System Requirement Warning

1.6.4. macOS® Installation

To install grandMA3 onPC on macOS®:

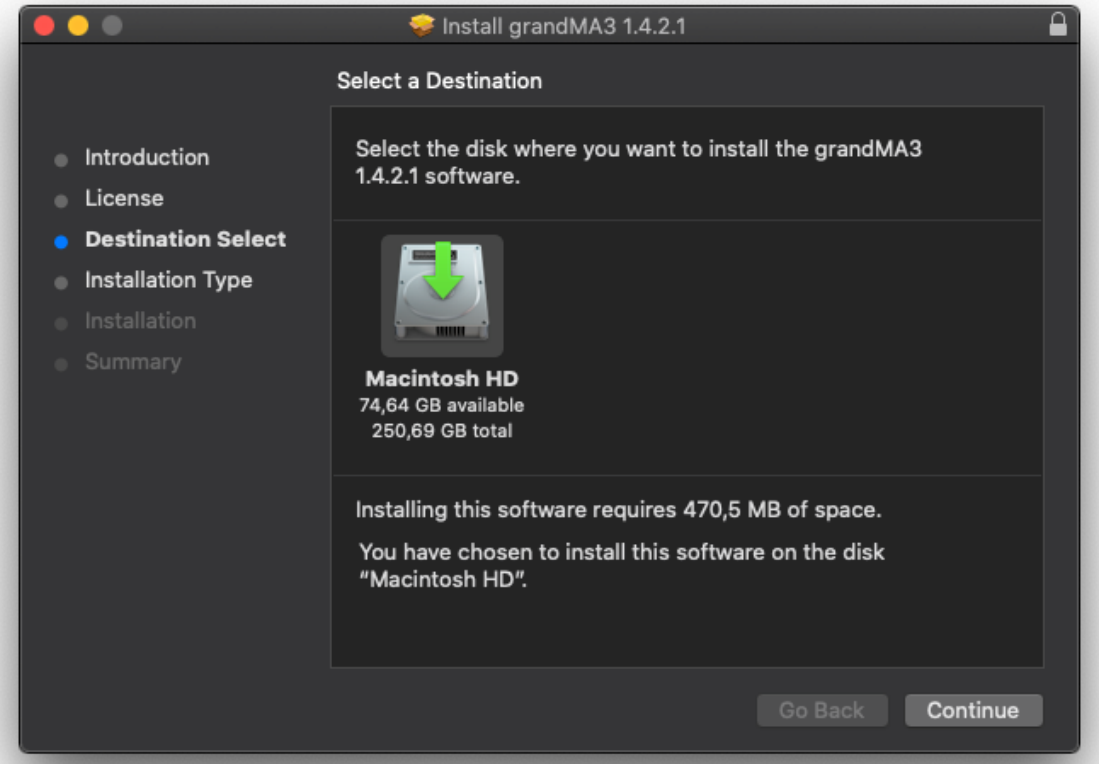
1. Download the installation file from **www.malighting.com**.
2. Click grandMA3 onPC Software x.x.x.x for macOS to download the installer.
3. Open the zip file, double-click it.
4. Go to the subfolder "ma".
5. Double-click the installation program grandMA3_onPC_x.x.x.x.pkg. The Installer opens:



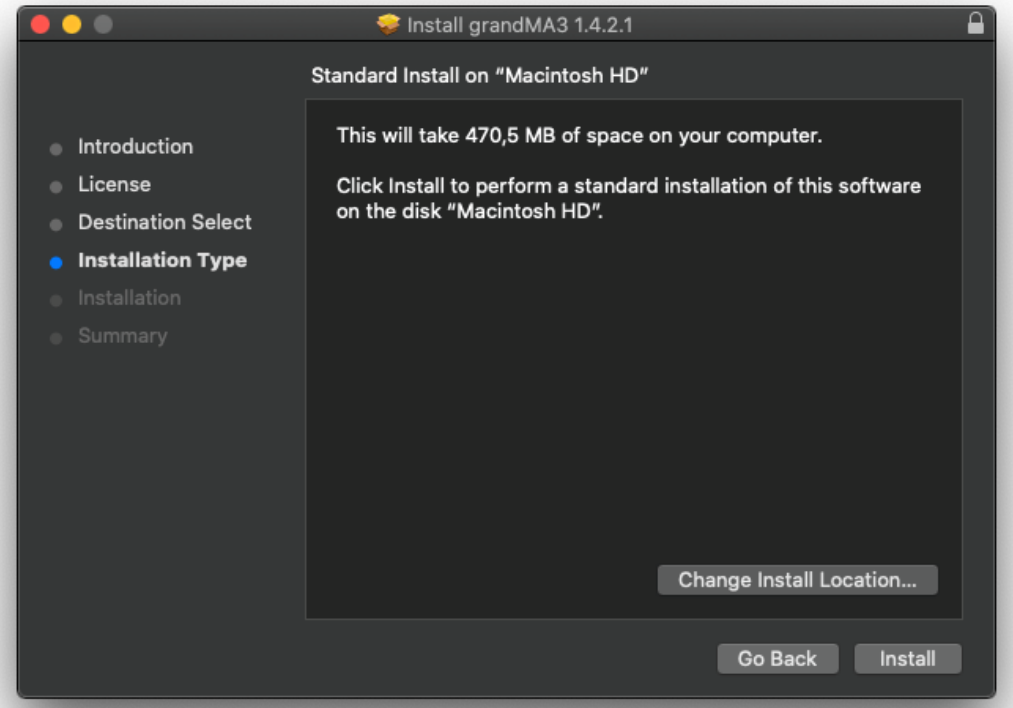
6. Click Continue. The software license agreement opens
7. Click on Continue and then on Agree.

The software is installed in the standard directory /Applications/. The resources are located in /Users/.

8. Optional: Change the destination disk, select the desired disk and click Continue:

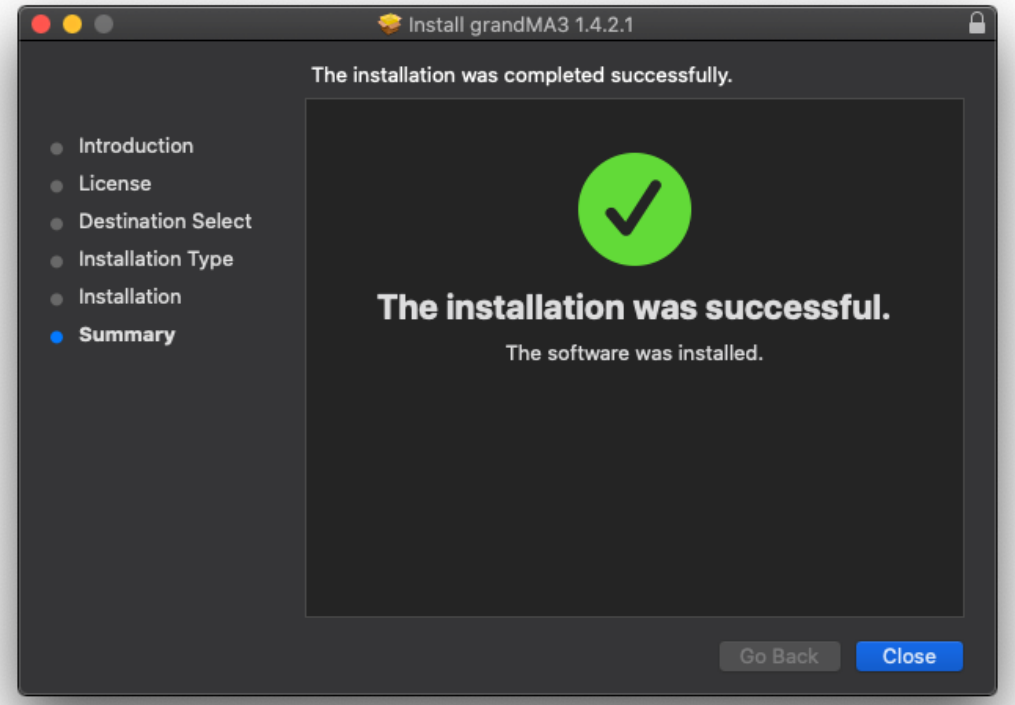


9. Click Install to install the application or Change Install Location to change the destination:



10. To install the software, click Install. During the installation, it is possible that you are asked for the administrator password.

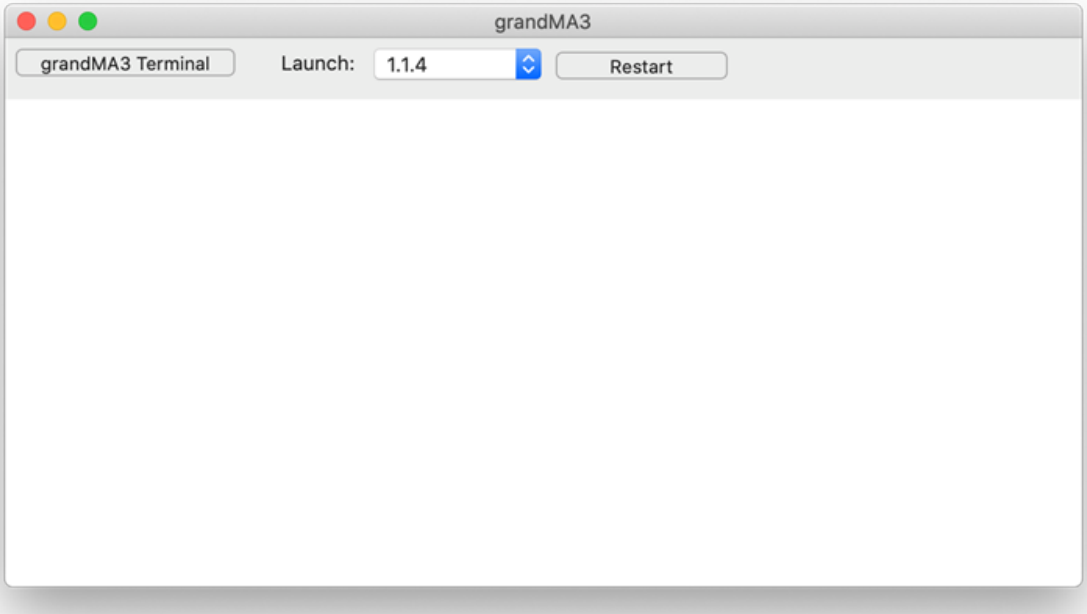
11. Click Close after the installation is complete:



	Hint:
	If you receive an "Installation failed" error message during the installation, move the ma folder outside the Downloads folder, for example, to the Desktop folder, and start the installation process again.

Start the Application

1. Go to Applications in the finder and double-click grandMA3.
2. The grandMA3 launcher starts.
3. The launcher will automatically start the last installed version.



4. To choose another version, select the desired version via the launch drop-down.
5. Click Restart in order to start the selected version.
6. Confirm the shutdown in the running application.

1.6.5. Optimize macOS®

The grandMA3 onPC software is available for macOS™® and **Windows™®**.

Please refer to the **system requirements topic** to learn more about the computer specifications.

For more information about the grandMA3 onPC software and the terminal app, read the **Installation of grandMA3 onPC topic**.

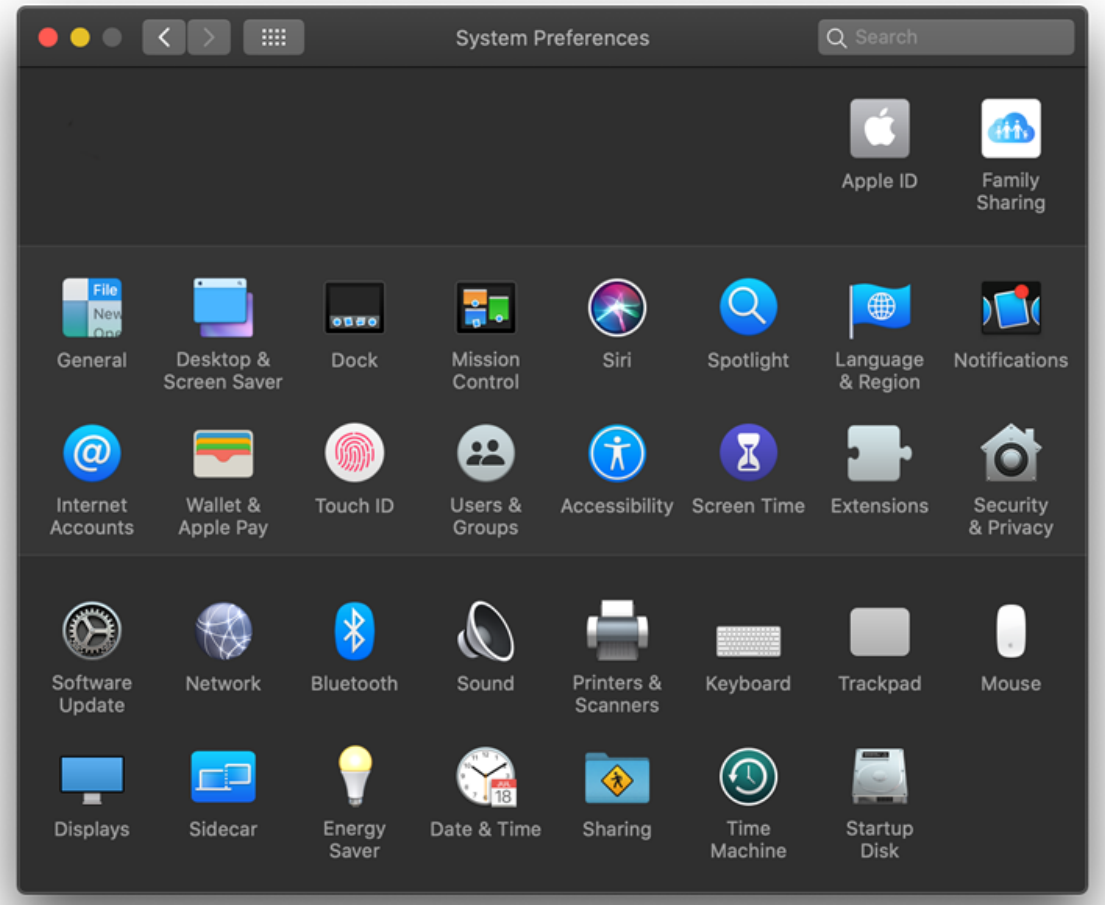
To run the grandMA3 onPC software even more efficiently, we recommend adjusting the following settings on your computer.

Follow the following steps for the best results.

	**Hint:**
	The same setting options may be available for other operating system versions, too.

Apple® macOS® Catalina 10.15

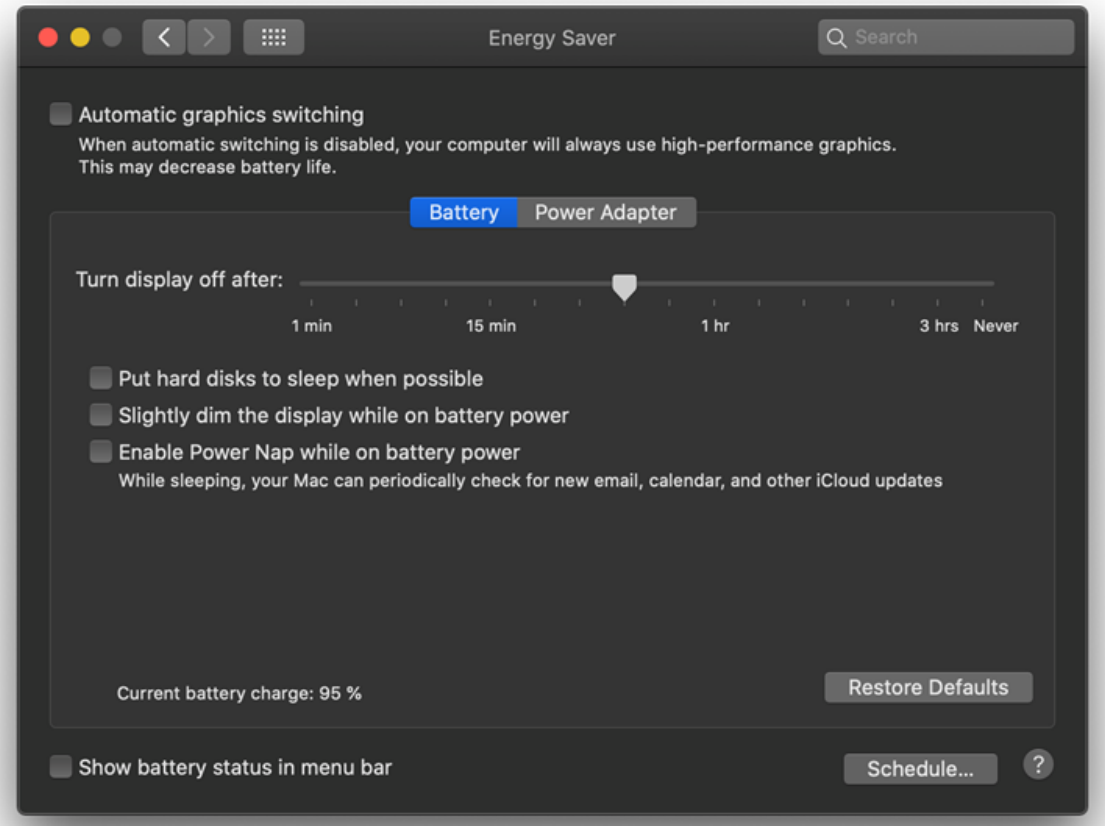
- Click Energy Saver in System Preferences.



System Preferences

Settings for Battery

- Automatic graphics switching: Off
- Turn display off after: Any time.
- Put hard disks to sleep when possible: Off
- Slightly dim the display while on battery power: Off
- Enable power nap while on battery power: Off

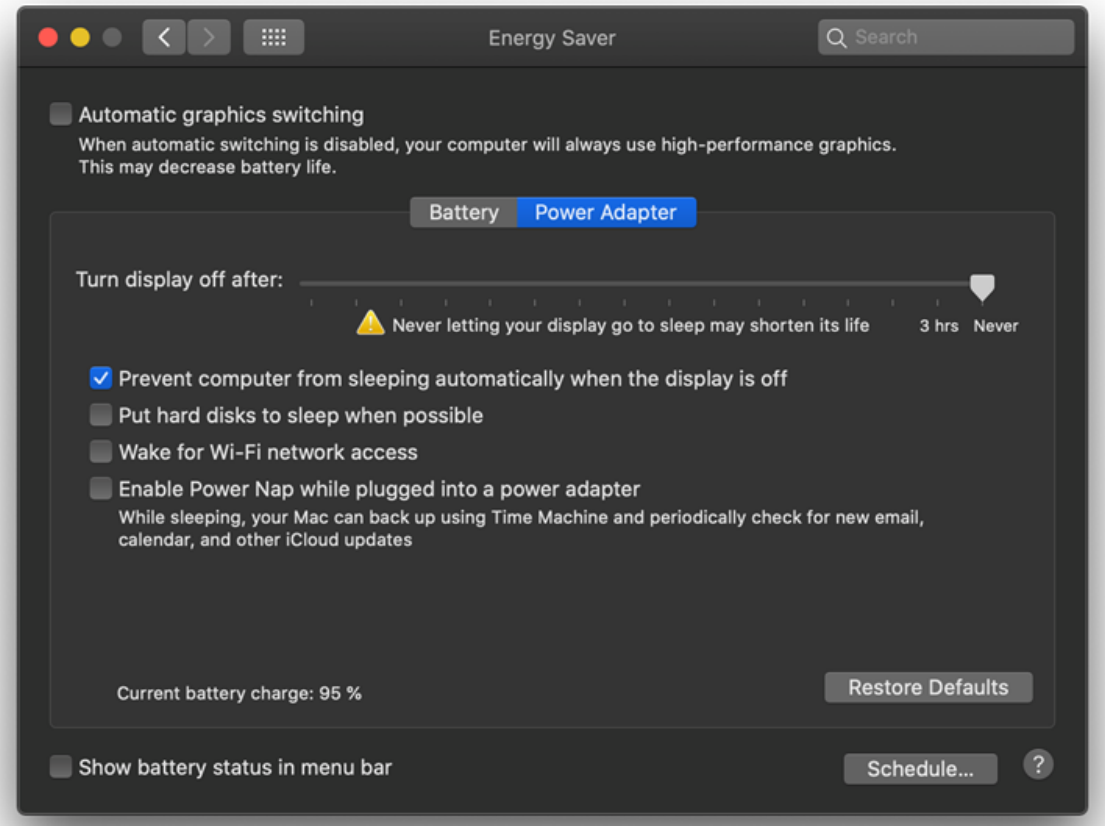


Battery

	Hint:
	Working with grandMA3 onPC under MacOS® using only the battery would also require high performance settings. It is recommended to always connect the computer to the power supply.

Settings for Power Adapter

- Automatic graphics switching: Off
- Turn display off after: Never.
- Prevent computer from sleeping automatically when the display is off: On
- Put hard disks to sleep when possible: Off
- Wake for Wi-Fi network access: Off
- Enable power nap while plugged into a power adapter: Off

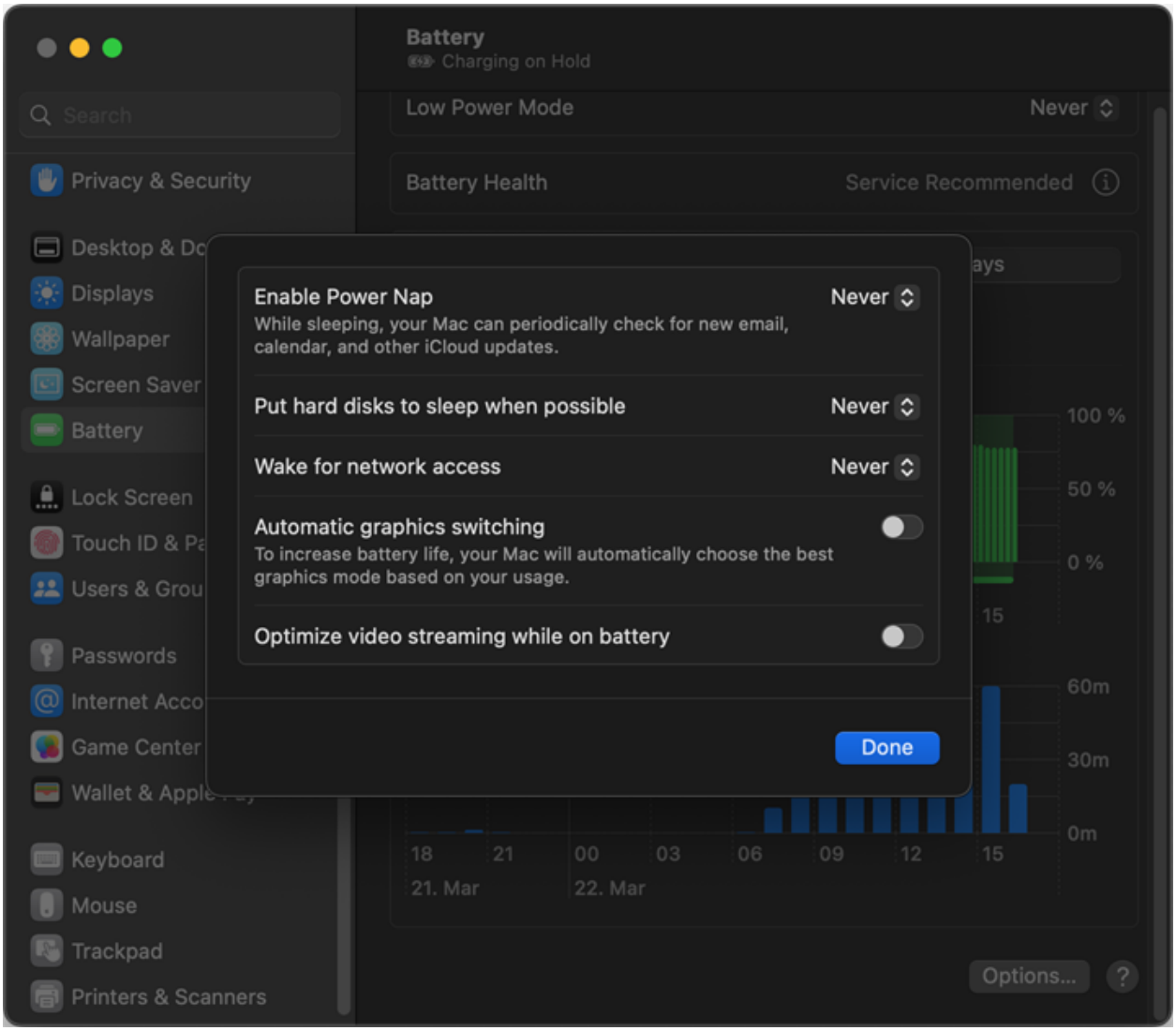


Power Adapter

Apple® macOS® Ventura 13

Settings for Battery

1. Click Battery in System Preferences and then click Options....

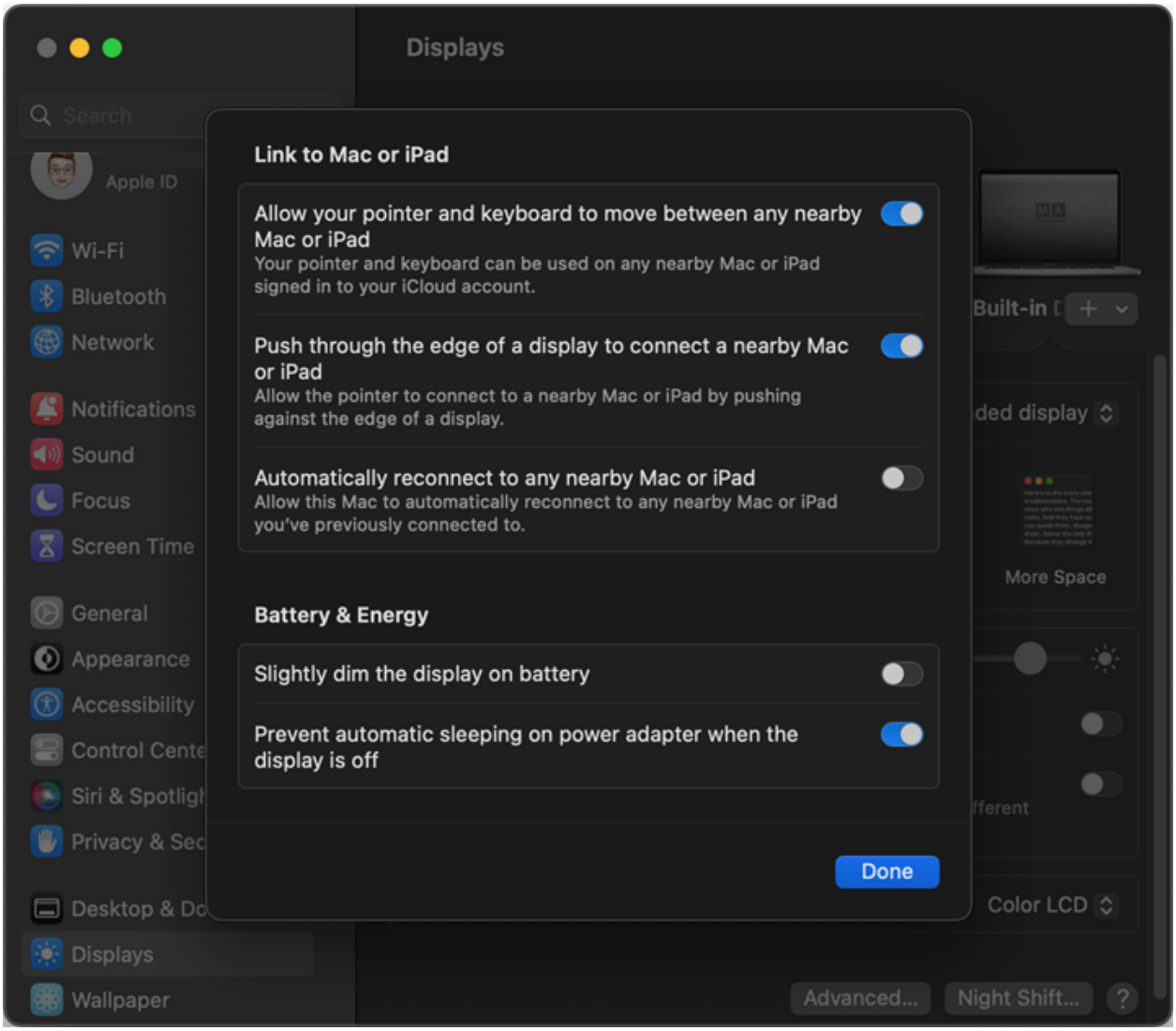


Battery settings

- Enable Power Nap: Never
- Put hard disks to sleep when possible: Never
- Wake for network access: Never
- Automatic graphics switching: Off
- Optimize video streaming while on battery: Off

Settings for Display

1. Click Displays in System Preferences and then click Advanced....

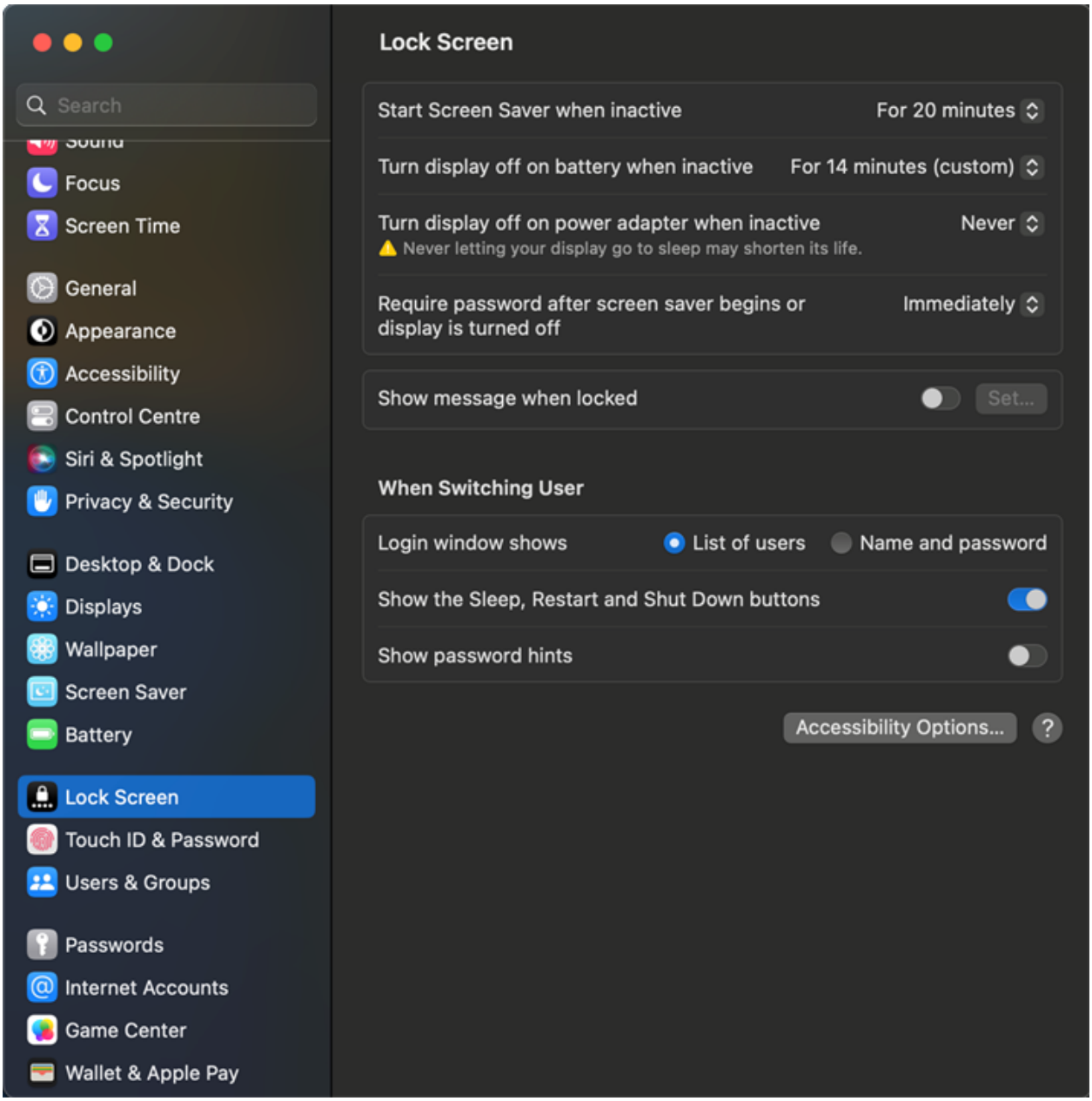


Display settings

- Slightly dim the display on battery: Off
- Prevent automatic sleeping on power adapter when the display is off: On

Lock Screen settings

1. Click Lock Screen in System Preferences.



Lock Screen settings

- Start Screen Saver when inactive: Any time
- Turn display off on battery when inactive: Any time
- Turn display off on power adapter when inactive: Never

1.6.6. onPC Terminal App

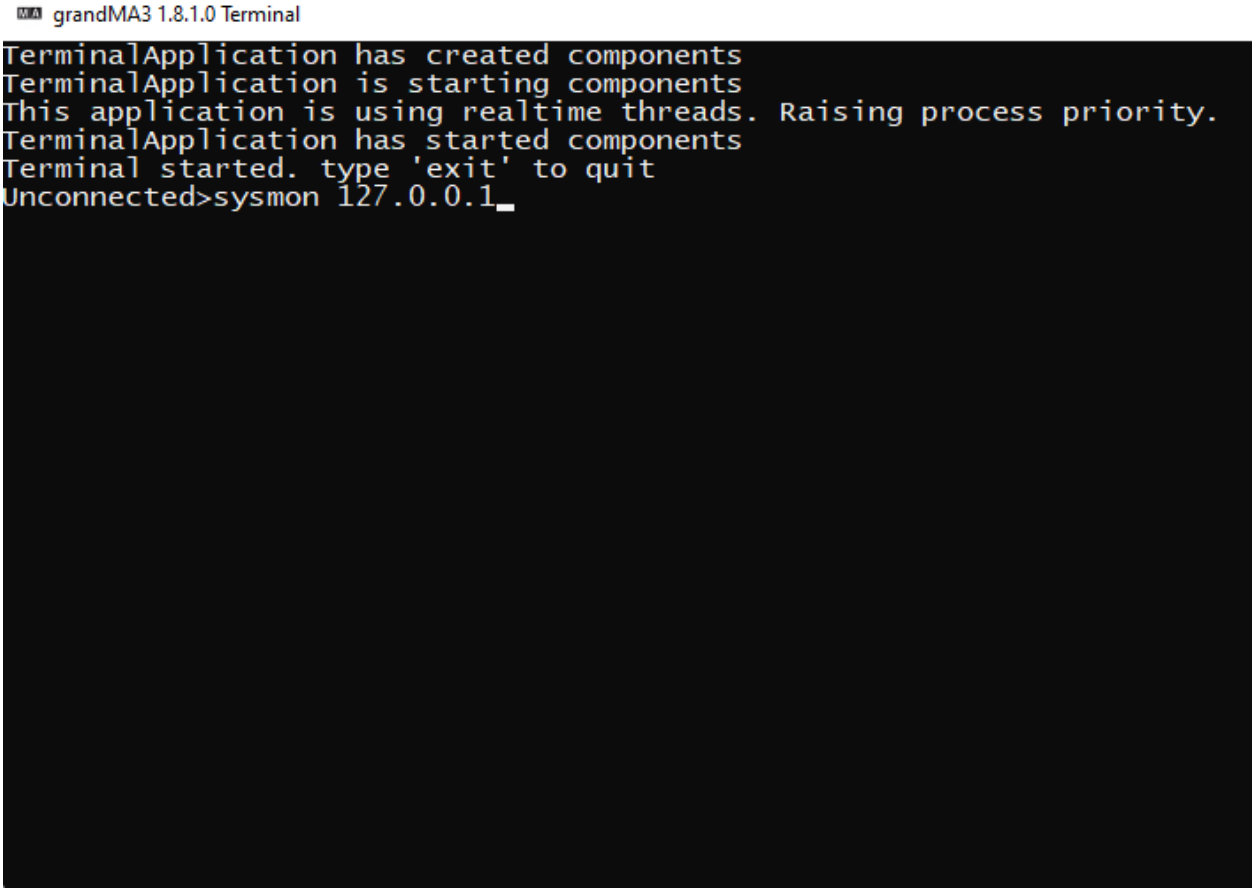
The terminal app can be used to connect to the same station or any other grandMA3 station inside your network with the command line or the system monitor of the main app using the Terminal.

- To monitor the software of a connected network station, start the terminal app.
- To connect to the system monitor of a station in the network, enter:

sysmon [IP Address]

Example:

1. Open the grandMA3 Terminal app on the computer.
2. Enter sysmon 127.0.0.1 in the command section in the grandMA3 Terminal app. Press Enter.

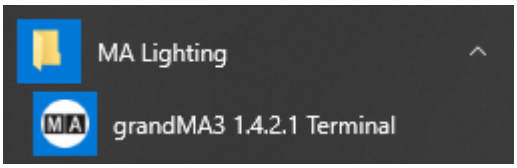


grandMA3 Terminal app with sysmon command

3. The network is now connected to the personal system monitor.

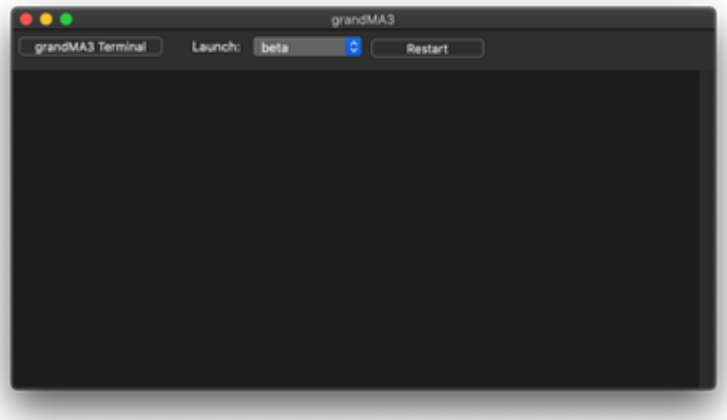
grandMA3 Terminal App on Windows

In Windows®, the grandMA3 Terminal app can be found in the start menu in the folder MA Lighting:



grandMA3 Terminal App on macOS

In macOS®, click grandMA3 Terminal to start the grandMA3 Terminal:



1.6.7. onPC Local Settings

The onPC Local Settings allow various settings for grandMA3 onPC. On the other hand, the **Local Settings** menu is available on the console.

To learn more about MIDI, read the topics **Connect MIDI** in **First Steps**, and **MIDI Remotes** in **Remote In and Out**.

To learn more about Sound, see **Sound**. For more information about viz-key, see **Update grandMA3 viz-key**.

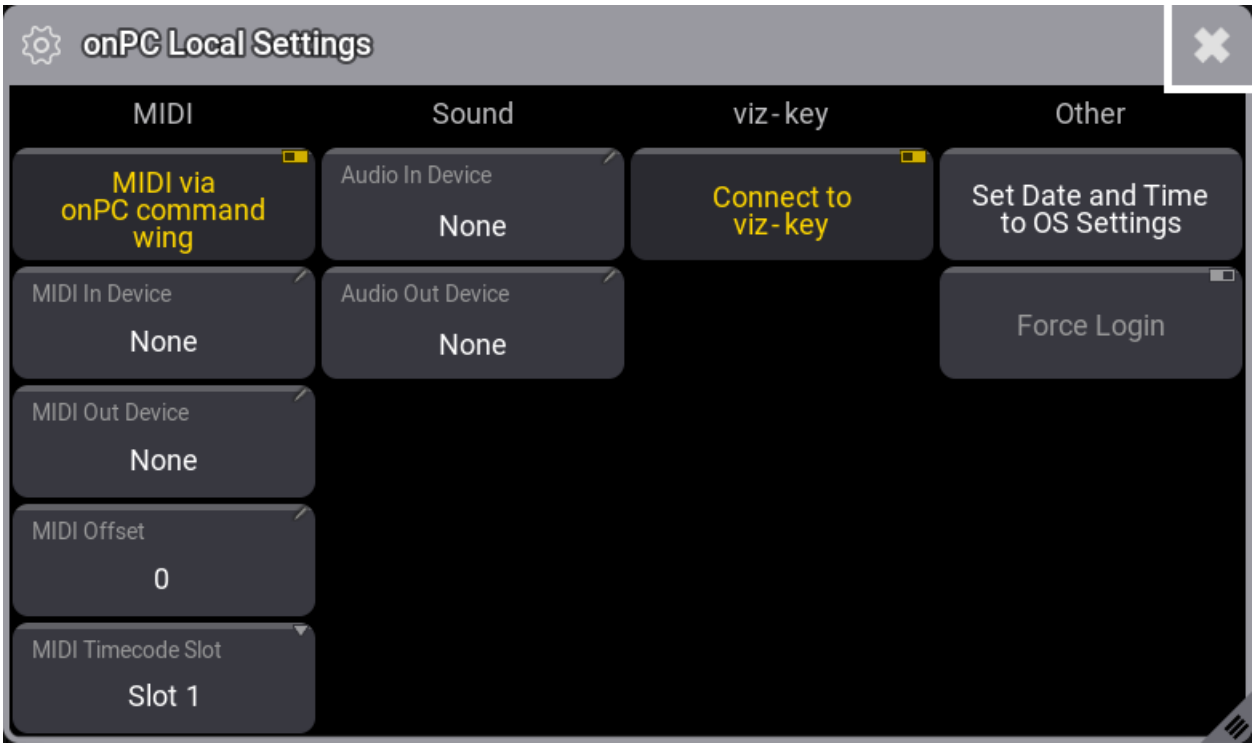
For more information about how to create a password, see **Create User** in **Single User and Multi User Systems**.

Open the onPC Local Settings

To open the settings menu on grandMA3 onPC software:

1. Click on the control bar. The Menu opens.
2. Click Settings. The Setting pop-up opens.
3. Click onPC Local Settings.

The onPC Local Settings menu opens.



onPC Settings menu

MIDI

- To enable or disable MIDI from grandMA3 onPC command wing, click MIDI via onPC command wing.
  If disabled, the device configured in MIDI In Device and MIDI Out Device is used.
  If enabled, the command wing MIDI and the device configured in MIDI In Device and MIDI Out Device are used.
- To select a MIDI In device, click MIDI In Device.
- To select a MIDI Out device, click MIDI Out Device.
- To adjust the MIDI Offset, click MIDI Offset.
- To adjust the MIDI Timecode Slot, click MIDI Timecode Slot.

Sound

- To select the Audio In device, click Audio In Device.
- To select the Audio Out Device, click Audio Out Device.

viz-key

- To attach the viz-key dongle to the onPC, enable Connect to viz-key.
- To attach the viz-key dongle to a visualizer that is running on the same computer as the onPC, disable Connect to viz-key.

Other

- To reset the date and time of the grandMA3 onPC application to the date and time of the operating system, click Set Date and Time to OS Settings.
- To force the user to enter a password after starting up the grandMA 3 onPC application, enable Force Login.

1.7. Show File Handling

The show file contains all the information that is related to the show including:

- Patch
- Fixture profiles
- Cues
- Timings
- 3D information
- Users and user profiles

	**Important:**
	Save the old show before loading or renaming a new one, to keep changes.

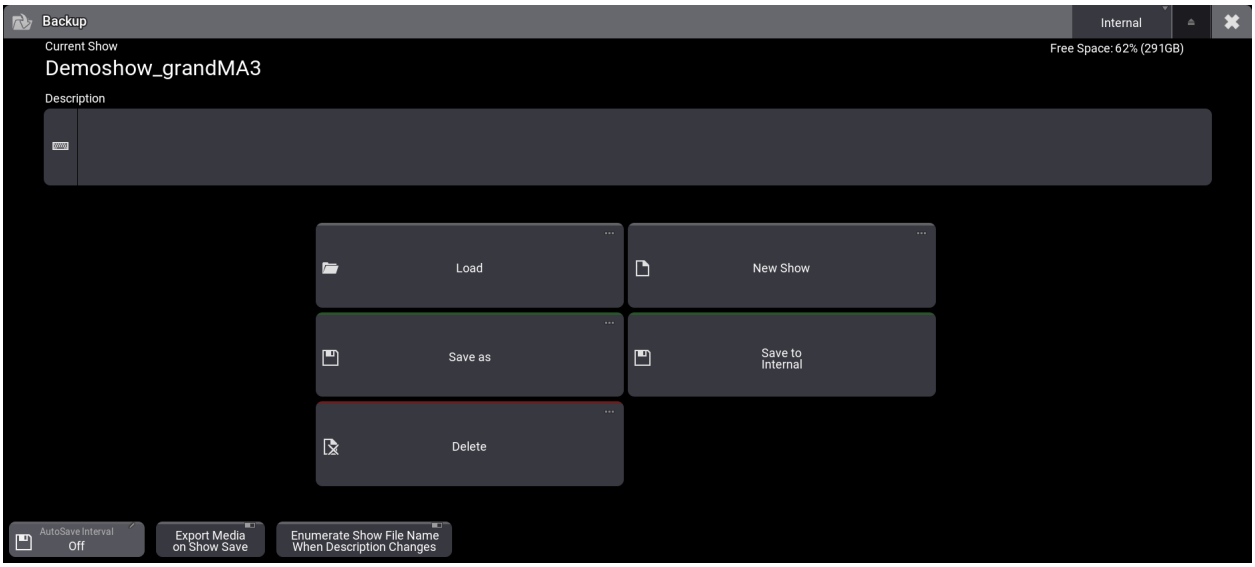
The software version of a show file can only be moved forward. A show saved on a USB stick that was programmed in a previous version can be loaded in a newer version. If the show is saved again on the USB stick in this new version, it cannot be taken back to the old software.

	**Important:**
	To keep the old software version, save the show with a new name.
	**Hint:**
	To learn more about the folder structure of Shows, Demo Shows, Backup Shows, and Template Shows, read the topics **File Management** and **Folder Structure**.

All the management of the show files is handled in the Backup menu.

1. To access the Backup menu, press Menu.
2. Tap Backup.

The Backup menu is open:



Backup menu.

Or use the **Menu command**, to call the Backup menu.

	Important:
	A loaded show file is limited to 10 GB of memory.
	Important:
	The size of all media pools in total, is limited to a maximum of 300 MB.
	Important:
	The show file name is limited to a maximum of 31 characters.
	Important:
	The show file name must not contain the following characters: \ " $ * ? ^ | / : < > `

To learn more about he current occupation of memory, read the **System Information** topic.

Select the Drive

The show files are stored on the internal drive or an USB stick. The internal drive is Drive 1 and the first USB stick is Drive 2. If more than one USB stick is connected the order of the connection selects the drive number. That means that the latest connected drive has the highest number.

	Hint:
	The order of connecting the USB drives to the console / onPC station determines the order of the drive numbers.

In the upper right corner of the Backup menu, it is possible to select the drive you are working on.
To select the drive:

1. Tap Internal to toggle between the different drives connected, saved shows or saved shows in older software versions.

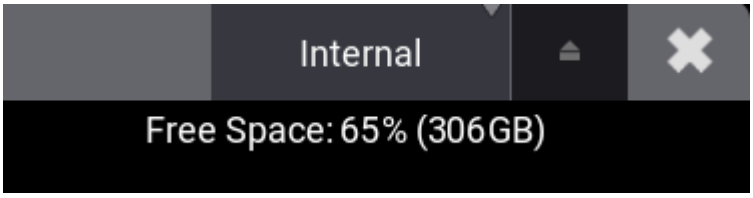
2. Tap and swipe right on the drive and a pop-up opens with the different opportunities.

The drive can also be selected via the command line, for example, the first USB drive:

User name[Fixture]> Select Drive 2

Free Disk Space

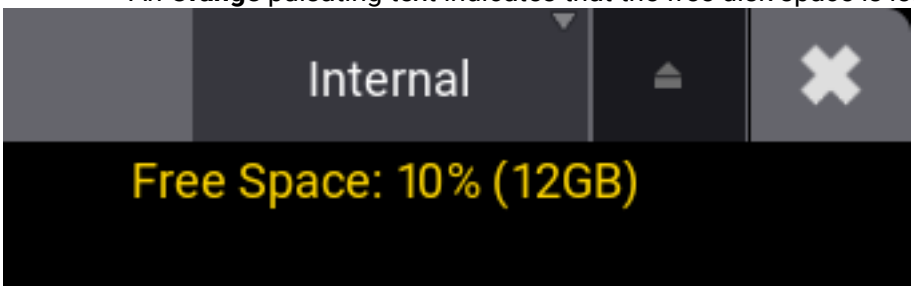
The Backup menu displays the free disk space of the selected drive in the upper right corner of the menu:



Showing free disk space in percent and gigabyte

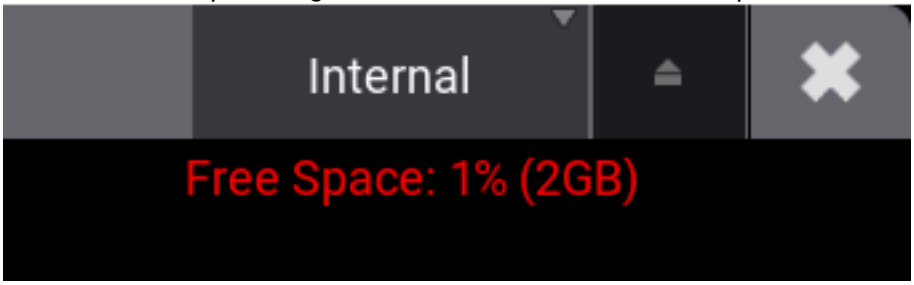
When free disk space is low:

- An **Orange** pulsating text indicates that the free disk space is less than 15 GB.



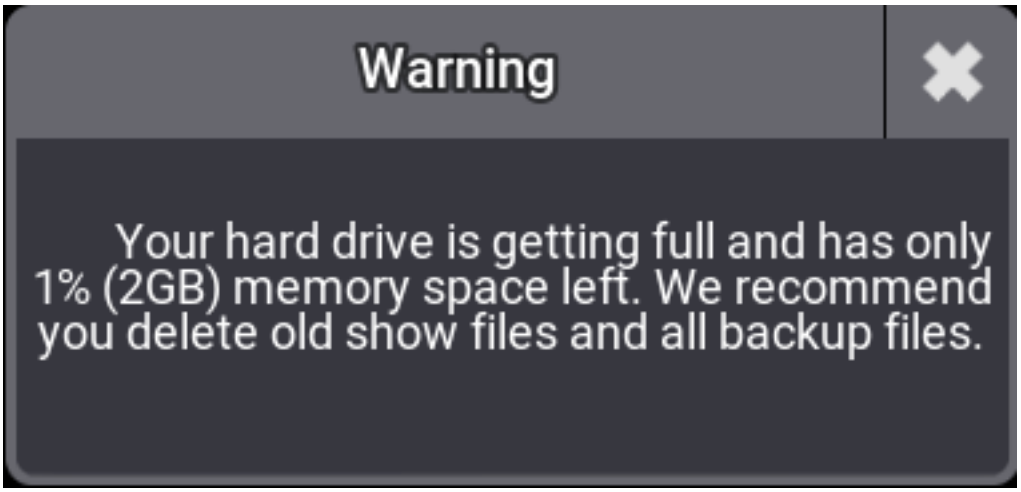
Orange indicator

- A **Red** pulsating text indicates that the free disk space is less than 5 GB.



Red indicator

- The following warning pop-up appears, when booting the software and the free disk space is less than 15 GB.



Warning pop-up.

Delete a Show File

Open the Backup Menu and tap Delete. Tap the show file to delete. Tap Delete in the bottom right corner and the show file is deleted.

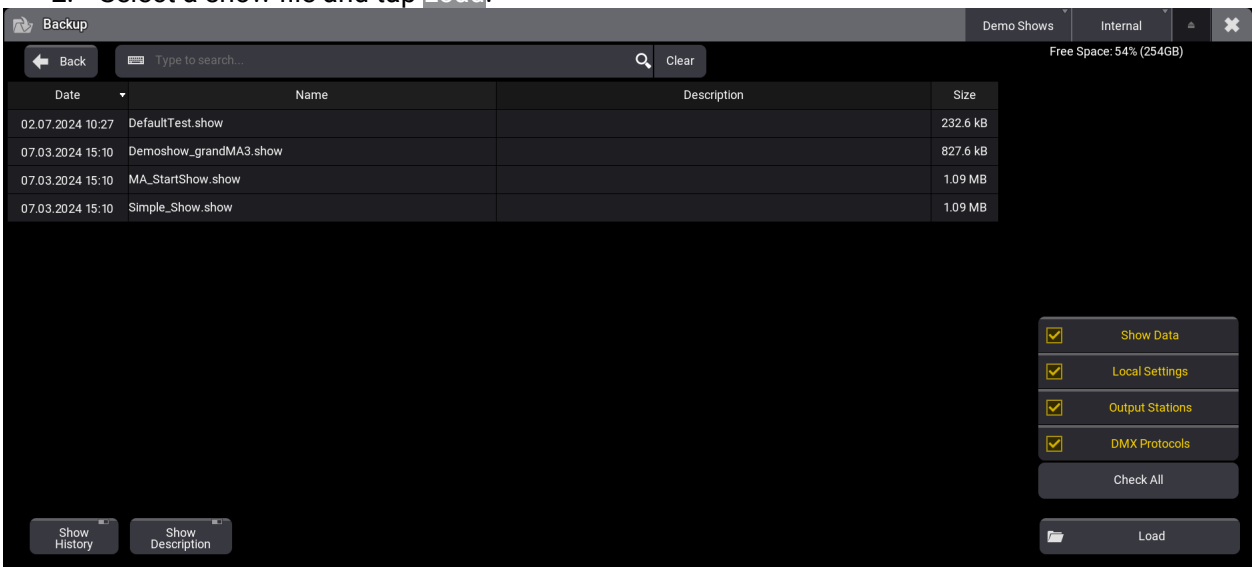
Subtopics

- **Load a Show File**
- **Save a Show File**
- **New Show File**
- **Backup, Demo and Template Show Files**
- **Organize Show Files**

1.7.1. Load a Show File

To load a show file:

1. Open the Backup menu and tap Load.
2. Select a show file and tap Load.



Load overlay in the Backup menu with the four checkbox buttons on the right side.

To load a show via the command line, use the **LoadShow keyword**.

Load Show Segments

The following checkbox buttons define which segments of a show file are loaded:

- Show Data: Loads the show-relevant data from the show file. This includes the patch, pool objects, user, and user profiles. Show Data is enabled by default.
- Local Settings: Loads the local settings. For example, this includes web remote settings and onPC settings.
- Output Stations: Includes the setup of all DMX ports, the SMPTE settings, and the MIDI settings.
- DMX Protocols: Loads the configuration of all available DMX protocols (Art-Net and sACN).
- Check All is a fast selection for all checkboxes.
- Load: The show file will be loaded.

	Hint:
	If no checkbox button is enabled, Load is disabled.
	Hint:
	The data of the currently loaded show file will be kept for the segments, where the checkbox buttons are not enabled.

	Important:
	The show file needs to be saved first with grandMA3 v1.9 or higher. Show files that were saved with grandMA3 v1.8.8.2 or prior do not contain Local Settings, Output Configuration, and DMX Protocols saved inside.

1.7.2. Save a Show File

Quick Save

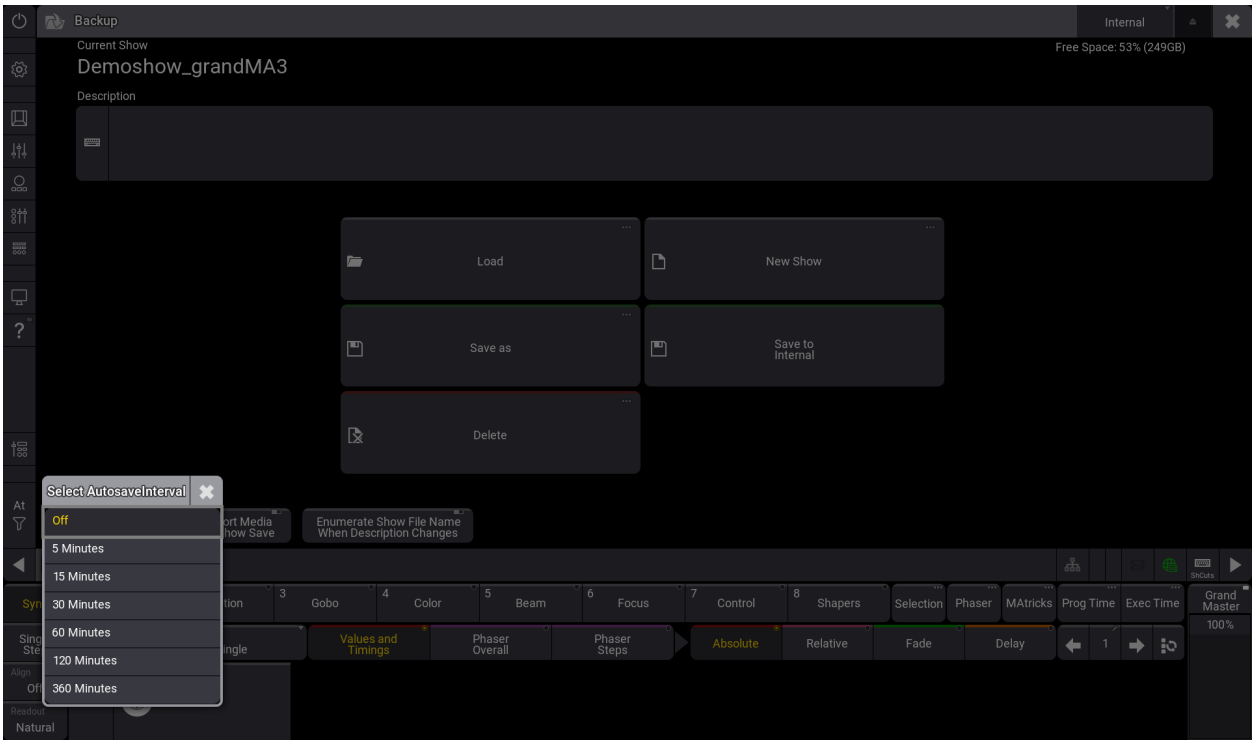
To save the show file quickly:

- Press Menu and tap Quick Save.
- Pressing quickly 2 x Menu also makes a Quick Save.

Auto Save

To save the show automatically in regular intervals:

1. Activate the Auto Save function. Tap Backup and AutoSave Interval.
2. Select the desired Auto Save Interval.



Auto Save Interval

Save Show As

To save a show with another name, open the Backup Menu and tap Save As. Write the new name of the show file in the Name field. Tap Save as ... and confirm pop-up window with OK. The show file is saved by its new name.

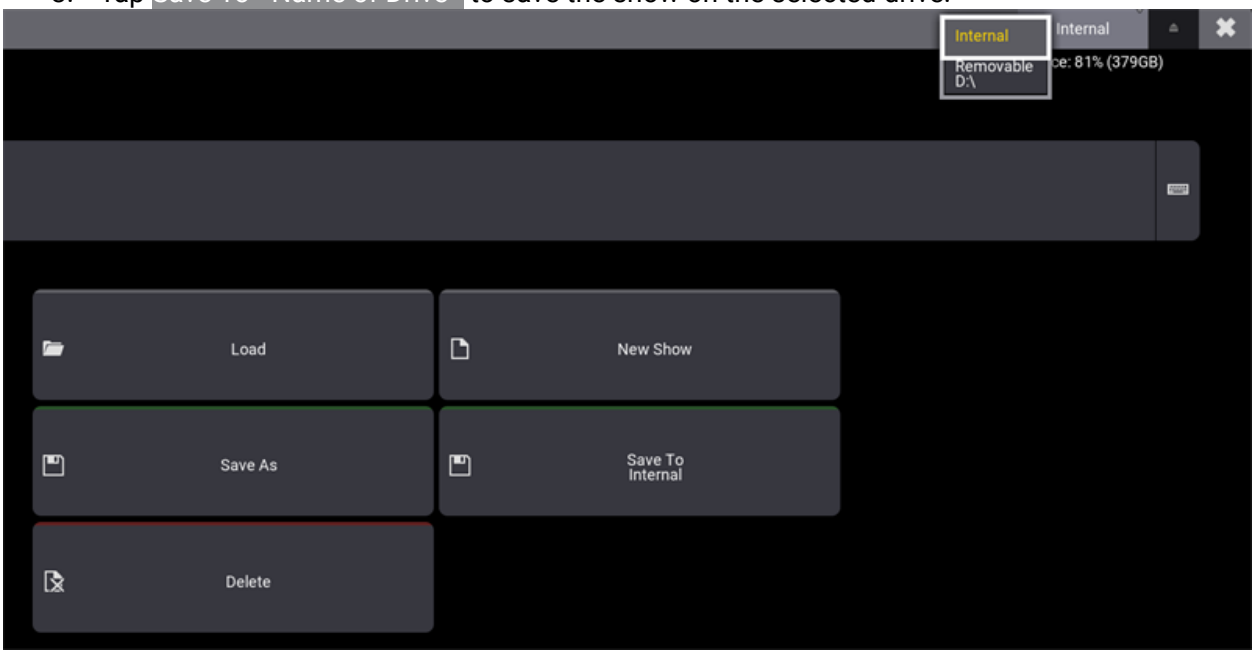
To save a show by the command line, use the **SaveShow keyword**.

Save To Internal

	Hint:
	Saving a show file to an external drive automatically saves a show file to the internal drive.

To select and change the internal/external drive to save a show file on:

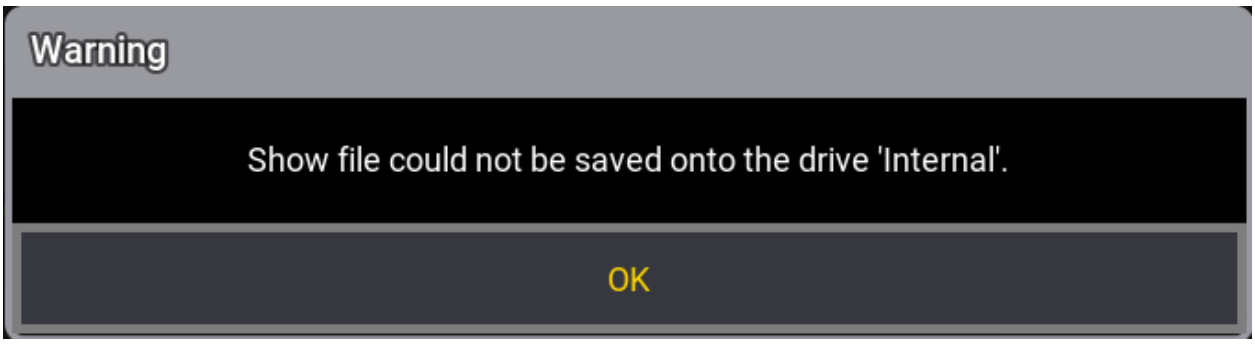
1. Tap Internal in the top right corner in the Backup Menu.
2. The name of Save To Internal button will automatically adjust to the name of the drive.
3. Tap Save To <Name of Drive> to save the show on the selected drive.



Save a show on an internal selected drive

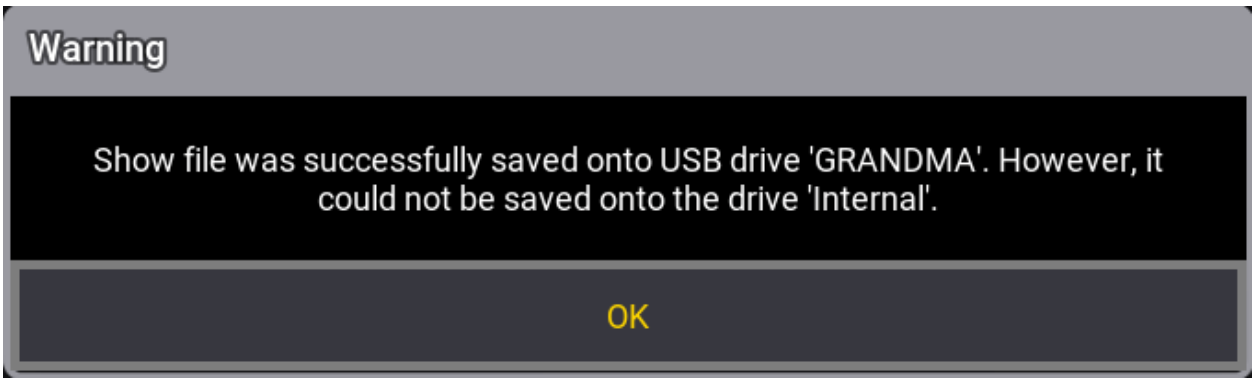
Full Disk Warning

If the show file cannot be saved due to a full disk, a warning pop-up will appear.



Warning pop-up for an 'Internal' save failure

To indicate that the show file is partially saved, a pop-up appears.

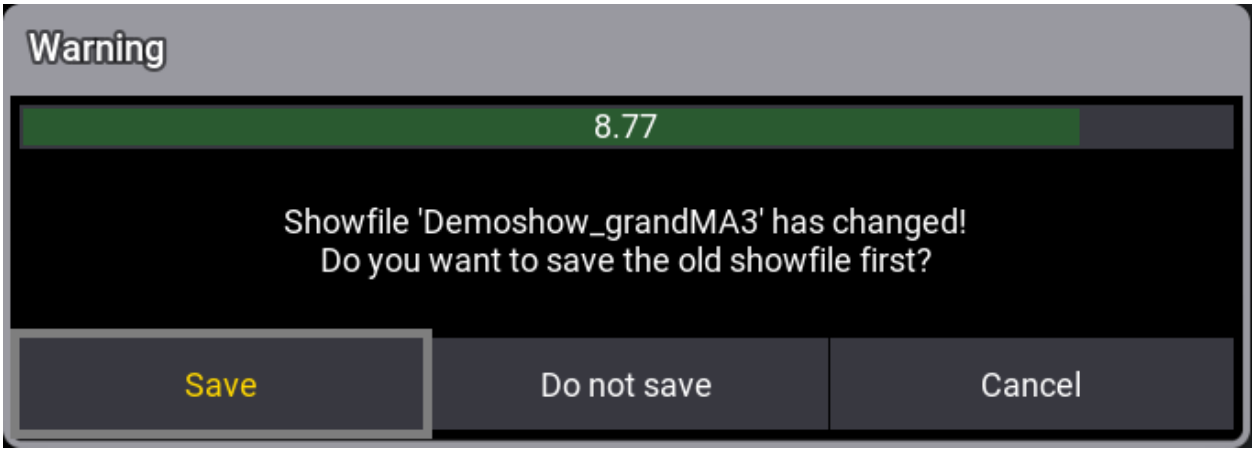


Warning pop-up for a partially saved show file

For more information about Free Disk Space, see **Show File Handling**.

Shutting Down the System

If an attempt is made to shut down before saving the show file, a Warning pop-up appears:



Save Show File pop-up

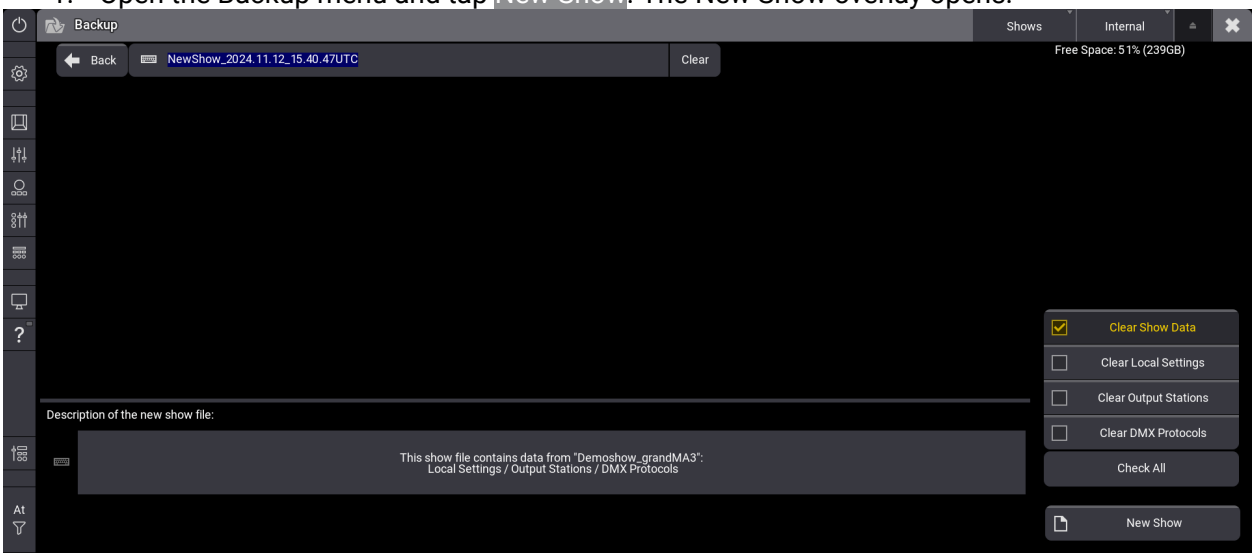
- **Save:** Saves the show file and then shuts down.
- **Do not save:** Shuts down without saving the show file.
- **Cancel:** Cancels the process.

For more information about shut down, see **Shut Down the System**.

1.7.3. New Show File

To create a new show file:

1. Open the Backup menu and tap New Show. The New Show overlay opens:



New Show overlay in the Backup menu with automatically generated show file name

2. Type to rename the file or keep the automatically generated show file name.
3. Tap New Show in the bottom right corner.
4. When the currently loaded show file has changes, a Warning pop-up appears. To continue, choose an option.

   -or-

   When the currently loaded show file does not have any changes, the Create New Show pop-up appears. Tap New Show.

	Hint:
	It is possible to use local letters in the filename, like æ, õ, and ä.
	Hint:
	Use the **NewShow keyword** to create a new show from the command line.

Clear Show Segments

1. To define which segments of the current show file should be cleared for the new show, use the four checkbox buttons on the right side:
   - Clear Show Data: Clears the show-relevant data from the show file. Includes the patch, pool objects, user, and user profiles. This checkbox is marked as default.
   - Clear Local Settings: Clears the local settings. For example, this includes web remote settings and onPC settings.
   - Clear Output Stations: Clears the setup of all DMX ports, the SMPTE settings, and the MIDI settings.

- Clear DMX Protocols: Clears the configuration of all available DMX protocols (Art-Net and sACN).

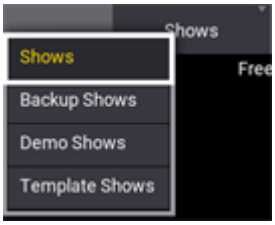
2. Selected data will be cleared after tapping New Show.
   - Check All is a fast selection for all checkboxes.
   - The data of the currently loaded show file will be kept for the segments, where the checkbox buttons are not enabled.
   - An automatically generated description text of the kept show segments can be found in the Description area.

	**Hint:**
	Make sure to at least select one checkbox button to enable New Show.

1.7.4. Backup, Demo and Template Show Files

To load a backup file or a demo show file:

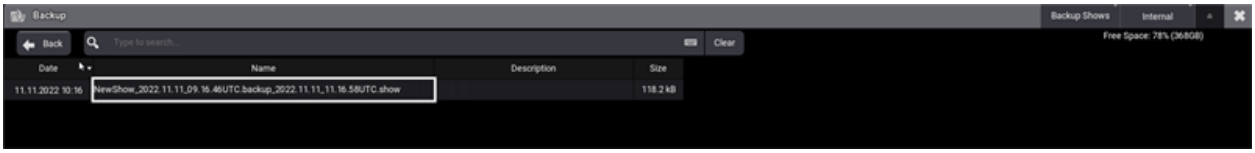
1. Open the Backup Menu and tap Load.
2. Tap Shows to toggle between Shows, Backup Shows, Demo Shows and Template Shows.



It is possible to load a previously saved backup or a demo show. It is not possible to save or create a new show file in the Backup Shows or Demo Shows folder.

It is possible to store 10 backup files from the same show file. A backup is automatically generated when saving a show file.
A backup file contains:

- Showfile Name
- Backup
- Date and Time in UTC format



Backup Files

Template Shows

The show files in the Template Shows folder can be used as a template.

	Important:
	The Template Shows folder is read-only, it can note be overwritten.

To store or save a template show, show files can be placed into a library folder. For example, use MALightingTechnology/gma3_library/templateshows on windows or a SFTP connection to a console.

For more information, see **Folder Structure**.

1.7.5. Organize Show Files

Show File History

To show the show file history:

1. Tap Load in the backup menu. Enable Show History in the bottom left corner.
2. Select a show file. The history of the selected show file is displayed at the bottom of the backup menu.

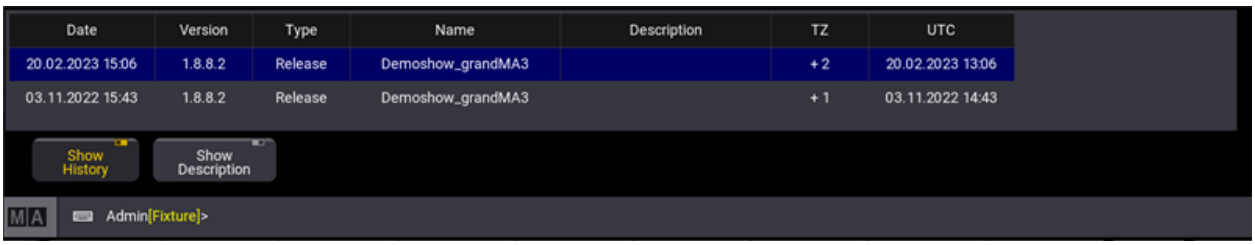
When loading the show file, the history gives information regarding:

- Date, time, and time zone (TZ)
- Used software version
- Software type
- Name

A new entry within the history will be entered when the show file is saved again and one of these criteria has changed:

- Version
- Type
- Name
- Time zone
- Description

The newest entry is on top of the history list.

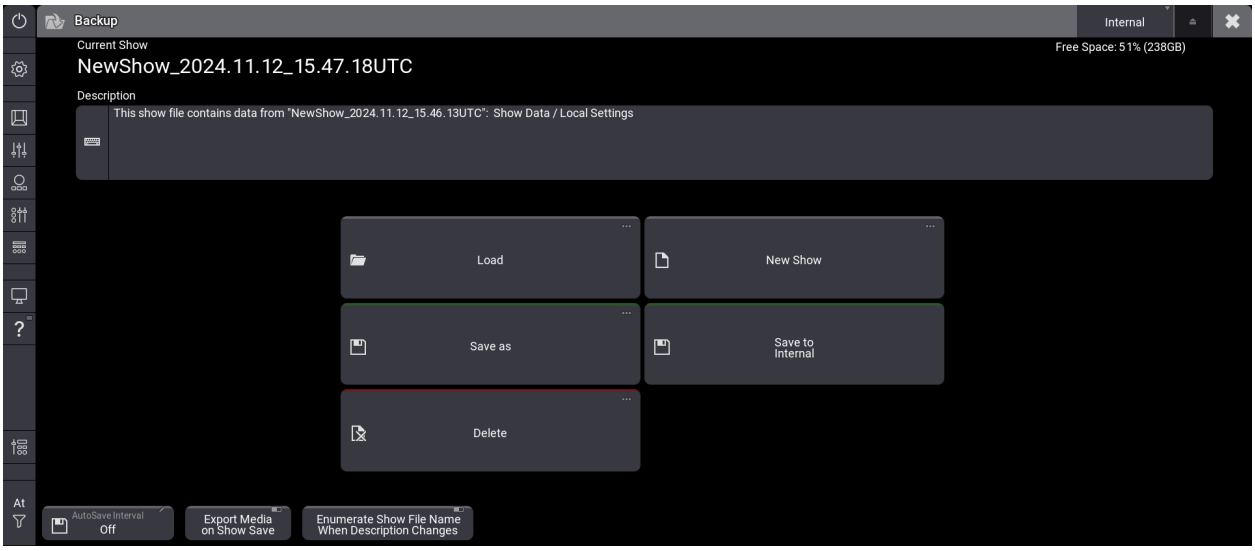


Show file name with description and time stamp

Add a Description

To add a description to the show file, use the description area at the top of the backup menu. To add another line, press Please.

Modifying or entering a description will only be applied to the show file when saving it afterward.



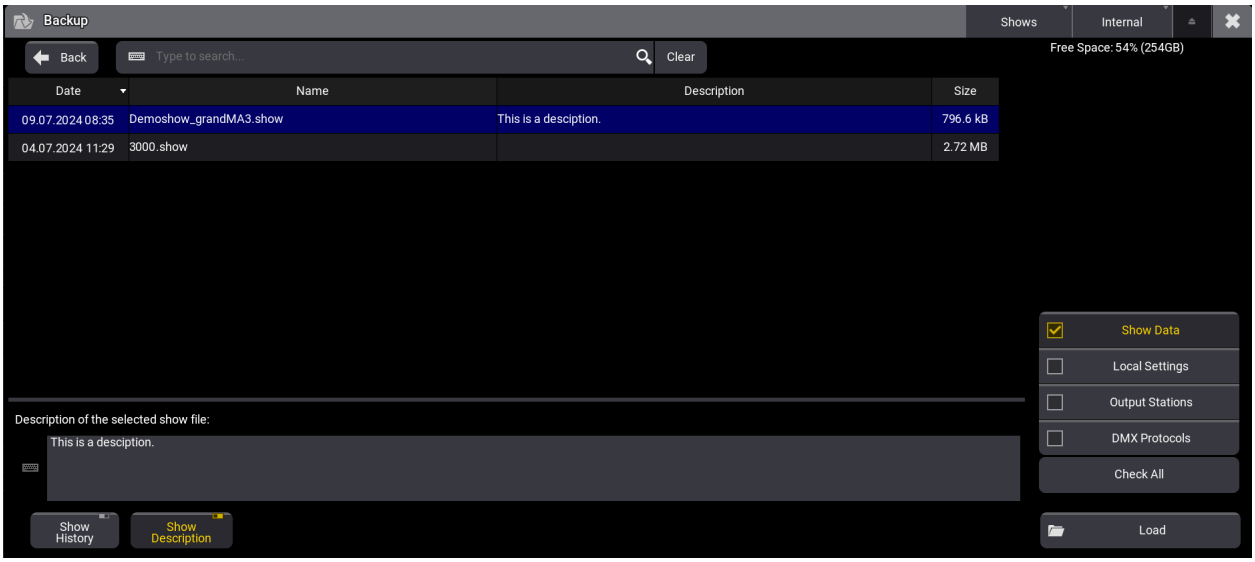
Show file name with description line

	Hint:
	When you create a new show file with kept show segments, the defined show segments are displayed in the Description area. For more information, see **New Show File**.

See the Description

To see the description of a show file:

1. Press Menu and then tap Backup. The Backup menu opens.
2. Tap Load. The first line of the description is shown in the Description column.
3. Enable Show Description and select a show file. The whole description of the selected show file is shown.



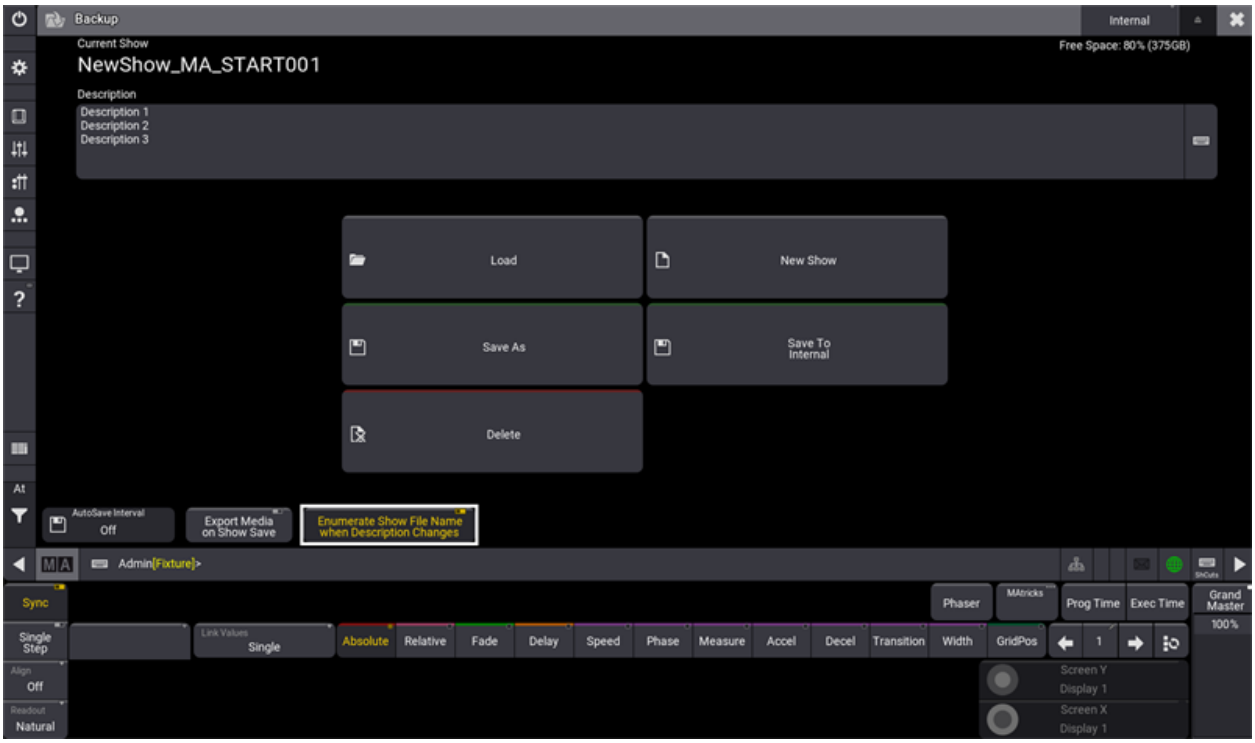
Backup menu - Show file description

Add Automatic Numbering

It is possible to add 3-digit numbers to the show file name.

To separate the show file name from the number, the user can add a hyphen (-) or an underscore (_) or any other character to separate the name from the numbers at the end of the show file name.

The setting Enumerate Show File Name when Description Changes in the Backup menu provides an automatic increase of a 3-digit number. The number will be added to the end of the show file name if the description is changed when saving the show.



Show File name with automatic numbering

If the original show file name is too long to add the number, the command line will return an error. In this case, the former show file name will be used again.
Enumerating a show file can also be called with the /Enumerate option.
For more information see **/Enumerate option keyword**.

Load a Show File During Startup

It is possible to specify a show file or a plugin that shall be loaded upon starting the onPC application.

To do so, add the SHOW or RUNPLUGIN parameter to the shortcut that launches the onPC application:

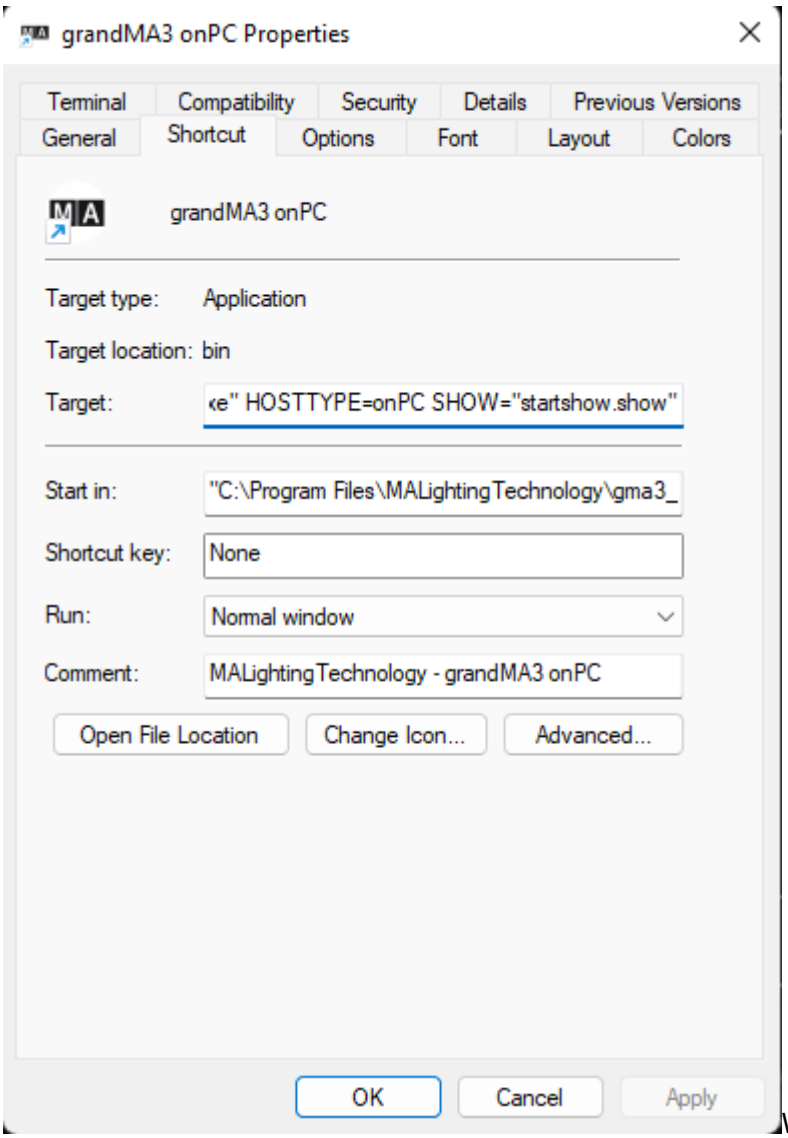
- "C:\Program Files\MALightingTechnology\gma3_x.x.x\bin\app_system.exe" HOSTTYPE=onPC SHOW="startshow.show"
- "C:\Program Files\MALightingTechnology\gma3_x.x.x\bin\app_system.exe" HOSTTYPE=onPC RUNPLUGIN=\"startplugin.xml\"-y
    - x represents the grandMA3 version number.

- y represents the number of the Lua component inside the plugin, that shall be started, e.g., 1.
- It is not recommended to specify a path for the plugin although the plugin may be put into a subfolder.

In addition, the optional parameters NOLOAD and CLEANSTART are also available when starting onPC.

- NOLOAD will not load the last show file but start with an empty show. Other device related configurations are kept.
- CLEANSTART will reset device related configuration back to default and come up with an empty show file.

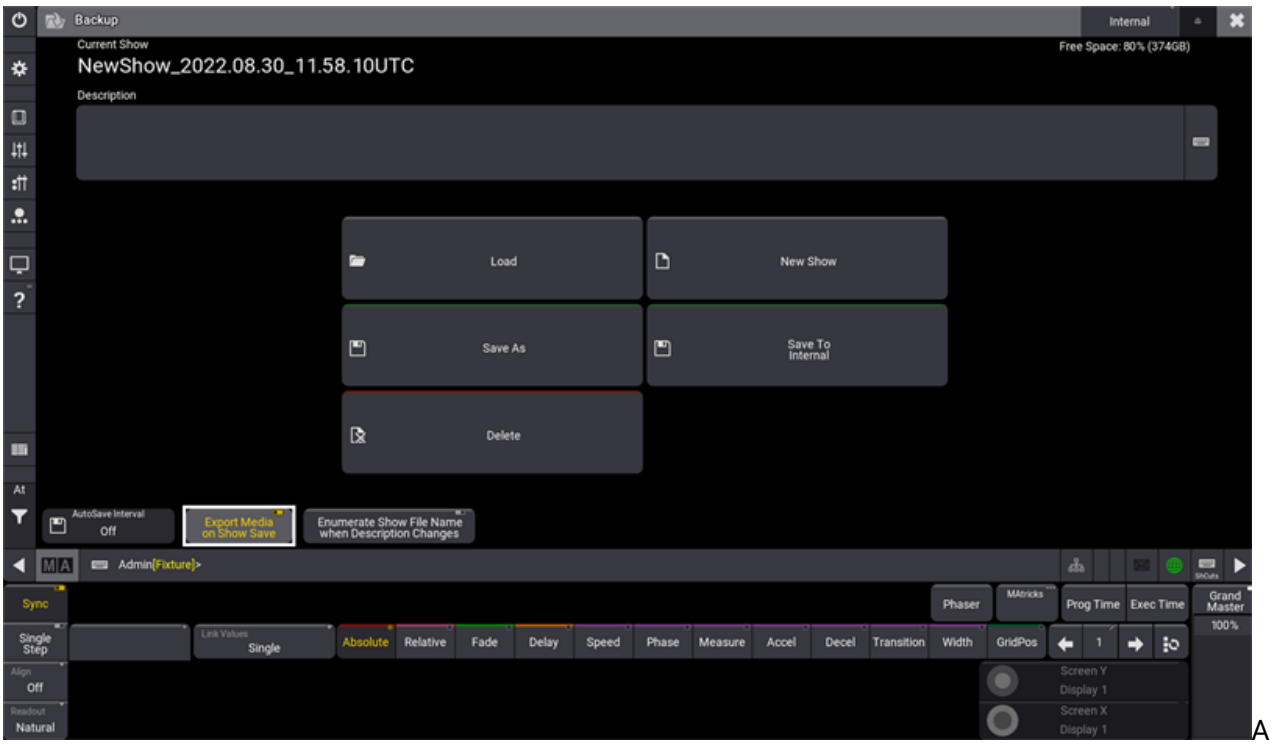
	Hint:
	Do not use spaces in a show file name when loading a show during startup.



Windows 11 properties are shown as an example for the load show process.

Export Media on Show Save

In order to save media files (e.g. images, videos) to the **gma3_library folder**, activate the Export Media on Show Save function. Tap Backup and Export Media on Show Save.



A show file with Export Media on Show Save

---

grandMA2 to grandMA3 Show File Converter

To learn more about how to convert show files from grandMA2 to grandMA3, see **grandMA2 to grandMA3 Show File Converter**.

1.8. Workspace

In general, **Workspace** deals with the visual elements on screens and input modes such as:

- Windows
- Views
- Coloring
- Executor bar
- Encoder bar
- Gestures
- Keyboard shortcuts

It represents the usage of the working area in the grandMA3 console.

Subtopics

- **User Interface**
- **Gestures**
- **Command Area**
- **Oops Overlay**
- **Master Controls**
- **Playback Controls**
- **Displays in grandMA3 onPC**
- **Encoder Bar**
- **Calculator**
- **Playback Bar**
- **command wing Bar**
- **Colors**

1.8.1. User Interface

This section covers the control elements that help you operate and control the grandMA3 in an efficient and effective way.

Subtopics

- **Configuration of Displays**
- **Desk Lock**
- **User-Defined Area**
- **Command Line**
- **Control Bar**
- **Tables in General**
- **View Bar and View Buttons**
- **Adjustable Columns**
- **Temporary Filtering**
- **Trackpad Window**

1.8.1.1. Configuration of Displays

The displays are populated with a combination of a user-defined area and a collection of additional display elements. The additional elements offer quick access to helpful features, and their availability differs from screen to screen.

The following video gives you an overview of the display configuration.

Pressing Menu then tapping Configure Display on any screen (excluding screen 8, screen 9, screen 10, screen 11, and screen 12) opens the Configure Display dialog for that screen. The number of the display to be configured is displayed in the title bar of the pop-up. This dialog can also be opened by tapping any user-defined area on the desired screen and tapping Configure in the upper-right corner of the Add Window dialog.

Tap areas to activate or deactivate them

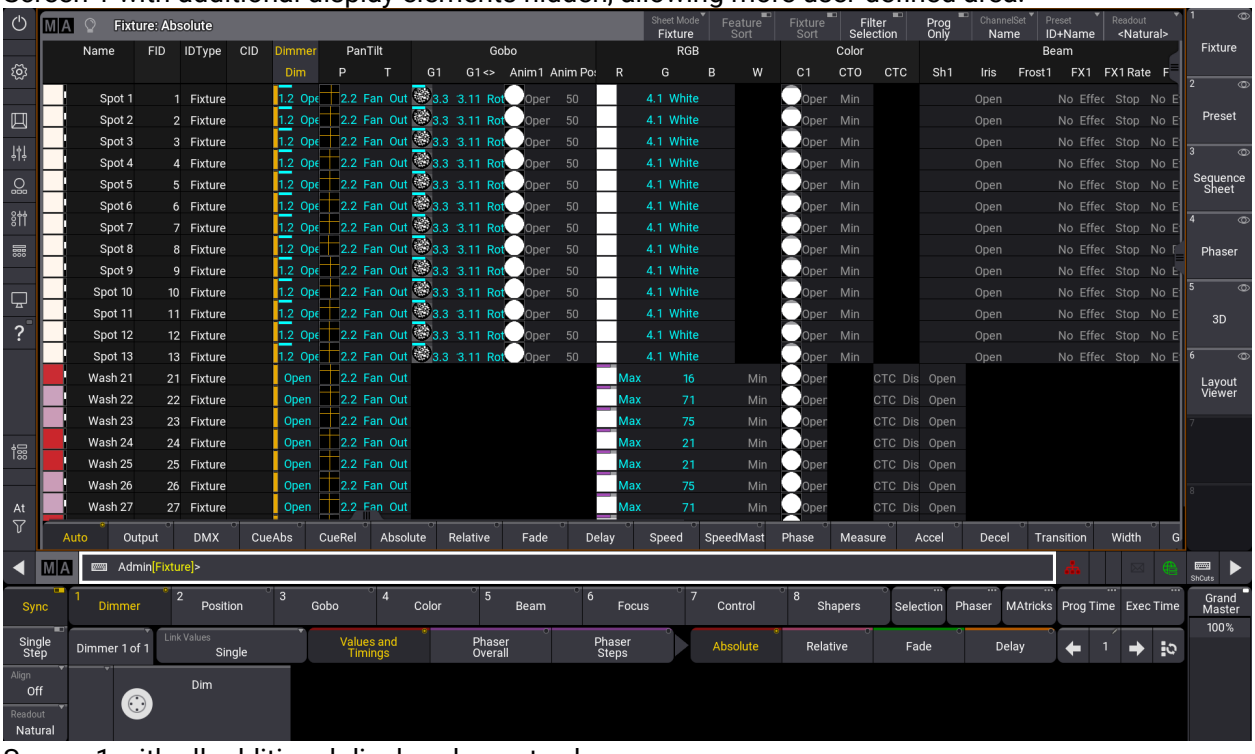
This pop-up is used to configure or change the appearance of the display. It has different areas:

- **Show Title Bar:** Activate this button to see the window frame in the grandMA3 onPC. This is not an option in the MA hardware.
- **Show Control Bar:** Activate this to see the control bar on the left side of the display.

- **Appearance:** Tap this to open the Select Appearance pop-up and select one of the existing appearances for this display. The appearance is added as a background for the screen area. Read more in the **Appearance section**.
- **Reset size** Tap this to reset the size of the user-defined area. The size can be changed using the **Width** and **Height**. Allowing each user to set the size of the user-defined area in the display. A user-defined area bigger than the display will appear within a brown frame. A user-defined area that is smaller than the screen will gray out the unused area.
- **Scale:** The scale can be used to change the scale of the interface. This option is only available on onPC and on external screens. It is very useful with high DPI screens or screens that have a lower resolution than Full HD.
- **Show Feedback:** Tap to show or hide a small feedback overlay at the bottom of the display. This feedback appears briefly when the show is saved.
- **Show View Bar:** This option hides or displays the column of view buttons. For more information on the View Bar, see **View Bar and View Buttons**.
- **Show Command Line:** Activate this to see the command line input bar at the bottom of the display.
- **Show Encoder Bar or Show Playback Bar:** This hides or shows the encoder bar or the playback bar at the bottom of the display.
- The in the title bar closes the pop-up.

These additional elements can be hidden to increase the amount of available user-defined area or shown in different combinations on most screens, whether internal screens or external monitors.

Screen 1 with additional display elements hidden, allowing more user-defined area.



Screen 1 with all additional display elements shown.

Command line with additional feedback overlay.

Screen 1, Screen 2, and Screen 3

Screen 1, screen 2, and screen 3 are the main screen workspaces on the console. The displays on these screens offer the most available user-defined area on any of the internal screens. They can be configured with a selection of additional display elements.

By default, screen 1 includes the command line and view bar. It can also be configured to display the control bar and encoder bar.

By default, screen 2 and screen 3 include the view bar. In addition, the grandMA3 compact XT includes the command line on screen 2. They can also be configured to display the command line, control bar, and playback bar.

Screen 4 and Screen 5

Screen 4 and screen 5 are on the optional external monitors. The displays on these screens offer a large amount of user-defined area. They can be configured with a selection of additional display elements. They can be configured to display the view bar, command line, control bar, and playback bar.

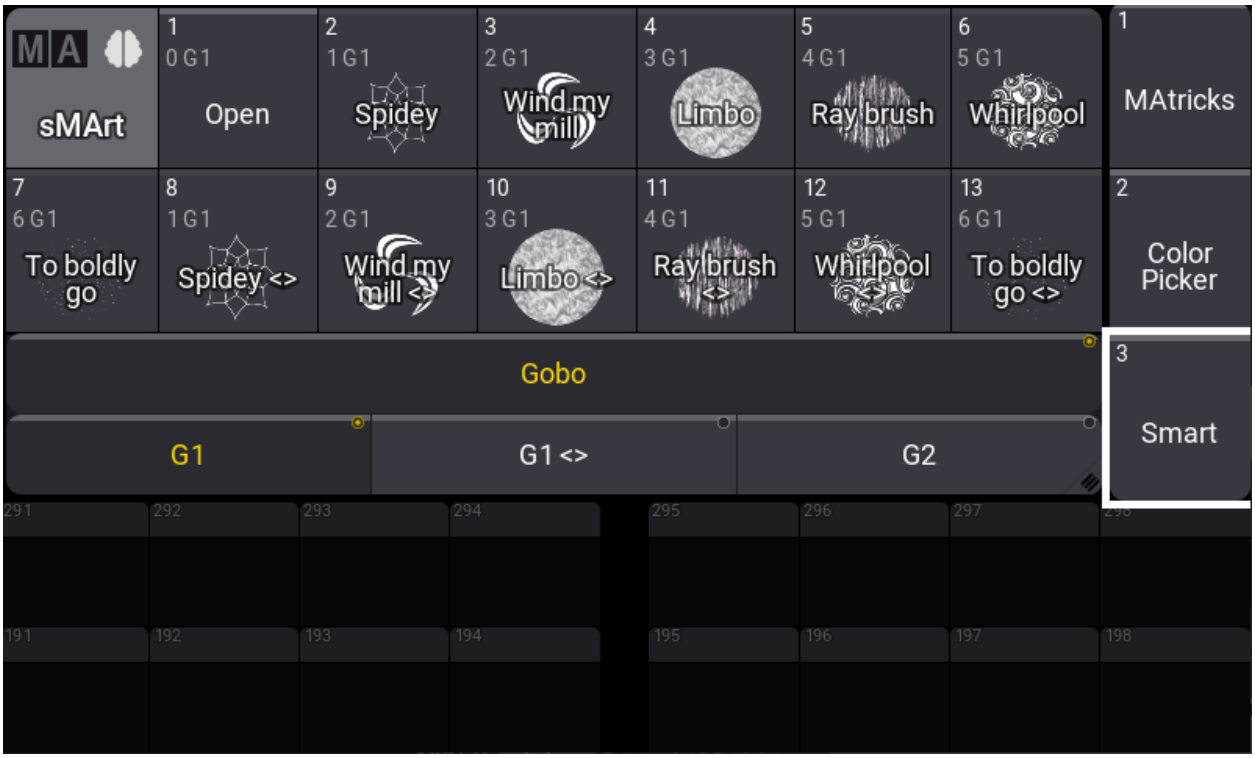
The scale option is available on screen 4 and screen 5 to take advantage of monitors of different resolutions.

Screen 6 and Screen 7

Screen 6 and screen 7 are the right command screen and left command screen, respectively. The displays on these screens offer a small amount of user-defined area. They can be configured with a selection of additional display elements.

Screen 6 can be configured to display the view bar, command line, and control bar.

Screen 7 can be configured to display the view bar, command line, control bar, and playback bar. The playback bar available on screen 7 shows the status of objects assigned to the Xkeys.



Screen 7 with view bar and playback bar visible, showing the assignments of the Xkeys.

Screen 8, Screen 9, Screen 10, Screen 11, and Screen 12

Screen 8, Screen 9, Screen 10, Screen 11, and Screen 12 are the letterbox screens, which are available on the grandMA3 full-size, light, and extension. The displays on these screens offer no user-defined area. The configurations of these displays cannot be changed. These screens are dedicated to specific display information and user input.

Screen 8 displays the encoder bar and grand master.

Screen 9 displays the playback bar and master section.

Screen 10, Screen 11, and Screen 12 display the playback bar and custom section encoders and wheels.

1.8.1.2. Desk Lock

To lock the desk, use one of these options:

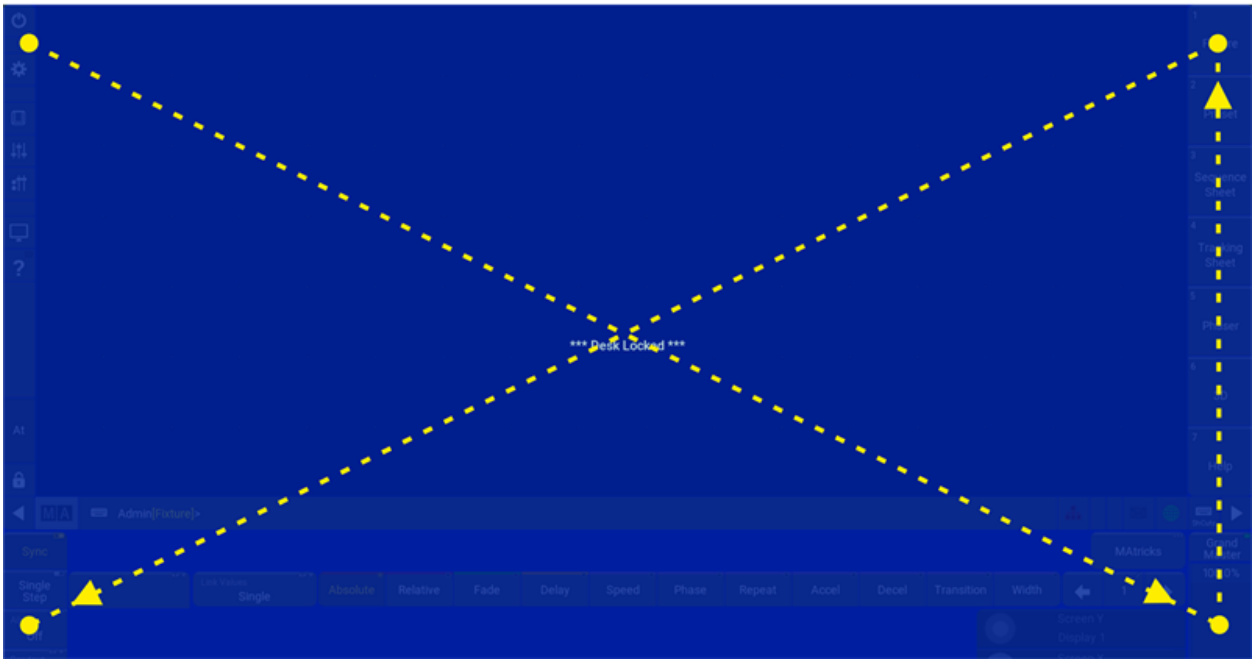
- Press MA + MA + Pause.
- Press Pause on the keyboard.
- Press F9 on the keyboard.
- Tap the icon at the top of the **control bar** to open the **shutdown menu**. Then tap the icon.

Desk Locked

The desk is locked.

To unlock the desk, use one of these options:

- Press MA + MA + Pause.
- Press Pause on the keyboard.
- Press F9 on the keyboard.
- Tap the upper left corner, lower right corner, upper right corner, and lower left corner of any screen.



Touch pattern to unlock the desk.

The desk is unlocked.

1.8.1.3. User-Defined Area

The user-defined area is located on screens 1 through 7. It appears as a grid of small dots.

Empty user-defined area with a grid of small dots

This is the area where windows can be added and arranged to be stored and recalled as views. For information on how to manage windows in the user-defined area and views, see **Windows, Views, and Menus**.

The Configure Display menu allows adjustment of the grid size within the user-defined area. It is possible to configure the grid of the user-defined area to a size larger or smaller than the available display space. For more information on the Configure Display menu, see **Configuration of Displays**.

Grid of the user-defined area configured larger than the available display space

If the grid is larger than the available display space, a dark orange outline appears around the edge of the user-defined area. Horizontal and vertical scroll bars appear, allowing access to the full space of the grid.

The 3 finger scroll gesture also scrolls the user-defined area within the visible portion of the display. For more information on gestures, see the **Gestures** topic.

Grid of the user-defined area configured smaller than the available display space

If the grid is smaller than the available display space, a partially transparent gray overlay appears over the excess area. Any previously stored views that utilize this area will still recall as stored. The overlay does allow interaction with the contents of windows underneath, but windows cannot be resized or created outside of the user-defined area.

1.8.1.4. Command Line

- **Show or Hide the Control Bar**
- **Open the Command Line History**
- **Open the Virtual Keyboard**
- **Information in the Command Prompt**
- **Change the Default Keyword**
- **Set the Cursor in the Command Line**
- **Power Status**
- **Battery Status Pop-up**
- **Session Status and Quick Access to the Network Menu**
- **Phaser Values Workload**
- **FlowControl**
- **Open the Message Center**
- **World Server Indicator**
- **Enable or Disable Keyboard Shortcuts**
- **Show or Hide the View Bar**
- **Hidden Progress Bar**

The main purpose of the command line is to display active command syntax as it is entered and to display any errors generated by failed syntax. The command line also includes several helpful status indicators and quick-access buttons to frequently used features.

By default, the command line is located at the bottom of screen 1. It can also appear at the bottom of screens 2, 3, 4, 5, 6, and 7.

Command line as it appears on screen 7

Show or Hide the Control Bar

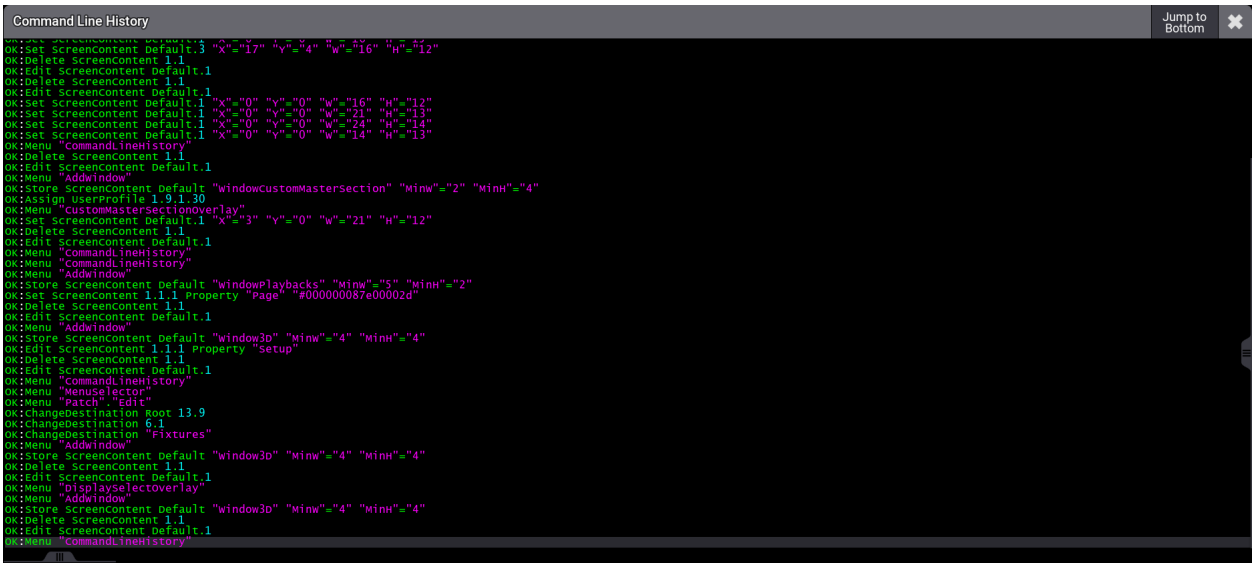
Tap the left arrow at the left edge of the command line to show or hide the control bar along the left edge of the screen.

Open the Command Line History

To open a temporary window showing the Command Line History:

- Tap MA on the left side of the command line

The Command Line History opens.



Command Line History

To close the window:

- Tap MA once again

-or-

- Tap in the upper right corner of the window

To open the window version of the Command Line History:

1. Open the Add Window dialog
2. Tap Common + Command Line History.

To show or hide the Command Line in the Command Line History window:

1. To open the Command Line History Window Settings, tap MA.
2. Enable Show Command Line.

To select a recent command in the command line history:

1. Tap and hold a recent command in the history. The command gets selected, a pop-up opens, and the Command Line History background turns red.
2. Select one of the following options from the pop-up:
    a. Copy: Copies the command in the clipboard.
    b. Execute Again: Executes the command again.
    c. Copy and Overwrite to Command Line: Copies the command into the command line. Existing command line content will be discarded.
    d. Add to Command Line: Copies the command to the end of existing command line content.
3. Tap Jump To Bottom to set the Command Line History to the latest entry.

	Hint:

	The pop-up can be opened by pressing Please or by pressing the screen encoder.

The Command Line History can display a maximum number of 2 048 lines.

For more information, see **Add Window**.

Open the Virtual Keyboard

To open a temporary command line window with a virtual keyboard:

- Tap on the left of the command line.

The window opens with a virtual keyboard.

Open virtual keyboard

To minimize the virtual keyboard:

- Tap .

To close the virtual keyboard:

- Tap in the upper right corner of the window.

To change the language on the keyboard:

- To cycle through the languages, tap the key that displays the language, for example, English, on the keyboard. For more information, see **Keyboard Keyword**.

Information in the Command Prompt

The command prompt is the point where the entered syntax appears. The prompt ends with the '>' symbol, but it contains two pieces of helpful information before the symbol.

- **User Name**
  The user name of the logged in user appears first in the command prompt.

- **Default Keyword**
  The default keyword appears in square brackets after the user name. The console uses the displayed default keyword for any numerical syntax entered without a specified object keyword.

	Important:
	It is possible to navigate through the directory structure of the console using the command line. In this case, the current location in the directory replaces the default keyword displayed in the command line. When returning to the root directory, the default keyword replaces the directory path display.

Change the Default Keyword

Possible default keywords of the command line can include:

- **Fixture**
- **Channel**
- **Universal**
- **Houselights**
- **NonDim**
- **Media**
- **Fog**
- **Effect**
- **Pyro**
- **MArker**
- **Multipatch**
- **PSR**

	Hint:
	Some of the listed keywords can be customized. For more information, see **Custom ID**.

To change the default keyword:

1. Enter the desired keyword into the command line. For example, press the Channel key.
2. Press Please.

Set the Cursor in the Command Line

When entering command line syntax using a keyboard, it is necessary to bring the keyboard focus to the command line.

To set the cursor in the command line:

- Tap the command line

- or-

- Press Esc one or more times. Once all open menus and pop-ups are closed, the keyboard focus returns to the command line.

- or-

- Press and hold MA and subsequently press Please

Set cursor in the command line

The cursor is set and starts blinking after the prompt and a white box surrounds the command line.

	Hint:
	A triangle symbol ( ) appears if the prompt is longer than the command line.

For more information, see **Command Syntax and Keywords**.

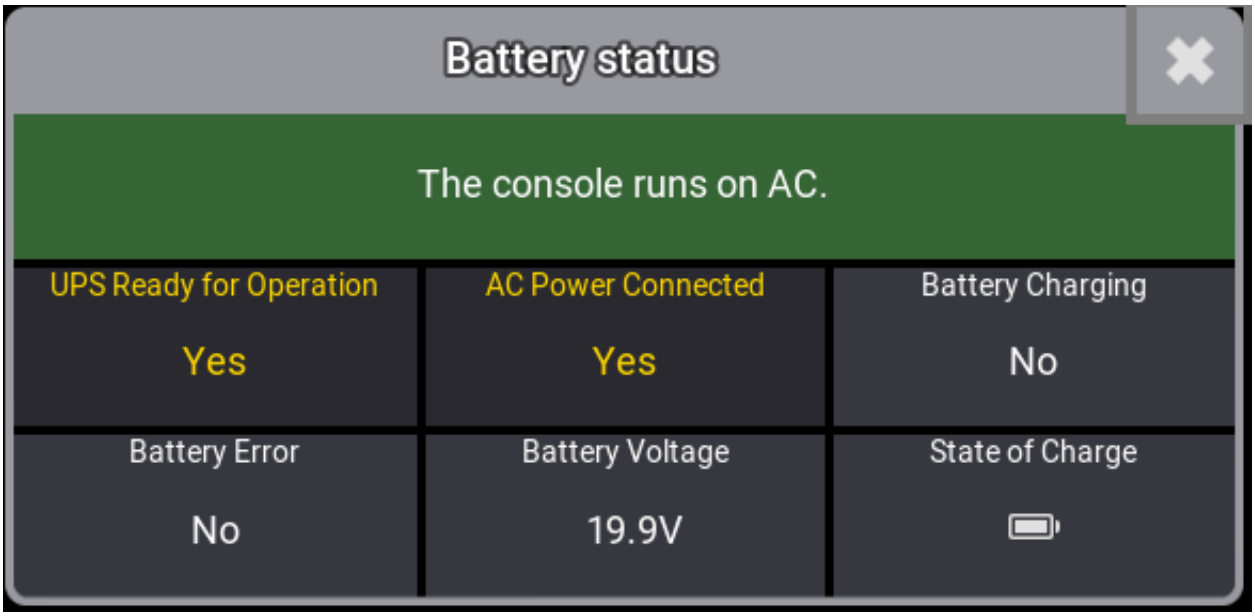
Power Status

On full-size, light, full-size CRV, and light CRV consoles, which are equipped with an internal battery backup, there is an indicator of power status on the right side of the command line. The battery percentage is always displayed at the bottom of the indicator. Status indications include the following states:

-  Power is connected and the battery is full.
-  Power is connected and the battery is not full.
-  Power is disconnected and the battery is discharging.

Battery Status Pop-up

Tap the power status indicator to open the Battery status pop-up.



Battery status pop-up

The Battery status pop-up displays multiple relevant metrics about the battery as well as A/C power.

For more information about the Power Status, see **UPS Battery**.

Session Status and Quick Access to the Network Menu

An indicator of network session status appears on the right side of the command line. Tapping the indicator toggles the network menu open or closed. The color of the network icon indicates one of the following session states:

- **Red:** Standalone
- **Black:** Startup
- **Gray:** IdleMaster
- **Light Blue:** GlobalMaster
- **Green:** Connected

For more information about networking, see the **Networking** topic.

Phaser Values Workload

The left bar between (network) and (message center) in the command line shows the current processing workload dedicated to phaser calculation.

- To show the workload of phasers, a green indicator increases with the number of parameters with actively running phasers. The exact number of phasers used is displayed as a tooltip.

For more information on Phasers, see the **Phasers** topic.

FlowControl

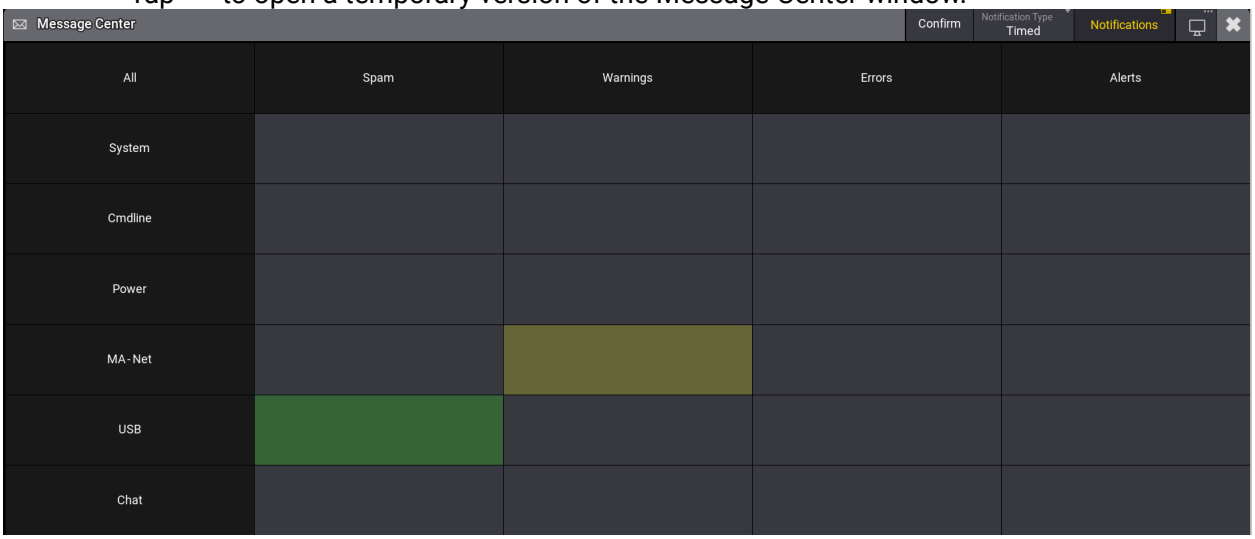
The right bar next to the Phaser indicator shows the current workload of the network. The exact intensity of the flow control is displayed as a tooltip on a scale of 0 to 255.

	Hint:
	The **Flow Control Level** is also indicated in the Network menu. For more information, see **Session**.

Open the Message Center

The envelope icon ( ) on the right side of the command line changes color to indicate new messages in the message center. The colors (red, orange, green) are based on the different message categories **Spam**, **Warnings**, **Errors**, and **Alerts**.

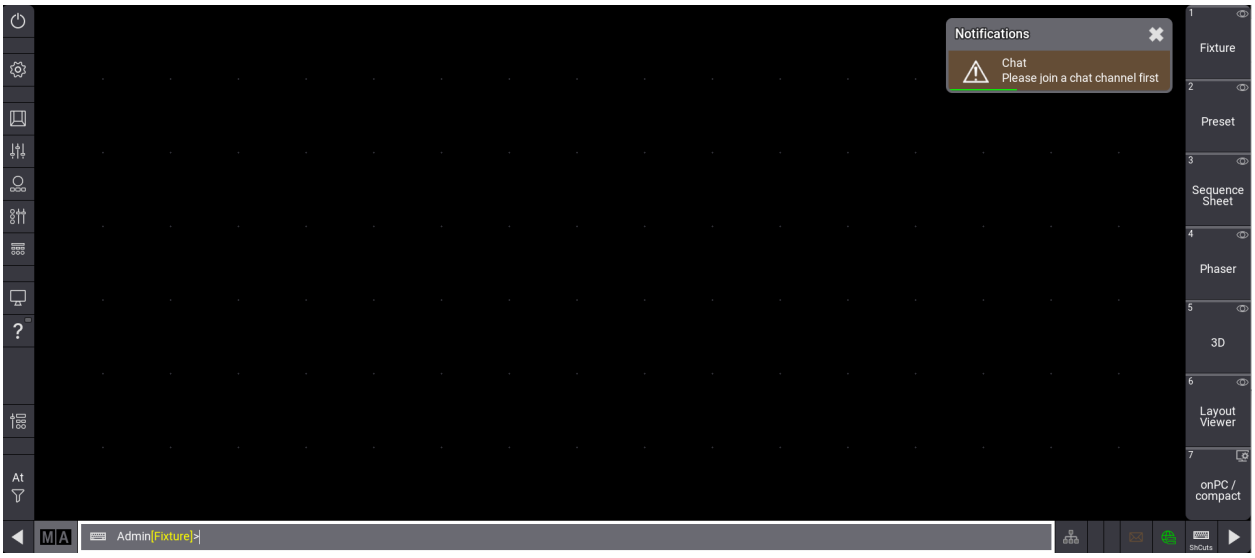
- Tap to open a temporary version of the Message Center window.



Open message center

- To view specific information about a message category, tap it. A window opens with a timestamp, sender information, and message section.
- To confirm all messages, tap Confirm. All current messages will be deleted.
- To open the message center on another screen, tap .;
- To close the message center, tap the message center icon in the command line once again or tap in the upper right corner of the window

A notification pop-up for System Errors, MA-Net Warnings, and all Chat messages will also be displayed in the upper right corner of all big screens.



Notification pop-up.

These notifications have the same background color as the indication in the message center. Messages can have different priorities: Alert (), Error (), Spam (), Warning ().

- To disable the message pop-up for all notifications, disable Notifications in the title bar of the message center.
- Notification Type:
  - To display notifications only for a span of 5 seconds, select **Timed**.
  - To display the notifications permanently until closing them, select **Permanent**.

World Server Indicator

The globe icon on the right side of the command line changes color from black to green to indicate an active connection to the world server.

For more information on the world server, see the **World Server** topic.

Enable or Disable Keyboard Shortcuts

Tapping the keyboard icon, with the text "ShCuts" beneath it, on the right side of the command line toggles the keyboard shortcuts on or off.

- A white color indicates that keyboard shortcuts are disabled.
- A yellow color indicates that keyboard shortcuts are enabled.
- A red color indicates that keyboard shortcuts are enabled but temporarily blocked by a menu requiring normal keyboard input.

For more information on keyboard shortcuts, see the **Keyboard Shortcuts** topic.

Show or Hide the View Bar

Tap the right arrow at the right edge of the command line to show or hide the bar of view buttons along the right edge of the screen.

Hidden Progress Bar

The process, e.g. when sending an update from a grandMA3 onPC station to a grandMA3 console, is indicated by a circle with a number in it on the right side of the command line. The number inside the circle indicates how many processes are running in the background.

Command LIne - Hidden Progress Bar Tap the circle to show or hide the regular progress bar.

1.8.1.5. Control Bar

The control bar allows fast access to the shutdown command, essential menus, additional displays (in grandMA3 onPC), online help, and At filter.

On console displays, the control bar is hidden on all screens by default. Within the onPC application, the control bar is shown along the left edge of screens 1 through 5 by default. In consoles and the onPC application, the control bar can appear along the left edge of screens 1 through 7.

To show or hide the control bar, tap Show Control Bar in the configure display menu for the desired screen or press the left arrow at the left side of the command line.

Another quick and easy way to toggle the control bar is to tap on the  icon in the command line.

Control bar

- Tap the icon at the top of the control bar to open the **shutdown menu**.

- Tap the  icon in the control bar to open the main menu. For more information, see **Menus**.
- Tap the  icon in the control bar to open the command section overlay. For more information, see **Command Section**.
- Tap the  icon in the control bar to open the master controls overlay. For more information, see **Master Controls**.
- Tap the  icon in the control bar to open the custom master section overlay. For more information, see **Special Executors**.
- Tap the  icon in the control bar to open the playback controls overlay. For more information, see **Playback Controls**.
- Tap the  icon in the control bar to open the Encoder Bar overlay.
- Tap the  icon in the control bar for access to all of the available displays within the grandMA3 onPC application. This option is only available in grandMA3 onPC. For more information, see **Displays in grandMA3 onPC**.
- Tap the  icon in the control bar to enter the Help keyword into the command line. For more information, see **Help keyword**.
- Tap the  icon in the control bar to toggle the main encoder bar to the command wing bar in the grandMA3 onPC application. This option is only available in grandMA3 onPC. For more information, see **Command Wing Bar**.
- Tap the  icon in the control bar to open the at overlay. For more information, see below.

At Overlay

The at overlay allows quick access to various programmer functions and special values.

At overlay

- Tap At Filter to open the At Filter Overlay. For more information, see the **At Filter** topic.
- Tap Cut Programmer, Copy Programmer, and Paste Programmer to cut, copy, and paste the contents of the programmer. For more information, see the **What is the Programmer** topic.

- Tap Delete Steps to delete the current phaser step or steps from the programmer. For more information, see the **Phasers** topic.
- Tap Clear to access the same three clear keywords available by pressing the Clear key. For more information, see the **Clear Key** topic.
- Tap Full, Zero, Default, Normal, On, Off, Release, or Remove to apply the desired special attribute value to the selection. On and off apply to the selection. Full, zero, default, release, and remove apply to the dimmer attribute of the selection. For more information, see the Specials section in the **Calculator** topic.
- Tap and hold to open the At Filter Overlay. For more information, see **At Filter Window**.

1.8.1.6. Tables in General

Compared to sheets that serve as an overview, tables is where you can actively enter, change, and modify values. Tables can be found in the Patch menu, for example.

Tables have parents and children in form of new "object lines".

- A parent is the main object
- A child is the sub object of the parent
- A child can have several children

To disable or enable children in tables, tap 'New object' line.

Simplify operation and edit in tables:

- **Move/Select**
  - To move the focus in a table, tap to select an object and use the arrow buttons of the integrated or virtual keyboard.
  - To move the selection from one column to another, tap the heading of the corresponding column.
  - Tap the heading twice to select the entire column.
- **Inertial scrolling**
  - Use two fingers to tap and quickly brush the table in either direction.
  - Depending on the momentum, the scroll slows down after a while until it comes to a halt.

1.8.1.7. View Bar and View Buttons

Screens can have View Bars visible on screens. The View bar can be on the right side of the screen or at the top of the screen. The View Bar contains View Buttons.

The view buttons can be used to store and recall views and clear the screen.

Other objects than views can be assigned. The following pool objects are supported:

- Encoder Bars
- Macros
- MAtricks
- Plugins
- Quickeys
- Screen Configurations
- Timers
- Users
- Views

If a store action without a specified object is performed on an empty view button, the default object is a new view.

The screen size might limit how many view buttons can be seen, but the view bar can be scrolled. If the view bar is scrolled, then there is an arrow icon (or) to scroll back to the first buttons.

The screen on the consoles can show 10 view buttons in a single column with the view bar on the right side and 19 view buttons on a row when at the top.

Configure the View Bar

Editing an empty area on the user-definable area opens the Configure Display pop-up:

Configure Display Pop-up

This pop-up can also be opened using the **Menu "DisplayConfig"** command. There are other methods to open the pop-up.

The right side of the pop-up has the option to toggle the view bar by tapping Show View Bar. There is an arrow icon ( ) that toggles the position of the view bar between the right side of the screen and the top of the screen. There is also the option to add up to three columns or three rows (depending on the view bar position) by tapping + or limiting again down to one by tapping - below the Show View Bar.

Each column or row begins with its own hundred number, so there can be a total of 300 view buttons on each screen.

The downward pointing triangle ( ) adds a button (or) at the last view button. Tapping this button scrolls the view buttons to the next set of view buttons.

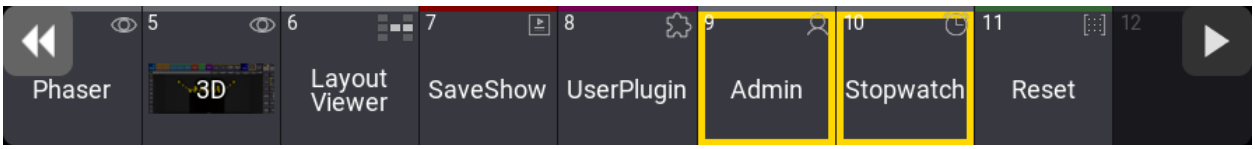
The view bars on each screen work independently, meaning that the view bar can have different sizes, locations, and scroll positions on each screen. View button 4 on screen 1 is "ViewButton 1.4". View button 4 on screen 2 would be "ViewButton 2.4".

Tapping an empty view button clears the screen.

Tapping and holding a view button activates the store view function, which opens the Store View pop-up. Read more about storing views in the **Store and Recall Views topic**.

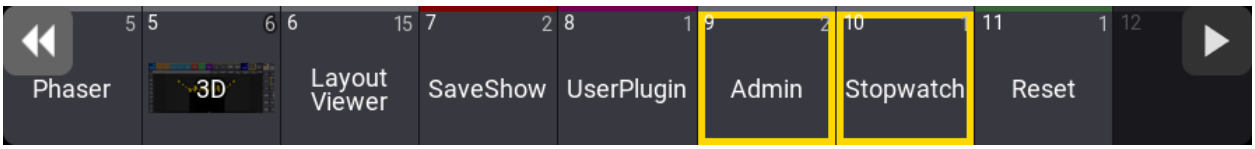
View Button Information

The view buttons show an icon in the upper right corner and a colored bar at the top indicating the object type.



The example above shows different objects in the view bar.

Pressing and holding MA show the object number instead of the object icon.



Notice that the icon for the view object is only shown if the view contains information about just one screen. If the view contains information for two or more, then the icon is replaced by an icon showing which screens it relates to - see view button 6 above.

Tapping objects in the view bar is the same as tapping the object in the respective pool - It executes the default pool action. For instance, tapping view button 9 in the image above selects user number 2 in the user pool.

1.8.1.8. Adjustable Columns

In certain windows and sheets, it is possible to move columns around and to show or hide columns.

	**Hint:**
	To save and recall your arrangements, store them as Views or use Preferences. For more information, see **Store and Recall Views** and **Store Settings and Store Preferences**

The following windows and sheets in the Add Window dialog use gestures to adjust columns.
See **gestures** for more information.

- Content Sheet.
- Fixture Sheet.
- Track Sheet.
- Layouts and Layout Editor.
- Timecode Editor and Timecode Viewer.

In addition to the gesture feature, the following windows and sheets use **column sets** to make adjustments:

- Agenda Viewer.
- RDM Devices Viewer.
- Sequence Sheet.
- Track Sheet.

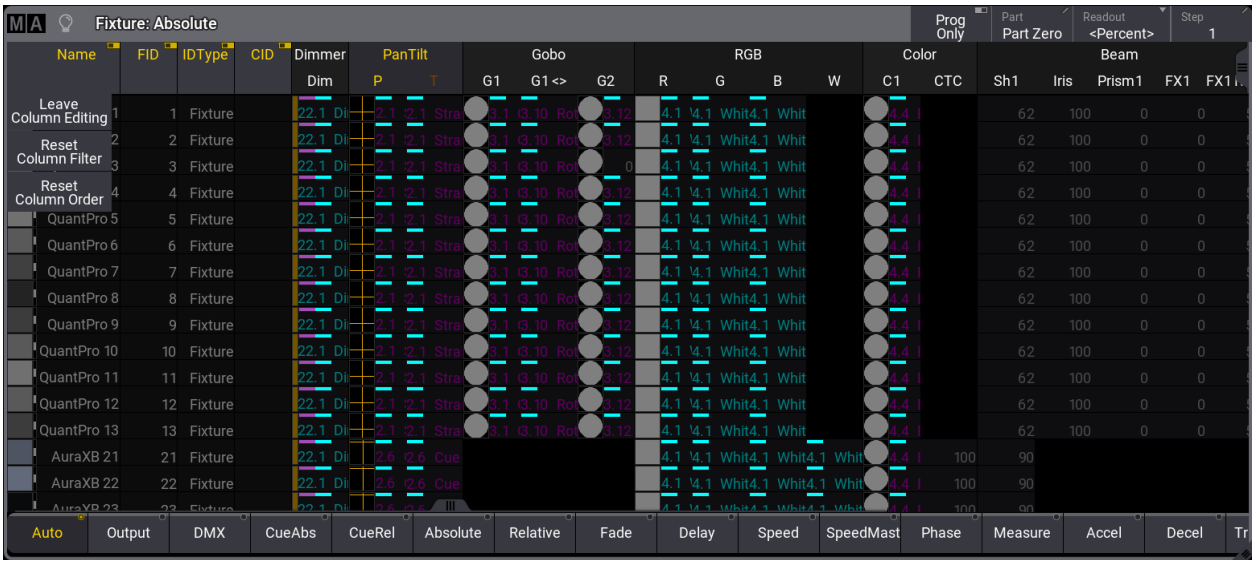
The following video provides a basic overview of the column editing feature using gestures.

Open Column Editing

To edit specific columns, a column edit mode has to be opened in the corresponding window.

To open column editing:

1. Open the corresponding window, for example, the fixture sheet via the **Add window dialog**.
2. Perform a 2 finger scroll gesture in the header row of the window. A gear icon () appears in the top left corner of the header.
3. Tap  within three seconds. Three available buttons for column editing are shown.
4. Tap Enter Column Editing. The contents of the window will be displayed in a light gray color and the columns will now be ready for editing as shown in the image below.



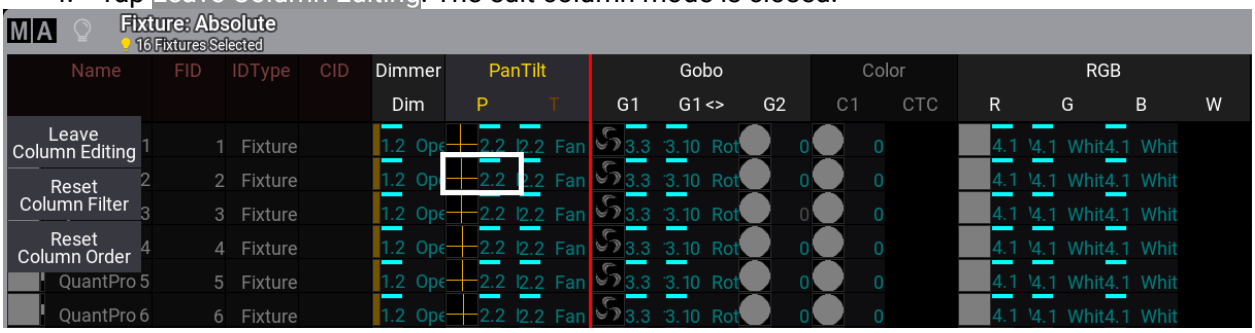
Fixture Sheet - Column Editing

Edit Columns

A drag & drop gesture is used, to change the column order.

To change the column order in the corresponding window, for example, the fixture sheet:

1. Tap and hold a column header with one finger, for example, the Color header.
2. Drag the column left or right to the desired position. A red vertical line indicates the potential position of the column if releasing the finger.
3. Release the finger. The column has now moved to the new position. To reset all columns back to the initial position, tap Reset Column Order.
4. Tap Leave Column Editing. The edit column mode is closed.



Fixture Sheet - Column Editing mode with a red vertical line

	Hint:
	To move a column without entering column editing, long press a column in the header area until the red vertical line appears.
	Restriction:
	Columns have certain zones in which they can be moved. Excluded zones are marked in red dark color. For example, as shown in the image above, it is not possible to move the column Color next to the column Name.

To filter columns in a corresponding window, for example, in the fixture sheet:

1. Open column editing.
2. Tap a column header to disable it, for example, Name. The column is greyed out.
3. Tap Leave Column Editing. The edit column mode is closed and an  icon in the top right corner of the window indicates that a column is hidden.
4. To reset all filters back to the initial position, tap Reset Column Filter in the column edit mode.

Column Sets

In certain windows and sheets (**see above**), columns can be adjusted and saved as column sets. This can be done in a column set editor. The editor can be found in the Columns tab of the corresponding window settings.

In the title bar of the corresponding window, the created column set can be selected. For more information about how to make the column selector visible in the title bar, see **Title Bar Configuration**.

The Column editor is divided into a scrollable grid on the left side and a menu of buttons on the right side.

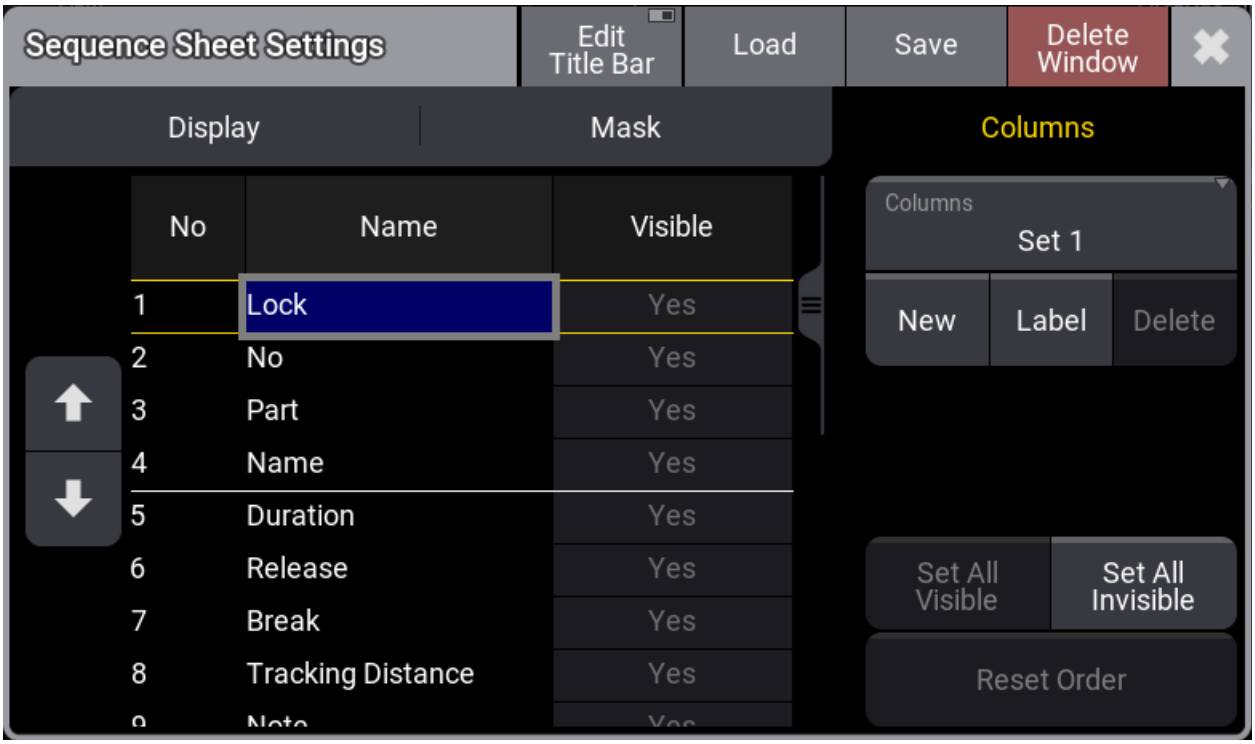
The following video provides a basic overview of how to customize columns using the column set editor.

To open the Column set editor follow these steps:

1. Open a window via the add window dialog, for example, the Sequence sheet.
2. Tap MA in the title bar of the window. The window settings open.
3. Tap the Columns tab. The Column set editor is open.

or

- Tap and hold Columns in the title bar. A temporary column set editor opens.



Sequence Sheet - Columns tab

To create a new column set:

1. Tap, hold, and swipe Columns. A pop-up opens.
2. Drag and drop your finger over New at the bottom of the pop-up. A new column set is created.

or

- Tap New. A new column set is created

	Hint:
	The currently selected column set is used as a template for a new column set.
	Important:
	The Column Set Editor offers only columns that were visible in the sheet at the time.

To enable or disable columns in the window:

1. Two-finger edit Yes/No in the Visible column.
2. Tap Set All Visible to enable all columns.
3. Tap Set All Invisible to disable all columns.

To change the order of the columns:

1. Select a specific column in the grid on the left.
2. Tap or to move the column up or down in the grid.
3. Reset Order will reset the order.

	Restriction:

	A horizontal white line divides the grid into two areas. It is not possible to move a column to the other area.

To select an existing column set:

1. Tap, hold, and swipe Columns. A pop-up opens.
2. Drag and drop your finger over the corresponding column set. The column set is selected.

To label a column set:

1. Select a column set.
2. Tap Label. The label editor opens.
3. Enter a name. Tap Please. A column set is named.

or

1. Enter label into the command line.
2. Tap Columns in the title bar of the corresponding sheet.
3. The label pop-up for the selected column set opens.

To delete a column set:

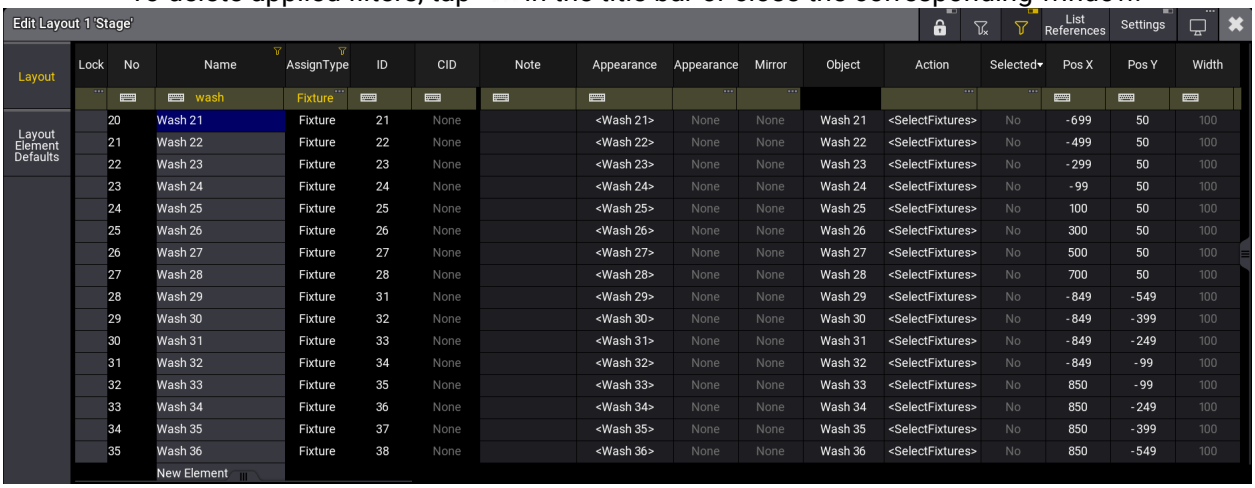
1. Select a column set
2. Tab Delete. The column set is deleted.

1.8.1.9. Temporary Filtering

Certain menus and windows, such as the Patch Menu or Layout Editor, allow you to temporary filter and search for specific terms in the grid.

	Hint:
	You can use several column filters at the same time to specify the results.

- The filter area is displayed as a yellow row between the header row and the first row of the grid. Tapping in the title bar of the window enables or disables the filter area.
- The in the filter area marks text input fields and the three dots (...) mark the selection fields.
- Applied filters are indicated with a yellow in the header of the corresponding column.
- To delete applied filters, tap in the title bar or close the corresponding window.



Edit Layout - Filter

Example

To select a filter in the Layout Editor:

1. Open the grandMA3 Demoshow.
2. Open the Layout Editor. For more information about Layout, see **Edit Layout**.
3. Make sure is enabled in the title bar.
4. Tap ... in the **Assign Type** column. A pop-up opens.
5. Select Fixture. The pop-up closes and the grid shows only the rows with Assign Type set to Fixture.

To use the text input field filter in the Layout Editor:

1. Follow steps 1-3 from above.
2. Tap to clear all filters.
3. Tap in the **Name** column. A text input field opens.
4. Type **Wash** and press Please. The pop-up closes and the grid shows the rows with "Wash" in their name.

or

1. Tap the area to the right of the in the **Name** column. A text cursor is set.
2. Start typing. The Name column is filtered as you type.

	Hint:
	If you disable the filter area ( ) while a filter is set and then enable it again, the previously set filter will still be active.

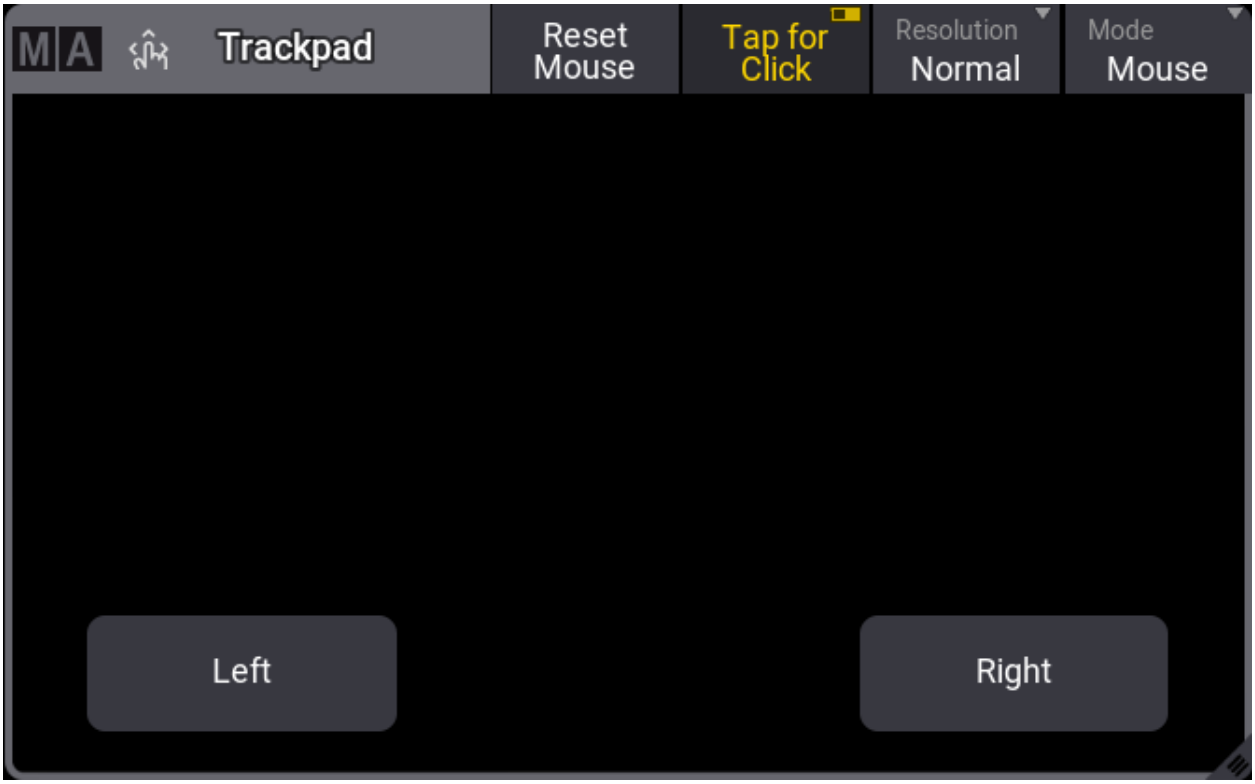
1.8.1.10. Trackpad Window

The trackpad window allows precise control of a mouse cursor as well as an alternate method for controlling the position of selected fixtures.

The trackpad window can appear within the user-defined area of screens 1 through 7. Views can store and recall the appearance and settings of the trackpad window. For more information on adding windows to the user-defined area, see the **Add windows** topic.

Tap the Mode button in the upper-right corner of the trackpad window to toggle between mouse mode and pan/tilt mode, or tap and swipe to see a menu of available modes.

Mouse Mode

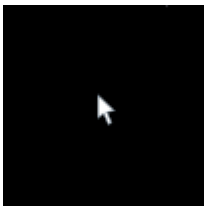


Trackpad window in mouse mode

Trackpad Mouse Cursor

While in mouse mode, tapping inside the main area of the trackpad window and swiping in any direction presents a dedicated cursor and moves that cursor in the direction of the swipe.

The main cursor in the user interface of the console appears as a black arrow with a white outline. This cursor is not affected by the trackpad window. A separate cursor appears when using the trackpad in mouse mode. The trackpad mouse cursor appears as a white arrow with a light gray outline.


Trackpad mouse cursor

The trackpad cursor initially appears slightly larger than normal. After three seconds, it reverts to its normal size. After five seconds of idle time, the trackpad cursor disappears.

The trackpad mouse cursor can move freely across the user interface of the console within screens 1 through 5. Although the trackpad window can appear on screen 6 and screen 7, the trackpad cursor cannot appear on these screens.

Trackpad Window Buttons in Mouse Mode

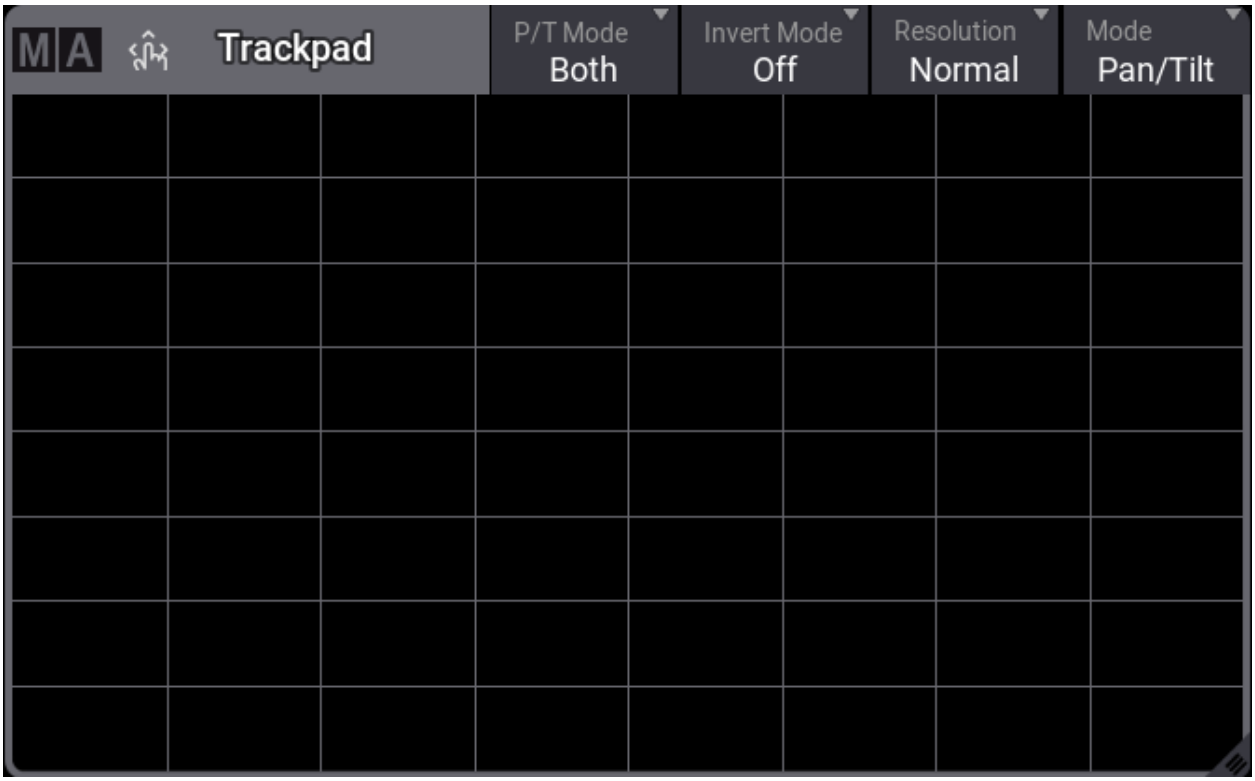
The main area of the trackpad window in mouse mode includes two buttons:

- Left: Tap for a left mouse click. This selects the item under the point of the trackpad cursor. Tap and hold while swiping in the main area of the trackpad window with a second finger to mimic a 1 finger swipe gesture.
- Right: Tap for a right mouse click. This generates the same result as a 2 finger edit gesture.

The title bar of the trackpad in mouse mode includes tools, which control the behavior of the trackpad. These tools include:

- Reset Mouse: Tap to respawn the trackpad mouse cursor in the center of screen 1.
- Tap for Click: When enabled, tapping anywhere in the main area of the trackpad window, except for the Right button, executes a left mouse click. When disabled, only a tap on the Left button executes a left mouse click.
- Resolution: Tap to cycle through the available resolution options for the trackpad mouse, or tap and swipe to see a menu of available resolution options. This resolution setting is separate from the resolution setting in pan/tilt mode.

Pan/Tilt Mode



Trackpad Pan/Tilt mode

While in pan/tilt mode, horizontal swipes within the main area of the trackpad window adjust the pan attribute of any selected fixtures. Vertical swipes adjust the tilt attribute. Diagonal and curved swipes adjust both pan and tilt accordingly. The main area of the trackpad window in pan/tilt mode displays a simple grid as a visual guide.

The title bar of the trackpad in pan/tilt mode includes tools, which control the behavior of the trackpad. These tools include:

- P/T Mode: Tap to cycle through the available pan/tilt modes, or tap and swipe to see a menu of available pan/tilt mode options. These options include:
  - Pan Only: This mode only allows adjustment of the pan attribute, ignoring any vertical movement.
  - Tilt Only: This mode only allows adjustment of the tilt attribute, ignoring any horizontal movement.
  - Both: This mode allows simultaneous adjustment of both the pan and tilt attributes.
- Invert Mode: Tap to cycle through the available invert modes, or tap and swipe to see a menu of available invert mode options. These options include:
  - Off: Both pan and tilt are adjusted normally.
  - Pan Invert: Pan inputs are interpreted in reverse of the normal direction. Tilt inputs remain normal.
  - Tilt Invert: Tilt inputs are interpreted in reverse of the normal direction. Pan inputs remain normal.
  - Both: Both pan and tilt inputs are interpreted in reverse of the normal direction.
- Resolution: Tap to cycle through the available resolution options for the trackpad control of pan and tilt, or tap and swipe to see a menu of available resolution options. This resolution setting is separate from the resolution setting in mouse mode.

	Hint:

	The **TrackpadMode**, **Mouse Resolution**, **P/T Resolution**, **P/T Invert Mode**, and **P/T Mode** settings are also available in the **Trackpad Window Settings** pop-up. To open this pop-up, tap MA in the upper-left corner of the **Trackpad** window.

1.8.2. Gestures

Use gestures on the touchscreens of the grandMA3 console to quickly navigate and adjust settings.

- **Tap**

Briefly tap the surface with a fingertip.

- **Tap & Hold**

Tap the surface and hold it for at least one second.

- **Zoom**

Tap the surface using two fingers and move them apart or toward each other.

- **Resize**

1. Tap the title bar or the title field using any finger of one hand and hold it.
2. Briefly tap anywhere on the surface using a second finger.
3. Release both fingers.
4. The window is resized.

- **1 Finger Swipe**

Tap and brush in any direction.

- **2 Finger Scroll**

To scroll within a window, tap the surface with two fingertips and brush the surface in the desired direction.

	Hint:
	If using a mouse, hold down the right and left buttons and move the mouse.

- **3 or More Finger Scroll**

Requirement:

- Set the height and/or width of the user-defined area larger than the available screen space. For more information see the **Configuration of Displays** topic.

To scroll the user-defined area within the available screen space, tap the surface with three or more fingertips and brush the surface in the desired direction.

	Hint:
	When using a mouse, press and hold the option key (on macOS) or Alt key (on Windows), then hold down the right and left buttons and move the mouse.

- **Drag & Drop**

Use this gesture to move and resize windows within the user-defined area as well objects and elements within certain graphical editors (such as the 3D, Layout, Phaser Editor, or Timecode window).

1. Tap and hold with one finger.
2. Move the finger to the desired position.
3. Release the finger.

- **2 Finger Edit**

In sheets and other similar grids:
The **Precise Edit** user profile setting defines the behavior of the **2 Finger Edit** gesture when it is used within sheets and other similar grids (such as the macro editor or patch menu).

Precise Edit disabled:

1. Tap the desired cell.
2. Tap with two fingers anywhere within the same window of the desired cell.

Precise Edit enabled:

1. Tap and hold one finger in the desired cell.
2. Tap anywhere on the same screen with a second finger.
3. Release both fingers.

In pools:
The **Precise Edit** user preference setting does not change the behavior of the **2 Finger Edit** gesture when it is used to edit objects within pools. To edit a pool object, follow the steps above under "**Precise Edit** enabled."

For more information about user profile settings, such as **Precise Edit**, see the **User Settings** topic.

- **On-screen Encoder and Rotating Knob**

The on-screen encoders and rotating knobs types each have a setting in **User Profile settings** that defines how the on-screen element work.

The User Profile settings have all the settings that apply to the user profile. See all the settings in the **User Settings topic**.

Rotate Style

To change the value using a gesture in the user interface:

1. Tap and hold the desired encoder or knob icon. The encoder icon turns dark.
2. Draw circles around the center of the encoder or knob in the direction of the desired change. Turning clockwise will increase the value, and turning counter-clockwise will decrease the value.

3. If desired, drag the pointer to another area of the screen. This moves the virtual center of the on-screen encoder to this new point. Larger circles result in finer control resolution.

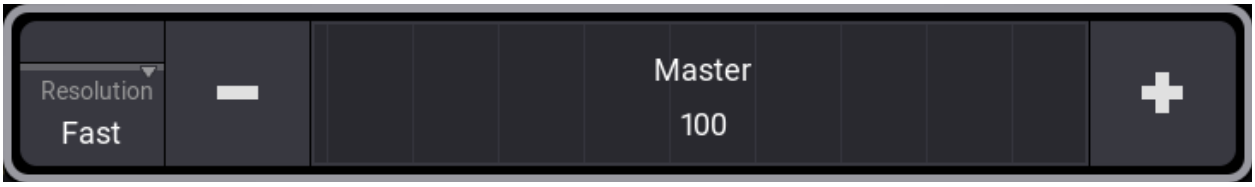
For more information about the on-screen encoder, see the **Encoder Toolbar** topic. For more information about the on-screen knobs, see the **Executor** topic.

Drag Style

Here is how to operate the encoders (including knobs) on a screen using the fingers:

1. Tap and hold the encoder, drag it outside the encoder, and release the finger.
2. A fader pop-up opens. Depending on the drag direction, it opens horizontally or vertically (see example below).
3. Tap into the fader area and hold it.
4. Moving the finger in the direction of the bar, the value changes.
5. When the desired value is set, release the finger.
6. The pop-up closes automatically when releasing the finger outside of the encoder.
7. Tapping the encoder fader opens a calculator.
8. If the position is held for more than one second, the fader value gets locked.

This is an example of the horizontal drag encoder pop-up:



Horizontal version of the Drag rotating knob style

There are small differences between the knob and encoder styles.

The buttons - and + in the small pop-up allow changing the value by 10%. If there are assigned commands to the knobs left and right rotation, then these are on the - and + buttons instead.

Tapping Resolution changes the value between the available resolutions. The knobs have **Slow** and **Fast**. The encoders have the **resolutions available** on the physical encoders. Depending on the chosen value, the value will change slower or faster when moving the fader in the pop-up.

The encoders have multiple resolutions. Typically, they are **Coarse**, **Fine**, **Increment**, and **Native**. Learn more in the **Encoder Resolution topic**.

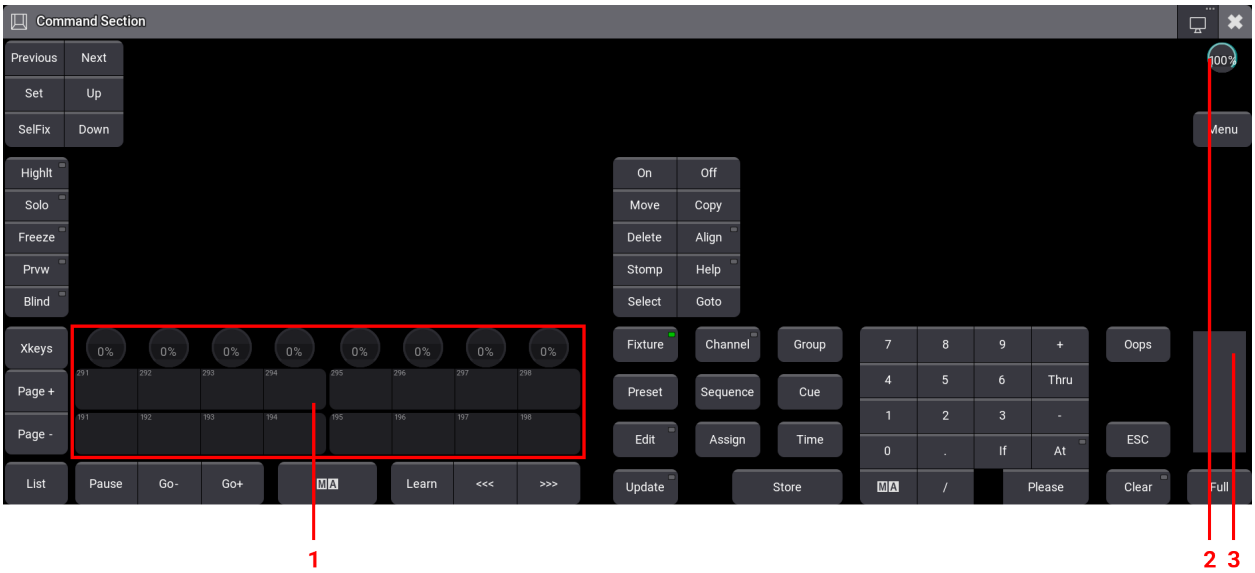
Each encoder pop-up displays which fader function is assigned to the encoder.

1.8.3. Command Area

This menu displays the Command Area of a console.

Consoles have a command area with physical hard keys. This on-screen version of the command area is useful when using an onPC.

To open the Command Section, tap in the control bar on the left of the screens 1 to 7 or F3 on a keyboard.



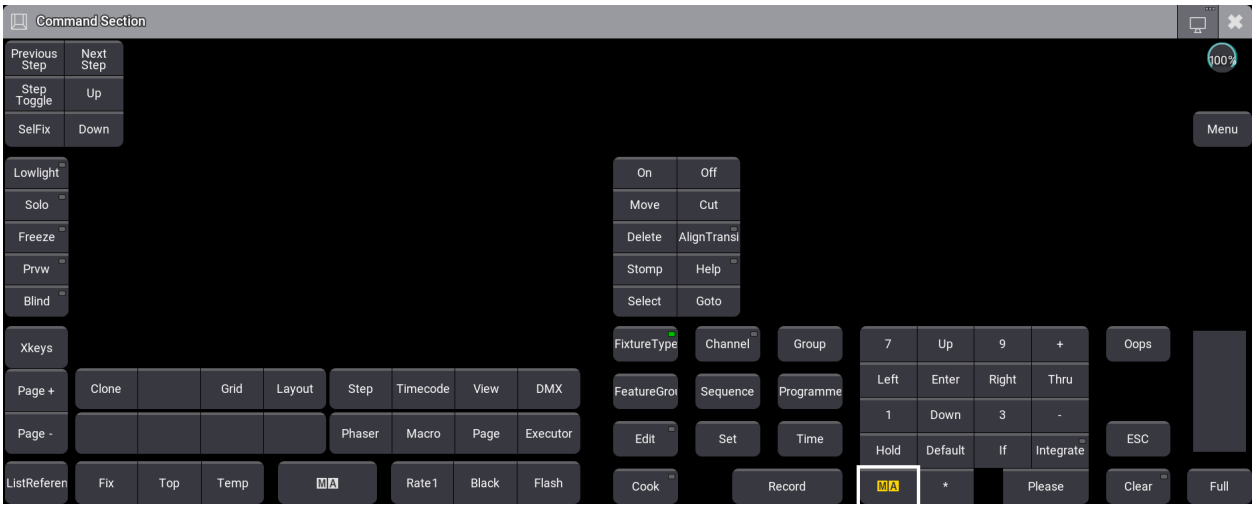
Command area

Apart from the command area, this menu also displays:

1. **Xkeys**
2. **Grand master**
3. **Level wheel**

The on-screen level wheel in this command area can be scrolled using a mouse with a scroll wheel. Move the pointer above the level wheel and scroll the wheel up or down to change the level.

To use the latch function, tap and swipe a MA key. The MA key turns yellow, and the keys of the Command Section changed, as shown below:



To unlatch, tap the latched MA key.

1.8.4. Oops Overlay

The Oops overlay displays the history of completed actions. Completed actions are displayed in form of a list and in chronological order. The action at the bottom of the overlay shows the latest entry.

The latest action or multiple actions can be Oopsed. It is also possible to use Oops filters for different kinds of actions.

For more information about Oops, see **Oops keyword** and **Oops hardkey**.

	Important:
	All Oops entries are cleared when changing the Patch or loading a show file.

To open the Oops overlay:

- Press and hold Oops.

or

- Execute the command **Menu "OopsOverlay"**.

Oops overlay

The overlay is separated into two columns:

- **Name**: Shows the executed command or the action that has been done.
- **Elapsed Time / Session Time**: Shows the bygone time since the action has been executed.

To toggle between different time units in the overlay, tap Columns in the title bar:

- Elapsed Time: Shows the passed time since the command has been executed.
- Session Time: Shows the time off the day the command has been executed.

Oops Actions

	Hint:
	Only the last action or a coherent amount of actions can be Oopsed. A selection of actions always includes the latest one.
	Hint:
	A blue background in the overlay indicates the selected action.

Requirements:

- Change at least two values in the show file to create Oops events, for example, by setting dimmer, position and color values for fixtures.

To undo the last action:

1. The action at the bottom is selected, when opening the Oops overlay.
2. Tap Oops Last Action located the bottom right corner in the Oops menu. The latest action is undone and is vanished in the overlay.

To undo multiple actions:

1. Use **2 Finger Scroll** or the scrollbar to scroll to to a particular event in the Oops overlay.
2. Tap on an any listed event to be undone. The latest event and all other events to the taped event are selected.
3. TapOops x Actions. The x stands for the selected number of actions.
4. The selected actions are undone and are vanished from the overlay.

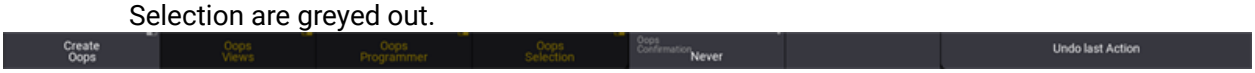
Toggle Oops Filters

The following filters in the Oops overlay allow to filter events:

- Create Oops
- Oops Views
- Oops Programmer
- Oops Selection

	Hint:
	The four filters operate the corresponding settings of the User Configuration. For more information, see **User Settings**.
	Hint:
	/NoOops does not generate oops events when executing commands. For more information, see **/NoOops Option Keyword**.

- To create no events at all, disable Create Oops. Oops View, Oops Programmer and Oops Selection are greyed out.



Create Oops

Oops Confirmation

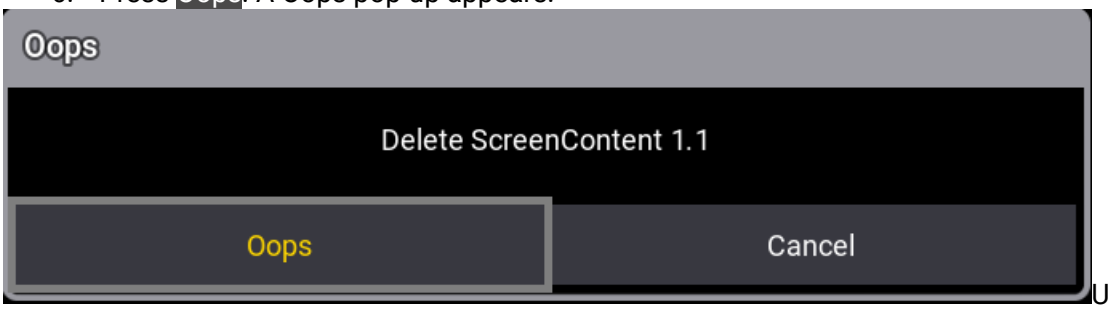
The setting Oops Confirmation defines when an Oopsed event has to be confirmed via a pop-up.

Oops Confirmation can be set to the following options:

- Always: Any Oopsed event needs to be confirmed.
- Main: Important actions needs to be confirmed, for example, storing, moving, or deleting objects.
- Never: No confirmation is needed. Oops Configuration is set to Never by default.

To change the Oops Confirmation to Always:

1. At least one Oops event is created.
2. Press and hold Oops. The Oops overlay opens.
3. Tap, hold and swipe Oops Confirmation. A dropdown menu opens.
4. Select Always. The dropdown menu closes.
5. Close the Oops Overlay.
6. Press Oops. A Oops pop-up appears:



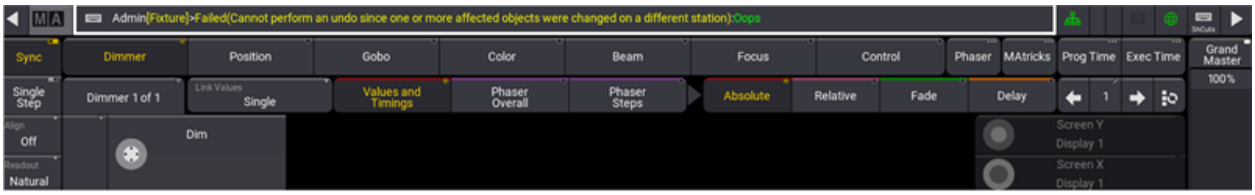
Undo pop-up

7. Tap Oops to confirm. The last action is undone.

Oops in a Session

Oopsing an action is possible in a session with multi-users. For more information about multi-users in sessions, see **Multi User Systems** and **Session**.

Investigations about already changed objects by other stations are made, when Oops is used in a session.
The first station displays a failure message in the command line, if the object was already changed by another station:

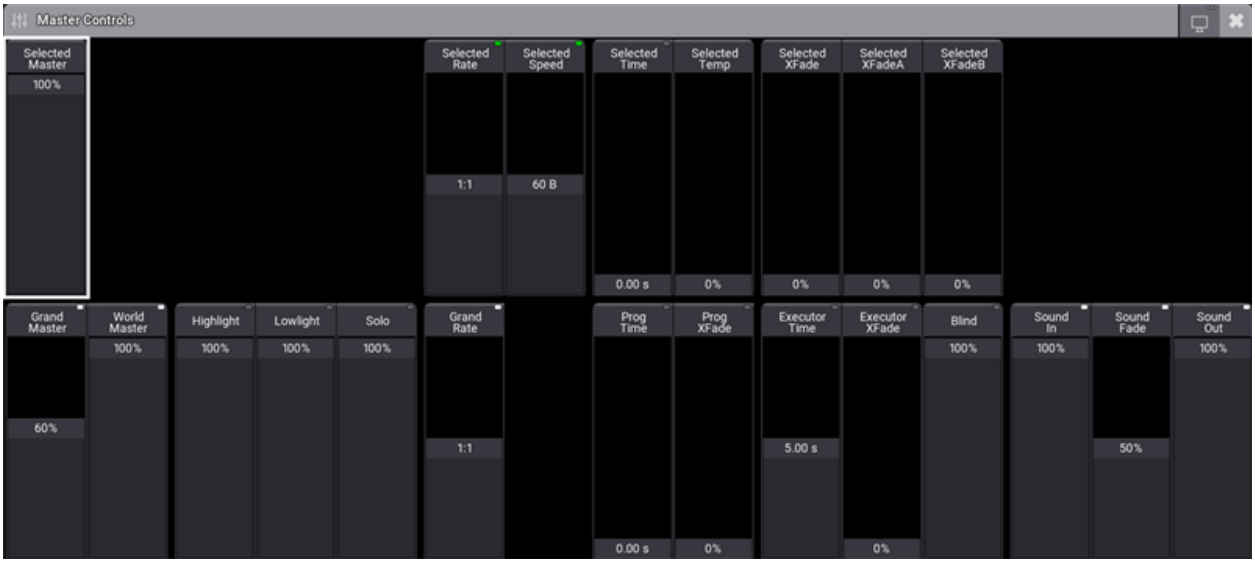


Error message

1.8.5. Master Controls

The menu Master Controls displays all selected and grand masters.

To open Master Controls, tap in the control bar on the left of screens 1 to 7 or press F4 on a keyboard.



Master Controls

1. Upper section is used in the selected sequence.
2. Lower section is used for global settings.

Adjust the value of the on-screen faders depending on their type:

- On-screen faders with indicators in the right corner of the title bar:
  - To change modes, see **Masters**.
  - To adjust values, tap and slide upward or downward.

- On-screen faders without indicators:
  - To adjust values, tap and slide upward or downward.

	Hint:
	You can enable or disable the on-screen faders **Highlight**, **Lowlight**, **Solo**, and **Blind** using HighIt, MA + HighIt, Solo, and Blind on the console.

1.8.6. Playback Controls

The Playback Controls menu shows an on-screen version of the physical faders of a console and/or wings.

To open Playback Controls, tap in the control bar on the left of screens 1 to 7 or F5 on a keyboard.

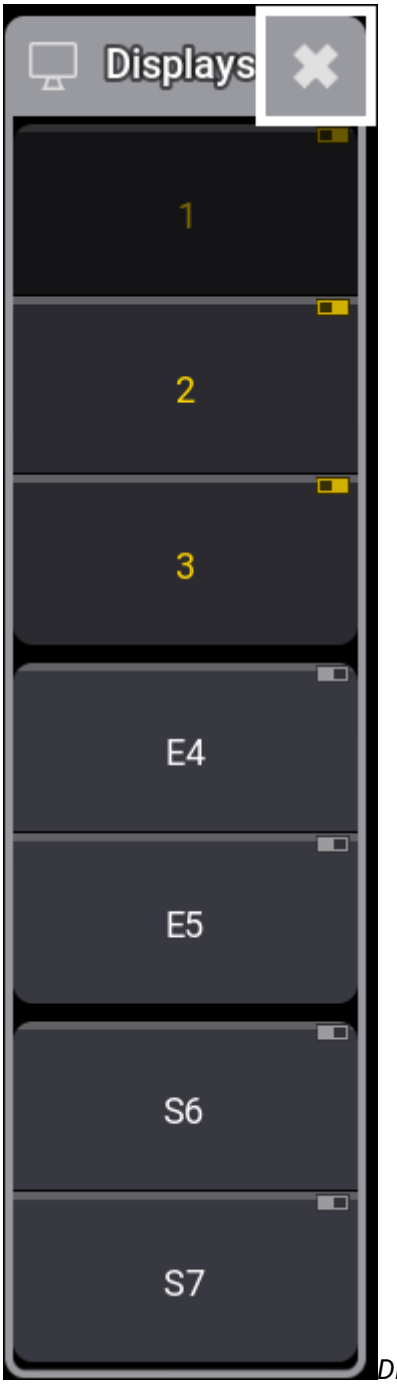
For more information on the functionality of executors, see **Executors**.

Playback Controls from the Demoshow

1.8.7. Displays in grandMA3 onPC

The Displays pop-up allows toggling showing, or hiding all 7 screens on the grandMA3 onPC.

To open the Displays pop-up on the grandMA3 onPC, tap in the **control bar**.



Displays

Displays can be toggled On or Off by tapping the relevant buttons.

The number and the indicator turn yellow on the displays that are shown.

The pop-up can be closed by tapping the icon again in the control bar or by tapping the X in the upper right corner of the pop-up.

For information on the location of menus, see **Change Menu Locations**.

1.8.8. Encoder Bar

Encoder Bar

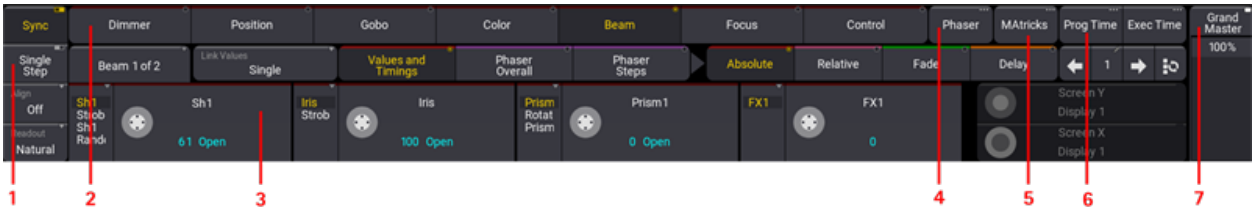
The encoder bar displays the attributes or functions linked to the encoders. The encoder bar also includes shortcuts for quick access to a selection of user profile settings and programming tools.

On the grandMA3 full-size, the grandMA3 full-size CRV, the grandMA3 light, and the grandMA3 light CRV; the encoder bar appears on screen 8. By default, the encoder bar appears at the bottom of screen 1 on all other grandMA3 consoles and the grandMA3 onPC software.

To show or hide the encoder bar on screen 1 on any grandMA3 console or the grandMA onPC software, tap Show Encoder Bar in the configure display menu on screen 1. For more information on the configure display menu, see the **Configuration of Displays** topic.

The Encoder Bar is available as a window under the **More** and **All** tabs in the **Add Window** pop-up. For more information on adding windows, see the **Add Windows** topic.

The Encoder bar is also available as an overlay. To open the overlay, tap in the control bar.



Encoder bar

The encoder bar includes the following main sections:

1. User profile settings and align mode.
    - Tap Sync to disable or enable synchronized playback of phasers in the programmer. For more information on sync, see the **Phasers** topic.
    - Tap Single Step to enable or disable single step. For more information on the single step setting, see the **User Settings** topic. For more information on using steps in general, see the **Phasers** topic.
    - Tap Align to cycle through align modes or tap and swipe to open a pop-up menu containing all of the available align modes. For more information on align, see the **Align** topic.
    - Tap Readout to cycle through attribute value readout options or tap and swipe to open a pop-up menu containing all of the available readout options. For more information on readout, see the **User Settings** topic.
2. The encoder bank buttons appear along the top of the encoder bar. For more information, continue to the following sub-topics.

3. The encoder toolbar occupies most of the area in the encoder bar. The layer toolbar is part of the encoder toolbar. For more information, **Feature Group Control Bar** and **Encoder Toolbar**.
4. Tap Phaser to open a temporary version of the Phaser Editor window. For more information on the Phaser Editor window, see the **Phaser Editor** topic.
5. Tap MAtricks to open a temporary version of the MAtricks window. The background, top bar, and three dots in the upper-right corner of this button will change color to indicate that MAtricks is active. For more information on MAtricks, see the **MAtricks and Shuffle** topic.
6. Time Control. Two on-screen fader controlling two different timing elements. They are called **Prog Time** (Program Time) and **Exec Time** (Executor Time). For more information, see **time control**.
7. The grand master appears at the right edge of the encoder bar. For more information on the grand master, see the **Grand Master** topic.

Encoder Bar Window Settings

The elements in the Encoder Bar window can be enabled or disabled.

To open the Encoder Bar Window Settings

1. Open the Add window dialog, open the More tab and then tab Encode Bar. The Encoder Bar window opens.
2. Tap MA in the title bar of the window. The Encoder Bar window settings opens.

Tap the following settings to show or hide various elements of the Encoder Bar window:

- **Title Bar:** Press MA + MA (or Ctrl + Alt) to temporarily show the title bar.
- **User Profile Settings:** Buttons on the left.
- **Encoder Bank:** The buttons top left.
- **Tool Pop-Ups:** The buttons top right.
- **Timing Buttons:** The buttons Prog Time and Exec Time.
- **EncoderPage:** The encoder page selector.
- **Layer Toolbar:** The layer toolbar in the middle.
- **Step Buttons:** The step buttons on the right.
- **Encoder Label:** The active value and the name above the encoder.
- **Function Selector:** The channel function area.
- **Fader Encoders:** Changes the encoder to a fader style in the graphical user interface.
- **Grand Master:** The grand master fader.

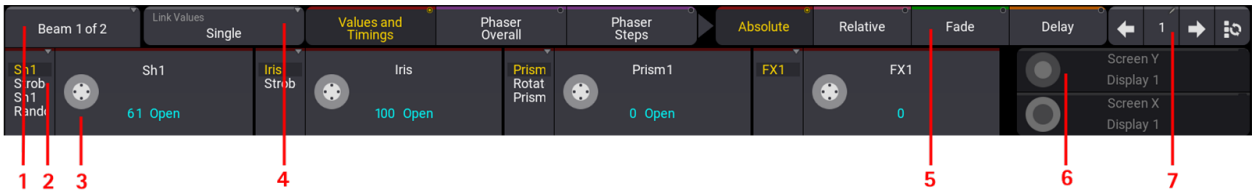
Subtopics

- **Encoder Toolbar**
- **Default Feature Group Control Bar**
- **Encoder Bar Pool**

1.8.8.1. Encoder Toolbar

Encoder Toolbar

The encoder toolbar represents the largest portion of the encoder bar. The encoder toolbar is context-sensitive and displays different sets of controls as different editors become active. The default set of controls in the encoder toolbar provides access to and display of attribute encoder information.



Encoder toolbar

Attribute controls available in the encoder toolbar include:

1. **Encoder Page:** This button displays the name of the current feature. It also indicates the total number of pages available in the current encoder bank.
   Tap to cycle through features or tap and swipe to open a pop-up menu with available features.
   Encoder Pages can be customized, see **Encoder Bar Pool**.
   For more information about features and feature groups, see the **Feature Group** topic.
2. **Channel function of the attribute:** These buttons display the current channel function of the attribute displayed directly to the right. They also display additional channel functions when available.
   Tap to cycle through subattributes or tap and swipe to open a pop-up menu with available subattributes.
3. **Attribute encoder display:** These buttons display the names of the attributes currently linked to the encoders, and show the values of those attributes.
   The center of the encoder icon includes an additional image, which changes depending upon the current resolution of the encoder. For more information on encoder resolution, see the **Encoder Resolution** topic.
   To open the calculator, tap the attribute name field or the attribute value field. For more information about attributes and subattributes, see the **Attribute Definitions** topic.
   For more information about the gesture for rotating a virtual encoder, see the **Gestures** topic.
4. **Link button:** In cases where the same value, timing, phaser, or resolution adjustments should apply to multiple attributes, the link button offers a few options for defining multiple, simultaneous attribute destinations. For more information about the link button, see the section below.
5. **Layer toolbar:** The default layer is absolute. Tap any layer button in the toolbar to access the desired layer on the attribute encoders. For more information about the layer toolbar, see the **section below**.

6. **Screen encoder:** If the screen encoder is enabled in the user profile settings of the current user, this area displays the basic functionality of the screen encoder. If the screen encoder is disabled in the current user profile, the functionality of the fifth dual encoder will be similar to that of the other four, and the display in this area will also be similar to the displays above the other four encoders. For more information about the screen encoder, see the **User Settings** topic.

	**Hint:**
	Setting Screen Encoder to **No** in the User Profile settings enables a fifth encoder for controlling attributes.

7. **Step bar:** The left and right arrows allow navigation between phaser steps in the programmer. The step number button displays the current phaser step in the programmer. Tap the step number button to open a calculator to choose a specific step. Tap the button to the right of the right-arrow to select all phaser steps. For more information about phaser steps, see the **Phasers** topic.

Link Button

The link button maintains different link settings for the different types of layers found in the layer toolbar as well as encoder resolution. The text in the link button will update automatically to show the link setting of the current layer type. The available link settings are:

- **Single:** Adjustments made on one encoder apply only to the attribute on that encoder.
- **Feature:** Adjustments made to one encoder within a feature apply simultaneously to all attributes in the feature.
- **AtFilter:** Adjustments made to one encoder apply simultaneously to all attributes currently enabled in the at filter.

Link Values

Value layers include:

- Absolute
- Relative

When a value layer is selected, tap Link Values to cycle through the link options or tap and swipe to open a pop-up menu with all of the available link options. The available link options for value layers include:

- Single
- Feature

Link Timing

Timing layers include:

- Fade
- Delay

When a timing layer is selected, tap Link Timing to cycle through the link options or tap and swipe to open a pop-up menu with all of the available link options. The available link options for timing layers include:

- Single
- Feature
- AtFilter
- **Active Only ():** If enabled, only attributes with active values in the programmer will get new values.
- **MultiStep Only ():** If enabled, new values are only applied to attributes with two or more active steps.

Link Phaser Overall

Phaser layers include:

- Speed
- SpeedMaster
- Phase
- Measure

Link Phaser Steps

- Accel
- Decel
- Transition
- Width

When a phaser layer is selected, tap Link Phaser to cycle through the link options or tap and swipe to open a pop-up menu with all of the available link options. The available link options for phaser layers include:

- Single
- Feature
- AtFilter
- Active Only
- MultiStep Only

Link Resolution

While the MA key is pressed and held, tap Link Resolution to cycle through the link options or tap and swipe to open a pop-up menu with all of the available link options. The available link options for encoder resolution include:

- Single
- FeatureGroup

Layer Toolbar

Presets and cues can store and recall multiple layers of data for each attribute. The layer toolbar provides access to all available layers. Layers are color-coded. Markers and text backgrounds using matching colors in the attribute encoder displays, encoder bank buttons, fixture sheet, and sequence sheet denote active or stored data for the corresponding layer. For more information about colors, see the **Colors** topic.

Tap a button in the layer toolbar to access data for the desired layer. The attribute encoders and any sheet with a layer selection set to auto will display the desired layer. For more information about value and timing layers, see the **What is the Programmer** topic. For more information about phaser layers, see the **Phasers** topic. For more information about using GridPosition to create MAgic presets, see the **Create New Presets** topic.

When the programmer is clear, the bars across the top of the attribute encoder displays and the channel function displays are gray. The selected layer group returns to **Values and Timings** and the **Absolute** layer. The layer selection in Phaser Overall and Phaser Steps stays as selected after clearing the programmer. As the selection changes, the bars across the top of any attributes and channel functions available in the current selection change color. The color coordinates with the color of the current layer in the layer toolbar.
The lights under the dual-encoders also follow the same behavior.

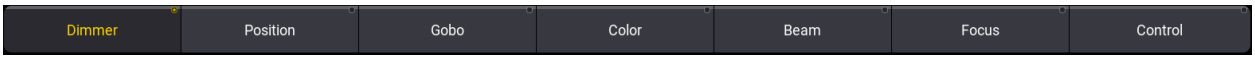
Hint:

The colors for each phaser-related layer can be changed in the color theme in the ColorGroup "ProgLayer". For more information, see **color theme**.

1.8.8.2. Default Feature Group Control Bar

The default feature group control bar appears along the top of the encoder bar. Each feature group used in the show file appears in the form of a radio button.

The feature group control bar is the MA default encoder bar. To create a customized Encoder bar, see the **Encoder Bar Pool** topic.



Feature group control bar with Dimmer selected

Adding fixtures, which use additional feature groups, to the patch automatically adds the necessary feature groups to the feature group control bar.

	Hint:
	When the programmer is clear, the bars across the top of the feature group buttons are grey. As the selection changes, the bars across the top of any feature groups available in the current selection change color. The color coordinates with the color of the current layer in the layer toolbar.

Select a Feature Group

To select a feature group in the control bar, tap the desired feature group. The radio button is enabled, and the encoder toolbar adjusts to display the attributes of the selected feature group.

To change the feature group using a shortcut, press and hold Preset + press one of the numbers from 1 to 9 on the numeric keypad. For example, to change to the color feature group, press Preset + 4.

Active Programmer Values

To activate the attributes of a feature group in the programmer, tap twice on the desired feature group. The information activates and a colored marker appears on the feature group button. To deactivate, tap the feature group once again.

	Important:
	The color of the marker or markers on the feature group button corresponds to the active layer or layers of information. These markers will appear as long as there are active values, no matter how the values were activated.
	Hint:
	To activate information in a specific layer, tap the desired layer in the layer control bar in the encoder toolbar before tapping the feature group.

Feature Group as the Object of a Command

Many function keywords (including On, Off, Park, Unpark, Remove, Release, Stomp, and Default) can execute their commands using feature groups as their destination objects.

Example

To remove all dimmer values within the current selection from the programmer:

- Press Off and tap Dimmer.

The dimmer values of the current selection are removed from the programmer.

1.8.8.3. Encoder Bar Pool

The Encoder Bar Pool can be used to create customizable encoder bars. The customization can be made in an editor, for example, defining the functionality per encoder, and defining the number of encoder banks and their encoder objects.

To always have a backup with the original feature group structure, the encoder bar pool object 1 is set as default and can not be deleted. For more information, read the **Feature Group Control Bar** topic.

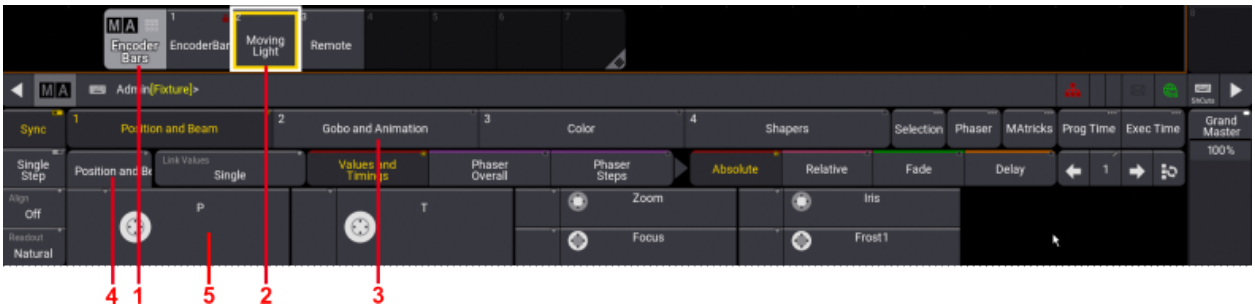
	Hint:
	Copy the default encoder bar to an empty pool object to use it as a template. Simply press Copy, tap the default encoder bar pool object and then tap an empty pool object.

When tapping on a pool object, the Encoder Bar and the Encoder Bar window change the user interface according to the settings previously made for those pools.

- To address Encoder pages directly, use the **EncoderBank Keyword**.
- To switch between encoder bars, use the **EncoderBar Keyword**.

Encoder Bar Structure

The following graphic and the numbered list displays the basic structure and terminology of the encoder bar:



1. **Encoder Bar Pool:** This pool contains all the encoder bar pool objects
2. **Encoder Bar Pool Object:** A pool can have several pool objects. For more information on how to setup pool objects, see **Pool Windows.**
3. **Encoder Bank Button:** There can be several encoder banks inside an encoder bar pool object. Allows to link to Special dialog tap.
4. **Encoder Page:** Each encoder bank can have several encoder pages. Encoders get arranged in Encoder Pages.
5. **Encoder:** An encoder page can contain up to five dual encoders.

Edit the Encoder Bar Pool

To open the Encoder Bars pool:

1. Open the Add Window dialog.
2. Tap Pools and then tap Encoder Bars. The Encoder Bars pool is open.

To create a new Encoder Bar Pool object:

- Press Edit and tap an empty encoder bar pool object. The Encoder Bar editor opens.

To copy the default encoder bar pool object to an empty pool object:

- Press Copy , tab Encoder BarPool Object 1 and then tap an empty encoder bar pool object in the Encoder Bars Pool.

To assign an Encoder Bar pool object to a ViewButton, see **View Bar and View Buttons**.

	Hint:
	To switch between different encoder bar pool objects, press, for example, Preset + 2.

Create New Encoder Bar Pool Object

To create a new Encoder Bar Pool object:

1. Press Edit and tap an empty encoder bar pool object 3.
2. The Encoder Bar editor opens. For more information, see **Edit Encoder Bar Pool** above.

Enable Settings in the title bar to see the five horizontal buttons on top of the editor:

- **Name:** Opens the name field.
- **Scribble:** Set a scribble. For more information, see **Scribbles**.
- **Appearance:** Opens the appearance dropdown. For more information, see **Appearances**.
- **Note:** Opens the note pop-up. For more information, see **Notes**.
- **Lock:** Locks the pool object. For more information, see **Lock and Unlock Pool Objects**.

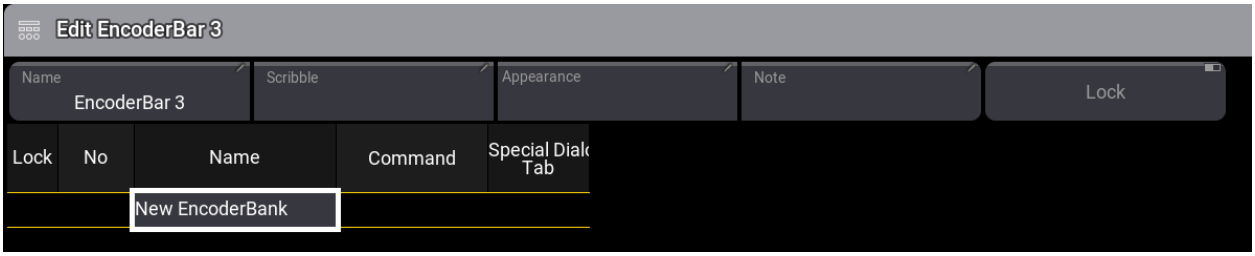
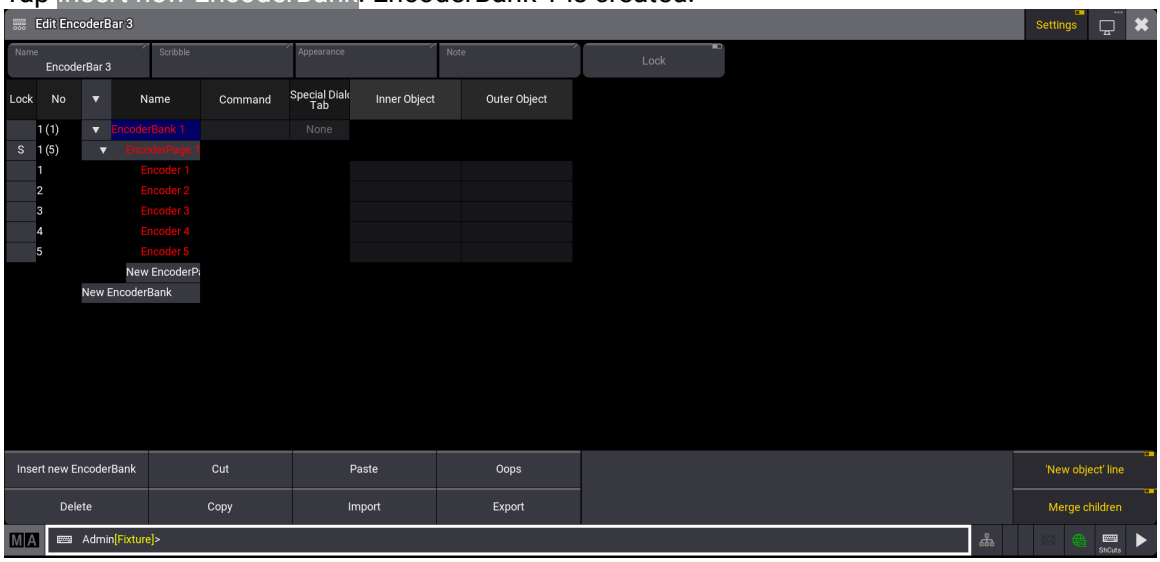


Image section of the Encoder Bar editor

To create a new Encoder Bank:

1. Tap Insert new EncoderBank. EncoderBank 1 is created.



Encoder Bar Pool

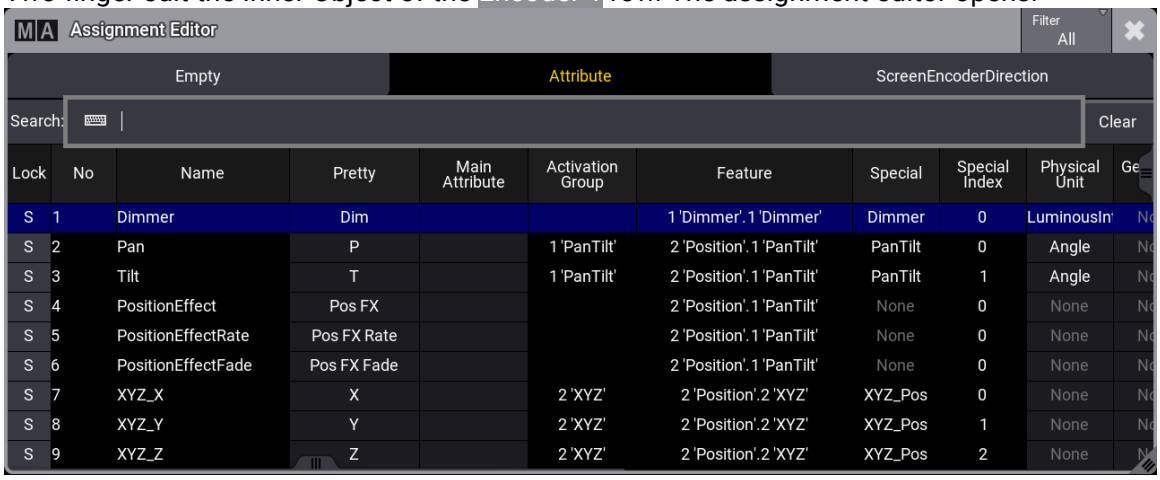
2. Two-finger Edit EncoderBank 1. The keyboard opens.
3. Rename the encoder bank, for example, Dimmer. Press Please.
4. Repeat steps 2-3 and rename EncoderPage 1, for example, to Dim. A new EncoderBank is created.

	Hint:
	Red text in the editor means that no fixture attribute is selected and the 'empty' encoder is not displayed in the user interface.

Assign Functions to Encoders

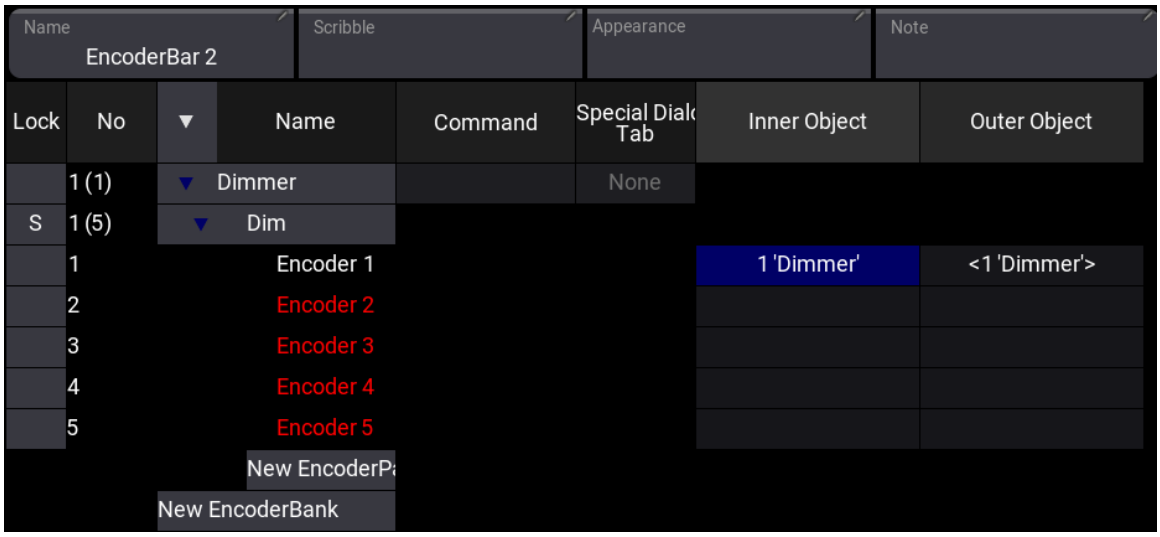
To assign functions to an encoder object:

1. Two-finger edit the Inner Object of the Encoder 1 row. The assignment editor opens.



Assignment Editor

2. Select an Attribute, for example, Dimmer. The assignment editor closes and the Inner Object is assigned to Dimmer.



Encoder Bar Pool Object

	Hint:
	Selecting an attribute for the inner object automatically sets the same attribute for the outer object (<Attribute>) and vice versa.

Using the Assignment Editor

To filter the assignment editor, tap and hold Filter in the Assignment Editor. A drop-down menu opens with the following options.

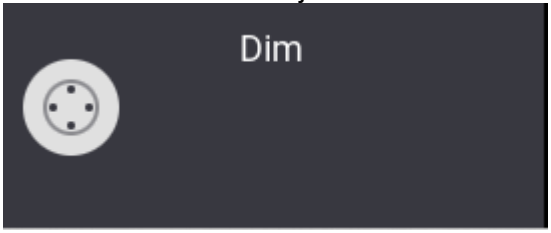
- **All:** Shows all attributes.
- **Used:** Shows all attributes used in the show.
- **Unused:** Shows all attributes not used in the show.
- **Selection:** Shows the attributes of the selected fixtures.

Another way to look for attributes is to use the search bar in the assignment editor.

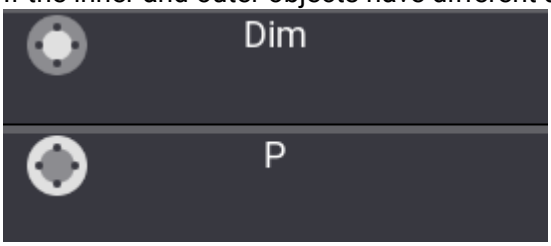
To clear the assignment of an encoder, tab Empty in the assignment editor.

To use encoders to scroll through the screens, tap ScreenEncoderDirection and select Screen Y for vertical scrolling and Screen X for horizontal scrolling.

If the inner and outer objects have the same attribute, the encoder is displayed in white:



If the inner and outer objects have different attributes selected, the encoder is separated:



Encoder Bank Command

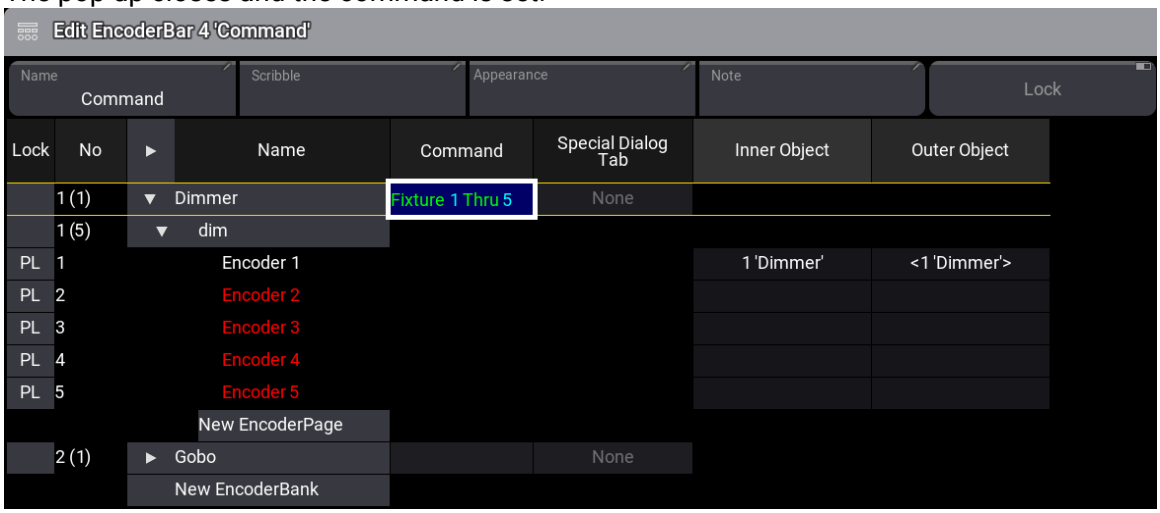
It is possible to trigger a command when entering a pre-defined encoder bank.

See the video below to get an overview of this feature:

To pre-define an encoder bank command:

1. Press Edit and tap an encoder bar pool object.
2. Two-finger edit the Command row next to an encoder bank. A pop-up opens.

3. Type a command, for example, **Fixture 1 Thru 5** and then press Please.
4. The pop-up closes and the command is set.



Encoder Bar Editor - Set a command

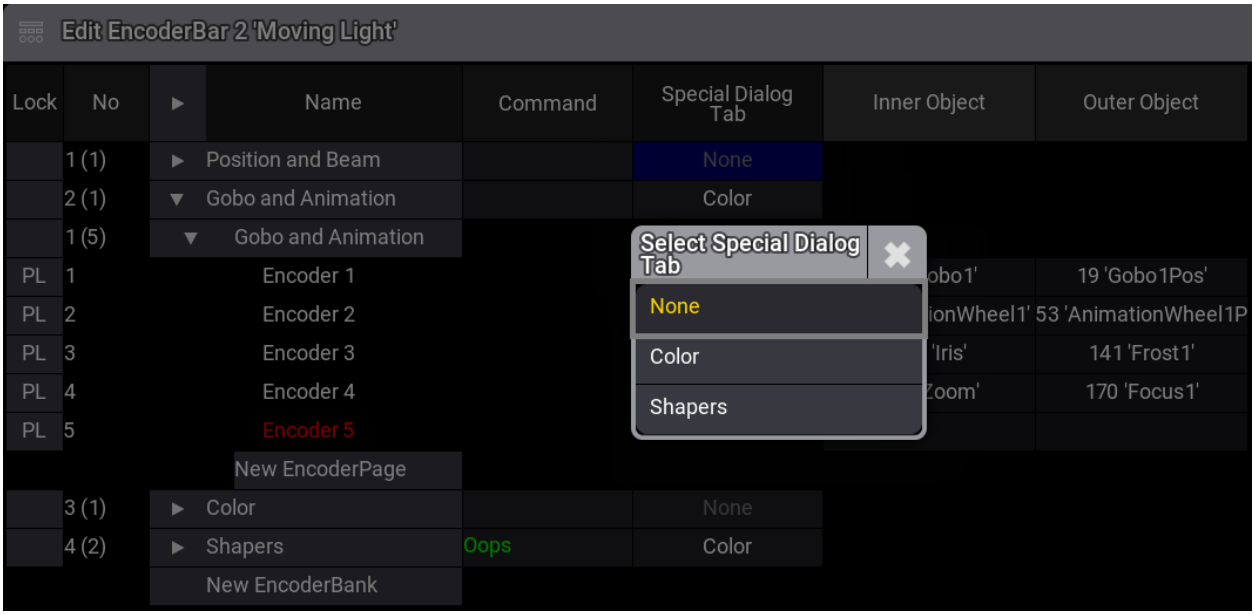
	Hint:
	If an encoder bank is already active, tapping the active encoder bank again will not execute the command.

- **Preview Variables:** For more information, see **Command Editor**.
- **Resolve Executor Assignments**: For more information, see **User Profile Settings**.

Special Dialog Tab

To automatically switch the Special Dialog to a specific tab when entering the defined encoder bank:

1. Two-finger edit Special Dialog Tab in the grid. A dropdown menu opens.
2. Select a Special Dialog tab, for example, Color. The dropdown menu closes.
3. Open the Special Dialog window.
4. Enable Link Encoder Bank in the Special Dialog settings. For more information about this setting, see **Special Dialog**.
5. Enter the previous defined encoder bank, for example, Gobo and Animation. The Special Dialog tab switches to Color.



Encoder Bar Pool - Special Dialog Tab

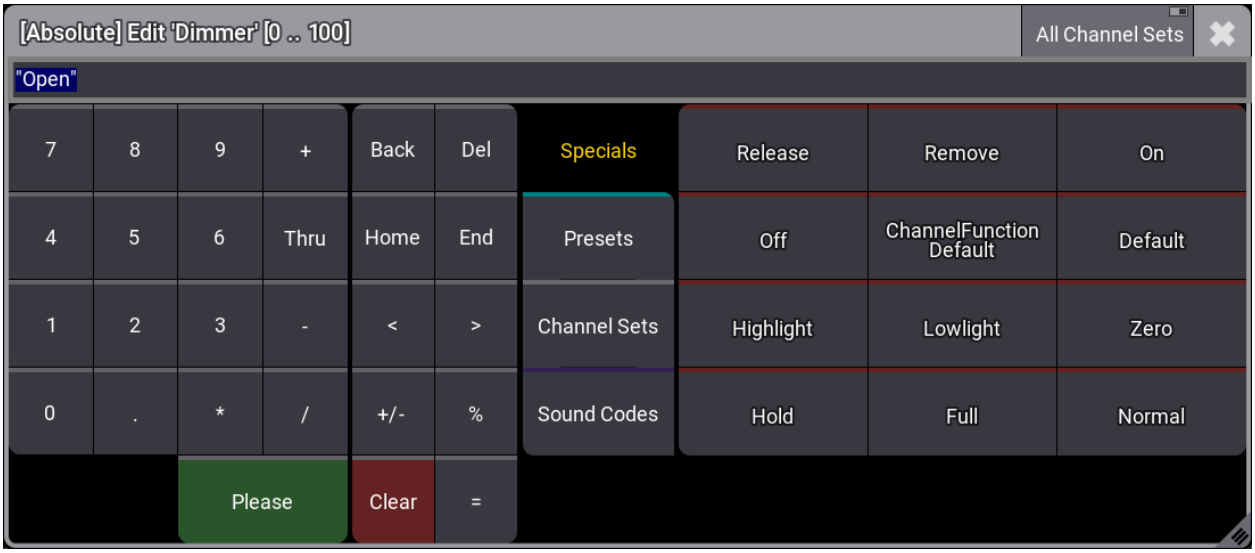
1.8.9. Calculator

When editing a field that accepts numeric values, the calculator appears. This includes pressing one of the dual encoders while it is displaying an editable value.

The calculator is a dynamic window, and its appearance changes depending upon the edited item and the type of value to be entered.

Example

- Edit a dimmer attribute



The calculator editing a dimmer attribute

The title bar displays several useful pieces of information about what the calculator is currently editing:

- **Layer:** If applicable, the first set of brackets contains the value layer.
- **Attribute** or **parameter:** The name of the attribute or parameter appears after the word "Edit."
- **Value Range:** The last set of brackets contains the allowed range of input values.

All Channel Sets: Shows the channel sets of all channel functions per attribute. For more information, see **Encoder Resolution**.

: A filter icon in the title bar indicates a filter selection.

Input Field

The input field appears below the title bar. When the calculator opens, the input field displays the current value of the edited object. The text of this value is completely selected so that any immediate entry will replace the existing value. Alternatively, the text can be deselected and retained as part of the entry.

Entering values with a percentage sign (%) in front of the value scales the former value by the new value. For example, the old value is 50, and entering % 50 results in a value of 25.

Standard Buttons

The following buttons appear in all calculators.

Number Pad

The calculator displays a number pad on the left. These buttons share the same functions in the calculator as the number keypad in the command section of the console.

Function Buttons

Function buttons appear to the right of the number pad. These buttons include:

- Back: Tap to delete characters to the left of the cursor.
- Del: Tap to delete characters to the right of the cursor.
- Home: Tap to set the cursor at the beginning of the input field.
- End: Tap to set the cursor at the end of the input field.
- <: Tap to move the cursor to the left.
- >: Tap to move the cursor to the right.
- +/-: Tap to insert a negative or a positive value. For more information see **- [Minus] Key** or the **+ [Plus] Key**.
- %: Tap to enter the percent sign.
- =: Tap to enter the equals sign.
- Clear: Tap to delete the entire entry.
- Please: Tap to confirm and apply the value. The calculator closes.

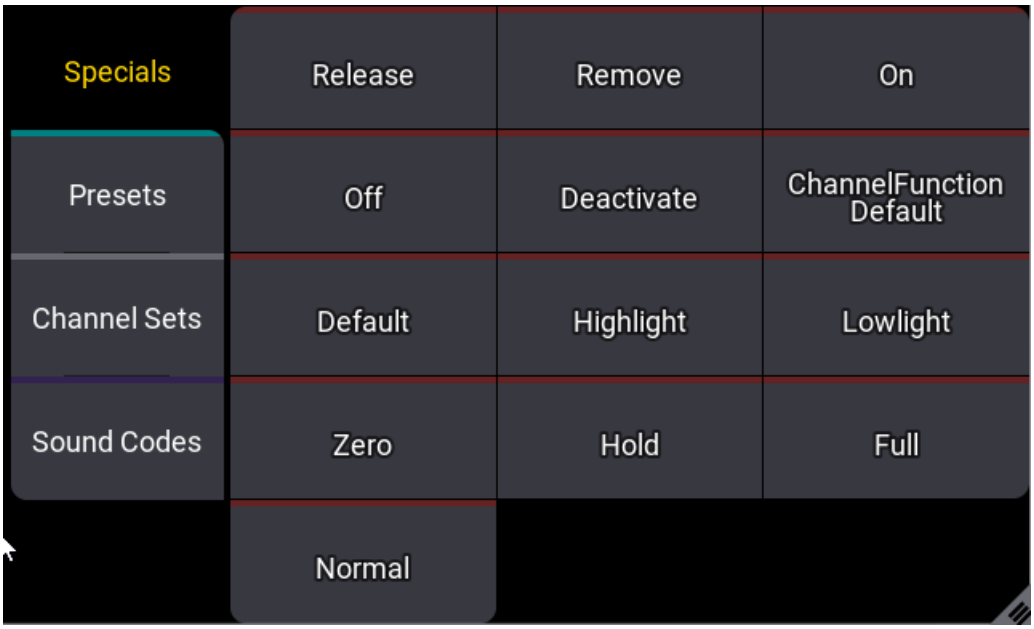
	**Hint:**
	Input from a standard keyboard, whether integrated under the console armrest or connected via USB, also populates the input field of the calculator. With one exception, keyboard shortcuts are disabled when entering text into a calculator. The exception is that T will enter the Thru keyword.
	**Hint:**
	When the input readout is set to one of the hexadecimal options, an additional row of buttons appears below the number pad, allowing entries of hexadecimal values A, B, C, D, E, and F.

Additional Input Options

Four tabs of additional input options appear to the right of the function buttons.

Specials

To open the special buttons tab, tap Specials.



Specials in the calculator

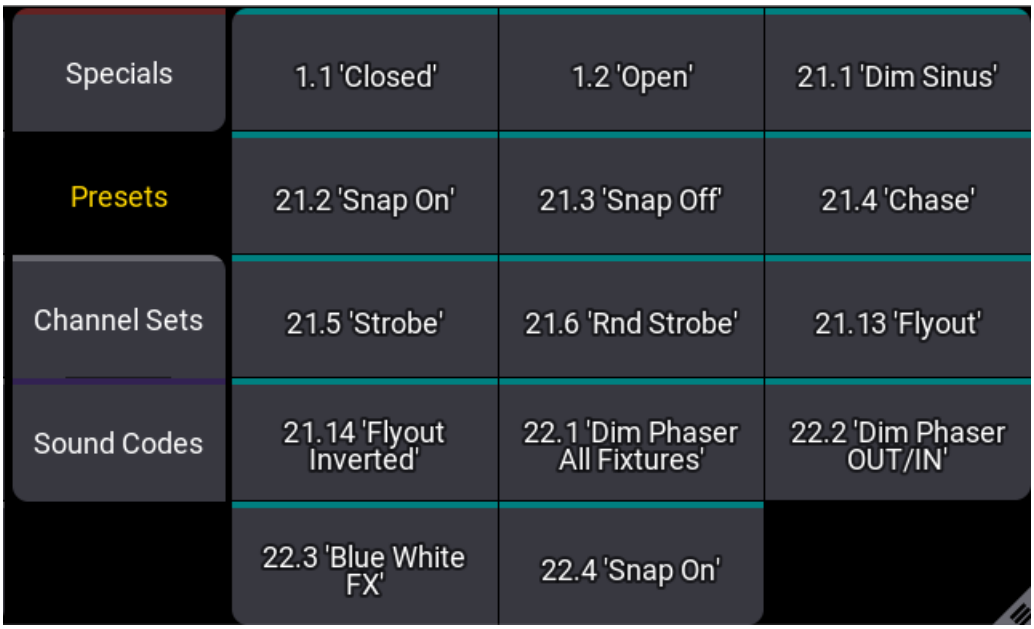
Use these buttons to enter special commands for attribute values:

- Release: Tap to enter a release value. For more information see the **Release Keyword**.
- Remove: Tap to enter a remove value. For more information see the **Remove Keyword**.
- On: Tap to activate values in the programmer without changing the value.
- Off: Tap to clear the values in the programmer.
- Deactivate: Tap to deactivate active data in the programmer. For more information see the **Deactivate Keyword**.
- ChannelFunctionDefault: Tap to set the attribute to the default value of the selected channel function. For more information see **ChannelFunctionDefault Keyword**.
- Default: Tap to enter the default value defined for the fixture type. For more information see the **Default Keyword**.
- Highlight: Tap to load the highlight value of the selected fixture into the programmer. For more information see the **Highlight Keyword**.
- Lowlight: Tap to load the lowlight value of the selected fixture into the programmer. For more information see the **Lowlight Keyword**.
- Zero: Tap to set the values to 0. For more information see the **Zero Keyword**.
- Hold: Tap to prevent an MIB action while the dimmer is closed. For more information see the **Hold Keyword**.
- Full: Tap to set values to 100%. For more information see the **Full Keyword**.
- Normal: Tap to set the normal value stored in the user profile. For more information see the **Normal Keyword**.

The at overlay in the control bar also offers quick access to limited versions of these special values. For more information, see the **Control Bar** topic.

Presets

To open the presets tab, tap Presets.



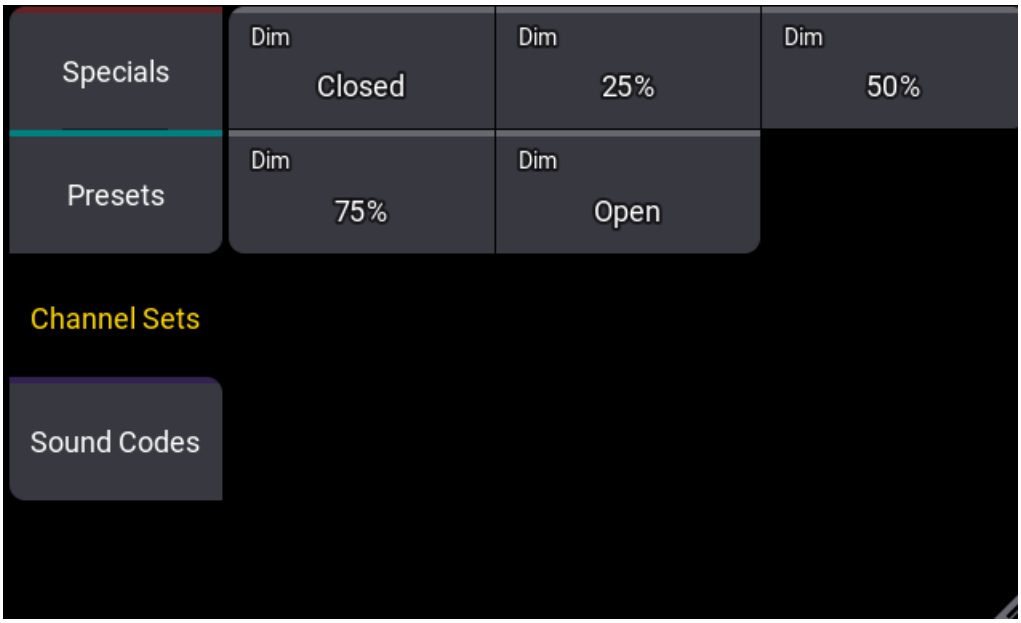
Presets in the calculator

Use these buttons to call presets for attribute values. This tab displays any preset from any preset pool in the currently selected data pool, which contains data that can be applied to the attribute currently edited by the calculator. Each button displays the preset number, using the format: [Feature Group].[ID], as well as the preset label. For more information on presets, see the **Presets** topic.

	Important:
	Only the buttons in the preset tab recall presets in the calculator. All other value sources in the calculator produce unlinked values that may be more difficult to reproduce or update.

Channel Sets

To open the channel sets tab, tap Channel Sets.



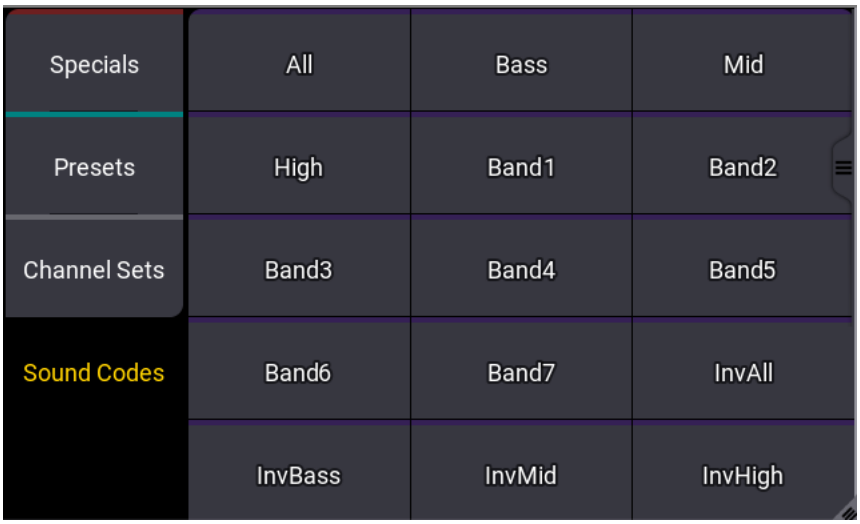
Channel Sets in the calculator

Use these buttons to enter attribute values based on channel sets defined for the fixture type. The channel sets displayed in this tab are the same as those displayed in the smart window. For more information on the smart window, see the **Smart View** topic.

	Hint:
	When editing attributes with multiple channel functions, a button for each channel function appears between the title bar and the input field of the calculator. Selecting a channel function shows only the relevant channel sets within the channel sets tab.

Sound Codes

To open the sound codes tab, tap Sound Codes.



Sound codes in the calculator

Use these buttons to link the attribute values to an incoming audio signal. Values can reference the total volume of the audio input or they can reference smaller frequency bands within the signal. The available options include:

- All: Use for the total volume of the incoming signal.
- Bass, Mid, and High: Use to reference one of three broad frequency bands.
- Band1 through Band7: Use to reference one of seven narrower frequency bands.
- Inv...: Use to reference the inverse value of any of the above options.

	Hint:
	Use the SoundIn and SoundFade masters to adjust how the console reacts to incoming audio signals.

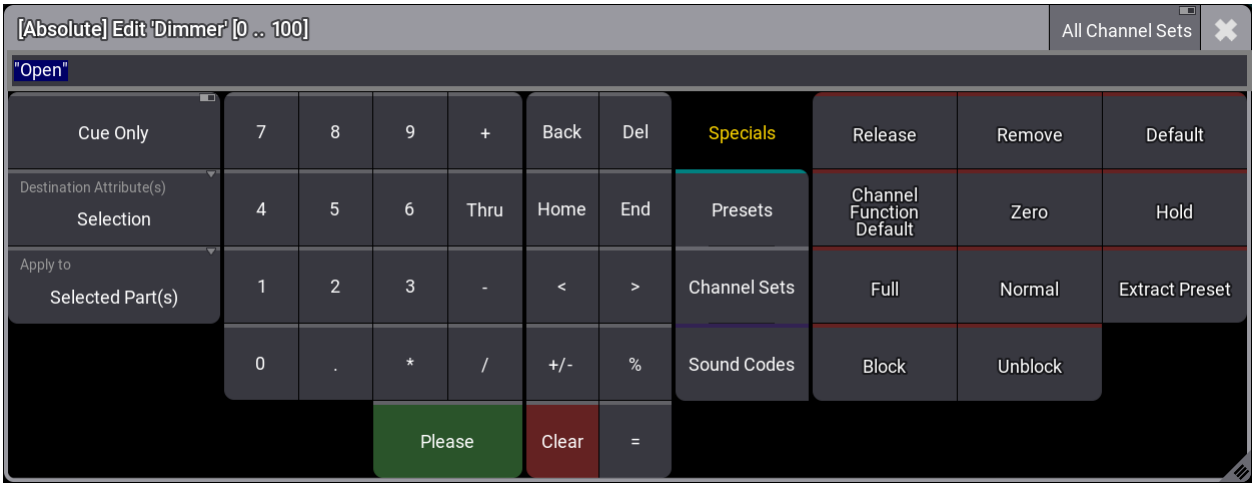
For more information about using sound input, see the **Sound Window** topic.

Track Sheet Calculator

When editing an attribute within the sequence sheet with the track sheet option enabled, the calculator appears with additional options to the left of the number pad. To edit attributes in the track sheet, drag the cursor across the desired attributes and up or down over the desired cues or cue parts. Then, use a 2-finger tap to edit the highlighted cells. For more information on the track sheet mode of the sequence sheet, see **Track Sheet Mode**.

Example

- Edit a dimmer attribute in the track sheet



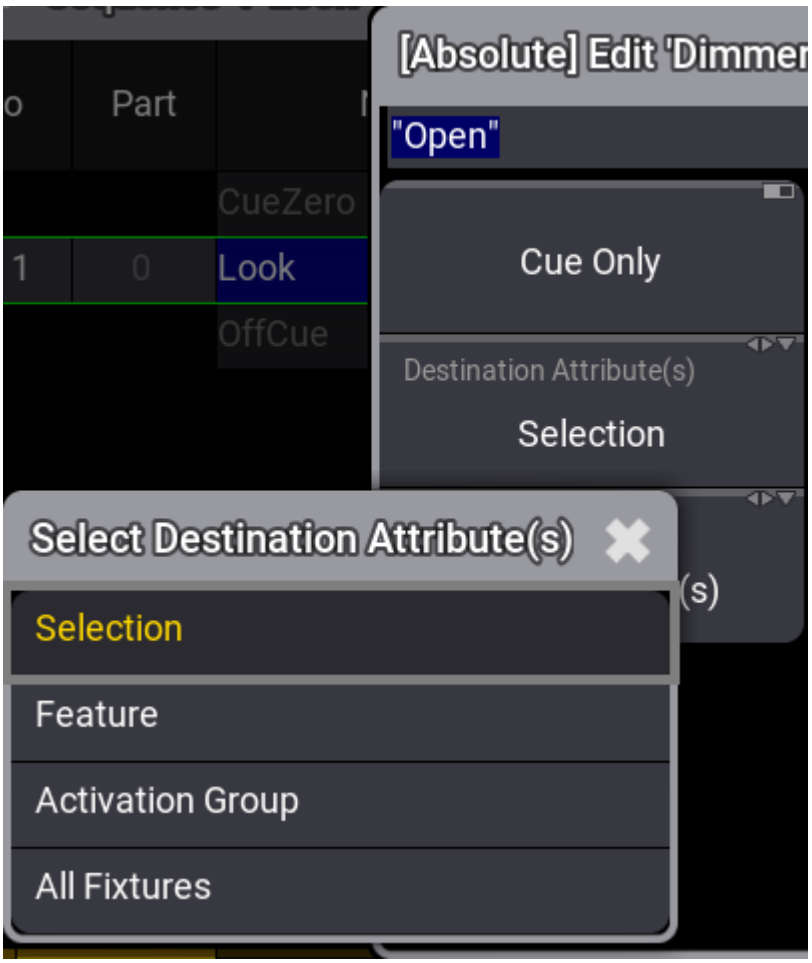
Calculator editing a dimmer in the track sheet

Cue Only

The cue only toggle appears at the top of the additional options to the left of the number pad. Toggle Cue Only off to allow changes to track forward. Toggle Cue Only on to prevent changes from tracking beyond the range of cues defined by the edit options. For more information on cue only, see the **Store Cues** topic.

Destination Attribute(s)

Tap Destination Attribute(s) to cycle through the available attribute destination options, or tap and swipe to see a menu of all available attribute destination options.



Attribute destination options

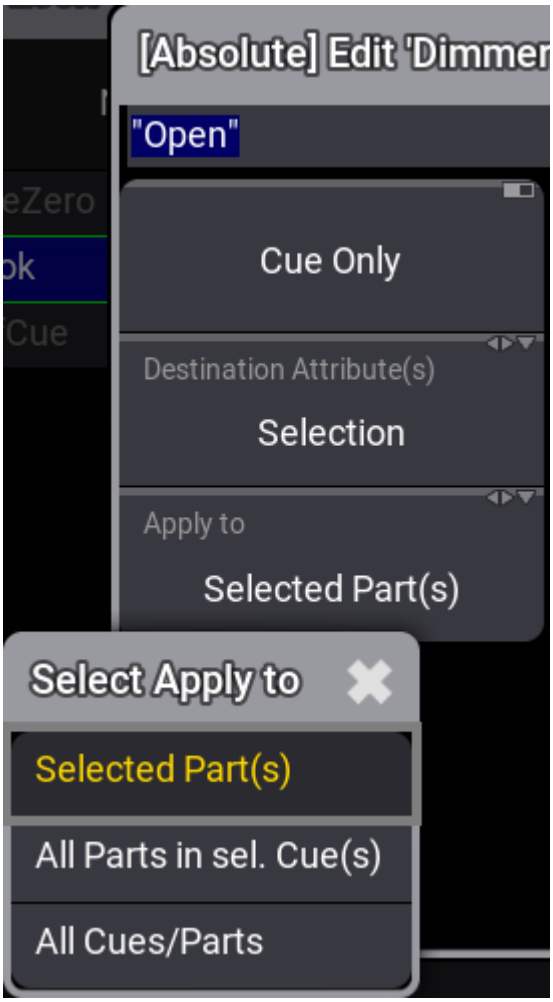
The select destination attributes menu includes the following options:

- Selection: Edits will only apply to the selected attribute cells.
- Feature: Edits will apply to all attributes within the feature group of the selected attribute cells, staying within the selected fixtures.
- Activation Group: Edits will apply to all attributes within the activation group of the selected attribute cells, staying within the selected fixtures.
- All Fixtures: Edits will only apply to all attributes of all fixtures within the cue part.

	Hint:
	Think of this option as expanding the effect of the edit horizontally within the track sheet.

Apply to

Tap Apply to to cycle through the available cue and part options, or tap and swipe to see a menu of all available cue and part options.



Apply to options

The apply to menu includes the following options:

- Selected Part(s): Edits will only apply to the selected cue part cells.
- All Parts in sel. Cue(s): Edits will apply to all parts within the selected cues.
- All Cues/Parts: Edits will apply to all parts of all cues within the sequence.

	Hint:
	Think of this option as expanding the effect of the edit vertically within the track sheet.

Time Calculator

When editing times, the calculator appears with different readout options and additional buttons.

Example

- Edit the wait time for a line in a macro

Calculator editing the wait time for a line in a macro

Additional buttons appear below the number pad. These buttons allow for quick entry of time units:

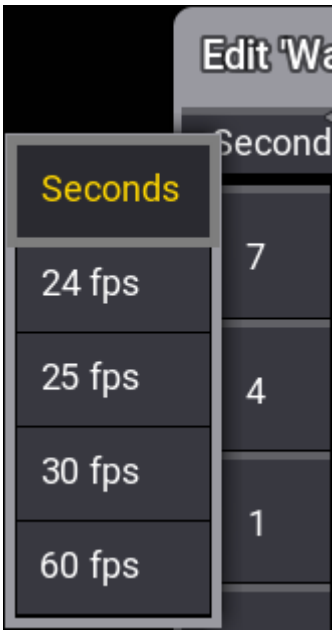
- D: Day
- H: Hour
- M: Minute
- S: Second
- F: Frame

Additional Input Options

Additional input options appear to the right of the function buttons. When editing a cell that can also contain triggers, this area displays trigger options.

Time Readouts

The readout selection appears to the left of the input bar. Tap the readout to cycle through the available readout options, or tap and swipe to see a menu of all available readout options.



Time readout options

1.8.10. Playback Bar

The playback bar displays the objects currently assigned to the corresponding executors.

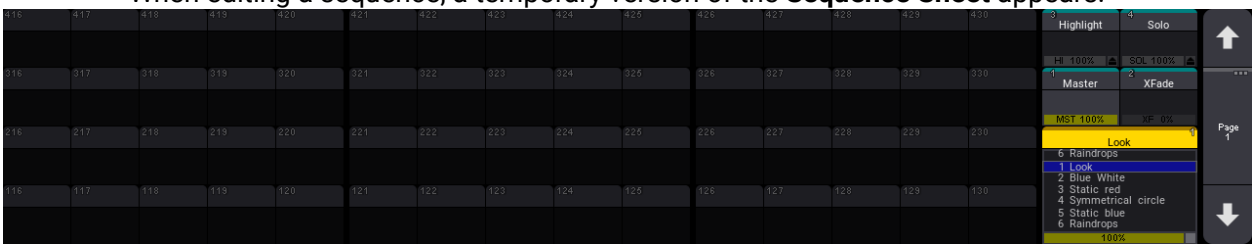
The bar includes:

- The playback status of each object
- The available controls assigned to the executor keys, faders, and knobs
- In some cases:
  - **Master Area**
  - **Custom Area**
  - Page navigation controls
  - Xkeys

Playback bar with custom area and page navigation

- To open the **Assign Menu** for an executor, tap the display of that executor in the playback bar.
  For more information about the **Assign Menu**, see the **Assign Object to an Executor** topic.

- To edit an object, tap and hold the display of that object in the playback bar.
  When editing a sequence, a temporary version of the **Sequence Sheet** appears.



Playback bar with master area and page navigation

The playback bar appears permanently on some screens and optionally on other screens.
To show or hide the playback bar on a screen, tap Show Playback Bar in the **Configure Display** pop-up.

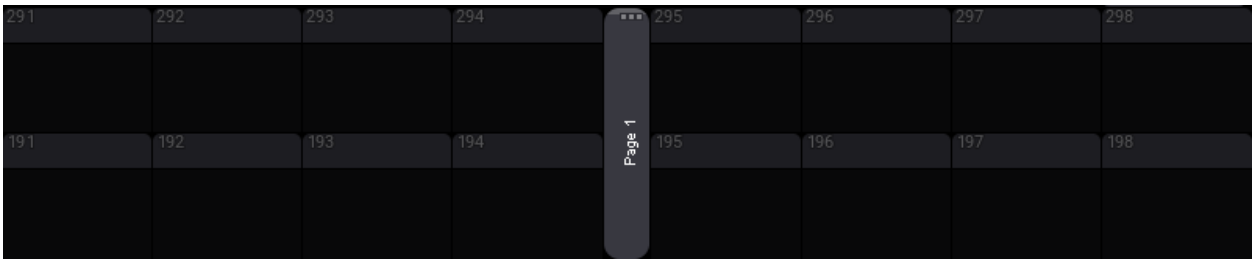
Depending on where an instance of the playback bar appears, the bar contains different information.

The following table provides an overview of which version of the playback bar appears on which screen:

Additional elements in the playback bar	Screen
• Assignment and status of the master area • Page navigation	9 (permanent) 2 (optional)
• Assignment and status of the custom area • Page navigation	10 (permanent) 11 (permanent) 12 (permanent)

Additional elements in the playback bar	Screen
	3 (optional)
	4 (optional)
	5 (optional)
• Assignment and status of the Xkeys	7 (optional)

For more information about screen allocation, see the **Screen Allocation** topic.



Playback bar with Xkeys.

A window showing the assignment and status of the Xkeys is available under the **Common** and **More** tabs in the **Add Window** pop-up. Additionally, a version of the playback bar is also available as a window under the **Common** and **More** tab in the **Add Window p**op-up. The **Playback** window does not include the custom area, master area, or the same page navigation controls as the **Playback Bar**. For more information on adding windows, see the **Add Windows** topic.

	Important:
	For more information about executors, page navigation, and the playback window and settings, see the **Executors** topic.

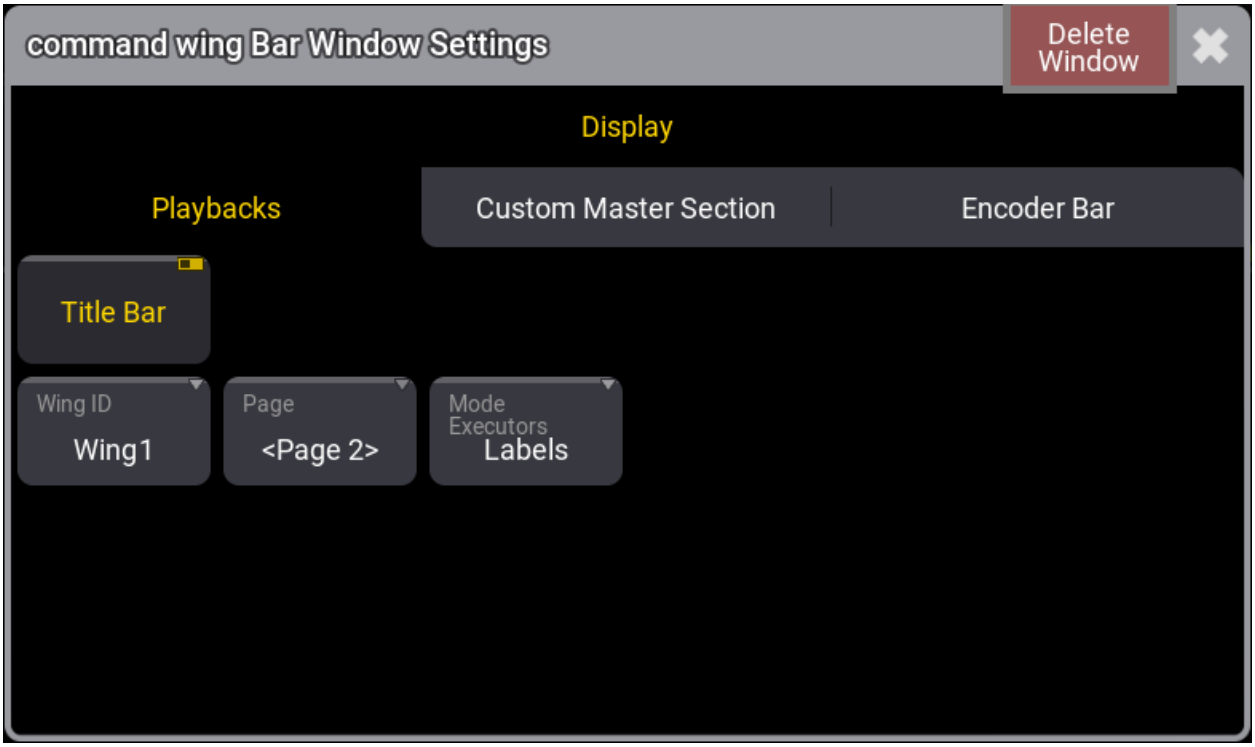
1.8.11. command wing Bar

The **command wing Bar** combines a small version of the **Playback Bar**, including two sections of executors and the **Master Area**, with a compressed version of the **Encoder Bar**. This combined bar is helpful for optimizing screen space and minimizing the views necessary to move between programming and playback, especially when using the grandMA3 onPC software. The executors displayed on the playback side of the bar coincide with the executor sections available on the grandMA3 onPC command wing and command wing XT.

In the grandMA3 onPC software, tap the icon in the **Control Bar** to toggle the **Encoder Bar** to the **command wing Bar**. **The command wing Bar** is also available as a window under the More and All tabs in the **Add Window** pop-up. For more information on adding windows, see the **Add Window** topic.

command wing Bar

Tap MA in the upper-left corner of the **command wing Bar Window** to access the **command wing Bar Window Settings** pop-up.



command wing Bar Window Settings pop-up

This pop-up includes the following tabs:

Playbacks:

- **Title Bar:**
  This shows or hides the window's title bar. It is On by default. If it is Off, then the title bar can be shown temporarily by pressing both MA keys in the control area. In grandMA3 onPC, the title bar can be temporarily shown by pressing Ctrl + Alt on Windows and Ctrl + Option on Mac.
- **Wing ID:**
  Defines which wing the window displays. Tap this setting to open a small **Select WingID** pop-up where the desired wing can be selected.
- **Page:**
  It is used to change which executor page the window relates to.
- **Mode Executors:**
  Tap to toggle the display of the executors in the window between Labels and Hardware Buttons.

Custom Master Section:

- **Title Bar:**
  This shows or hides the window's title bar. It is On by default. If it is Off, then the title bar can be shown temporarily by pressing both MA keys in the control area. In grandMA3 onPC, the title bar can be temporarily shown by pressing Ctrl + Alt on Windows and Ctrl + Option on Mac.
- **Master Section:**
  This toggle button hides or shows the Master Section.
- **Mode Masters:**
  Tap to toggle the display of the special masters in the window between Labels and Hardware Buttons.

Encoder Bar:

This tab includes the same settings as the encoder bar window settings. For more information on this, see **Encoder Bar Window Settings**.

1.8.12. Colors

grandMA3 uses several colors to provide information about states of values and their origin. There are two editable color themes included in the software, and it is possible to create, import, and export additional color themes. The following topics give an overview of the colors defined in the default color theme, as well as an introduction to the color theme editor.

	**Important:**
	Use caution when editing the color theme, as it can make information on the screens difficult or impossible to read; for example, selecting the same color for both text and the background behind the text.
	**Hint:**
	MA + MA + Clear activates the default color theme and resets all values of that theme to the manufacturer defaults.

Subtopics

- **System Colors**
- **Markers**
- **Color Theme**
- **Compare Tool**

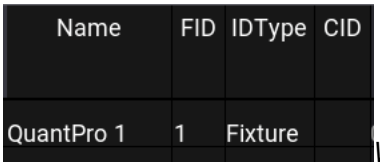
1.8.12.1. System Colors

	Important:
	The colors described here are included in the default color theme. For more information about alternate color themes, see the **Color theme** topic.

Sheet Colors for Fixture Names and IDs

The fixture names and numbers in sheets have several different color combinations.

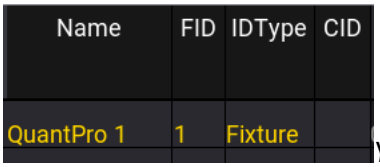
White:



White names and numbers with a dark gray background

White fixture names and IDs on a dark gray background indicate fixtures that are not selected. It is the default color combination.

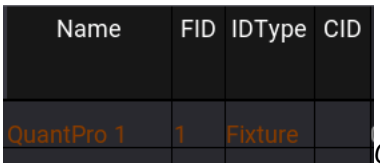
Yellow:



Yellow names and numbers on a medium gray background

Yellow fixture names and IDs on a medium gray background indicate fixtures that are selected.

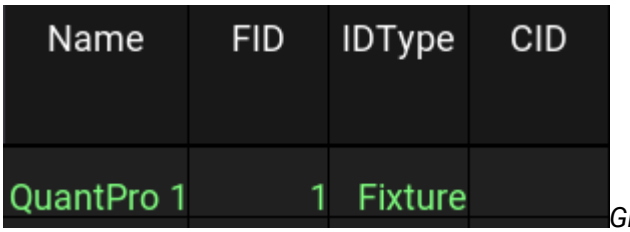
Dim Orange:



Orange names and numbers on a medium gray background

Orange fixture names and IDs on a medium gray background indicate fixtures that are in the current selection, but not the current sub-selection. For example, multiple fixtures are selected and Next has been pressed.

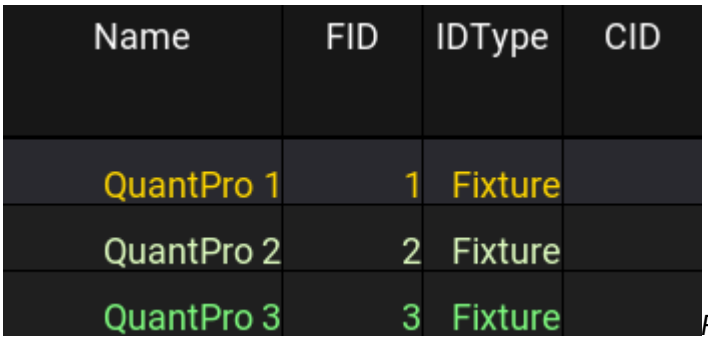
Green:



Green names and numbers on a medium gray background

Green fixture names and IDs on a medium gray background indicate fixtures with inverted encoder control of at least one attribute, based on selection and MAtricks settings. For more information about the MAtricks, see the **MAtricks and Shuffle** topic.

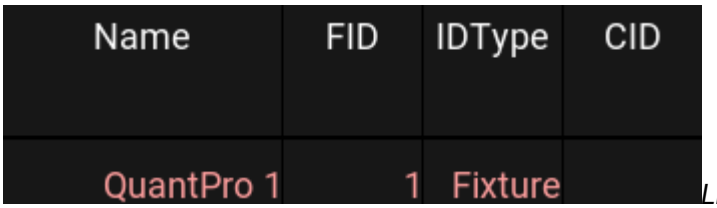
Light Green:



Fixture 2 appears with light green name and number on a medium gray background

Light green fixture names and IDs on a medium gray background indicate edge fixtures, based on selection and MAtricks settings. Edge fixtures do not follow all values. For example, when creating a mirrored circle with three fixtures and two wings, the edge fixture in the center will only tilt, but not pan. This is done in order to keep the symmetry. For more information about the MAtricks, see the **MAtricks and Shuffle** topic.

Fading between Yellow and Light Red:



Light red names and numbers on a medium gray background

Fixture names and IDs fading between yellow and light red on a medium gray background indicate at least one multipatch instance of this primary fixture is selected. For more information about the multipatch fixtures, see the **Add Multipatch Fixtures** topic.

Sheet Colors for Absolute Attribute Values

The cells showing attribute values in the fixture sheet and sequence sheet, with the track sheet mode enabled, have several combinations of text and background colors.

Black:

Name	FID	IDType	CID	RGB			
				R	G	B	W
QuantPro 1	1	Fixture		100	100	100	

Black background for on the W attribute

There is no object to be edited. In this case, this fixture does not include a white color mixing attribute.

Light gray text with a gray background:

Name	FID	IDType	CID	RGB			
				R	G	B	W
QuantPro 1	1	Fixture		100	100	100	

Light gray text with a gray background on R, G, and B attributes

The values of these attributes are at their default levels.

Red text with gray background:

Name	FID	IDType	CID	RGB			
				R	G	B	W
QuantPro 1	1	Fixture		100	100	100	

Red text with gray background on R, G, and B attributes

These attributes have values in the programmer, but they are not active. By default, they will not be included in any store or update actions.

White text with a red background:

Name	FID	IDType	CID	RGB			
				R	G	B	W
QuantPro 1	1	Fixture		100	100	100	

White text with a red background on R, G, and B attributes

These attributes have active values in the programmer. By default, they will be included in any store or update actions.

Cyan text with a gray background:

Name	FID	IDType	CID	RGB			
				R	G	B	W
QuantPro 1	1	Fixture		100	100	100	

Cyan text with a gray background on R, G, and B attributes

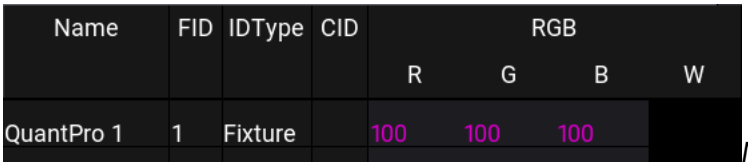
These attributes have new values in the current cue outputting from the selected sequence. A dimmer value displayed with cyan text indicates a new, higher value in the current cue outputting from the selected sequence.

Green text with a gray background:

Green text with a gray background on R, G, and B attributes

A dimmer value displayed with green text indicates a new, lower value in the current cue outputting from the selected sequence. This color is only used for descending dimmer values.

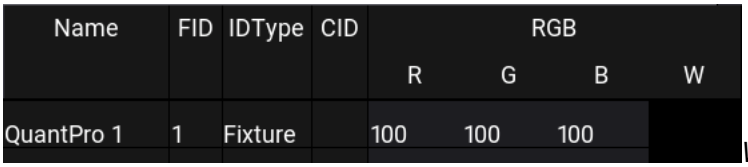
Magenta text with a gray background:



Magenta text with a gray background on R, G, and B attributes

These attributes have tracked values in the current cue outputting from the selected sequence.

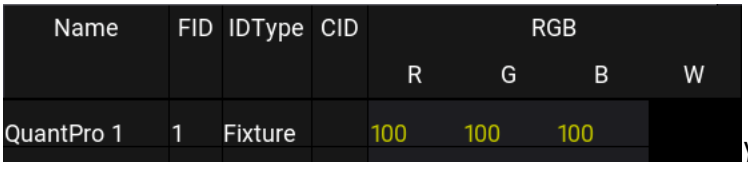
White text with a gray background:



White text with a gray background on R, G, and B attributes

These attributes have blocked values in the current cue outputting from the selected sequence.

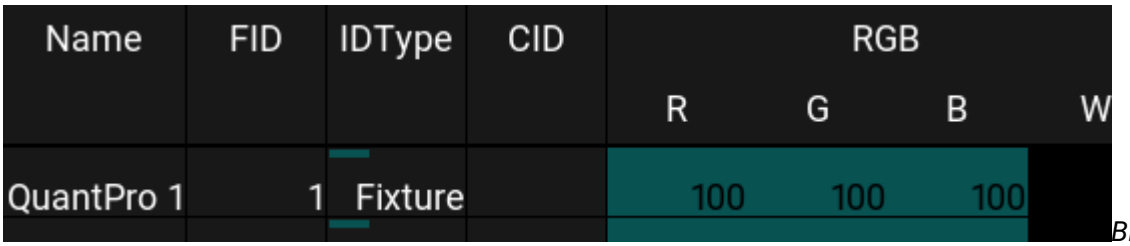
Yellow text with a gray background:



Yellow text with a gray background on R, G, and B attributes

These attributes have new or blocked values in the current cue outputting from a sequence that is not selected. This color combination does not appear in the sequence sheet.

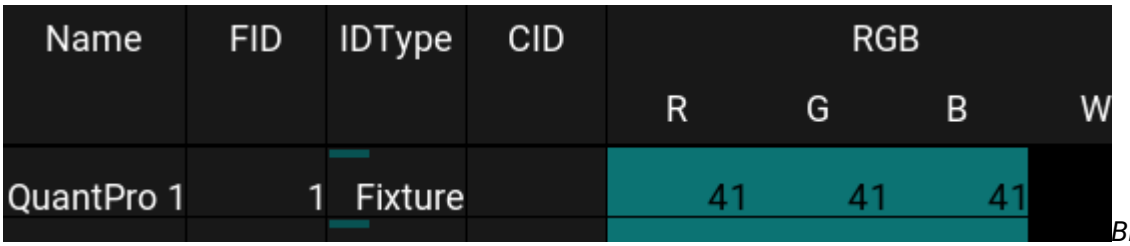
Black text with a deep sea green background:



Black text with a deep sea green background on R, G, and B attributes

These attributes have completed a move in black. For more information on move in black, see the **Move in black** topic.

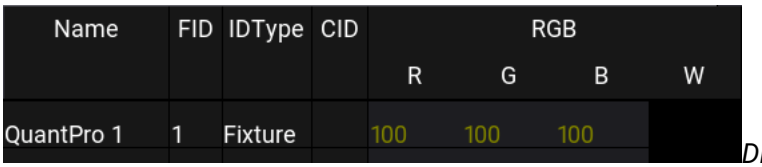
Black text with a sea green background:



Black text with a sea green background on R, G, and B attributes

These attributes are currently fading through a move in black command. This color combination does not appear in the sequence sheet. For more information on move in black, see the **Move in black** topic.

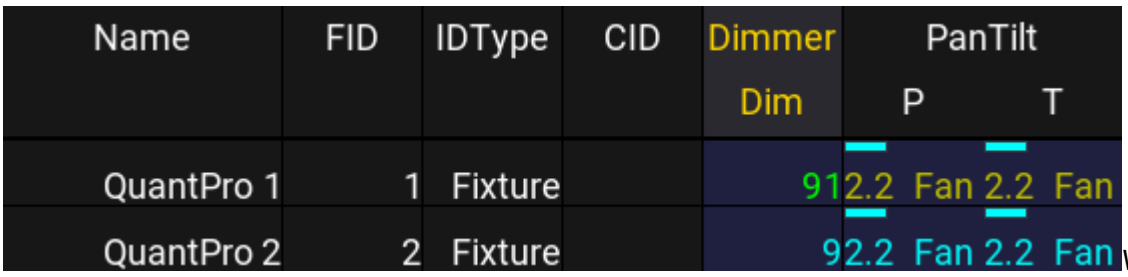
Dim yellow text with a gray background:



Dim yellow text with a gray background on R, G, and B attributes

These attributes have tracked values in the current cue outputting from a sequence that is not selected. This color combination does not appear in the sequence sheet.

Dark blue background:



Various colors of text with a dark blue background on Dim, P, and T attributes

These attributes are currently fading through cue transitions. Text colors continue to specify the source of the data. This background color combination does not appear in the sequence sheet.

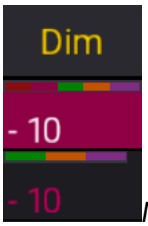
Sheet Colors for Additional Attribute Layers

Along with the absolute value, other layers of values may influence the output of an attribute. The color of the values from these layers matches the colors in the layer control bar in the encoder toolbar. For more information on the encoder toolbar, see the **Encoder Toolbar** topic.

The combinations of text and background colors for the additional layers all use the following pattern:

- White text on a background of the layer's color shows a value in the programmer.
- The text of the layer's color on a gray background shows a value from the current cue in the selected sequence.
- Yellow text on a gray background shows a value from a cue outputting from a sequence that is not selected.

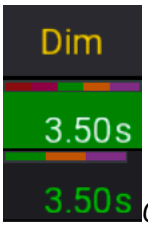
Mauve background or text:



Mauve background and text.

There is a relative value on the dimmers of these two fixtures. The top value with a mauve background comes from the programmer. The bottom value with mauve text comes from the current cue in a selected sequence.

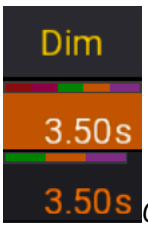
Green background or text:



Green background and text.

There is an individual fade time on the dimmers of these two fixtures. The top value with a green background comes from the programmer. The bottom value with green text comes from the current cue in a selected sequence.

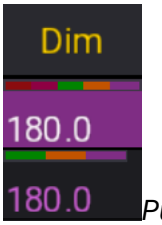
Orange background or text:



Orange background and text.

There is an individual delay time on the dimmers of these two fixtures. The top value with an orange background comes from the programmer. The bottom value with orange text comes from the current cue in a selected sequence.

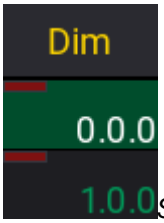
Purple background or text:



Purple background and text.

There is a phaser value on the dimmers of these two fixtures. All layers of phaser values share the same purple color. The top value with a purple background comes from the programmer. The bottom value with purple text comes from the current cue in a selected sequence.

Shamrock background or text:



Shamrock background and text.

There is a GridPos value on the dimmers of these two fixtures. Both values come from the programmer. The top value with a shamrock background shows an active GridPos value. The bottom value with shamrock text shows a deactivated GridPos value.

	Hint:
	Darkened versions of any of the programmer colors above indicate data from a programmer part that is not the current part shown in the sheet. For more information about programmer parts, see the **What is the Programmer** topic.

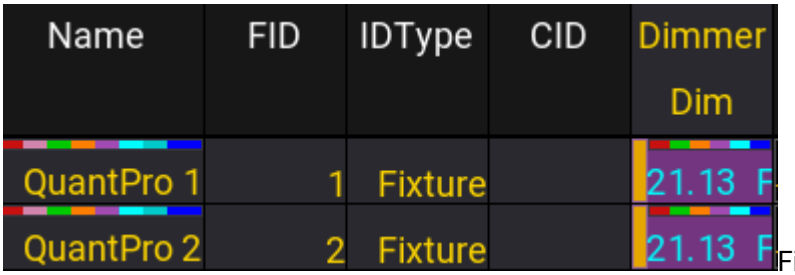
1.8.12.2. Markers

Markers are colored indicators in the form of a bar or a thin line. Markers appear in:

- Fixture Sheet
- Sequence Sheet with the Track Sheet option enabled
- Feature group control bar
- Attribute encoder display

The name column in the fixture sheet, the feature group control bar, and the attribute encoder display all show color-coded markers indicating values, individual timing, and phasers in the programmer. Attribute columns in the fixture sheet show markers indicating values, individual timing, and phasers either in the programmer or currently outputting from a sequence. Attribute columns in the sequence sheet show markers indicating values, individual timing, and phasers stored within the displayed cues.

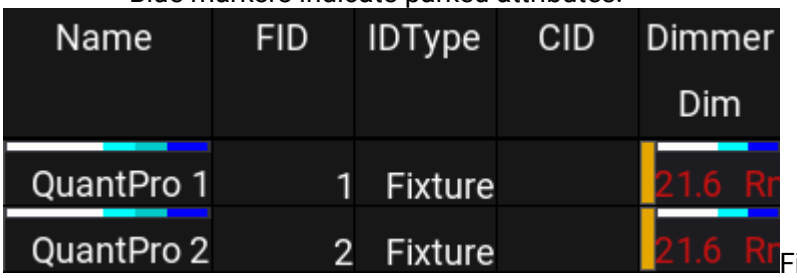
Marker colors:



Fixture sheet with markers on one fixture name and both dimmer attributes

Markers above the name and dimmer value of fixture 1 indicate active information in the programmer. Markers only above the dimmer value of fixture 2, but not above the name, indicate information output from a sequence.

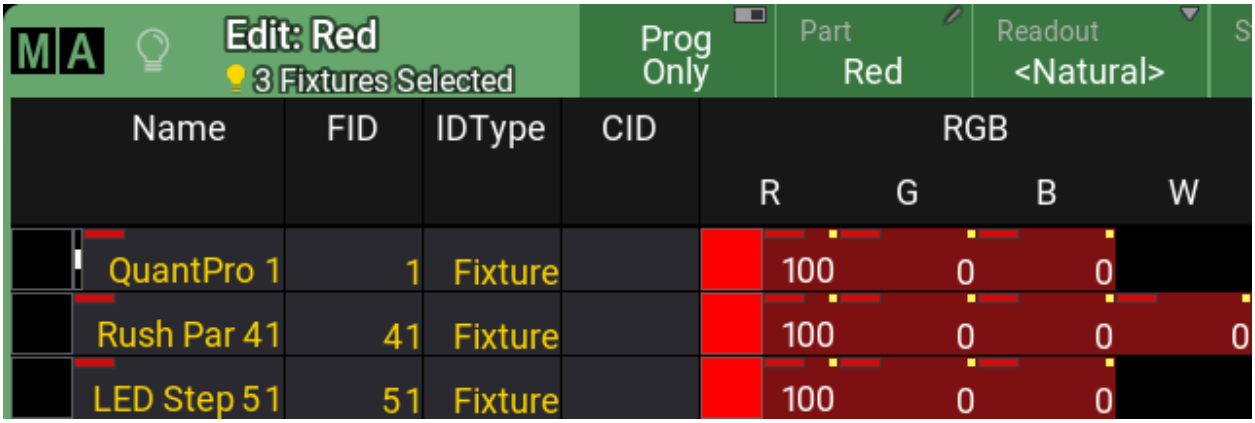
- Red markers indicate active absolute programmer values.
- Mauve markers indicate active relative programmer values.
- Green markers indicate individual fade times.
- Orange markers indicate individual delay times.
- Pink markers indicate phaser values.
- Bright cyan markers indicate absolute values from a preset.
- Dark cyan markers indicate relative values from a preset.
- Blue markers indicate parked attributes.



Fixture sheet with white markers

White markers indicate information in the programmer, which will not be stored when storing active values.

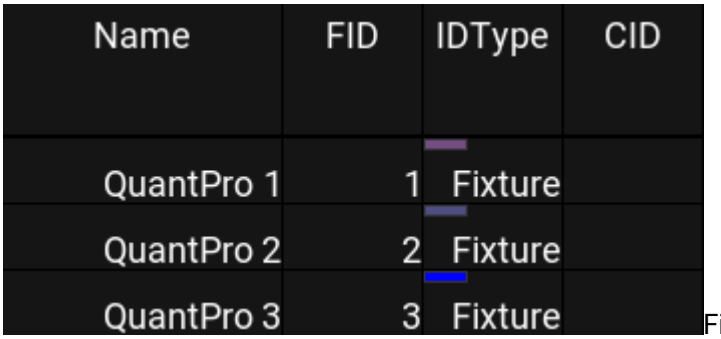
Small, yellow markers when editing a preset:



Fixture sheet while editing a global preset

Small, yellow markers in the fixture sheet while editing a global preset indicate an attribute holding the global value for that attribute for all fixtures of the same type within the preset. For more information on global presets, see the **Presets** topic.

Markers above the IDType:



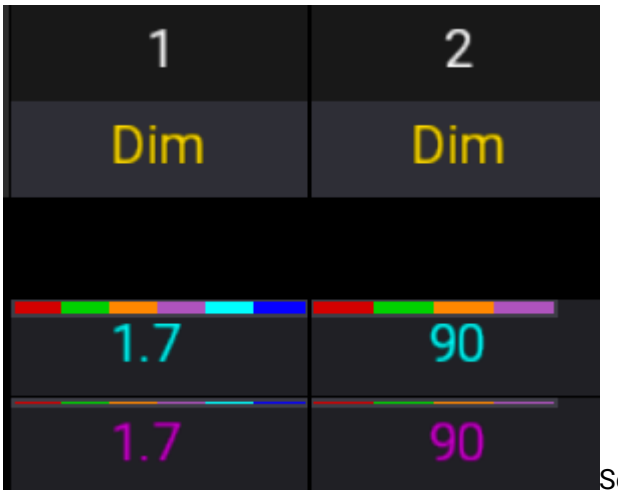
Fixture sheet with markers above the IDType.

Markers above the IDType in the fixture sheet indicate a group master is currently influencing the output of the marked fixture.

- Light pink markers indicate fixtures that are currently influenced by a Positive, Negative, or Scaling group master.
- Lavender markers indicate fixtures that are currently influenced by an Additive group master.
- Blue markers indicate fixtures with the Master React option set to "None."

For more information about group masters, see the **Group Masters** topic. For more information about the Master React option, see the **Add Fixtures to the Show** topic.

Markers in the Sequence Sheet with the Track Sheet option enabled:



Sequence sheet attribute columns with markers

Markers in the sequence sheet include an additional color and two different heights.

- Markers in the form of a bar indicate a new value stored in the cue.
- Markers in the form of a thin line indicate a tracked value.

1.8.12.3. Color Theme

Color themes allow for the customization of a wide variety of colors within the graphical user interface of the console. Alternate color themes can be activated quickly, changing the look of the interface on demand. Color themes can be exported to and imported from external sources. The console initially includes two color themes. The "default" color theme provides optimal readability when using the console under most lighting conditions. The "defaultDAYLIGHT" color theme creates a bolder interface for use in brighter conditions, such as daylight.

Activating Alternate Color Themes

Color themes can be imported using the menu.

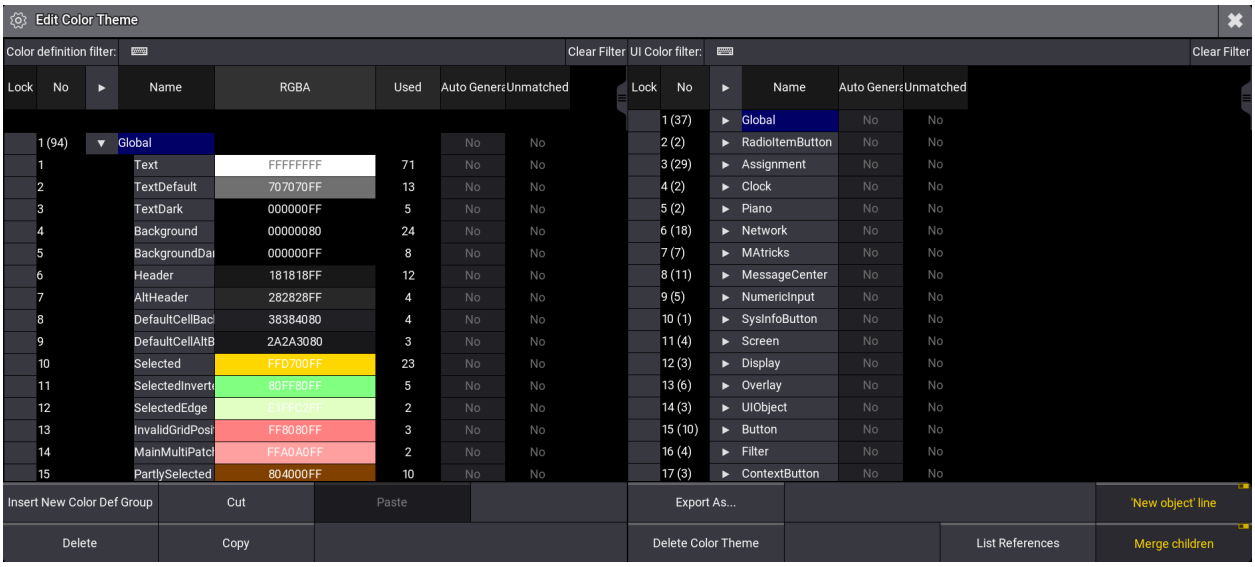
1. Open the Menu.
2. Tap Desk Lights & Color Theme in the top-right corner.
3. Tap Active Color Theme. This button will also display the name of the current color theme. For example, 'default'.
   The Import Color Theme pop-up opens.
4. Tap the desired color theme. If the color theme is on a USB drive, tap Internal in the title bar to select the drive.

Color themes can also be imported using the command line. For example, use the following command to activate the "defaultDAYLIGHT" color theme:

```
User name[Fixture]>Import ColorTheme Library "defaultDAYLIGHT.xml" AT ColorTheme /NoConfirmation
```

Editing the Current Color Theme

In the main menu, tap Desk Lights & Color Theme and then tap Edit in the bottom-right corner. The color theme editor opens:



Color theme editor

The left side of the color theme editor includes a list of color definitions. These function like color presets for user interface elements. Any changes to color definitions will be reflected in all of the UI elements referencing those definitions. This list shows the name of the color definition, the hex value of the color (including red, green, blue, and alpha values), and the number of UI elements referencing each color definition.

The right side of the color theme editor includes a list of user interface elements. This list shows the name of the UI color, a reference to a color definition, and the actual color defined by that reference. Changing the color definition reference to reference another color definition changes the color used for that element without altering any other elements. Deleting the reference for a UI element allows for a direct color definition for that element. This is analogous to storing a cue with hard values instead of referencing a preset.

Tap Export As... at the bottom of the menu to export the current color theme, either to the internal drive or to an external USB drive.

1.8.12.4. Compare Tool

The color theme compare tool is part of the Desk Lights & Color Theme menu.

The compare tool is a useful feature when it comes to:

- Comparing color themes with each other.
- Easily find distinctions when comparing color themes via color indicators and colored rows.
- Transfer colors between one color theme and another.
- Live color editing.

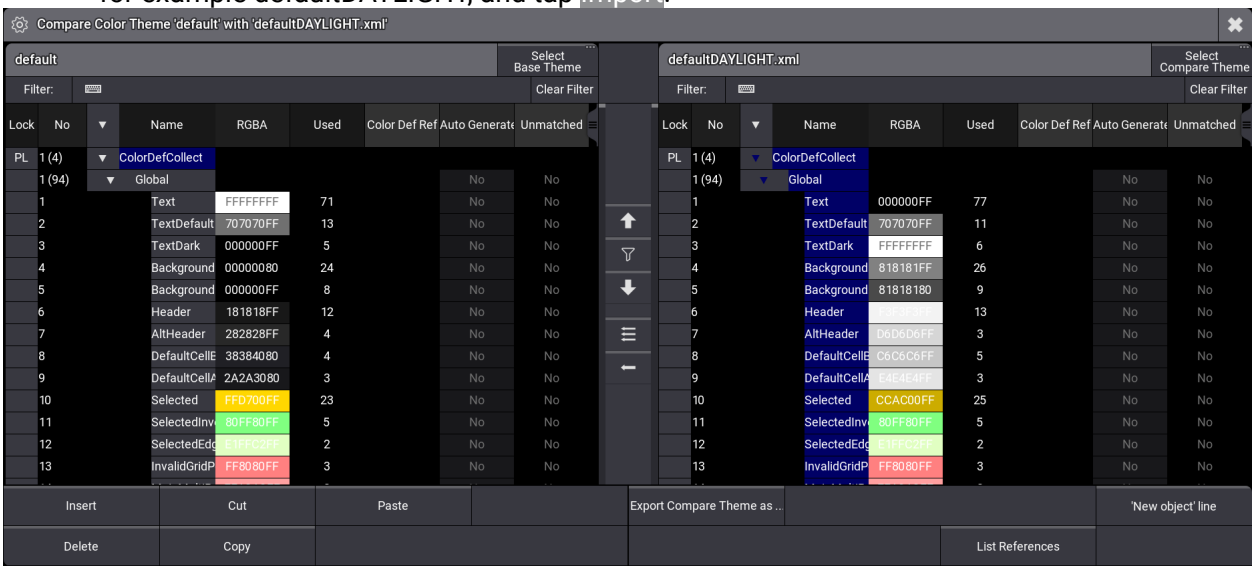
The color indicators and colored rows are defined as the following:

- **Orange**: This color exists in the base color theme, but is missing in the compared color theme, or vice versa.
- **Green**: This color was transferred from the base to the compared color theme, or vice versa.

For more information about color themes, see **Color Theme** topic.

To open the compare tool:

1. Open the Menu.
2. Tap Desk Lights & Color Theme.
3. Tap Compare. The compare tool opens. The currently active color theme will be taken as the base for comparison.
4. When opening the compare tool for the first time, a import pop-up opens. Select a color theme, for example defaultDAYLIGHT, and tap Import.



Compare tool is open

The different buttons of the compare tool are described below:

- Filter:
  Filters the color name.
- Clear Filter:
  Clears the filter.
- Select Base Theme:
  Opens the import base color theme pop-up.
- Select Compare Theme:
  Opens the import compare color theme pop-up.
-  (Arrow up):
  Shows previous missing or merged color in the theme.
-  (Object Filter):
  Shows only unmatched colors.
- (Arrow down):
  Shows next missing or merged color in the theme.
-  (Merge all left):
  Merges all unmatched colors from the right into the left theme.
-  (Merge all right):
  Merges all unmatched colors from the left into the right theme.
- (Merge left):
  Merges only selected unmatched colors from the left into the right theme.
- (Merge right):
  Merges only selected unmatched colors from the right into the left theme.

	**Hint:**
	To change the direction of the merge arrows, tap into the grid area of the specific color theme.

The grid is categorized in different columns:

- **Lock**:
  When it is set to "Yes", no color values can be changed.
- **No**:
  This is the color index number. Value in the bracket (x) shows the amount of colors in the particular structure tree.
- **Name**:
  Color name.
- **RGBA**:
  Shows the color definition.
- **Used**:
  This is the amount of color related references.

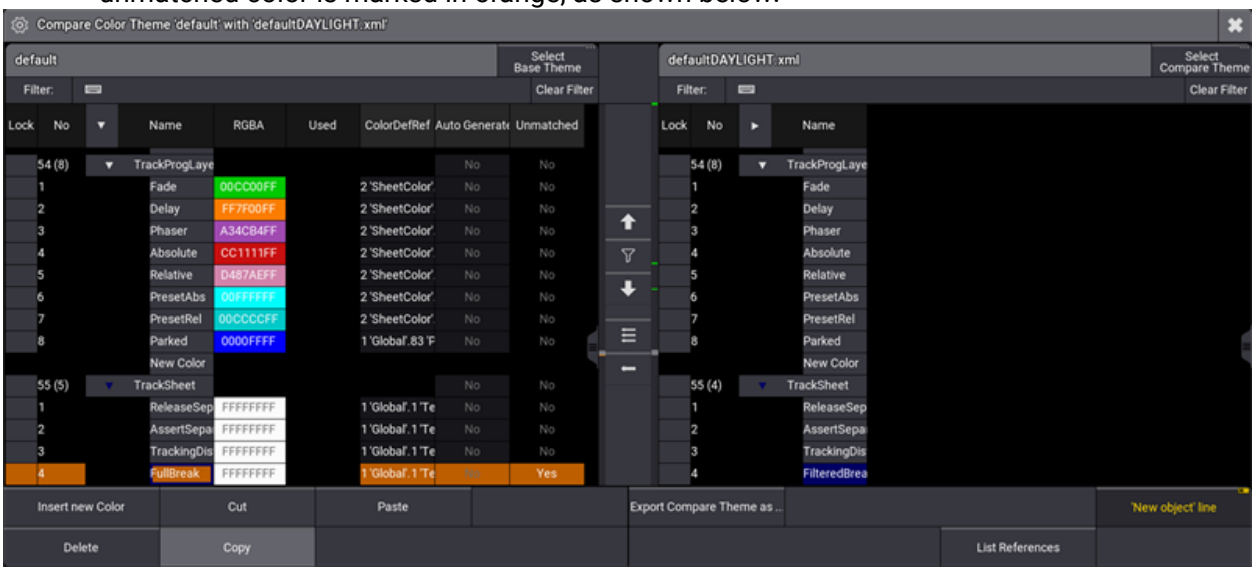
- **ColorDefRef**:
  See **Color Theme**.
- **Auto Generated**:
  Shows "Yes" when a color is merged.
- **Unmatched**:
  Is set to "Yes" when a color is not matching the compared theme.

Example for a Color Theme Merge

Requirements:

- Default is selected as base color theme.
- DefaultDAYLIGHT is selected as compare color theme.
- A color is deleted in the DefaultDAYLIGHT color theme. In the example below, FullBreak.

1. To quickly move the scroll bar to the next unmatched color in the color theme, tap . The unmatched color is marked in orange, as shown below:

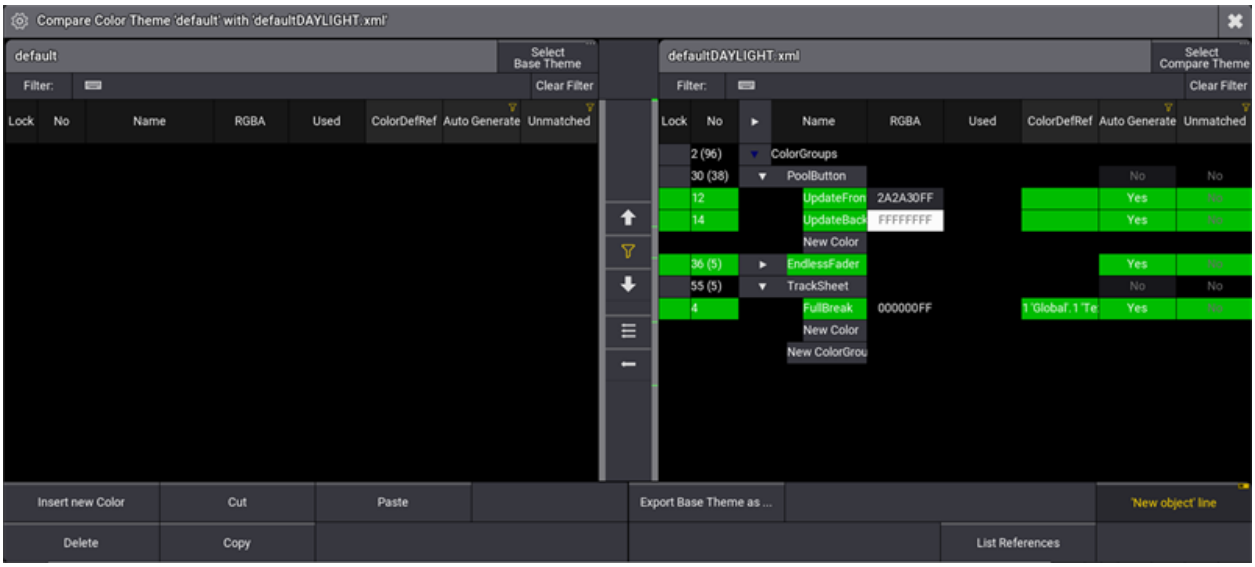


Unmatched colors are marked in orange.

2. To merge all unmatched colors, tap .
   All merged colors are displayed in green rows. Notice that the column Auto Generated is set to Yes and the column Unmatched is set to No automatically in the defaultDAYLIGHT color theme as shown below:

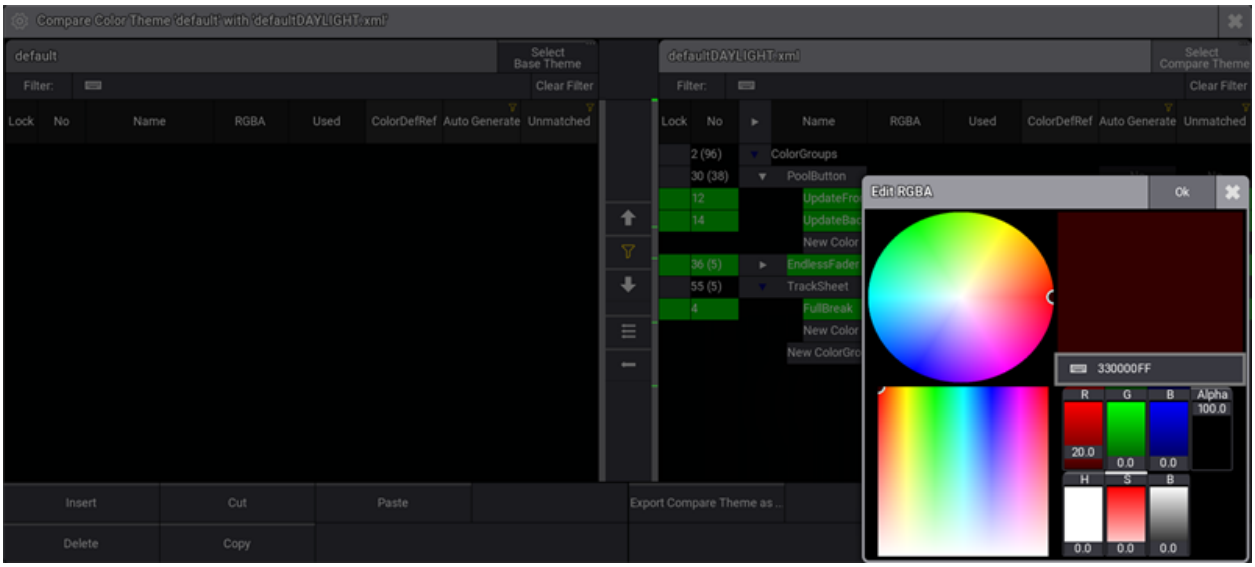
Missing colors merged into defaultDAYLIGHT color theme are shown in green.

3. To show only Auto Generated and Unmatched colors, tap . When enabling filter, the button turns yellow and the columns names "Auto Generated" and "Unmatched" get a yellow filter icon as well:



Filter is active.

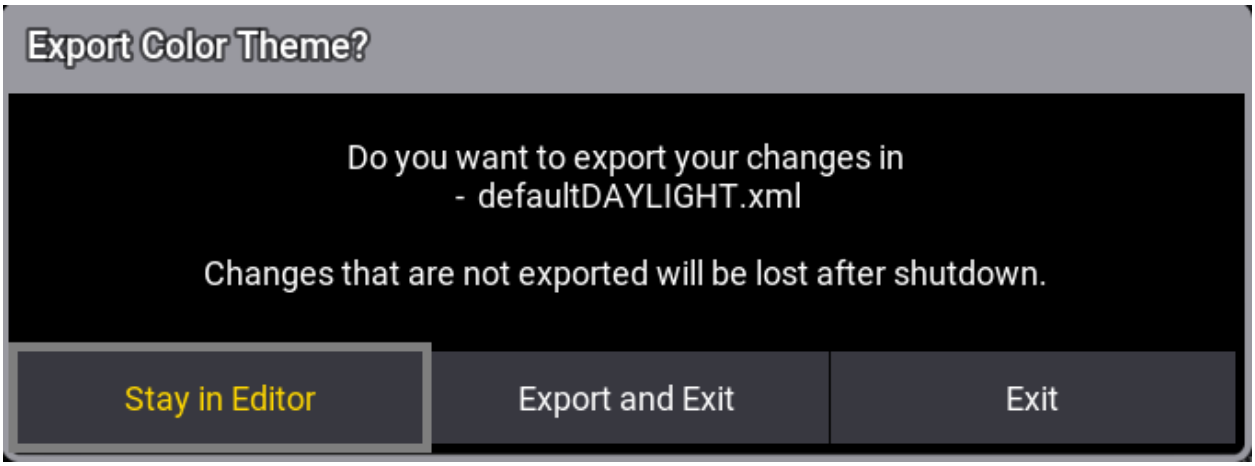
4. To change RGBA values for "FullBreak", right click the color in the RGBA column. The color editor opens.
5. Change the R value to 20 by using the dynamic color scroll bar. Tap Ok to close the editor. The color value has changed:



Colors are editable in the color theme compare tool, too.

Export Color Theme

If closing the color editor without exporting the color theme first, the following pop- up with opens:



Export color theme pop-up.

- Stay in Editor:
  Pop-up closes and the color theme editor stays open.

- Export and Exit:
  Changes are exported to the color theme listed above.

- Exit:
  Closes the color theme editor. Changes will get lost after a shutdown.

To export the defaultDAYLIGHT.xml color theme as a new theme:

1. Tap into the right grid area of the compare tool to link the Export button to the corresponding defaultDAYLIGHT.xml file.
2. Tap Export Compare Theme as .... The Export Compare Theme pop-up opens.

3. Type defaultDAYLIGHT_new.xml and then tap Export. The color theme is exported.

1.9. Command Syntax and Keywords

Function of the Command Line

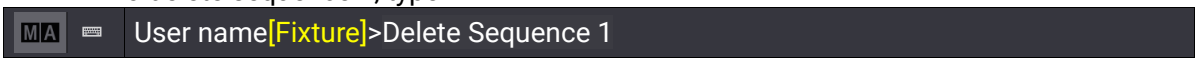
The command line is an essential way of communication between the console and its operator.

Using keywords, special characters, and numerical identifiers is how the operator tells the console what to do.

	Important:
	Depending on the use case, some commands can be executed using the **Please key**.

Example

- To delete sequence 1, type:



- Or use the keys in the command section of the console and press:

Delete + Sequ + 1 + Please

Keywords

The keywords that are mostly used have their own corresponding keys.

For more information on the multiple functions of the keys see **Keys**.

	Important:
	All keywords can be entered in the command line using the internal or any other connected keyboard.

To view the commands issued to the console, tap MA in the left corner of the command line.

The **Command Line History** opens.

For more information on the usage of the command line in detail see **Command Line**.

Subtopics

- **Syntax Rules**

- **Parent Child Concept**
- **General Keywords**
- **Option Keywords**
- **Extended Command Line Syntax Options**

1.9.1. Syntax Rules

The command line syntax is used to create valid commands.

Object keywords are used to allocate objects in the show file.

Help keywords are used to create a relation between functions and objects.

Playback keywords provide control over playback functionalities.

Fader keywords are related to anything that has to do with faders.

For more information see **Command Line History**.

General Rules

The general rules are:

	**Important:**
	The basic syntax is as follows: **[Function] [Object]**

- All objects have a default function which is used if no function is given.
- Most functions have a default object or an object type which is used if no object is given.
- Objects are arranged in a hierarchical tree structure.
- If an object does not support the function applied, the function is passed on to a child or parent object.

Terminology

- [Square brackets]:
  Description of non-literal content.
- (Parentheses):
  Description of optional content.
- "Quotation marks"/'Quotation marks':
  Quotation marks are used to enter a definite name or content. If the line ends after the word in quotation marks, the quotation marks at the end may be omitted. If the name or content is not a keyword and does not contain special characters, the quotation marks can be left out altogether.

	**Important:**
	If an option or any other part of the keyword command requires two types of quotation marks - single (') and double quotation marks (") - make sure to always use an equal pair. For more information see the **RemoteCommand keyword**.

- Capitalization:
  In general, capitalization is important. Only when using keyword commands, the console does

not distinguish between upper case and lower case. In such topics, capitalization is used to improve readability only.

Use the Command Line

It is possible to abbreviate all the commands using the shortcuts of the corresponding keywords.

Each keyword has its own shortcut. Every keyword topic also names the respective shortcut.

For more information see **General Keywords**.

Example

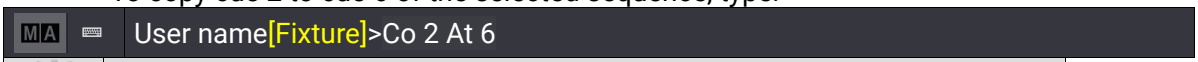
- To store cue 20 in sequence 8 using the overwrite function, type:

Full version of the syntax:

Abbreviated version of the syntax:

Very short version of the syntax:

- To copy cue 2 to cue 6 of the selected sequence, type:



Hint:
The examples use the full version in the manual.

1.9.2. Parent Child Concept

The general selection syntax is not only applying to fixtures (Parent) and their levels of sub-fixtures (Child), but furthermore the general selection syntax applies to all types of objects that use a hierarchic structure. Objects that use hierarchic structures are, for example:

- Preset Pools with Presets.
- Pages with Executors.
- Macro with Macro lines.

	**Hint:**
	To learn more about selecting fixtures and its sub-fixtures using syntax commands, see **Select Fixtures** topic.

To gain a faster and more logical way (for example to select, copy, move or delete objects) in the hierarchical structure, the dot (.) is an important tool in the selection syntax:

- The dot (.) ends the selection on the current level and then steps one level down in the hierarchical structure.
  Instead of using the dot (.) as a separator, dedicated keywords for the objects in the next level of the hierarchy can be used. For example, **Page 1 Executor 201**.

To restart at the top level of the hierarchy within one command, the starting keyword (for example Fixture) has to be entered again. For example, **Fixture 5.2 Fixture 3** .

To select fixtures in the same hierarchic structure within one command, (+) can be used. For example, **Fixture 5.2 + 3**.

Dot Selection Behavior

It is not necessary to recall the subcategory of an object in a syntax command, after the subcategory is already called using the dot (.). This behavior is shown in the examples below.

Do not use the following command:

Clone Fixture 101 At Fixture 102 If Preset 21.101 Thru 21.105

Use a command with a omission of "21." instead:

Clone Fixture 101 At Fixture 102 If Preset 21.101 Thru 105

Do not use the following command:

Move Preset 1.1 Thru 1.3 At 1.5

Use a command with a omission of "1." instead:

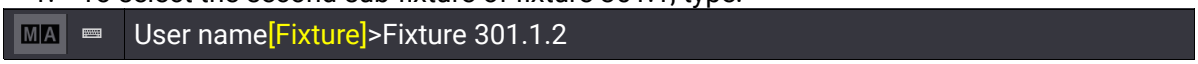
Move Preset 1.1 Thru 3 At 1.5

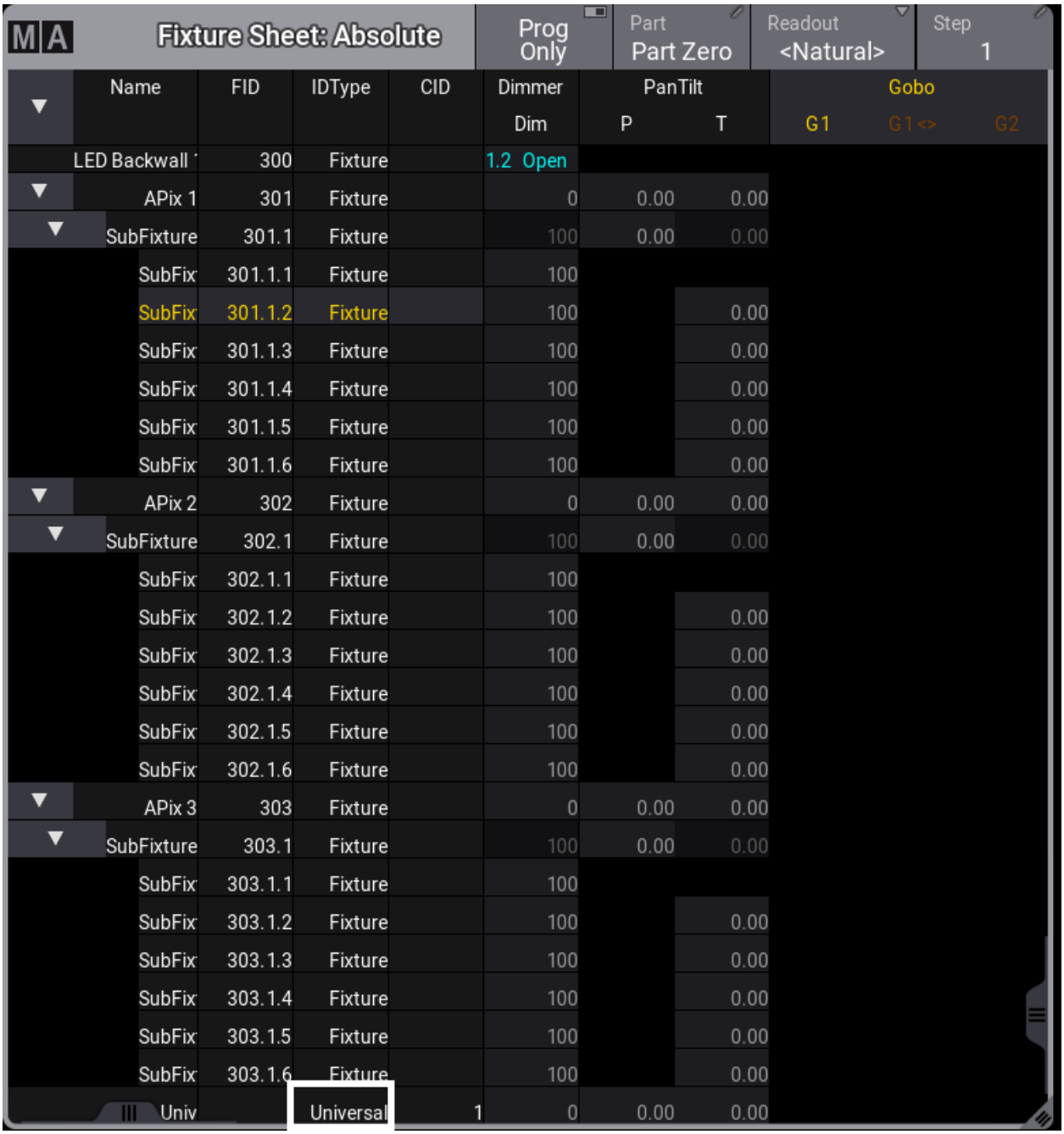
Examples for Selection Syntax

To recreate the examples, the following should be prepared:

- The grandMA3 demo show file is loaded.
- A fixture sheet window is open.
- The following fixture type is patched:
  - Ayrton Alienpix - RS Ex 16 Bit
  - Quantity: 3
  - FID: 301 - 303

1. To select the second sub-fixture of fixture 301.1, type:



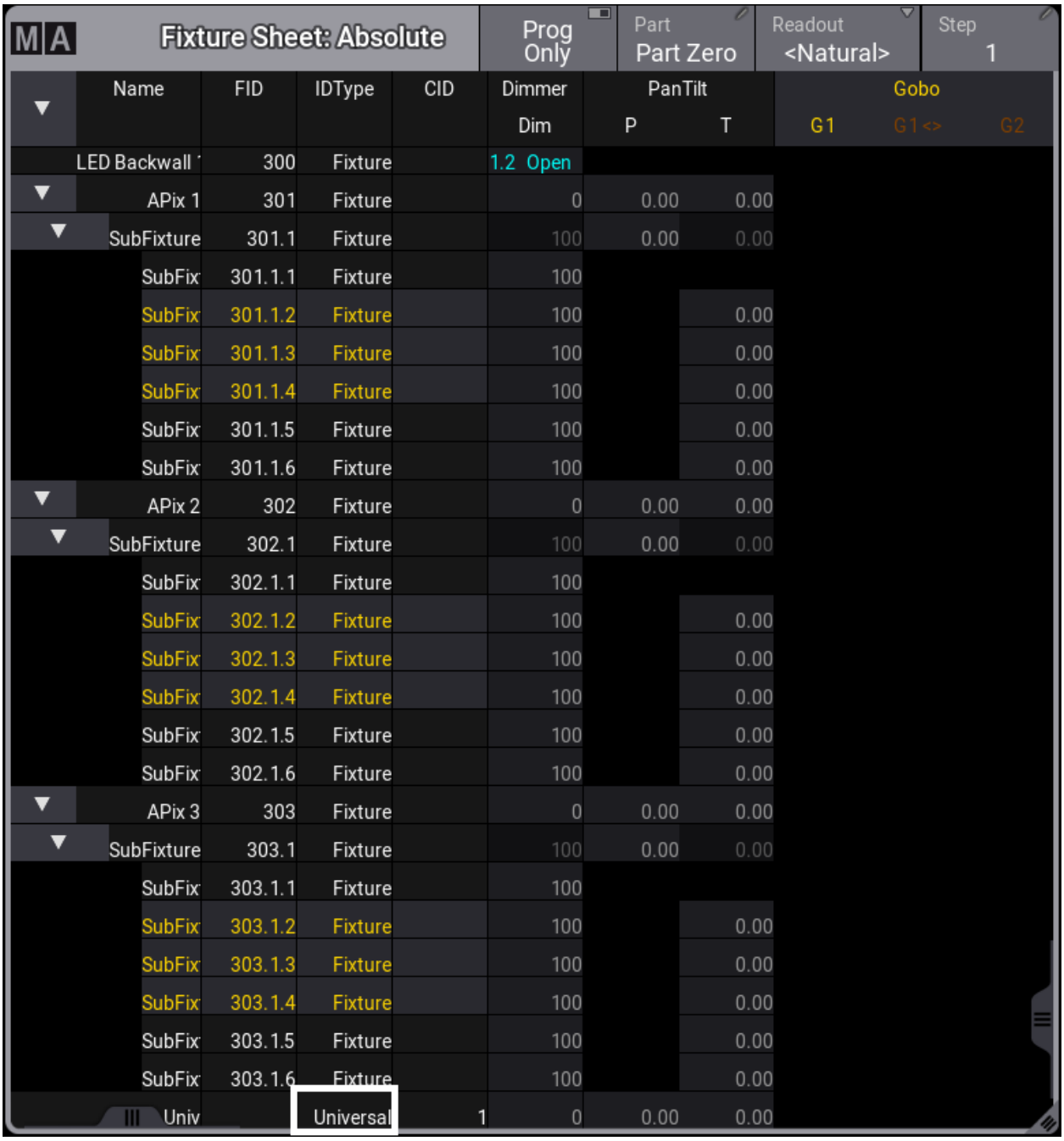


Second sub-sub-fixture of fixture 301 is selected

2. To select only the sub-sub-fixtures 2 to 4 of fixtures 301 to 303, type:

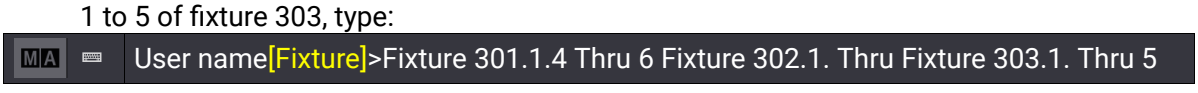
In this example all fixtures on the top level have to be selected: **Fixture 301 Thru 303.1.**.

And after the dot, the sub-fixtures are selected: **2 Thru 4**.



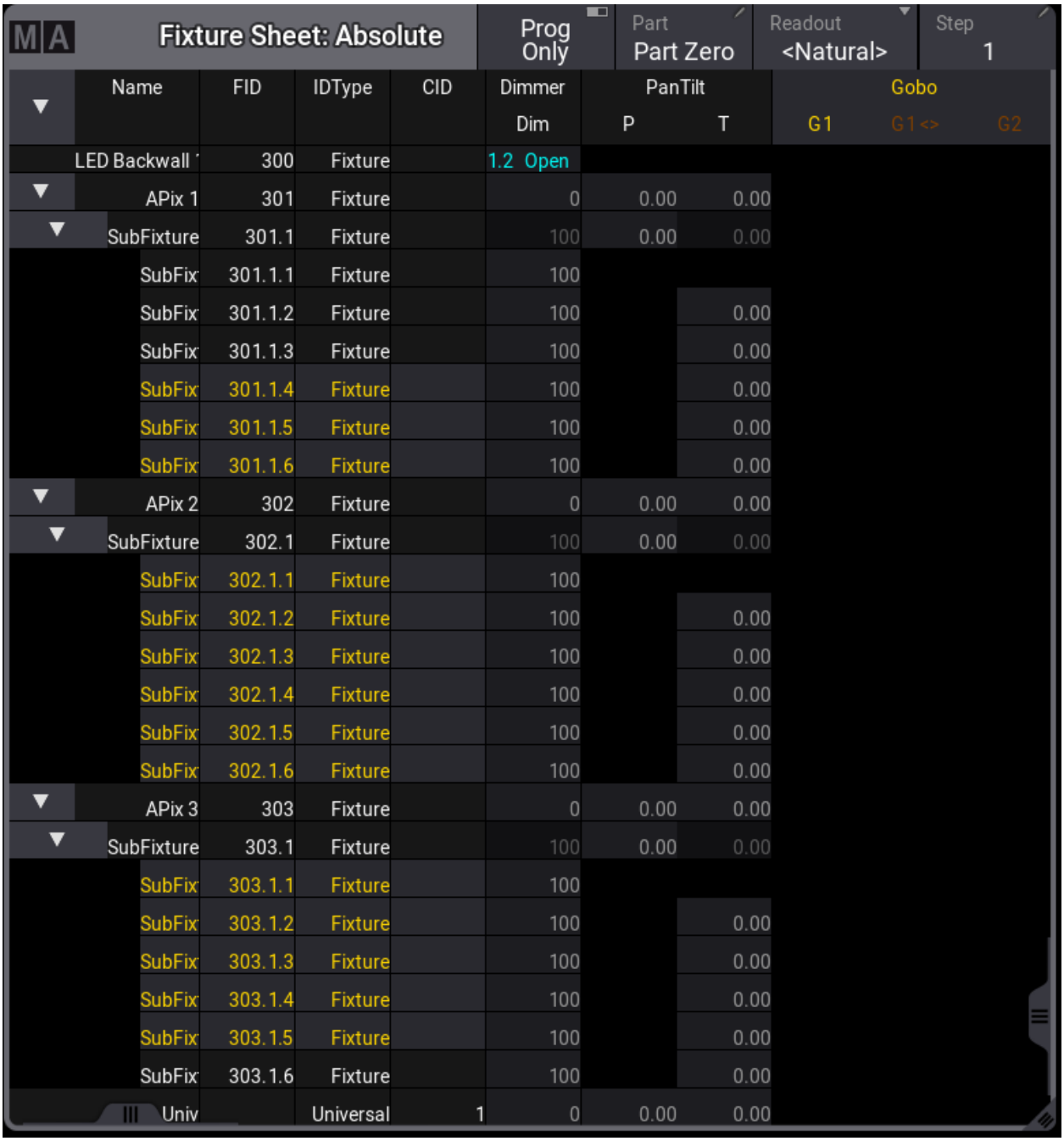
Sub-sub-fixtures 2 to 4 are selected.

3. To select sub-sub-fixtures 4 to 6 of fixture 301, all sub-fixtures of fixture 302, and sub-sub-fixtures 1 to 5 of fixture 303, type:



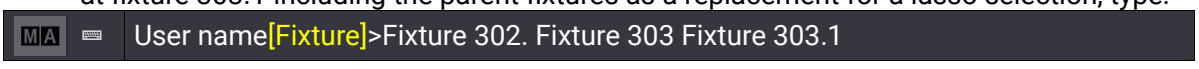
In this example the . 4 in **Fixture 301.1.4** steps down to the sub-fixture level. **Fixture 302.1.** needs to be entered to restart at the top level with fixture 302.

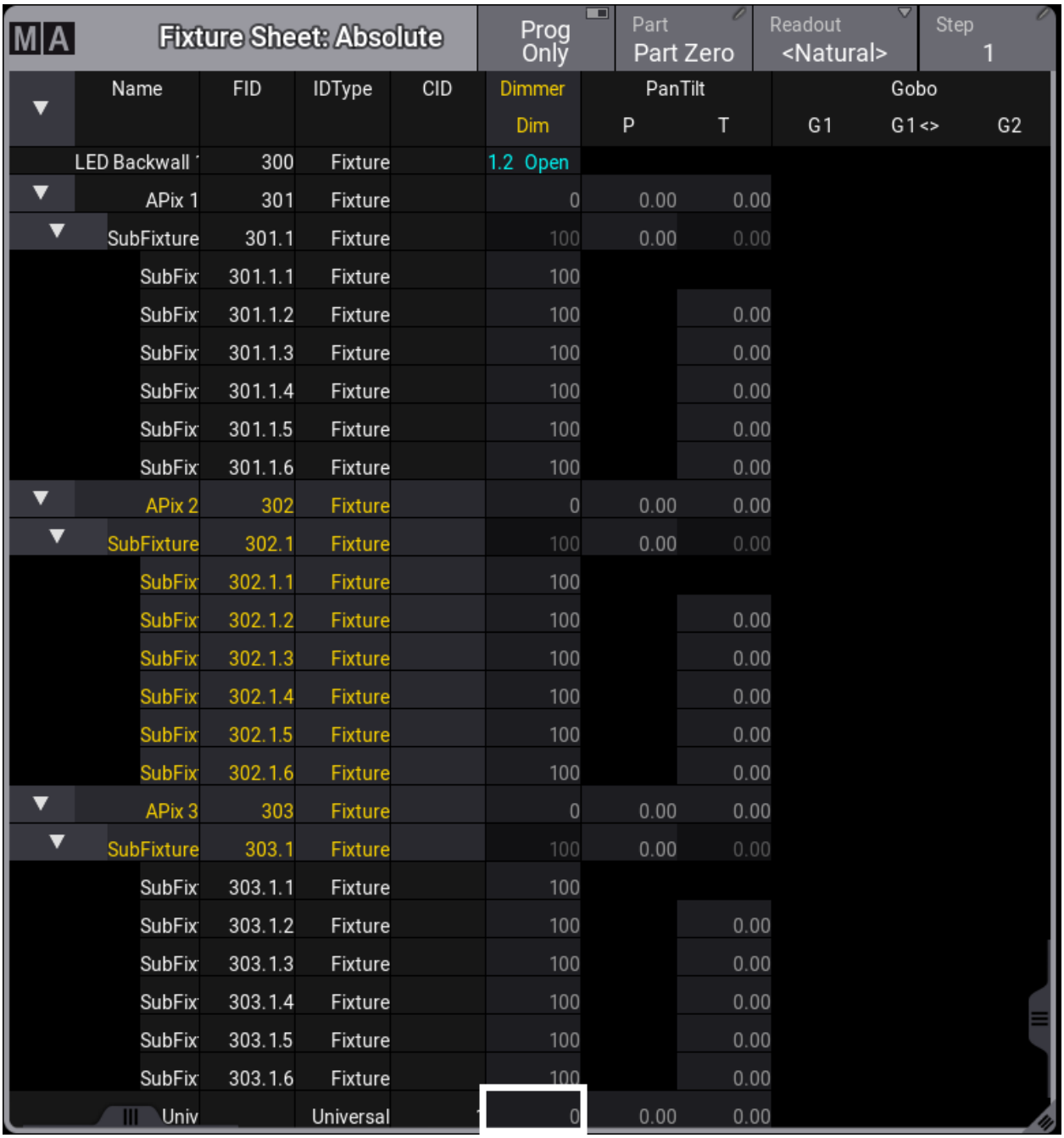
If **Fixture 302** would not be entered, the second sub-fixture of sub-fixtures 4 to 6 would be selected.



Different sub-sub-fixtures are selected for each fixture type

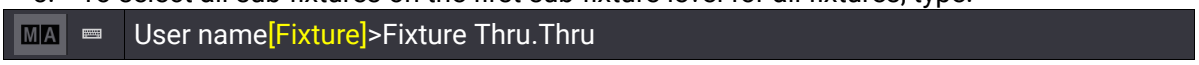
4. To select the fixtures as a replacement for a lasso selection, starting at fixture 302 and ending it at fixture 303.1 including the parent fixtures as a replacement for a lasso selection, type:





This fixture selection is a replacement for a lasso selection.

5. To select all sub-fixtures on the first sub-fixture level for all fixtures, type:

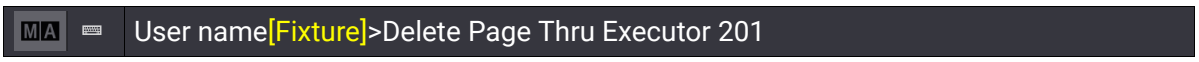


Deleting Syntax

- The following examples are used in general, and are not in correspondence with the demo show file.

To delete executor 201 on all pages, type:

or type:



This second example describes the additional rule of the first rule from above: Instead of using Page x.y, the Executor keyword is used to address the executors.

1.9.3. General Keywords

Using keywords provides one of many possibilities to operate the grandMA3 console.

The following subtopics cover each of the available grandMA3 keywords. These are arranged in alphabetical order.

Each topic describes various means of entering the keyword into the command line, provides definition, proper syntax, and depicts examples of the keywords.

Subtopics

- **Chat**
- **ChatJoin**
- **ChatLeave**
- **CleanUp**
- **Clear**
- **ClearActive**
- **ClearAll**
- **ClearSelection**
- **Clone**
- **CommandDelay**
- **Collect**
- **Collection**
- **Color**
- **ColorDefinition**
- **ColorTheme**
- **Configuration**
- **Console**
- **Cook**
- **Copy**
- **Cue**
- **CueAbsolute**
- **CopyCrashLog**
- **CueDelay**
- **CueFade**
- **CueInDelay**
- **CueInFade**
- **CueOutDelay**
- **CueOutFade**
- **CueRelative**
- **CueUpdate**
- **CurrentEnvironment**
- **CurrentUser**
- **CurrentUserProfile**
- **Cut**
- **DataPool**
- **Deceleration**
- **Decimal8**
- **Deactivate**
- **Decimal16**
- **Decimal24**
- **Default**
- **Delay**
- **DeleteGlobalVariable**
- **Delete**
- **DeleteOtherVersion**
- **DeleteUserVariable**
- **Disconnect**
- **Dismiss**
- **Display**
- **DMXAddress**

- **DMXLayer**
- **DMXReadout**
- **DMXUniverse**
- **DoubleSpeed**
- **Down**
- **DumpLog**
- **Drive**
- **DropOwnership**
- **Echo**
- **Edit**
- **EditSetting**
- **Eject**
- **Effect**
- **EditRecipe**
- **EncoderBank**
- **EncoderBar**
- **EndIf**
- **Environment**
- **Exchange**
- **Executor**
- **Export**
- **Extension**
- **Extract**
- **Fade**
- **FaderCrossFade**
- **FaderCrossFadeA**
- **FaderCrossFadeB**
- **FaderMaster**
- **FaderRate**
- **FaderSpeed**
- **FaderTime**
- **FaderTemp**
- **FeatureGroup**
- **Filter**
- **Fix**
- **Fixture**
- **FixtureClass**
- **FixtureLayer**
- **FixtureType**
- **Flash**
- **Flip**
- **Fog**
- **Freeze**
- **Full**
- **Gel**
- **Generator**
- **GetGlobalVariable**
- **GetUserVariable**
- **Go+**
- **Go-**

- **Goto**
- **Grid**
- **GridPosition**
- **GridStore**
- **GoboImage**
- **Group**
- **HalfSpeed**
- **Help**
- **HelpLua**
- **Hex8**
- **Hex16**
- **HelpKeyword**
- **Hex24**
- **Highlight**
- **Hold**
- **Houselights**
- **Hz**
- **If**
- **IfActive**
- **IfProgrammer**
- **IfOutput**
- **Image**
- **Import**
- **Index**
- **Insert**
- **Integrate**
- **Interface**
- **Invert**
- **Invite**
- **IP**
- **JoinSession**
- **Key**
- **Keyboard**
- **KeyboardShortcut**
- **Kill**
- **KnockIn**
- **KnockOut**
- **Label**
- **Layout**
- **LearnSpeed**
- **LeaveSession**
- **Library**
- **List**
- **ListOwnership**
- **ListReference**
- **Load**
- **Loaded**
- **ListCrashLog**
- **LoadShow**
- **Lock**

- **LogIn**
- **LogOut**
- **Lowlight**
- **Lua**
- **LuaFile**
- **Macro**
- **MArker**
- **Master**
- **MAtricks**
- **Measure**
- **Media**
- **MemoryInfo**
- **Menu**
- **Mesh**
- **Move**
- **Multipatch**
- **MyRunningPreset**
- **MyRunningMacro**
- **MyRunningSequence**
- **MyRunningSoundFile**
- **MyRunningTimecode**
- **MyRunningTimer**
- **Natural**
- **NDI**
- **NewShow**
- **Next**
- **NextY**
- **NextZ**
- **NetworkNode**
- **NonDim**
- **Normal**
- **Note**
- **Off**
- **Offset**
- **On**
- **NetworkSpeedTest**
- **onPC**
- **OSC**
- **Oops**
- **OutputLayer**
- **Page**
- **Part**
- **Park**
- **Paste**
- **Patch**
- **Pause**
- **Percent**
- **PercentFine**
- **Phase**
- **Physical**

- **Plugin**
- **Preset**
- **PresetUpdate**
- **Press**
- **Preview**
- **Previous**
- **PreviousY**
- **PreviousZ**
- **Programmer**
- **Property**
- **ProcessingUnit**
- **PSR**
- **Pyro**
- **Quickey**
- **Rate1**
- **RDM**
- **Readout**
- **Reboot**
- **Recast**
- **Reconnect**
- **Record**
- **Relative**
- **Release**
- **ReloadAllPlugins**
- **ReloadUI**
- **Remote**
- **RemoteHID**
- **RemoteCommand**
- **Remove**
- **RenderQuality**
- **Reset**
- **Restart**
- **Root**
- **RealtimeChannel**
- **RunningMacro**
- **RunningPreset**
- **RunningSoundFile**
- **RunningSequence**
- **RunningTimecode**
- **RunningTimer**
- **SaveShow**
- **ScreenConfiguration**
- **ScreenContent**
- **Scribble**
- **Seconds**
- **Select**
- **Selection**
- **SelectFixtures**
- **Sequence**
- **SendMIDI**

- **SendOSC**
- **SendMVR**
- **Set**
- **SetGlobalVariable**
- **SetUserVariable**
- **Shuffle**
- **ShutDown**
- **SnapDelay**
- **SoftwareImport**
- **Session**
- **SoftwareUpdate**
- **SoundChannel**
- **Solo**
- **SpecialExecutor**
- **Speed**
- **Speed1**
- **SpeedMaster**
- **Stage**
- **Station**
- **StationSettings**
- **Step**
- **Stomp**
- **Store**
- **SwitchTograndMA2Software**
- **SwitchTograndMA3Software**
- **Swap**
- **Temp**
- **Thru**
- **Time**
- **Timecode**
- **TimecodeSlot**
- **Timer**
- **Toggle**
- **Top**
- **Tag**
- **TopUp**
- **Transition**
- **Type**
- **UIChannel**
- **UIGridSelection**
- **Unblock**
- **Universal**
- **Unlock**
- **Unpark**
- **Unpress**
- **Up**
- **Update**
- **UpdateContent**
- **User**
- **User1**

- **User2**
- **UserProfile**
- **Video**
- **Version**
- **View**
- **ViewButton**
- **Width**
- **World**
- **Zero**

1.9.3.1. ; [Semicolon]

To enter the ; [Semicolon] in the command line, press ;.

Description

The semicolon separates multiple commands.

Syntax

[Command]; [Command]

Example

- To turn off sequence 5 and delete group 3, type:

1.9.3.2. / [Slash]

To enter the / [Slash] in the command line, press /.

Description

The slash character has two functions:

- It is used to set options during a command.
  For more information see the **Option Keywords**.
- It is used as a separator in the calculator.

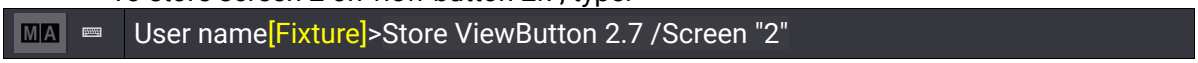
	**Important:**
	When adding option keywords to your commands, make sure to use the slash without spaces.

Syntax

[Command] /Option ["Option_Value"]

Example

- To store screen 2 on view button 2.7, type:



1.9.3.3. . [Dot]

To enter the . [Dot] in the command line, press .

Description

The dot is a separator which is used in fractional digits and hierarchic IDs of objects.

	Important:
	Make sure to use the dot without spaces.

Syntax

[Command] [Value].[Value_Fraction]

[Command] [Parent_Number].[Child_Number]

Examples

- To call the second color preset for fixture 31.2, type:

```
User name[Fixture]> Fixture 31.2 At Preset 4.2
```

- To set the pan attribute to 50.5, type:

```
User name[Fixture]>Attribute "Pan" At 50.5
```

1.9.3.4. = [Equal]

To enter the = [Equal] character in the command line, type **=**.

Description

The = [Equal] character is used when assigning a value to an object setting.

Many objects have different settings and these settings can be set using the setting name followed by the equal sign and then the desired value.

It is used in conjunction with the **Set keyword**.

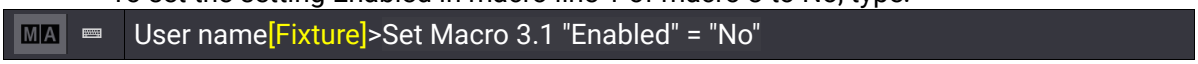
	**Hint:**
	The equal character can be omitted in the command. In case a command could be misinterpreted, use this character.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] ["Setting"] = ["Option"]

Example

- To set the setting Enabled in macro line 1 of macro 3 to No, type:



1.9.3.5. #[ ]

To enter the #[ ] character in the command line, type **# [ ]**.

Description

The #[ ] is used instead of object numbers or names in order to address an object. Using #[ ] makes it unnecessary to update macros or commands if the object was moved or the name of the object was changed.

Syntax

[Function] #[Object "Object_Name"]

[Function] #[Object Object_Number]

Examples

- To set the variable "MyHandle" to the handle of macro 1, type:

```
User name[Fixture]>SetUserVariable "MyHandle" #[Macro 1]
```

or:

```
User name[Fixture]>SetUserVariable "MyHandle" At Macro 1
```

- To go to the next cue in sequence 23 using the handle of the sequence, type:

```
User name[Fixture]>Go+ #[Sequence 23]
```

- To delete macro lines 1 to 5 of macro 42 using the handle of macro 42, type:

```
User name[Fixture]>Delete #[Macro 42].1 Thru 5
```

1.9.3.6. <<< [GoFastBackward]

To enter the <<< [GoFastBackward] keyword in the command line, use one of the options:

- Press <<<
- Type **<<<**
- Type the shortcut **<**

Description

The <<< [GoFastBackward] keyword is a playback keyword which is used to jump to the previous cue in a sequence, without using cue timing.

For more information on how to assign executors see **Assign Objects to an Executor**.

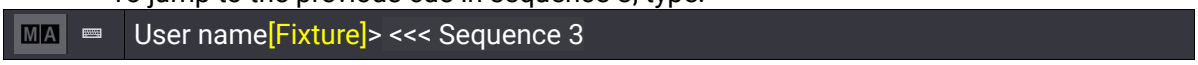
	**Hint:** Instead of Goto it is also possible to use <<< in order to go to a specified cue.

Syntax

<<< [Object] ["Object_Name" or Object_Number]

Examples

- To jump to the previous cue in sequence 3, type:



- To jump to the previous cue in the sequence assigned to executor 103 (on the current page), type:

- To jump to the previous cue on all sequences assigned to executors on page 4, type:

For information on the key see the **<<<** [GoFastBackward] key.

1.9.3.7. >>> [GoFastForward]

To enter the >>> [GoFastForward] keyword in the command line, use one of the options:

- Press >>>
- Type **>>>**
- Type the shortcut **>**

Description

The >>> [GoFastForward] keyword is a playback keyword which is used to jump to the next cue in a sequence without using cue timing.

For more information on how to assign executors see **Assign Objects to an Executor**.

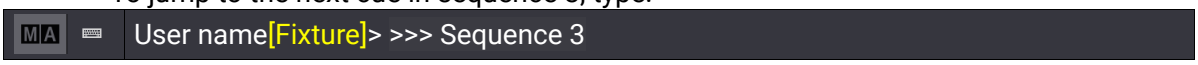
	**Hint:**
	Instead of Goto it is also possible to use >>> in order to go to a specified cue.

Syntax

>>> [Object] ["Object_Name" or Object_Number]

Examples

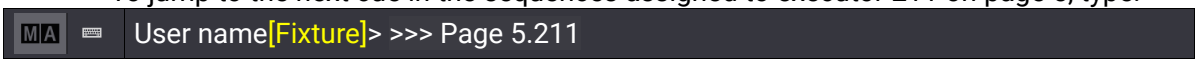
- To jump to the next cue in sequence 3, type:



- To jump to the next cue in the sequence assigned on executor 103, type:

- To jump to the next cue on all sequences assigned to executors on page 4, type:

- To jump to the next cue in the sequences assigned to executor 211 on page 5, type:



For more information on the key see the **>>>** [GoFastForward] key.

1.9.3.8. - [Minus]

To enter the - [Minus] keyword in the command line, use one of the options:

- Press -
- Type **-** [Minus]

Description

The - [Minus] keyword is used to remove objects from a list or to indicate negative values.

If the - [Minus] keyword is used in order to indicate values, it will indicate absolute or relative values:

- A space between the - [Minus] and the value is automatically added. The space makes the value relative
- To obtain an absolute value, remove the space between the - [Minus]

Syntax

([Attribute] ["Attribute_Name" or Attribute_Number]) At - [Value]

[Object] ["Object_Name or Object_Number] - [Object] ["Object_Name" or Object_Number] [Number]

Examples

- To reduce the percentage readout of pan by 10 percent, type:

```
User name[Fixture]>Attribute "Pan" At - 10
```

- To reduce 10 % from the current dimmer value in the selected fixtures, type:

```
User name[Fixture]>At - 10
```

- To select the entire group 5 without selecting fixture 2, type:

```
User name[Fixture]>Group 5 - Fixture 2
```

- To remove fixtures 5, 6, and 7 in the current selection of fixtures, type:

```
User name[Fixture]>- Fixture 5 Thru 7
```

1.9.3.9. * [Asterisk]

To enter the * [Asterisk] in the command line, use one of the options:

- Press and hold Shift + 8 on the internal keyboard
- Press and hold MA + /* on the numeric keypad of the console

Description

The * [Asterisk] is a placeholder which is used to substitute any other character or characters in a name.

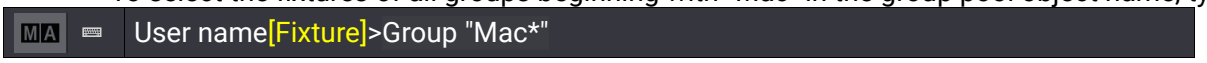
Syntax

[Command] "Name*"

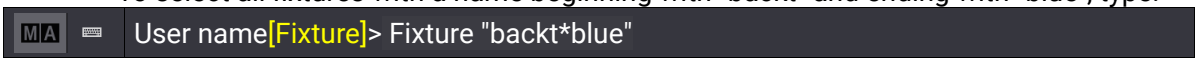
	**Hint:**
	The * [Asterisk] can be positioned anywhere within the name string.

Examples

- To select the fixtures of all groups beginning with "Mac" in the group pool object name, type:



- To select all fixtures with a name beginning with "backt" and ending with "blue", type:



1.9.3.10. % [Percent]

To enter the % [Percent] keyword in the command line, type %.

Description

The % [Percent] keyword is a helping keyword that can be used as a scaling operator.

Syntax

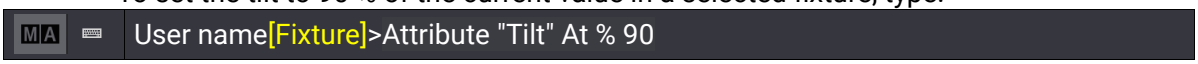
(Attribute ["Attribute_Name" or Attribute_Number]) At % [Value]

Examples

Requirement: Set the dimmer to 70 %

- To reduce the dimmer value from 70 % to 35 % in a selected fixture, type:

- To set the tilt to 90 % of the current value in a selected fixture, type:



1.9.3.11. + [Plus]

To enter the + [Plus] keyword in the command line, use one of the options:

- Press +
- Type **+**

Description

The + [Plus] keyword is a helping keyword that has various functions.

It is used to combine multiple objects in a list or to indicate relative values. When used as a relative indicator without a value, value 1 is used.

If used as a starting keyword, the + [Plus] keyword creates a selection list which is added to the current selection.

Syntax

(Attribute ["Attribute_Name" or Attribute_Number]) At + [Value]

[Function] [Object] ["Object_Name" or Object_Number] + ["Object_Name" or Object_Number]

Examples

- To delete cue 1 and 2 on the selected executor, type:

```
User name[Fixture]> Delete Cue 1 + 2
```

- To add 5 % to the current dimmer value, type:

```
User name[Fixture]>At + 5
```

- To add fixtures 5, 6 and 7 to the current selection, type:

```
User name[Fixture]>+ Fixture 5 Thru 7
```

1.9.3.12. Absolute

To enter the Absolute keyword in the command line, use one of the options:

- Press MA + Time + Time + Time + Time
- Type **Absolute**
- Type the shortcut **Ab**

Description

The Absolute keyword is used to address the absolute layer.

Syntax

Absolute

(Attribute ["Attribute_Name" or Attribute_Number]) At Absolute [Value]

Examples

- To set the layer to absolute, type:

- To set the color red to 10 in the programmer, type:

1.9.3.13. Acceleration

To enter the Acceleration keyword in the command line, use one of the options:

- Type **Acceleration**
- Type the shortcut **Acc**

Description

The Acceleration keyword is used to set the acceleration curve of a step in the phaser.

For more information see **Phasers**.

Syntax

Acceleration

(Attribute ["Attribute_Name" or Attribute_Number]) At Acceleration [Value]

Examples

- To set the layer to accelerate, type:

- To set the acceleration curve of the dimmer to 70, type:

1.9.3.14. Action

To enter the Action keyword in the command line, use one of the options:

- Type **Action**
- Type the shortcut **Actio**

Description

The Action keyword is used to call functions that do not have a designated keyword.

Syntax

Action ["Function"]

Examples

- To store the pan/tilt position to calibration point 1 of the currently selected fixtures, type:

```
User name[Fixture]>Action "StoreCalibrationPoint1"
```

- To call the pan/tilt position of the currently selected fixture of calibration point 2 into the programmer, type:

```
User name[Fixture]>Action "CallCalibrationPoint2"
```

- To solve the stage calibration, type:

```
User name[Fixture]>Action "SolveCalibration"
```

1.9.3.15. Agenda

To enter the Agenda keyword in the command line, use one of the options:

- Type **Agenda**
- Type the shortcut **Age**

Description

The Agenda keyword represents the single agenda events.

For more information see **Agenda**.

Syntax

[Function] Agenda ["Agenda_Name" or Agenda_Number] (/Option)

Option Keywords

The Agenda keyword uses the following option keywords:

- **/Date**

Examples

- To assign sequence 1 to the agenda event 1, type:

```
User name[Fixture]> Assign Sequence 1 At Agenda 1
```

- To give the first agenda event the name "Sunset", type:

```
User name[Fixture]> Label Agenda 1 "Sunset"
```

- To change the mode of the second agenda event to "Dawn", type:

```
User name[Fixture]>Set Agenda 2 Property "Mode" "Dawn"
```

1.9.3.16. ActivationGroup

To enter the ActivationGroup keyword in the command line, use one of the options:

- Type **ActivationGroup**
- Type the shortcut **AG**

Description

The ActivationGroup keyword represents attributes that are activated together.

For more information see **Activation Group**.

Syntax

[Function] ActivationGroup

Example

- To list all activation groups, type:

1.9.3.17. Align

To enter the Align keyword in the command line, use one of the options:

- Press Align
- Type **Align**
- Type the shortcut **Al**

Description

The Align keyword is a function keyword which is used to toggle through the align modes of the attribute encoders.

To toggle through the modes, repeatedly press Align.

Align Mode	Index Number
Off	0
/	1
<	2
>	3
><	4
<>	5

For more information on the align modes, see **Operate Fixtures - Align**.

Syntax

Align ["Align_Mode" or Align_Index_Number]

Examples

- To set the encoders to the first align mode, type:

```
User name[Fixture]>Align "<"
```

- To disable the align mode altogether, type:

```
User name[Fixture]>Align 0
```

1.9.3.18. AlignTransition

To enter the AlignTransition keyword in the command line, use one of the options:

- Type **AlignTransition**
- Type the shortcut **AlignT**

Description

The AlignTransition keyword is used to toggle between the align transition modes:

AlignTransition Mode	Index Number
Linear	0
Sinus	1
Slow	2
Fast	3

Syntax

AlignTransition ["AlignTransition_Name" or AlignTransition_Index_Number]

AlignTransition [Function]

Examples

- To set the AlignTransition to Sinus, type:

```
User name[Fixture]> AlignTransition "Sinus"
```

- To toggle the AlignTransition to the previous align mode, type:

```
User name[Fixture]>AlignTransition Previous
```

- To toggle the AlignTransition to the next align mode, type:

```
User name[Fixture]>AlignTransition Next
```

1.9.3.19. Appearance

To enter the Appearance keyword in the command line, use one of the options:

- Press MA + X4 + X4
- Type **Appearance**
- Type the shortcut **Ap**

Description

The Appearance keyword is used to assign an appearance to an object.

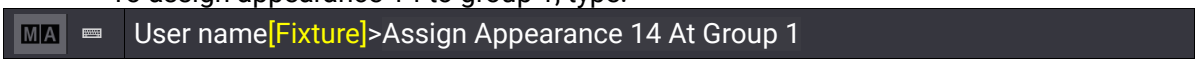
For more information see **Appearance**.

Syntax

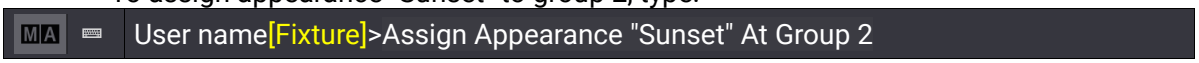
Assign Appearance ["Appearance_Name" or Appearance_Number] At [Object] ["Object_Name" or Object_Number]

Examples

- To assign appearance 14 to group 1, type:



- To assign appearance "Sunset" to group 2, type:



1.9.3.20. Assign

To enter the Assign keyword in the command line, use one of the options:

- Press Assign
- Type **Assign**
- Type the shortcut **As**

Description

The Assign keyword is a function which is used to assign objects to other objects.

To unassign objects in other objects, use the **Assign Off** command.

To set parameters for objects, use the **Set keyword**.

For more information see:

- **Assign macros to keys and buttons**
- **Create new presets**
- **Use appearances**
- **Store and recall views**

Syntax

Assign [Object] ["Object_Name" or Object_Number] At [Object] ["Object_Name" or Object_Number]

Assign Off [Object] ["Object_Name" or Object_Number] At [Object] ["Object_Name" or Object_Number]

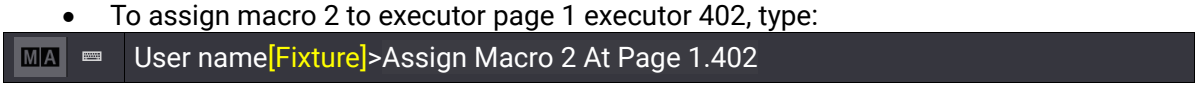
Option Keywords

The Assign keyword uses the following option keywords:

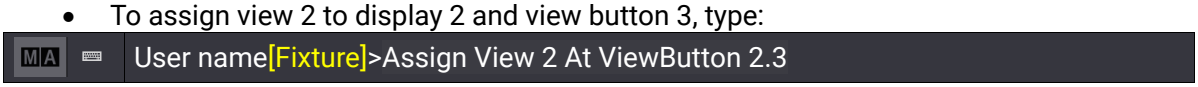
- **/Tab**

Examples

- To assign macro 2 to executor page 1 executor 402, type:



- To assign view 2 to display 2 and view button 3, type:



- To assign appearance 1 to group 5, type:

```
User name[Fixture]>Assign Appearance 1 At Group 5
```

- To unassign tag "ArgleBargle" in sequence 3, type:

```
User name[Fixture]>Assign Off Tag "ArgleBargle" At Sequence 3
```

- To mirror sequence 1 in sequence 2, type:

```
User name[Fixture]>Assign Sequence 1 At Sequence 2
```

- To revert the mirroring of sequence 2 in sequence 1 and keep all settings of sequence 2,

1. Type:

```
User name[Fixture]>Copy Sequence 2 At Sequence 3
```

2. After that, type:

```
User name[Fixture]>Delete Sequence 2
```

	Hint:
	In terms of playback, mirrored sequences are independent from one another. However, their content is linked.

Requirements:

1. Open the grandMA3 demo showfile.
2. Tap view button 1 "Fixture" on screen 1. The Fixture Sheet opens on screen 1.
   - To assign the filter "Only Dimmer" to the Fixture Sheet on screen 1, type:

```
User name[Fixture]>Assign Filter "Only Dimmer" At ScreenContent 1.1.1
```

- To assign the world 1 to the Fixture Sheet on screen 1, type:

```
User name[Fixture]>Assign World 1 At ScreenContent 1.1.1
```

1.9.3.21. At

To enter the At keyword in the command line, use one of the options:

- Press At
- Type **At**
- Type the shortcut **A**

Description

The At keyword is a function keyword and a helping keyword at once.

- As a function keyword it is used to apply values.
- As a helping keyword it is used along with other function keywords to indicate destination.

	**Hint:**
	At applies values live in the programmer. For information on how to apply values throughout the show file see the **Clone keyword**.

At is "the exception that proves the rule". At is one of the few functional keywords which accept objects before the function.

As a starting keyword, At is a function that applies values in the programmer to the current selection.

If value type Fade or Delay is used, the value list will be applied as individual fade/delay times.

Following an object list, At is a function that applies values to the object list. If the object list does not support the At function, the object list is resolved into a selection list and At applies values in the programmer.

Following an object list that follows a function, At is a helping keyword for the starting function.

General Syntax

At ([Value_Type]) [Value]

At [Object] ["Object_Name" or Object_Number]

[Object] ["Object_Name" or Object_Number] At ([Value_Type]) [Value]

[Object] ["Object_Name" or Object_Number] At [Object] ["Object_Name" or Object_Number]

[Destination] At [Source] (If [Object] ["Object_Name" or Object_Number]) (/Option)

Syntax as a Helping Keyword

[Function] [Object] ["Object_Name" or Object_Number] At [Object] ["Object_Name" or Object_Number]

Examples

- To set the MasterFader of the sequence 1 to 30 %, type:

```
User name[Fixture]>FaderMaster Sequence 1 At 30
```

- To set the dimmer attributes of the current selection to 75%, type:

```
User name[Fixture]>At 75
```

- To set the fixture selection to the values of cue 3 in the selected sequence, type:

```
User name[Fixture]>At Cue 3
```

- To set the pan attributes of the selected fixtures to 20%, type:

```
User name[Fixture]>Attribute "Pan" At 20
```

- To set an individual delay time of 2 seconds to attribute 2, type:

```
User name[Fixture]>Attribute 2 At Delay 2
```

- To copy group 4 to group 10, type:

```
User name[Fixture]>Copy Group 4 At 10
```

- To set a speed to 60 using the speed readout specified in the user profile (for example, BPM), type:

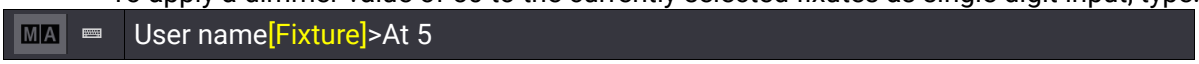
```
User name[Fixture]>At Speed 60
```

	Hint:
	If you use the At command without specifying additional attributes, the natural readout of the dimmer of the user profile will be used.

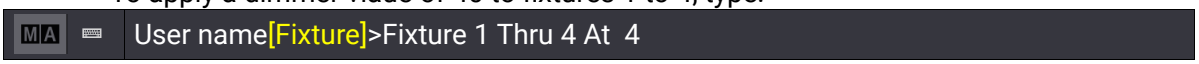
Requriement: Enable sinlge digit input first.

For more information on single digit input and how to enable it see **User Settings**.

- To apply a dimmer value of 50 to the currently selected fixutes as single digit input, type:



- To apply a dimmer vlaue of 40 to fixtures 1 to 4, type:



1.9.3.22. Attribute

To enter the Attribute keyword in the command line, use one of the options:

- Press Preset Preset
- Type **Attribute**
- Type the shortcut **Att**

Description

The Attribute keyword is an object keyword which is used to set attributes of a fixture.

The default function of attributes is Call. If you call an attribute, you can use the encoders to modify the values. Calling attributes also selects the attributes in the fixture sheet.

	**Important:**
	The number of an attribute may vary if new fixtures and attributes are added to the show file. We recommend you use the unique library name of attributes.
	**Hint:**
	• Attributes are organized by subattributes; • Subattributes are organized by features; • Features are organized by feature groups.

Syntax

[Function] Attribute ["Attribute_Name" or Attribute_Number]

Examples

- To view the list of attributes along with their corresponding names and numbers in the command line history, type:

```
User name[Fixture]> List Attribute
```

- To set the attribute "pan" to 120 degrees in the selected fixtures, type:

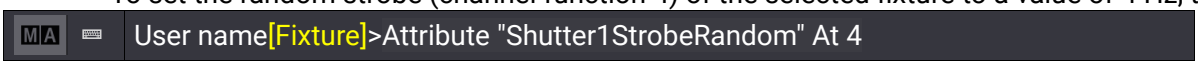
```
User name[Fixture]>Attribute "pan" At 120
```

- To knock out the first attribute, which is Dimmer, in the current selection, type:

```
User name[Fixture]> Off Attribute 1
```

Requirement: Load the demo show and select fixture 1.

- To set the random strobe (channel function 4) of the selected fixture to a value of 4 Hz, type:



1.9.3.23. AutoCreate

To enter the AutoCreate keyword in the command line, use one of the options:

- Type **AutoCreate**
- Type the shortcut **Au** or **Ac**

Description

The AutoCreate keyword creates objects depending on predefined source objects. It can, for example, create groups in the fixture types of the patched fixtures.

	**Hint:** If your selection is defined as source list, the active MAtricks settings will be taken into cosideration whenever executing AutoCreate.

Syntax

AutoCreate [Source_Object] ["Source_Object_Name" or Source_Object_Number] At [Destination_Object] ["Destination_Object_Name" or Destination_Object_Number] (/Option)

Option Keywords

The AutoCreate keyword uses the following option keywords:

- **/All**
- **/ChannelSet**
- **/Merge**
- **/Overwrite**
- **/Single**

Examples

- To create single fixture groups starting with group pool object 1 using all selected fixtures, type:

```
User name[Fixture]>AutoCreate Selection At Group 1
```

- To create single fixture groups for fixtures 1 to 10 starting with group pool object 1, type:

```
User name[Fixture]>AutoCreate Fixture 1 Thru 10 At Group 1
```

Requirement:

1. Insert at least ten fixture types in the show file and patch at least two fixtures of every fixture type.
   - To create a group in the group pool object 42 containing all fixtures of fixture type 10, type:

```
User name[Fixture]>AutoCreate FixtureType 10 At Group 42
```

Requirement:

1. Create layers and classes withtin the patch.
2. Set fixtures to these layers and classes.

Hint:

The demo show already uses these settings.

- To create a group in the group pool object 301 using all patched fixtures that are set to class "Spots" within the patch, type:

```
User name[Fixture]>AutoCreate FixtureClass "Spots" At Group 301
```

- To create single fixture groups starting with group pool object 401 using all fixtures of fixture type 9 that are also set to fixture class "Spots", type:

```
User name[Fixture]>AutoCreate FixtureType 9 + 10 If FixtureClass "Spots"
```

- To create universal dimmer presets with a value increment of 25%, with the first preset being the 11th dimmer preset, type:

```
User name[Fixture]>AutoCreate Universal 1 At Preset 1.11 "DimmerIncrement" 25
```

1.9.3.24. AutoStore

To enter the AutoStore keyword in the command line, use one of the options:

- Type **AutoStore**
- Type the shortcut **AS**
- Type the shortcut **Autos**

Description

The AutoStore keyword is used to directly store presets in a fixture type. This can be useful, for example, when you have to use a fixture type in a different show file, and want to save time by creating your own standard set of presets.

Syntax

AutoStore FixtureType ["FixtureType_Name" or FixtureType_Number]

AutoStore Fixture ["Fixture_Name" or Fixture_Number]

AutoStore Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] At FixtureType ["FixtureType_Name" or FixtureType_Number]

Option Keywords

The AutoStore keyword uses the following option keywords:

- **/Merge**
- **/NoConfirmation**
- **/Overwrite**

Examples

- To store your presets into the fixture type Mac Aura XB, type:

```
User name[Fixture]>AutoStore FixtureType "Mac Aura XB"
```

- To store all possible presets to fixture type 9, type:

```
User name[Fixture]>AutoStore Preset *.* At FixtureType 9
```

- To store possible color presets to the fixture type of fixture 1, merging existing fixture type presets, type:

```
User name[Fixture]>AutoStore Preset 4.1 Thru At Fixture 1 /Merge /NoConfirmation
```

1.9.3.25. Bitmap

To enter the Bitmap keyword in the command line, use one of the options:

- Type **Bitmap**
- Type the shortcut **Bi**

Description

The Bitmap keyword allows you to use media files such as images or videos, also called bitmap generator objects, and map them to the selection of fixtures.

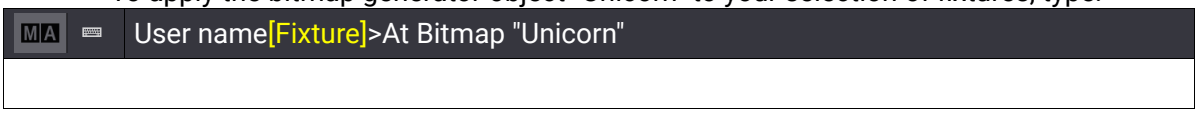
Syntax

At Bitmap ["Bitmap_Name" or Bitmap_Number]

Example

Requirement:
Select fixtures and arrange them in the selection grid. By doing so you create a "fixture canvas" which, on the other hand, is used to map the media file of this bitmap object to your selection of fixtures.

- To apply the bitmap generator object "Unicorn" to your selection of fixtures, type:



1.9.3.26. Black

To enter the Black keyword in the command line, use one of the options:

- Press MA + <<<
- Type **Black**
- Type the shortcut **Bla**

Description

Black is a playback keyword that is used to temporarily override the master level to zero on executing objects.

For more information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Black (On or Off) [Object] ["Object_Name" or Object_Number]

Examples

- To override the master level of executor 201 to zero, type:

MA | User name[Fixture]>Black On Executor 201

- To return the master level of executor 201 to the master fader, type:

MA | User name[Fixture]>Black Off Executor 201

1.9.3.27. Blind

To enter the Blind keyword in the command line, use one of the options:

- Press Blind
- Type **Blind**
- Type the shortcut **BI**

Description

The Blind keyword is used to suppress the output of the live programmer. It is possible to program without putting the content live. Disabling Blind activates the programmer and the changes made during the mode.

Use the **DMX sheet** to see that the output is set to 0.

	**Hint:**
	Blind is a toggle function which means that entering Blind without using a helping keyword enables or disables it.

Syntax

Blind (On or Off)

Example

- To enable Blind and to disable the output of the programmer in return, type:

1.9.3.28. Block

To enter the Block keyword in the command line, use one of the options:

- Type **Block**
- Type the shortcut **Blo**

Description

Block is a function used to add data and prevent them from tracking. Tracked values are converted to stored values.

If the object list does not contain any references to any cues, the Block function is applied to the selected sequence.

If syntax does not contain any selection list filter, all fixtures will be used.

If syntax does not contain any attribute list filter, all attributes will be used.

Syntax

Block (Sequence ["Sequence_Name" or Sequence_Number]) Cue ["Cue_Name" or Cue_Number] (If Fixture ["Fixture_Name" or Fixture_Number] FeatureGroup ["FeatureGroup_Name" or FeatureGroup_Number] or Attribute ["Attribute_Name" or Attribute_Number] EndIf)

Examples

- To block all parameters in cue 2 of the selected sequence, type:

```
User name[Fixture]>Block Cue 2
```

To unblock parameters, use the **Unblock keyword**.

- To block pan and tilt of fixture 4 in cue 5 of the selected sequence, type:

```
User name[Fixture]>Block Cue 5 If Fixture 4 FeatureGroup "Position" EndIf
```

1.9.3.29. BPM

To enter the BPM keyword in the command line, use one of the options:

- Type **BPM**
- Type the shortcut **Bp**

Description

The BPM keyword is used to set the speed of a fixture selection or a speed master using the unit BPM.

Syntax

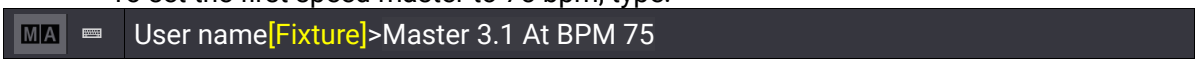
At Speed BPM [Value]

Master [Master_Number] At BPM [Value]

Examples

- To set the speed layer to 5 bpm, type:

- To set the first speed master to 75 bpm, type:



1.9.3.30. Call

To enter the Call keyword in the command line, use one of the options:

- Press On On
- Type **Call**
- Type the shortcut **Cal**

Description

The Call keyword is used to apply an object or its content.

Syntax

Call [Object] ["Object_Name" or Object_Number] (/Option)

Option Keywords

The Call keyword uses the following option keywords:

- **/Screen**

Examples

- To call view button 2.1, type:

- To call macro 2, type:

1.9.3.31. CancelSoftwareUpdate

To enter the CancelSoftwareUpdate keyword in the command line, use one of the options:

- Type **CancelSoftwareUpdate**
- Type the shortcut **Can**

Description

The CancelUpdateSoftware keyword is a function keyword which is used to cancel the running process of sending grandMA3 update files to other grandMA3 devices in the network.

Syntax

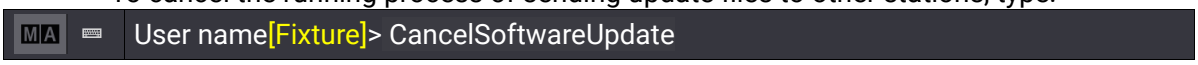
CancelSoftwareUpdate

Example

Requirement:

- Device must be in the network;
- Device must send update files to other stations.

- To cancel the running process of sending update files to other stations, type:



1.9.3.32. Camera

To enter this keyword into the console, use one of the options:

- Type **Camera**
- Type the shortcut **Cam**

Description

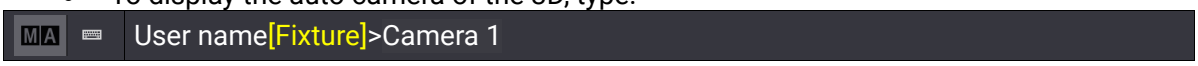
The Camera keyword selects a camera in the camera pool.

Syntax

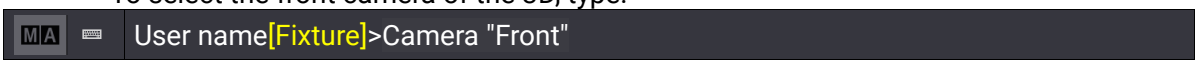
([Function]) Camera ["Camera_Name" or Camera_Number]

Examples

- To display the auto camera of the 3D, type:



- To select the front camera of the 3D, type:



1.9.3.33. Capture

To enter the Capture keyword in the command line, use one of the following options:

- Press Stomp Stomp
- Type **Capture**
- Type the shortcut **Cap**

Description

The Capture keyword is a command keyword which is used to activate the current output values of specified parameters.

Capturing parameters during a running phaser or cue fade activates the actual values at the moment the command is executed. The result is a single step of static values resembling a freeze frame of the output.

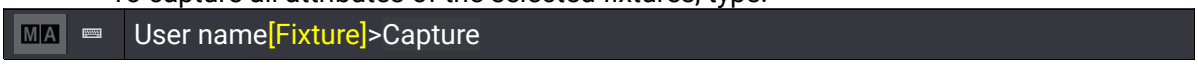
	**Important:**
	The capture command produces only numeric values, losing any existing preset references.
	**Important:**
	The capture command translates the output of each parameter into the appropriate value on the absolute layer while activating a 0 value on the relative layer. For any parameter where the output was previously composed of a combination of absolute and relative values, the output will appear the same while relying only on the absolute layer with no relative offset.

Syntax

Capture [Object] ["Object_Name" or Object_Number]

Examples

- To capture all attributes of the selected fixtures, type:



- To capture only the position attributes of the selected fixtures, type:

1.9.3.34. ChangeDestination

To enter the ChangeDestination keyword in the command line, use one of the options:

- Type **ChangeDestination**
- Type the shortcut **CD** or **Chang**

Description

The ChangeDestination keyword is a function keyword used to change the current destination of the command line.

Syntax

ChangeDestination ["Element_Name" or Element_Number]

ChangeDestination Root

ChangeDestination ..

Examples

- To enter the first element of the current destination, type:

```
User name[Fixture]>ChangeDestination 1
```

Result:

```
User name@MessageCenter>
```

- To enter the element of the current destination called "Sequence", type:

```
User name[Fixture]>ChangeDestination "Sequence"
```

- To leave the destination "Sequence", type:

```
User name@Menus> ChangeDestination Root
```

Result:

```
User name[Fixture]>
```

- To go one level back in the tree structure, type:

```
User name@MessageCenter/Undefined> ChangeDestination ..
```

Result:

1.9.3.35. Channel

To enter the Channel keyword in the command line, use one of the options:

- Press Channel
- Type **Channel**
- Type the shortcut **Chann**
- Type **F** + **1**

Description

The Channel keyword is an object keyword used to select fixtures of the ID type Channel.

Use the Channel keyword as a default keyword for the command line when programming with channels. For more information see **Workspace - Command Line - Change the Default Keyword**.

Syntax

Channel ["Channel_Name" or Channel_Number]

Example

- To select channel ID 10, type:

1.9.3.36. ChannelFunctionDefault

To enter the ChannelFunctionDefault keyword in the command line, use one of the options:

- Type **ChannelFunctionDefault**
- Type the shortcut **ChannelF**

Description

The ChannelFunctionDefault keyword is used to set all attributes to the default value of the first channel function.

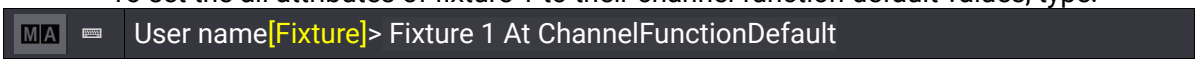
Syntax

ChannelFunctionDefault

Fixture ["Fixture_Name" or Fixture_Number] At ChannelFunctionDefault

Example

- To set the all attributes of fixture 1 to their channel function default values, type:



1.9.3.37. ChannelSet

To enter the ChannelSet keyword in the command line, use one of the options:

- Type **ChannelSet**
- Type the shortcut **ChannelS**

Description

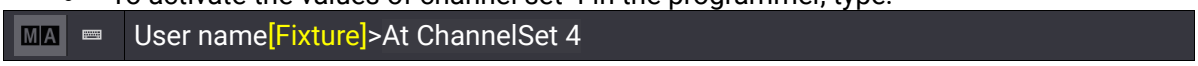
The ChannelSet keyword is used to take channel sets, such as gobos, actively into the programmer.

Syntax

At ChannelSet ["ChannelSet_Name" or ChannelSet_Number]

Example

- To activate the values of channel set 4 in the programmer, type:



1.9.3.38. Chat

To enter the Chat keyword in the command line, use one of the options:

- Type **Chat**

Description

The Chat keyword is used to send messages via the world server in chat channels that were already joined. These chat messages are displayed in the message center.

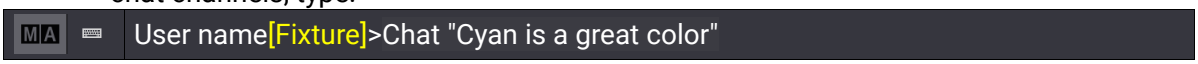
Syntax

Chat "Text"

Example

Requirement: Join a chat channel using the **ChatJoin Keyword**

- To send the text "Cyan is a great color" as a chat message to all devices that joined the same chat channels, type:



1.9.3.39. ChatJoin

To enter the ChatJoin keyword in the command line, use one of the options:

- Type **ChatJoin**
- Type the shortcut **ChatJ**

Description

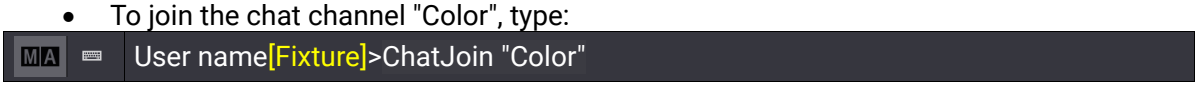
The ChatJoin keyword is used to join a chat channel.

Syntax

ChatJoin ["ChatChannel_Name"]

Example

- To join the chat channel "Color", type:



1.9.3.40. ChatLeave

To enter the ChatLeave keyword in the command line, use one of the options:

- Type **ChatLeave**
- Type the shortcut **ChatL**

Description

The ChatLeave keyword is used to leave a chat channel.

Syntax

ChatLeave ["ChatChannel_Name"]

Example

Requirement: Join a chat channel first

- To leave the chat channel "Color", type:

1.9.3.41. CleanUp

To enter the CleanUp keyword in the command line, use one of the options:

- Type **CleanUp**
- Or type the shortcut **Clean**

Description

The CleanUp keyword is a command keyword that is used to delete objects that are not used and do not contain references in the show file. For example, you can clean up sequences that are not assigned to an executor.

Syntax

CleanUp [Object] ["Object_Name" or Object_Number] (/Option)

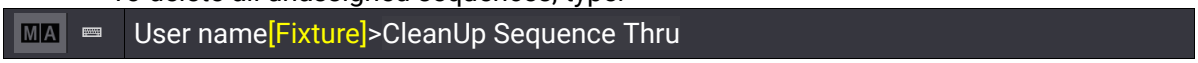
Option Keywords

The CleanUp keyword uses the following option keywords:

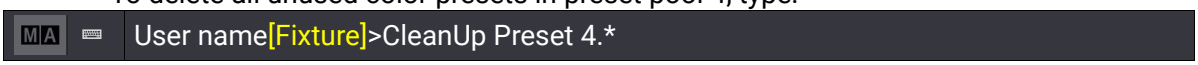
- **/Recipe**
- **/Selective**
- **/Type**

Examples

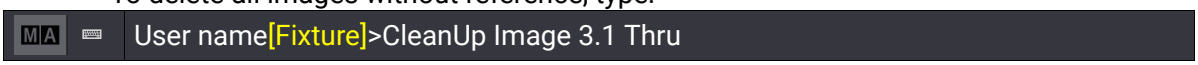
- To delete all unassigned sequences, type:



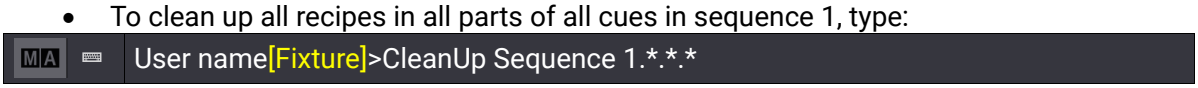
- To delete all unused color presets in preset pool 4, type:



- To delete all images without reference, type:



- To clean up all recipes in all parts of all cues in sequence 1, type:



1.9.3.42. Clear

To enter the Clear keyword in the command line, use one of the options:

- Press Clear
- Type **Clear**
- Type the shortcut **CL**

Description

The Clear keyword is a function keyword which is used to clear the selection, active values, or the programmer.

Depending on the status of the programmer the Clear keyword successively:

1. Clears selection (deselects all fixtures);
2. Clears active values (deactivates all values);
3. Clears all (empties the entire programmer).

For information on the key see the **Clear Key**.

For more information on each function of the Clear keyword see **ClearSelection**, **ClearActive**, **ClearAll**.

Syntax

Clear

Example

- To clear the selection, active values or the programmer depending on the status or the content of the programmer, type:

1.9.3.43. ClearActive

To enter the ClearActive keyword in the command line, use one of the options:

- Type **ClearActive**
- Type the shortcut **Ca**

Description

The ClearActive keyword is used to deactivate all values in the programmer.

For information on the key see the **Clear Key**.

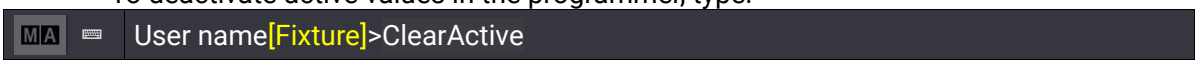
For more information on the additional functions of the Clear keyword see **Clear**, **ClearAll**, **ClearSelection**.

Syntax

ClearActive

Example

- To deactivate active values in the programmer, type:



1.9.3.44. ClearAll

To enter the ClearAll keyword in the command line, use one of the options:

- Type **ClearAll**
- Type the shortcut **ClearA**

Description

The ClearAll function clears the selection and discards all values in the programmer.

For information on the key see the **Clear Key**.

For more information on the additional functions of the Clear keyword see **Clear**, **ClearActive**, **ClearSelection**.

Syntax

ClearAll

Example

- To clear the programmer, type:

1.9.3.45. ClearSelection

To enter the ClearSelection keyword in the command line, use one of these options:

- Type **ClearSelection**
- Type the shortcut **Cs**

Description

The ClearSelection keyword is a function keyword which is used to deselect the selected fixtures.

For information on the key see the **Clear Key**.

For more information on the additional functions of the Clear keyword see **Clear**, **ClearActive**, **ClearAll**.

Syntax

ClearSelection

Example

- To deselect selected fixtures, type:

1.9.3.46. Clone

To enter the Clone keyword in the command line, use one of the options:

- Press MA + X1 | Clone
- Type **Clone**
- Type **Clo**

Description

Clone replicates data from one fixture or selection to another throughtout the show file. For more information on applying values live in the programmer only, see the **At keyword**.

Syntax

Clone [Object] ["Object_Name" or Object_Number] At [Object] ["Object_Name" or Object_Number] (If [Object] ["Object_Name" or Object_Number] /Option)

Option Keywords

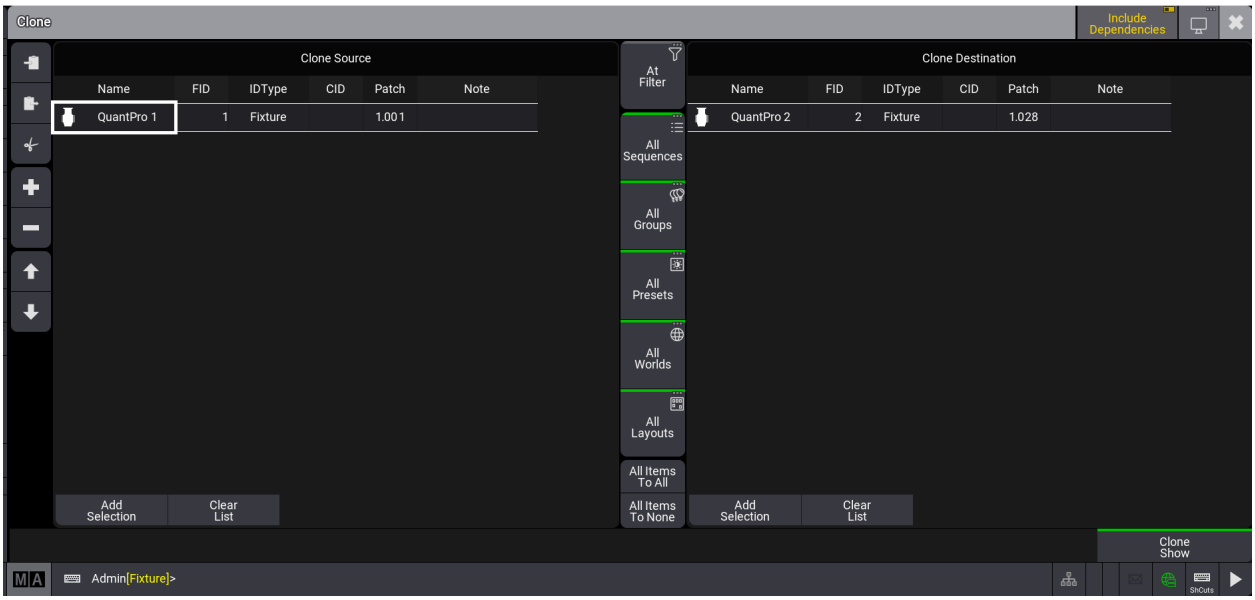
The Clone keyword uses the following option keywords:

- **/MergeLowPriority**
- **/MergeHighPriority**
- **/NoDependencies**
- **/Overwrite**

Examples

- To clone the entire show data from fixture 1 to fixture 2, type:

The Clone window opens.



Clone Window

For more information on cloning in the general user interface see **Clone**.

- To clone data from fixtures 1 and 2 to fixtures in group 10 within sequence 1 thru 10C only, type:

```
User name[Fixture]>Clone Fixture 1 + 2 At Group 10 If Sequence 1 Thru 10
```

- To clone all objects within data pool 1 from fixtures 1 thru 12 to fixtures 101 thru 112, type:

```
User name[Fixture]>Clone Fixture 1 Thru 12 At Fixture 101 Thru 112 If DataPool 1
```

For more information on data pools, see the **Data Pools**.

- To clone the pan attribute of fixture 1 to the tilt attribute of fixture 2, type:

```
User name[Fixture]>Clone Fixture 1 Attribute "Pan" At Fixture 2 Attribute "Tilt"
```

Hint:

When cloning presets and cue data, all presets referenced by the destination objects are automatically cloned in addition.

Hint:

The syntax:
Clone [Source_Object] ["Object_Name" or Object_Number] At [Destination_Object] ["Object_Name" or Object_Number] (/Option)
produces the same results as:
[Destination_Object] ["Object_Name" or Object_Number] At [Source_Object] ["Object_Name" or Object_Number] (/Option)
All options described above work the same for both syntax structures.
When cloning without the **Clone** keyword, keep in mind that the source and destination change places within the syntax.
For more information on the **At** keyword, see the **At keyword**.

1.9.3.47. CommandDelay

To enter the CommandDelay keyword in the command line, use one of the options:

- Type **CommandDelay**
- Type the shortcut **CommandD**

Description

The CommandDelay sets the delay time when executing commands in a cue.

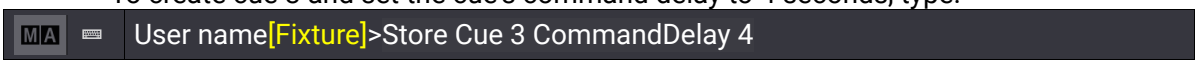
Syntax

CommandDelay [CommandDelay_Value]

([Function]) (Sequence ["Sequence_Name" or Sequence_Number]) Cue ["Cue_Name" or Cue_Number] CommandDelay [CommandDelay_Value]

Example

- To create cue 3 and set the cue's command delay to 4 seconds, type:



1.9.3.48. Collect

To enter the Collect keyword in the command line, use one of the options:

- Press Select Select
- Type **Collect**
- Type the shortcut **Coll**

Description

The Collect keyword is used to add objects of the same type to a collection. For more information on collection see **Collection Keyword**.

	**Hint:**
	It is possible to collect empty pool objects during import or when storing.

Syntax

Collect [Object] ["Object_Name" or Object_Number]

Examples

- To add the second preset of the color preset pool to the collection, type:

```
User name[Fixture]> Collect Preset 4.2
```

- To add a different color preset to the collection, type:

```
User name[Fixture]>Collect Preset 4.10
```

- To reset a collection, type:

```
User name[Fixture]>Collect
```

1.9.3.49. Collection

To enter the Collection keyword in the command line, use one of the options:

- Press Fixture + Fixture + Fixture
- Type **Collection**
- Type the shortcut **Collecti**

Description

The Collection keyword is an object keyword used to represent the current collection of objects.

	**Important:**
	A collection stays always at the same spot it was collected at.
	**Hint:**
	Pool objects that are part of a collection have an orange frame around them. They also display the order of the collection which is located on top of a pool object.

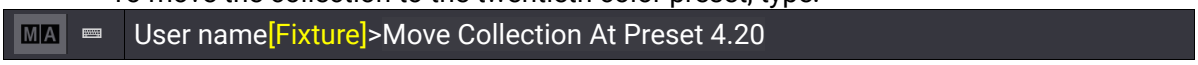
.Syntax

[Function] Collection At [Object] ["Object_Name" or Object_Number]

Examples

Requirement: Create a collection of color presets

- To move the collection to the twentieth color preset, type:



1.9.3.50. Color

To enter the Color keyword in the command line, type **Color**.

Description

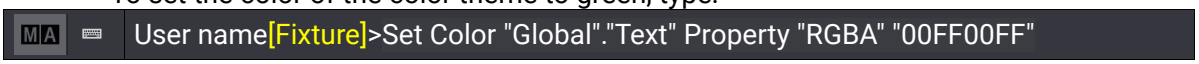
The Color keyword is an object keyword which is used, for example, to adjust the color of the color theme to your liking.

Syntax

Set Color ["ColorGroup_Name"."Text"] Property ["Property_Name" "PropertyReadout"]

Examples

- To set the color of the color theme to green, type:



1.9.3.51. ColorDefinition

To enter the ColorDefinition keyword in the command line, use one of the options:

- Type **ColorDefinition**
- Type the shortcut **ColorD**

Description

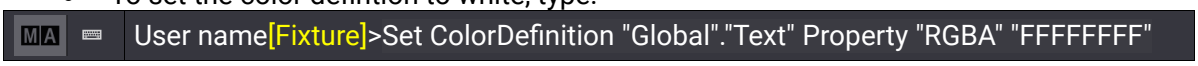
The ColorDefinition keyword is an object keyword which is used, for example, to adjust the color defintion of the color theme to your liking.

Syntax

[Function] ColorDefinition ["ColorGroup_Name"."Name"] (Property ["Property_Name"] ["Property_Value"])

Examples

- To set the color defintion to white, type:



1.9.3.52. ColorTheme

To enter the ColorTheme keyword in the command line, use one of the options:

- Type **ColorTheme**
- Type the shortcut **Col**

Description

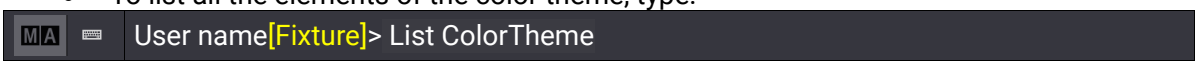
ColorTheme is an object keyword which is used, for example, to adjust the color theme.

Syntax

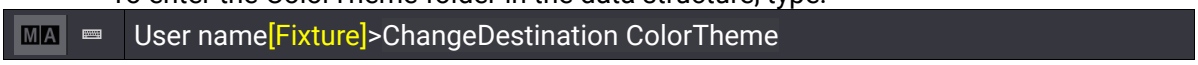
[Function] ColorTheme

Examples

- To list all the elements of the color theme, type:



- To enter the ColorTheme folder in the data structure, type:



1.9.3.53. Configuration

To enter the Configuration keyword in the command line, use one of the options:

- Type **Configuration**
- Type the shortcut **Con**

Description

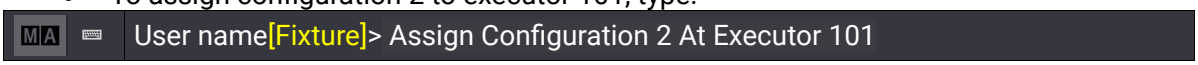
A configuration contains a set of different button, key, fader, and encoder functions. It can be assigned to an executor.

Syntax

[Function] Configuration ["Configuration_Name or Configuration_Number] At [Object] ["Object_Name" or Object_Number]

Example

- To assign configuration 2 to executor 101, type:



1.9.3.54. Console

To enter the Console keyword in the command line, use one of the options:

- Type **Console**
- Type the shortcut **Cons**

Description

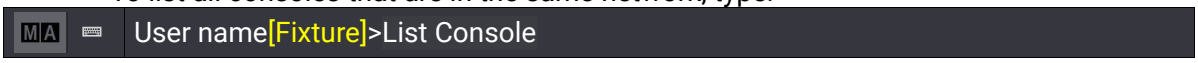
The Console keyword is an object keyword which is used to address all grandMA3 consoles in the network.

Syntax

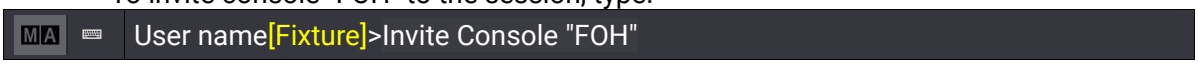
[Function] Console ["Console_Name" or Console_Number]

Examples

- To list all consoles that are in the same network, type:



- To invite console "FOH" to the session, type:



1.9.3.55. Cook

To enter the Cook keyword in the command line, use one of the options:

- Press MA + Update
- Type **Cook**

Description

The Cook keyword is a command keyword which is used to concoct recipes of objects without accessing the editor.

For more information see **Recipes**.

Syntax

Cook [Object] ["Object_Name" or Object_Number] (/Option)

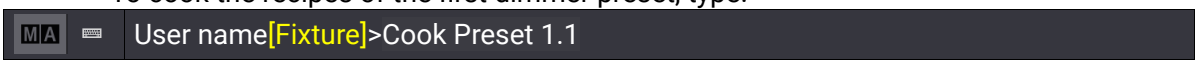
Option Keywords

The Cook keyword uses the following option keywords:

- **/Merge**
- **/MergeLowPriority**
- **/Overwrite**
- **/Remove**

Example

- To cook the recipes of the first dimmer preset, type:



For more information see **Recipes**.

1.9.3.56. Copy

To enter the Copy keyword in the command line, use one of these options:

- Press Copy
- Type **Copy**
- Type the shortcut **Co**

Description

The Copy keyword is a function keyword which is used to create copies of an object.

If no object type is given and the command line destination is root (no destination), the default object type **– Cue –** is used for this function.

Syntax

Copy [Object] ["Source_Name" or Source_Number] At ["Destination_Name" or Destination_Number] (/Option)

Copy [Object] ["Source_Name" or Source_Number] (/Option)

Option Keywords

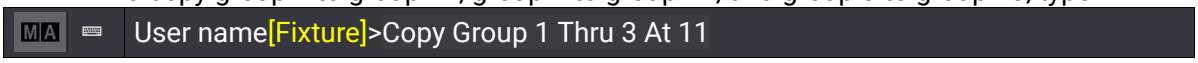
The Copy keyword uses the following option keywords:

- **/CopyCueDestination**
- **/CopyCueSource**
- **/Default**
- **/Merge**
- **/NoConfirmation**
- **/Overwrite**
- **/Release**

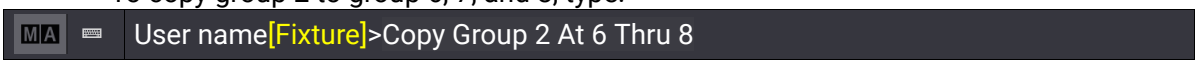
Examples

- To copy group 1 to group 5, type:

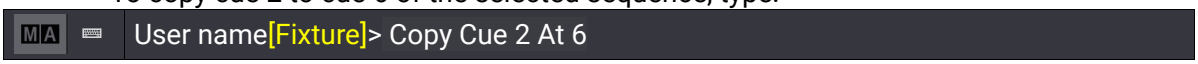
- To copy group 1 to group 11; group 2 to group 12; and group 3 to group 13, type:



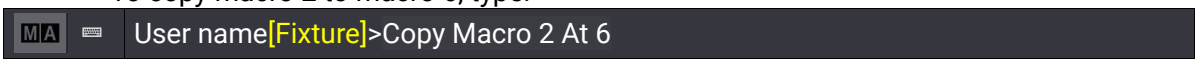
- To copy group 2 to group 6, 7, and 8, type:



- To copy cue 2 to cue 6 of the selected sequence, type:



- To copy macro 2 to macro 6, type:



1.9.3.57. Cue

To enter the Cue keyword in the command line, use one of the options:

- Press Cue
- Type **Cue**

Description

	**Important:**
	Cue is the only object type that accepts numerical IDs as decimal fractions. The ID which is allowed for cues ranges from 0.001 to 9999.999. In all other objects, a dot indicates the ID of a parent or a child object.

Cue is an object type holding a look on stage.

Cues are arranged in sequences and are divided into parts.

For more information see the **Cue and Sequence section**.

The **Cue** keyword is an object keyword. Object keywords must have a function keyword in front of them to create a complete command. For more information on how to use the command syntax, see **General Syntax Rules**.

Cue has a default function called **SelFix**. This selects all the fixtures that have stored values in the cue.

If a sequence is not specified, then the selected sequence is used in the command.

Syntax

[Function] Cue ["Cue_Name" or Cue_Number]

[Function] Sequence ["Sequence_Name" or Sequence_Number] Cue ["Cue_Name" or Cue_Number] ([Setting] ["Setting_Value"] (/Option)

Assign [Object] ["Object_Name" or Object_Number] At (Sequence ["Sequence_Name" or Sequence_Number]) Cue ["Cue_Name" or Cue_Number]

Settings

In some settings you have to assign an object. These can be assigned using the **Assign keyword**.

Other settings may contain a text option or a value. Use the **Set keyword** for these settings.

Cues can also contain settings for MAtricks and recipe values. These can be changed using the **Set Keyword**.

Option Keywords

The Cue keyword uses the following option keywords:

- **/Active**
- **/ActiveForSelected**
- **/AddNewContent**
- **/All**
- **/AllForSelected**
- **/Ask**
- **/CopyCueDestiation**
- **/CopyCueSource**
- **/CreateSecondCue**
- **/CueOnly**
- **/Default**
- **/DMX**
- **/GridMergeMode**
- **/Look**
- **/Merge**
- **/NoConfirmation**
- **/Output**
- **/Overwrite**
- **/Programmer**
- **/Release**
- **/Remove**

Examples

- To select the fixtures with values stored in cue 3 of a selected sequence, type:

```
User name[Fixture]>Cue 3
```

The default function for **Cue** is **SelFix** so **Cue 3** is the same as **SelFix Cue 3**.

- To delete cue 2.5 in the selected sequence, type:

```
User name[Fixture]>Delete Cue 2.5
```

- To store cue 2 in sequence 5, type:

```
User name[Fixture]>Store Sequence 5 Cue 2
```

- To store cue 3 in sequence 5 with a cue fade time of 7 seconds and an outfade of 11, type:

```
User name[Fixture]>Store Sequence 5 Cue 3 CueFade 7/11
```

1.9.3.58. CopyCrashLog

To enter the CopyCrashLog keyword in the command line, use one of the options:

- Type **CopyCrashLog**
- Type the shortcuts **Copyc**

Description

CopyCrashLog is a function keyword that is used to copy crash logs into the crash logs folder in the gma3_library folder on any device.

Syntax

CopyCrashLog (If Drive ["Drive_Name" or Drive_Number] /Option)

Option Keywords

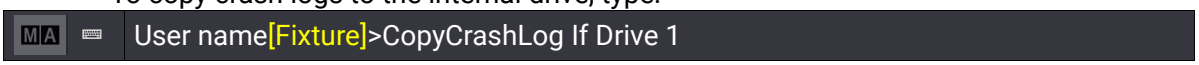
The CopyCrashLog keyword uses the following option keywords:

- **/Remove**

Examples

- To copy crash logs to one or more connected USB drives, type:

- To copy crash logs to the internal drive, type:



1.9.3.59. CueAbsolute

To enter the CueAbsolute keyword in the command line, use one of the options:

- Type **CueAbsolute**
- Type the shortcut **CueA**

Description

CueAbsolute is a layer keyword. It is used to change the selected layer to the CueAbsolute layer. This layer can show information in, for instance, the fixture sheet. The layer shows the cue ID number for each parameter that has an absolute value from the active cues.

If a fixture sheet is set to **Auto** layer, then it will automatically change to show the selected layer.

A cue is composed of **[Sequence_Number].[Cue_Number]:[Part_Index]**

Syntax

CueAbsolute

Examples

- To select the CueAbsolute layer, type:

1.9.3.60. CueDelay

To enter the CueDelay keyword in the command line, use one of the options:

- Press Time Time (If the Time Key Target is set to Cue. For more information see **User Settings** and **Time Key**.)
- Type **CueDelay**
- Type the shortcut **Cued**

Description

CueDelay can set both the indelay and the outdelay time of a cue. To do so, use a /. See examples further down.

For more information on how to set the delay times in objects see the **Delay Keyword**.

Syntax

(Cue ["Cue_Name" or Cue_Number]) CueDelay [CueDelay_Time]

(Cue ["Cue_Name" or Cue_Number]) CueDelay [CueInDelay_Time]/[CueOutDelay_Time]

Examples

- To enter a delay of 5 seconds in the current cue of the selected sequence, type:

```
User name[Fixture]>CueDelay 5
```

- To set an indelay of 6 seconds and an outdelay of 12 seconds in the current cue of the selected sequence, type:

```
User name[Fixture]>CueDelay 6/12
```

- To adjust the CueInDelay to 3 seconds, but leave the CueOutDelay as it was, type:

```
User name[Fixture]>CueDelay 3/
```

- To enter a delay of 5 seconds in cues 1 to 4 of the selected sequence, type:

```
User name[Fixture]>Cue 1 Thru 4 CueDelay 5
```

1.9.3.61. CueFade

To enter the CueFade keyword in the command line, use one of the options:

- Press Time (If the Time Key Target is set to Cue. For more information see **User Settings** and **Time Key**.)
- Type **CueFade**
- Type **Cuef**

Description

CueFade can set both the infade and the outfade time of a cue. To do so, use a /. See examples further down.

As a helping keyword for programming functions (for example Store), this keyword sets the fade time of a cue or a cue part.

Syntax

(Cue ["Cue_Name" or Cue_Number]) CueFade [CueFade_Time]

(Cue ["Cue_Name" or Cue_Number]) CueFade [CueInFade_Time]/[CueOutFade_Time]

Examples

- To enter a fade of 5 seconds in a cue, type:

```
User name[Fixture]>CueFade 5
```

- To set an infade of 6 seconds and an outfade of 12 seconds in the current cue of the selected sequence, type:

```
User name[Fixture]> CueFade 6/12
```

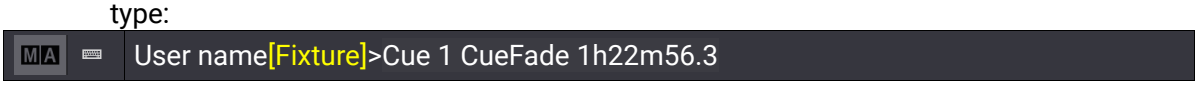
- To adjust the CueInFade to 3 seconds, but leave the CueOutFade as it was, type:

```
User name[Fixture]>CueFade 3/
```

- To enter a fade of 5 seconds in cues 1 to 4, type:

```
User name[Fixture]>Cue 1 Thru 4 CueFade 5
```

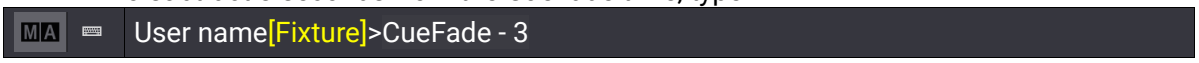
- To enter a fade time of 1 hour 22 minutes 56.3 seconds in cue 1 of the selected sequence, type:



Or press:

- To double the CueFade time, type:

- To subtract 3 seconds from the CueFade time, type:



1.9.3.62. CueInDelay

To enter the CueInDelay keyword in the command line, use one of the options:

- Type **CueInDelay**
- Type the shortcut **Cueind**

Description

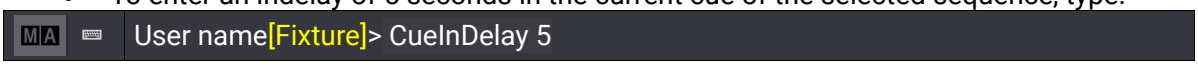
CueInDelay sets the delay time of a cue.

Syntax

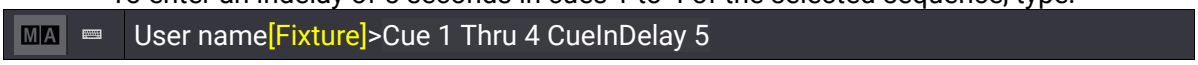
(Cue ["Cue_Name or Cue_Number]) CueInDelay [CueInDelay_Time]

Examples

- To enter an indelay of 5 seconds in the current cue of the selected sequence, type:



- To enter an indelay of 5 seconds in cues 1 to 4 of the selected sequence, type:



1.9.3.63. CueInFade

To enter the CueInFade keyword in the command line, use one of the options:

- Type **CueInFade**
- Type the shortcut **Cuein**

Description

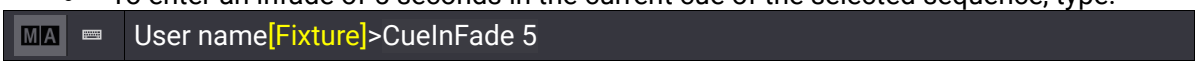
CueInFade sets the infade time of a cue.

Syntax

(Cue ["Cue_Name" or Cue_Number]) CueInFade [CueInFade_Time]

Examples

- To enter an infade of 5 seconds in the current cue of the selected sequence, type:



1.9.3.64. CueOutDelay

To enter the CueOutDelay keyword in the command line, use one of the options:

- Type **CueOutDelay**
- Type the shortcut **Cueoutd**

Description

CueOutDelay sets the outdelay time of a cue.

	Hint:
	CueOutDelay is only used by dimmer parameters that go to a lower value in the cue.

Syntax

(Cue ["Cue_Name" or Cue_Number]) CueOutDelay [CueOutDelay_Time]

Examples

- To enter an outdelay of 5 seconds in the current cue of the selected sequence, type:

```
User name[Fixture]>CueOutDelay 5
```

- To enter an outdelay of 5 seconds in cues 1 to 4 of the selected sequence, type:

```
User name[Fixture]>Cue 1 Thru 4 CueOutDelay 5
```

1.9.3.65. CueOutFade

To enter the CueOutFade keyword in the command line, use one of the options:

- Type **CueOutFade**
- Type the shortcut **Cueo**

Description

CueOutFade sets the outfade time of a cue.

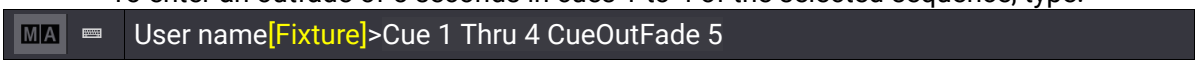
	**Hint:**
	CueOutFade is only used by dimmer parameters that go to a lower value in the cue.

(Cue ["Cue_Name" or Cue_Number]) CueOutFade [CueOutFade_Time]

Examples

- To enter an outfade of 5 seconds in the current cue of the selected sequence, type:

- To enter an outfade of 5 seconds in cues 1 to 4 of the selected sequence, type:



1.9.3.66. CueRelative

To enter the CueRelative keyword in the command line, use one of the options:

- Type **CueRelative**
- Type the shortcut **CueR**

Description

CueRelative is a layer keyword. It is used to change the selected layer to the CueRelative layer. This layer can show information in, for instance, the fixture sheet. The layer shows the cue ID number for each parameter that has a relative value from the active cues.

If a fixture sheet is set to **Auto** layer, then it will automatically change to show the selected layer.

A cue is composed of **[Sequence_Number].[Cue_Number]:[Part_Index]**

Syntax

CueRelative

Example

- To select the CueRelative layer, type:

1.9.3.67. CueUpdate

To enter the CueUpdate keyword in the command line, use one of the options:

- Type **CueUpdate**
- Type the shortcut **CueU**

Description

CueUpdate keyword is an object keyword which addresses all cues that can be updated.

Syntax

[Function] CueUpdate [CueUpdate_Number]

Examples

- To list all the cues that can be updated in the command line feedback, type:

```
User name[Fixture]>List CueUpdate
```

It is the same list shown in the **Update menu**.

	Important:
	The number relates to the first element in the list. It does not represent the cue number.

- To update the first cue in the list, type:

```
User name[Fixture]>Update CueUpdate 1
```

1.9.3.68. CurrentEnvironment

To enter the CurrentEnvironment keyword in the command line, use one of the options:

- Type **CurrentEnvironment**
- Type the shortcut **CurEnv**

Description

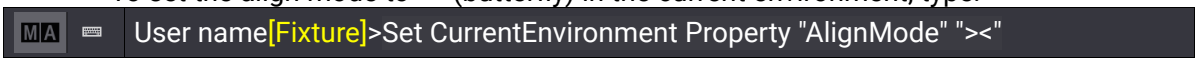
The CurrentEnvironment keyword addresses the currently selected environment of the current user - Live or Preview.

Syntax

[Function] CurrentEnvironment (Property ["Property_Name"] ["Value"])

Example

- To set the align mode to >< (butterfly) in the current environment, type:



1.9.3.69. CurrentUser

To enter the CurrentUser keyword in the command line, use one of the options:

- Type **CurrentUser**
- Type the shortcut **Cu**

Description

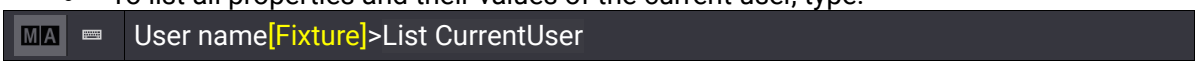
CurrentUser represents the user that is currently logged in.

Syntax

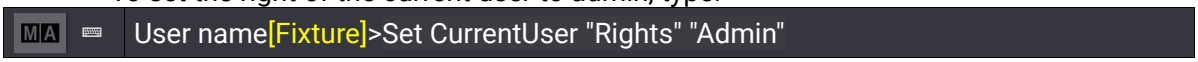
[Function] CurrentUser (["Property"] ["Value"])

Examples

- To list all properties and their values of the current user, type:



- To set the right of the current user to admin, type:



1.9.3.70. CurrentUserProfile

To enter the CurrentUserProfile keyword in the command line, use one of the options:

- Type **CurrentUserProfile**
- Type the shortcut **CUP**
- Or type the shortcut **CurrentUserP**

Description

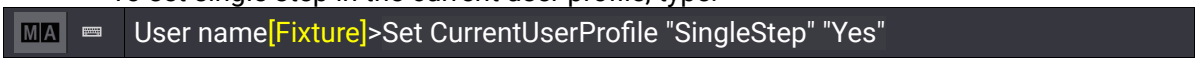
The CurrentUserProfile keyword represents the profile of the current user.

Syntax

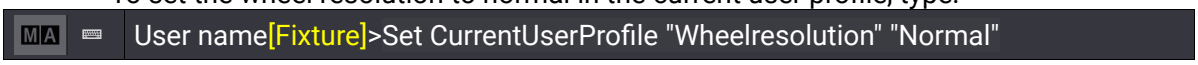
[Function] CurrentUserProfile (["Property"] ["Value"])

Examples

- To set single step in the current user profile, type:



- To set the wheel resolution to normal in the current user profile, type:



1.9.3.71. Cut

To enter the Cut keyword in the command line, type **Cut**.

Description

The Cut keyword is a function keyword used to specify the source objects for a two-step action.

Syntax

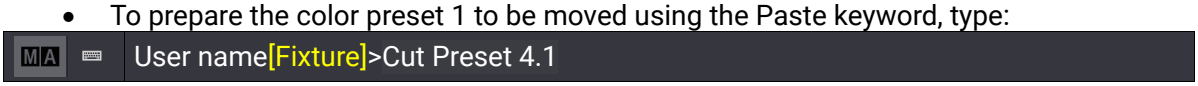
Cut [Object] ["Object_Name" or Object_Number]

The object is temporarily stored for later use as source object for the following Paste command.

For more information on Cut & Paste see the **Paste Keyword**.

Example

- To prepare the color preset 1 to be moved using the Paste keyword, type:



1.9.3.72. DataPool

To enter the DataPool keyword in the command line, use one of the options:

- Press MA + Preset + Preset
- Type **DataPool**
- Type the shortcut **Dat**

Description

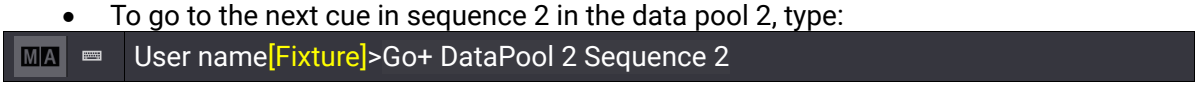
The DataPool keyword is used to address a data pool or objects outside the data pool you are currently in.

Syntax

[Function] DataPool [DataPool_Number] ([Object] ["Object_Name" or Object_Number])

Example

- To go to the next cue in sequence 2 in the data pool 2, type:



1.9.3.73. Deceleration

To enter the Deceleration keyword in the command line, use one of the options:

- Type **Deceleration**
- Type the shortcut **Dec**

Description

The Deceleration keyword is used to set the deceleration curve of a step in the phaser.

For more information see **Phasers**.

Syntax

Deceleration

[At] Deceleration [Value]

Examples

- To set the layer to deceleration, type:

- To set the deceleration curve of the dimmer to 85, type:

1.9.3.74. Deactivate

To enter the Deactivate keyword in the command line, use one of the options:

- Press MA + Off
- Type **Deactivate**
- Type the shortcut **Dea**

Description

The Deactivate keyword deactivates active data in the programmer. Values will not be stored when storing cues or presets as values are subject to store settings.

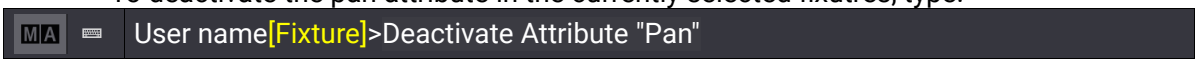
For more information see **Store Settings and Store Preferences**.

Syntax

Deactivate Attribute ["Attribute_Name" or Attribute_Number]

Example

- To deactivate the pan attribute in the currently selected fixutres, type:



1.9.3.75. Decimal8

To enter the Decimal8 keyword in the command line, use one of the options:

- Type **Decimal8**
- Type the shortcut **Deci**

Description

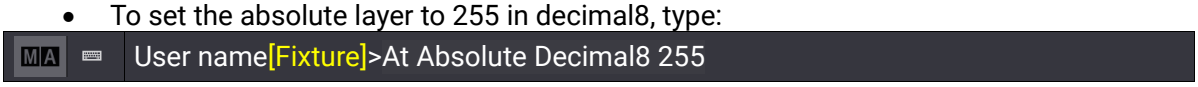
The Decimal8 keyword is used to set the values of a fixture selection using the 8bit decimal notation.

Syntax

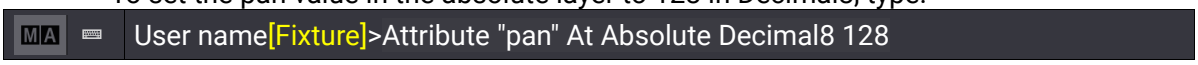
At [Layer] Decimal8 [Value]

Examples

- To set the absolute layer to 255 in decimal8, type:



- To set the pan value in the absolute layer to 128 in Decimal8, type:



1.9.3.76. Decimal16

To enter the Decimal16 keyword in the command line, use one of the options:

- Type **Decimal16**
- Type the shortcut **Decimal1**

Description

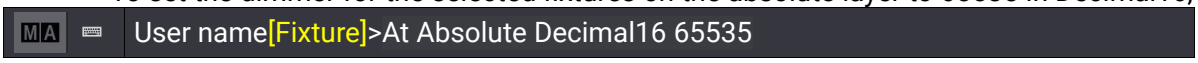
The Decimal16 keyword is used to set the values of a fixture selection using the 16bit hexadecimal notation.

Syntax

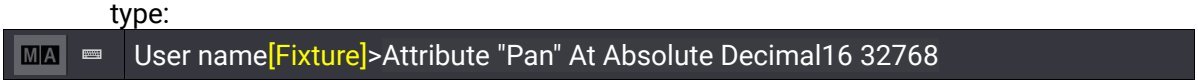
At [Layer] Decimal16 [Value]

Examples

- To set the dimmer for the selected fixtures on the absolute layer to 65535 in Decimal16, type:



- To set the pan attribute for the selected fixtures on the absolute layer to 32768 in Decimal16, type:



1.9.3.77. Decimal24

To enter the Decimal24 keyword in the command line, use one of the options:

- Type **Decimal24**
- Type the shortcut **Decimal2**

Description

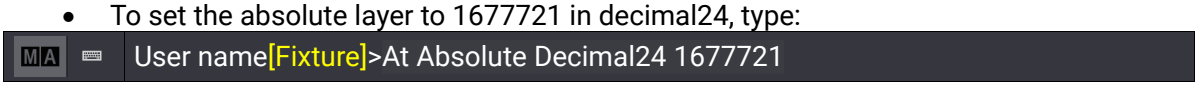
The Decimal24 keyword is used to set the values of a fixture selection using the 24bit decimal notation.

Syntax

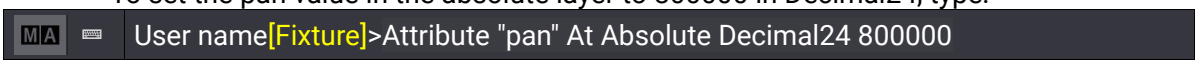
At [Layer] Decimal24 [Value]

Examples

- To set the absolute layer to 1677721 in decimal24, type:



- To set the pan value in the absolute layer to 800000 in Decimal24, type:



1.9.3.78. Default

To enter the Default keyword in the command line, use one of the options:

- Press MA + .
- Type **Default**
- Type the shortcut **Def**

Description

The Default keyword is used to reset the attributes of your fixture selection to default values. If there is no attribute list, all attributes of the selected fixtures will be set to their default values.

Syntax

Default

Fixture ["Fixture_Name" or Fixture_Number] At Default (FeatureGroup ["FeatureGroup_Name" or FeatureGroup_Number])

Store Default (/Option)

Examples

- To set the dimmer of fixture 1 to its default values, type:

```
User name[Fixture]> Fixture 1 At Default
```

- To set the position attribute of fixture 2 to default, type:

```
User name[Fixture]>Fixture 2 At Default FeatureGroup 2
```

Example: Store Special Values

Requirement:

Activate values in the programmer:

-Call a preset or turn the attribute encoders.

- To store the currently active values as default values, type:

```
User name[Fixture]>Store Default
```

- To reset the special values in the currently active values, type:

For more information on the special values see **Parameter List**.

1.9.3.79. Delay

To enter the Delay keyword in the command line, use one of the options:

- Press Time Time (If the Time Key Target is set to Fixture. For more information see **User Settings** and **Time Key**.)
- Press MA Time Time Time
- Type **Delay**
- Type the shortcut **Dela**

Description

Delay sets the delay time of an object.

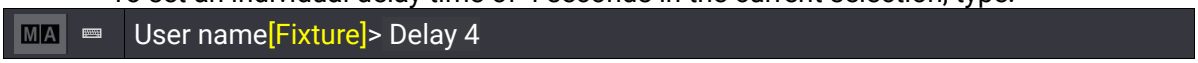
For more information on the delay times for cues see **CueDelay Keyword**.

Syntax

Delay [Delay_Time]

Example

- To set an individual delay time of 4 seconds in the current selection, type:



1.9.3.80. DeleteGlobalVariable

To enter the DeleteGlobalVariable keyword in the command line, use one of the options:

- Type **DeleteGlobalVariable**
- Type the shortcut **DeleteG**

Description

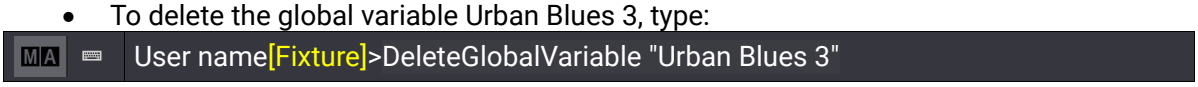
The DeleteGlobalVariable keyword is used to delete global variables in a show.

Syntax

DeleteGlobalVariable ["Name of Variable"]

Example

- To delete the global variable Urban Blues 3, type:



1.9.3.81. Delete

To enter the Delete keyword in the command line, use one of the options:

- Press Delete
- Type **Delete**
- Type the shortcut **D**

Description

The Delete keyword is a function keyword which is used to remove data in a show file.

Syntax

Delete [Object] ["Object_Name" or Object_Number]

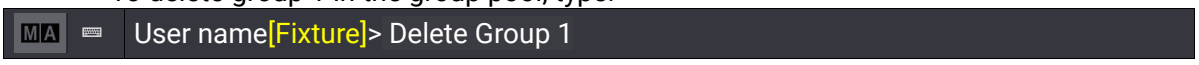
Option Keywords

The Delete keyword uses the following option keywords:

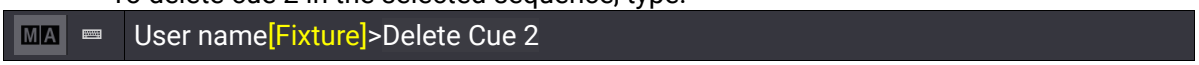
- **/NoConfirmation**

Examples

- To delete group 1 in the group pool, type:



- To delete cue 2 in the selected sequence, type:



1.9.3.82. DeleteOtherVersion

To enter the DeleteOtherVersion keyword in the command line, use one of the options:

- Type **DeleteOtherVersion**
- Type the shortcut **Deleteo**

Description

The DeleteOtherVersion keyword is a function keyword that is used to delete the installation files of the software versions in a grandMA3 device.

Syntax

DeleteOtherVersion "release_type_x.x.x.x.xml;/Path/to/MALightingTechnology/installation_packages"

Example

- To delete the installation files of the version 1.6.3.67 on your Windows® computer, type:

```
User name[Fixture]>DeleteOtherVersion
"release_stick_v1.6.3.67.xml;C:/ProgramData/MALightingTechnology/installation_packages"
```

1.9.3.83. DeleteUserVariable

To enter the DeleteUserVariable keyword in the command line, use one of the options:

- Type **DeleteUserVariable**
- Type the shortcut **DeleteU**

Description

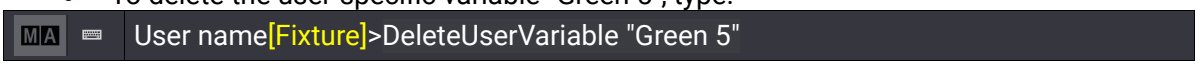
The DeleteUserVariable keyword is used to delete user-specific variables.

Syntax

DeleteUserVariable ["Name of Variable"]

Example

- To delete the user-specific variable "Green 5", type:



1.9.3.84. Disconnect

To enter the Disconnect keyword in the command line, use one of the options:

- Type **Disconnect**
- Type the shortcut **Disc**

Description

The Disconnect keyword disconnects the station from session. Contrary to Dismiss, the Disconnect keyword only disables the network of the station and does not reset the name of session and location of the device. Disconnect only works in devices that take part in the same session.

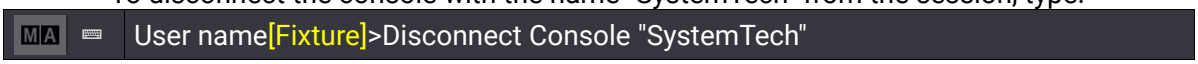
Syntax

Disconnect IP [IP_Number]

Disconnect [DeviceType] ["DeviceType_Name" or DeviceType_Number]

Example

- To disconnect the console with the name "SystemTech" from the session, type:



1.9.3.85. Dismiss

To enter the Dismiss keyword in the command line, use one of the options:

- Type **Dismiss**
- Type the shortcut **Di**

Description

The Dismiss keyword is used to throw stations out of your session.

Syntax

Dismiss [DeviceType] ["Device_Name" or Device_Number]

Dismiss IP [IP_Address]

Find the device number in the **Network menu**:

- To show the devices, unfold the device type.
- Along with other information, each device displays number, IP address, and name.

Examples

- To dismiss the console with the name "FOH2", type:

```
User name[Fixture]>Dismiss Console "FOH2"
```

- To dismiss the Processing Unit with the number 1, type:

```
User name[Fixture]>Dismiss ProcessingUnit 1
```

- To dismiss the station with the IP address 192.168.0.10, type:

```
User name[Fixture]>Dismiss IP 192.168.0.10
```

1.9.3.86. Display

To enter the Display keyword in the command line, use one of the options:

- Type **Display**
- Type the shortcut **Disp**

Description

Display is an object keyword which is used to manage the displays.

Syntax

[Function] Display

[Function] Display [Display_Number]

Examples

- To list all open displays, type:

- To close display 2, type:

- To retrieve display 2, type:

1.9.3.87. DMXAddress

To enter the DMXAddress in the command line, use one of the options:

- Press MA + X8 | DMX + X8 | DMX
- Type **DMXAddress**
- Type the shortcut **DMXa**

Description

The DMXAddress keyword is used to access DMX addresses directly using an absolute numbering method.

Syntax

[Function] DMXAddress [DMXAddress_Number]

DMXAddress [DMXAddress_Number] At ([Readout]) [Value]

Examples

- To select the fixture patched to universe 2, address 1, type:

```
User name[Fixture]>SelectFixtures DMXAddress 513
```

- To output 50% on the third DMX channel of universe 1 using the DMX testing function, type:

```
User name[Fixture]>DMXAddress 3 At 50
```

- To output 42% on the DMX channels 8 to 15 on universe 2 using the DMX testing function, type:

```
User name[Fixture]>DMXAddress 520 Thru 527 At 42
```

- To disable the DMX testing function on all DMX channels of all universes, type:

```
User name[Fixture]>Off DMXAddress Thru
```

1.9.3.88. DMXLayer

To enter the DMXLayer keyword in the command line, use one of the options:

- Type **DMXLayer**
- Type the shortcut **DMXL**

Description

DMXLayer is a layer keyword. It is used to change the selected layer to the DMXLayer. This layer can show information in, for instance, the fixture sheet.

The DMXLayer shows the values of the DMX sheet.

If a fixture sheet is set to **Auto** layer, then it will automatically change to show the selected layer.

Syntax

DMXLayer

Example

- To select the DMXLayer, type:

1.9.3.89. DMXReadout

To enter the DMXReadout keyword in the command line, use one of the options:

- Type **DMXReadout**
- Type the shortcut **DMXR**

Description

The DMXReadout keyword is used to set the readout of DMX values in the user profile. It affects the display of values when editing a fixture type or the normal value in the user profile.

Syntax

DMXReadout ["DMXReadout_Name" or DMXReadout_Number]

Example

- To set the DMXReadout to Hex8, type:

1.9.3.90. DMXUniverse

To enter the DMXUniverse in the command line, use one of the options:

- Press MA + X8 | DMX
- Type **DMXUniverse**
- Type the shortcut **DMX**

Description

The DMXUniverse keyword is used to access DMX universes or all DMX channels of a universe.

Syntax

[Function] DMXUniverse ["DMXUniverse_Name" or DMXUniverse_Number]

DMXUniverse ["DMXUniverse_Name" or DMXUniverse_Number] At ([Readout]) [Value]

Examples

- To patch all DMX channels of universe 1 to universe 11, type:

```
User name[Fixture]> Move DMXUniverse 1 At DMXUniverse 11
```

- To select the fixture patched to universe 2.001, type:

```
User name[Fixture]>SelectFixtures DMXUniverse 2.001
```

- To output 50% on the third DMX channel of universe 1 using the DMX testing function, type:

```
User name[Fixture]>DMXUniverse 1.3 At 50
```

- To output 42% on the DMX channels 8 to 15 on universe 2 using the DMX testing function, type:

```
User name[Fixture]>DMXUniverse 2.8 Thru 15 At 42
```

- To disable the DMX testing function on all DMX channels of all universes, type:

```
User name[Fixture]>Off DMXUniverse Thru
```

1.9.3.91. DoubleSpeed

To enter the DoubleSpeed keyword in the command line, use one of the options:

- Type **DoubleSpeed**
- Type the shortcut **DS** or **Dou**

Description

The DoubleSpeed keyword is a playback keyword which is used to change the speed of the speed masters, presets, or sequences to double the speed.

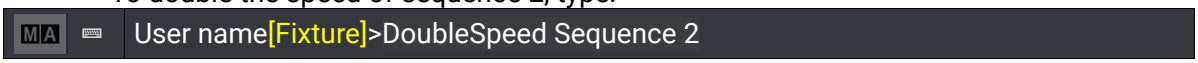
For more information on how to assign executors see **Assign Objects to an Executor**.

Syntax

DoubleSpeed [Object] ["Object_Name" or Object_Number]

Example

- To double the speed of sequence 2, type:



1.9.3.92. Down

To enter the Down keyword in the command line, use one of the options:

- Press Down
- Type **Down**
- Type the shortcut **Do**

For more information see the **Down Key**.

Description

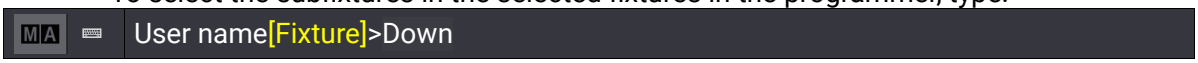
The Down keyword is a function keyword which is used to address the subfixture within the main fixture.

Syntax

Down

Example

- To select the subfixtures in the selected fixtures in the programmer, type:



1.9.3.93. DumpLog

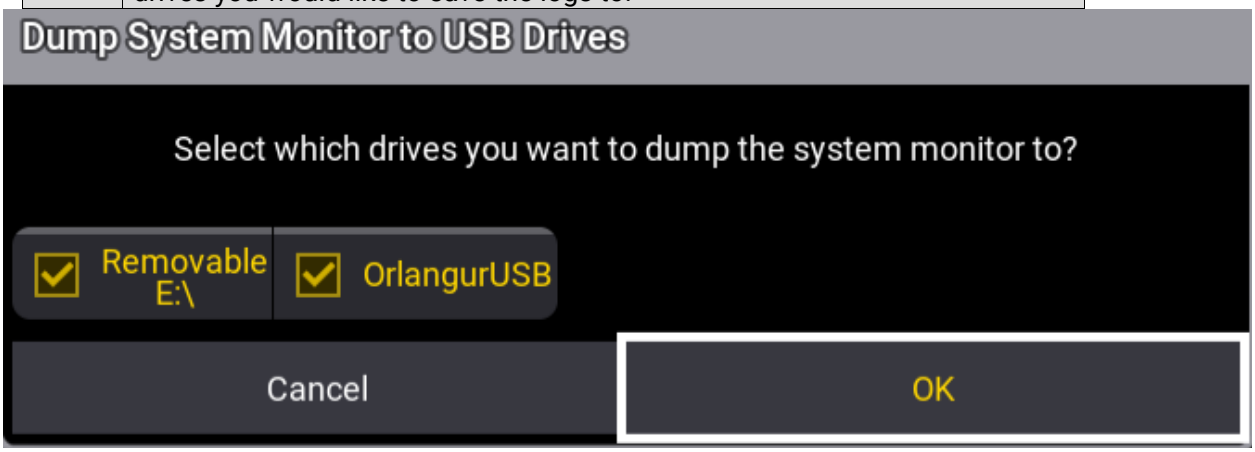
To enter the DumpLog keyword in the command line, use one of the options:

- Type **DumpLog**
- Type the shortcuts **Du**

Description

DumpLog is a function keyword that is used to deposit the output of the system monitor in form of a text file in case you should experience any inconsistencies or errors in the software. Hence, use this keyword if you are prompted by the tech support. Regardless of the location the files are stored to – internally or to a USB drive – the path is always **gma3_library.**

	Hint:
	If you insert one USB drive only, the logs will be automatically dumped onto this drive. If you insert more than one USB drive, you will have to specify the drives you would like to save the logs to.



Select Drive to Dump the Logs

Syntax

DumpLog

Option Keywords

The DumpLog keyword uses the following option keywords:

- **/Limit**

Example

- To deposit the output of the system monitor, type:

1.9.3.94. Drive

To enter the Drive keyword in the command line, use one of the options:

- Type **Drive**
- Type the shortcut **Dr**

Description

The Drive keyword is used to address physical drives, for instance, USB flash drives and the internal hard drive.

Syntax

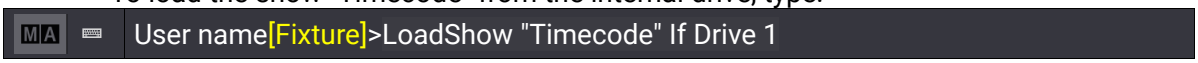
[Function] Drive ["Drive_Name" or Drive_Number]

[Command] If Drive ["Drive_Name" or Drive_Number]

Examples

- To display all available drives, type:

- To load the show "Timecode" from the internal drive, type:



1.9.3.95. DropOwnership

To enter the **DropOwnership** keyword in the command line, use one of the options:

- Type **DropOwnership**
- Type the shortcut **Dro**

Description

The DropOwnership keyword is used when a user wants to edit an object that is currently owned by a different user. DropOwnership sends a request to release the ownership of an object.

	Important:
	If a running process owns the object to protect data integrity, DropOwnwership might not work right away. If this should be the case, we recommend you use the command again.

Syntax

DropOwnership [Object] ["Object_Name" or Object_Number]

Examples

- To request dropping ownership of group 1, type:

```
User name[Fixture]> DropOwnership Group 1
```

- To request dropping ownership of macro 1, line 1, type:

```
User name[Fixture]>DropOwnership Macro 1
```

or type:

```
User name[Fixture]>DropOwnership Macro 1.1
```

or

```
User name[Fixture]>DropOwnership Macro 1.*
```

1.9.3.96. Echo

To enter the Echo keyword in the command line, use one of these options:

- Type **Echo**
- Type the shortcut **Ec**

Description

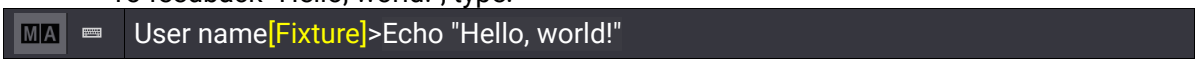
The Echo keyword is used to display feedback in the windows **Command Line History** and the **System Monitor**.

Syntax

Echo ["Feedback_Text"]

Example

- To feedback "Hello, world!", type:



Result:

1.9.3.97. Edit

To enter the Edit keyword in the command line, use one of the options:

- Press Edit
- Type **Edit**
- Type the shortcut **E**

Description

Edit is a function keyword which is used to modify values.

Syntax

Edit [Object] ["Object_Name" or Object_Number]

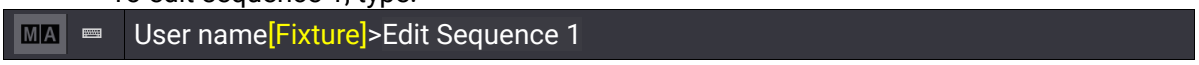
Option Keywords

The Edit keyword uses the following option keywords:

- **/Tab**

Example

- To edit sequence 1, type:



The Edit Sequence pop-up opens.

1.9.3.98. EditSetting

To enter the EditSetting keyword in the command line, use one of the options:

- Press Edit Edit
- Type **EditSetting**
- Type the shortcut **EditS**

Description

EditSetting is a function keyword which is used to modify the settings of an object.

Syntax

EditSetting [Object] ["Object_Name" or Object_Number]

Option Keywords

The EditSetting keyword uses the following option keywords:

- **/Tab**

Examples

- To edit the object settings of group 1, type:

```
User name[Fixture]>EditSetting Group 1
```

The group settings pop-up opens.

- To edit the settings of object 5 in preset pool 21, type:

```
User name[Fixture]>EditSetting Preset 21.5
```

The Preset settings pop-up opens.

- To change the Cue Part default, enter EditSetting into the command line and then tap the preset.
  The Preset settings pop-up opens.
- Tap Cue Part, enter the new value and apply it by tapping Please.

1.9.3.99. Eject

To enter the Eject keyword in the command line, use one of the options:

- Type **Eject**
- Type the shortcut **Ej**

Description

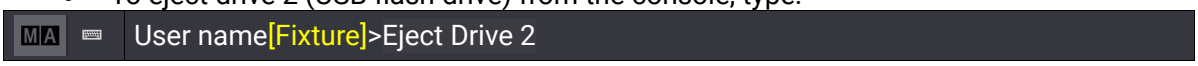
The Eject keyword is used to eject a USB flash drive.

Syntax

Eject Drive ["Drive_Name" or Drive_Number]

Examples

- To eject drive 2 (USB flash drive) from the console, type:



- To eject drive Enceladus, type:

1.9.3.100. EditRecipe

To enter the EditRecipe keyword in the command line, use one of the options:

- Press MA + Edit
- Type **EditRecipe**
- Type the shortcut **Editr**

Description

The EditRecipe is a function keyword that enables the recipe editor.

Syntax

EditRecipe ([Object] ["Object_Name" or Object_Number] or ["Recipe_Name" or Recipe_Number])

Examples

- To enable the edit mode in the recipe editor, type:

```
User name[Fixture]>EditRecipe
```

- To enable edit mode in cue 1, type:

```
User name[Fixture]>EditRecipe Cue 1
```

or type:

```
User name[Fixture]>EditRecipe 1
```

- To enable edit mode in the second position preset, type:

```
User name[Fixture]>EditRecipe Preset 2.2
```

- To enable edit mode in the running cue in sequence 1, type:

```
User name[Fixture]>EditRecipe Sequence 1
```

- To enable the edit mode in the running cue on executor 204, page 1, type:

```
User name[Fixture]>EditRecipe Page 1.204
```

1.9.3.101. Effect

To enter the Effect keyword in the command line, use one of the options:

- Press Channel until Effect appears in the command line
- Type **F + 7**
- Type **Effect**
- Type the shortcut **Ef**

	**Hint:**
	To use the Effect keyword, make sure you assign the Effect ID type and a custom ID to a fixture first.

Description

The Effect keyword is an object keyword which is used to call fixtures of the ID type Effect in the programmer.

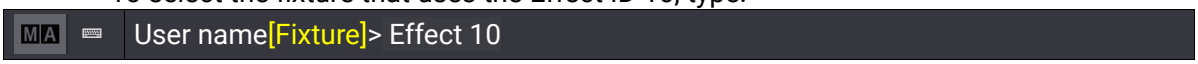
The name of this keyword can change because it is a custom ID type. For more information see **Custom ID Type**.

Syntax

Effect ["Effect_Name" or Effect_Number]

Example

- To select the fixture that uses the Effect ID 10, type:



1.9.3.102. EncoderBank

To enter the EncoderBank keyword in the command line, use one of the options:

- Press MA + X15 | Page + X15 | Page + X15 | Page
- Type **EncoderBank**
- Type the shortcut **EncoderBan**

	Hint:
	You can define commands for each encoder bank. The command will be executed once you select the encoder bank. To do so, edit the command cell in the Encoder Bar editor first.

Description

The EncoderBank keyword is an object keyword which addresses encoder banks.

Syntax

[Function] EncoderBank ["EncoderBank_Name" or EncoderBank_Number].(["EncoderPage_Name" or EncoderPage_Number])

Examples

- To select the encoder bank "Fancy Stuff", type:

```
User name[Fixture]>Select EncoderBank "Fancy Stuff"
```

- To switch to the encoder page "Song 2" of the encoder bank "Show", type:

```
User name[Fixture]>Select EncoderBank "Show"."Song 2"
```

- To only consider the attributes of filter 5 while knocking in the encoder bank 2, type:

```
User name[Fixture]>On EncoderBank2 If Filter 5
```

1.9.3.103. EncoderBar

To enter the EncoderBar keyword in the command line, use one of the options:

- Press MA + X15 | Page + X15 | Page
- Type **EncoderBar**
- Type the shortcut **En**

Description

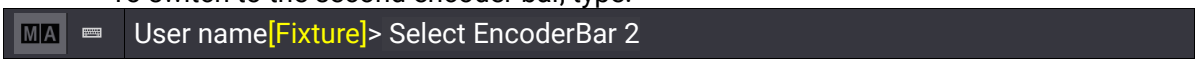
The EncoderBar keyword is an object keyword which is used to switch between the encoder bars.

Syntax

[Function] EncoderBar ["EncoderBar_Name" or EncoderBar_Number]

Example

- To switch to the second encoder bar, type:



1.9.3.104. EndIf

	Important:
	If the If keyword is already entered in the command line, pressing If again makes it an EndIf.

To enter the EndIf keyword in the command line, use one of the options:

- Type **EndIf**
- Type the shortcut **End**

Description

EndIf is a helping keyword that indicates the end of an If statement.

It enables If statements to be entered in the middle of a syntax. Upon processing, the If statement is moved to the end of the syntax, and is used as a filter or condition. This enables If syntax to be used in conjunction with pool items.

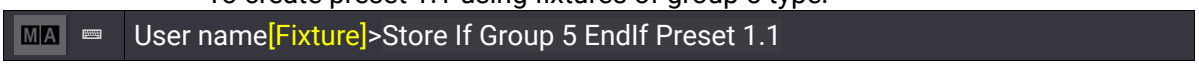
For more information see **If Keyword**.

Syntax

[Function] If [Object] ["Object_Name" or Object_Number] EndIf [Object] ["Object_Name" or Object_Number]

Example

- To create preset 1.1 using fixtures of group 5 type:



Result in the command line history:

1.9.3.105. Environment

To enter the Environment keyword in the command line, use one of the options:

- Type **Environment**
- Type the shortcut **Env**

Description

The Environment keyword is used to grant access to the two environments of the user profile - normal and preview. Each environment has its own programmer, selection, At filter and other.

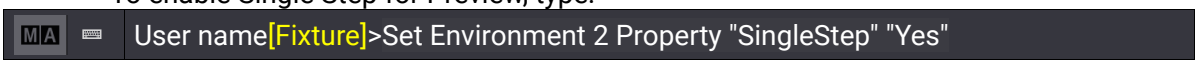
Syntax

[Function] Environment [Environment_Number] (Property ["Property_Name"] ["Property_Value"])

Examples

- To list the environments, type:

- To enable Single Step for Preview, type:



1.9.3.106. Exchange

To enter the Exchange keyword in the command line, use one of the options:

- Press Move Move
- Type **Exchange**
- Type the shortcut **Exc**

Description

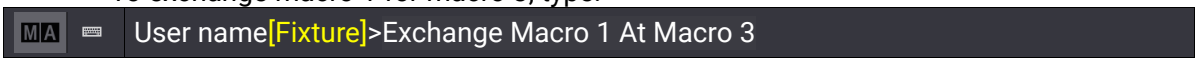
The Exchange keyword is a function keyword used to swap places.

Syntax

Exchange [Object] ["Object_Name" or Object_Number] At [Object] ["Object_Name" or Object_Number]

Example

- To exchange macro 1 for macro 3, type:



1.9.3.107. Executor

To enter the **Executor** keyword in the command line, use one of the options:

- Press MA + X16 | Exec
- Type **Executor**
- Type the shortcut **Ex**

Description

The Executor keyword is an object keyword used as a control handle for other objects.

The default function for Executor objects is **Select**. This means that calling executors without any function specified selects the object assigned to the executor. This selection is now also controllable with the 100 mm fader section.

If you apply a function or reference a property not supported by the Executor object, the command will be passed on to its child: key, fader, or the object assigned to the executor.

Syntax

Executor [Executor_ID]

Select Page [Page_ID] Executor [Executor_ID]

Set Executor [Executor_ID] [Setting] = [Setting_Option]

Settings

The following table displays the settings that can be set using the command line:

Setting	Setting Options	Description
Key	Go+ etc.	Executor key assignment
Fader	Master etc.	Executor fader assignment
Encoder	Master etc.	Executor encoder assignment
EncoderLeft	<<< etc.	Executor encoder assignment when turning the encoder counterclockwise
EncoderRight	>>> etc.	Executor encoder assignment when turning the encoder clockwise
KeyCmd	Go+ etc.	Command run when executor button is pressed
EncoderRightCmd	<<< etc.	Command run when the encoder is turned counterclockwise
EncoderLeftCmd	>>> etc.	Command run when the encoder is turned clockwise

Setting	Setting Options	Description
MAKey	Go+ etc.	Executor key assignment when pressing it together with the MA key
MAFader	Master etc.	Executor fader assignment when pressing the MA key
MAEncoder	Master etc.	Executor encoder assignment when the MA key is pressed
MAEncdoerRight	<<< etc.	Executor encoder assignment when turning the encoder counterclockwise while pressing the MA key
MAEncoderLeft	>>> etc.	Executor encoder assignment when turning the encoder clockwise while pressing the MA key
MAKeyCmd	Go+ etc.	Command run when executor button is pressed together with the MA key
MAEncoderRightCmd	<<< etc.	Command run when the encoder is turned counterclockwise together with the MA key
MAEncoderLeftCmd	>>> etc.	Command run when the encoder is turned clockwise together with the MA key
PrimaryAssignmentChanged		This is information only. See the description below
SecondaryAssignmentChanged		This is information only. See the description below
Width	1-5	Executor width
Height	1-4	Executor height
Object		The object the executor controls
Config		The executor configuration used by the executor
TotalPrimaryAssignmentChanged		This is information only. See the description below
TotalSecondaryAssignmentChanged		This is information only. See the description below

The four information settings are related to changes made in relation to the used executor configuration. Primary assignments are the assignments the executor has when the MA key is not pressed. The Secondary assignments are the assignments the executor has while the MA key is pressed. The two properties beginning with "Total" are for the entire combined executor. This is relevant when the executor is part of a combined executor with more than one executor in height and/or width. The information settings cannot be changed, they are automatically updated by the software.

For setting the executor assignments using the interface, please read the **Assign Object to an Executor topic** and the **Executor Configurations topic**.

Examples

- To remove executor 205 on the current page, type:

```
User name[Fixture]>Delete Executor 205
```

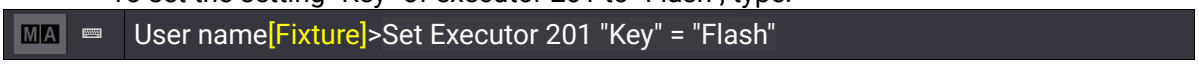
It does not delete the object assigned to the executor. It just deletes the assignment.

- To delete cue 3 of the sequence assigned to executor 205, type:

```
User name[Fixture]>Delete Executor 205 Cue 3
```

- To select executor 102 on page 4, type:

- To set the setting "Key" of executor 201 to "Flash", type:



For more information see **Executors**.

1.9.3.108. Export

To enter the Export keyword in the command line, use one of the options:

- Type **Export**
- Type the shortcut **Exp**

Description

Export is a function keyword which is used to save objects from the current show file as a smaller file.

If no file name is used in the command, the file name will use the name of the object.

By default, files will be exported to the relevant folder within the library folder structure, either on the local drive of the console or onPC station, or on a selected USB drive. For more information on grandMA3 folders see **Folder Structure**.

	**Important:**
	When exporting several objects at a time without indicating a file name for each pool object, a separate XML file will be exported. When exporting several objects at a time indicating a file name, all pool objects will be exported into a shared XML file.
	**Hint:**
	The Import and Export buttons offer a graphical user interface for import and export functions within the Show Creator Menu. For more information, see **Import/Export**.

Syntax

Export [Object] ["Object_Name" or Object_Number] (If Drive [Drive_Number]) (/Option) ("Option_Value")

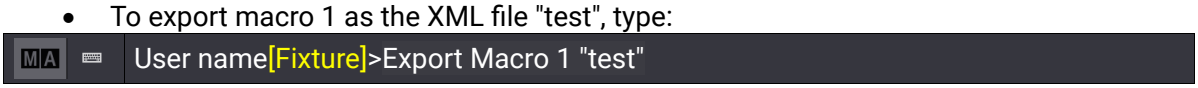
Option Keywords

The Export keyword uses the following option keywords:

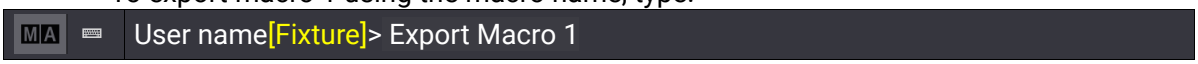
- **/Gaps**
- **/GDTF**
- **/NoDependencies**
- **/Path**
- **/Type**

Examples

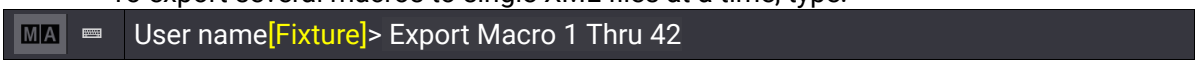
- To export macro 1 as the XML file "test", type:



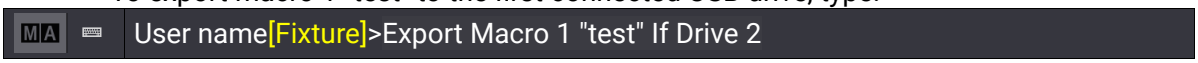
- To export macro 1 using the macro name, type:



- To export several macros to single XML files at a time, type:



- To export macro 1 "test" to the first connected USB drive, type:



1.9.3.109. Extension

To enter the Extension keyword in the command line, use one of the options:

- Type **Extension**
- Type the shortcut **Exte**

Description

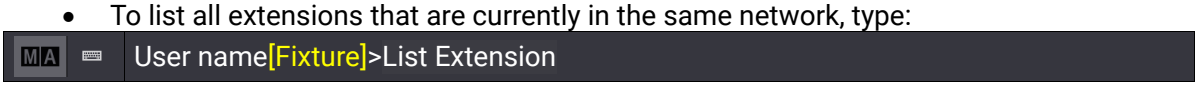
The Extension keyword is an object keyword which is used to address all extensions in the network.

Syntax

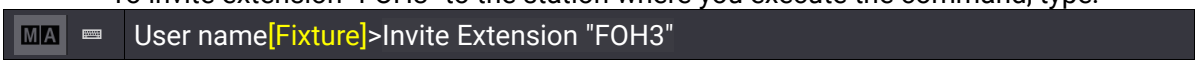
[Function] Extension ["Extension_Name" or Extension_Number]

Examples

- To list all extensions that are currently in the same network, type:



- To invite extension "FOH3" to the station where you execute the command, type:



1.9.3.110. Extract

To enter the **Extract** keyword in the command line, use one of the options:

- Press MA + At + At
- Type **Extract**
- Type the shortcut **Ext**

Description

The Extract keyword is a command keyword used to call the values of presets without references.

Syntax

Extract [Object] ["Object_Name" or Object_Number] (/Option)

Option Keywords

The Extract keyword uses the following option keywords:

- **/Single**

Example

- To call color preset 2, type:

1.9.3.111. Fade

To enter the **Fade** keyword in the command line, use one of the options:

- Press Time (If the Time Key Target is set to Fixture. For more information see **User Settings** and **Time Key**.)
- Press MA Time Time
- Type **Fade**

Description

The Fade keyword is a helping keyword which is used to indicate fade times.

As a helping keyword for playback functions (for example Goto), this keyword sets the time which is used to execute the function.

If it is used as a starting keyword, Fade will apply individual timing in the programmer for the current selection and attributes.

	**Hint:**
	As long as the command starts with a function, the Fade keyword and the fade value can be randomly used within the command.

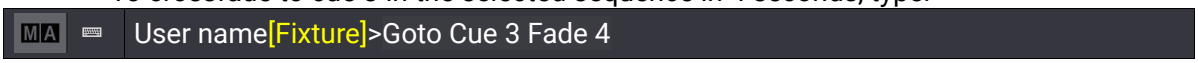
To set the fade times for cues, read more in the **CueFade keyword**.

Syntax

([Function] [Object] ["Object_Name" or Object_Number]) Fade [Fade_Value]

Examples

- To crossfade to cue 3 in the selected sequence in 4 seconds, type:



- To set the individual fade time of 2 seconds to the dimmer of the current selection, type:

- To set the dimmer value of the current fixture selection to 50 % and apply an individual fade time of 2 seconds to the dimmer of the current selection, type:

1.9.3.112. FaderCrossFade

To enter the FaderCrossFade keyword in the command line:

- Type **FaderCrossFade**
- Type the shortcut **FaderC** or **FaderX**

Description

The FaderCrossFade keyword represents the crossfade function of a sequence.

Crossfade gradually activates the next cue of a sequence in accordance with the position of the fader.

For more information see **Executors**.

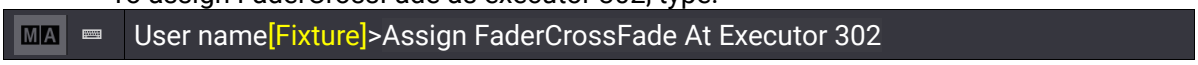
Syntax

[Function] FaderCrossFade At [Object] ["Object_Name" or Object_Number]

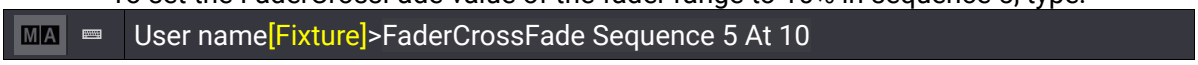
FaderCrossFade [Object] ["Object_Name or Object_Number] At [Value]

Example

- To assign FaderCrossFade as executor 302, type:



- To set the FaderCrossFade value of the fader range to 10% in sequence 5, type:



1.9.3.113. FaderCrossFadeA

To enter the FaderCrossFadeA keyword in the command line:

- Type **FaderCrossFadeA**
- Type the shortcut **FaderXA**

Description

The FaderCrossFadeA keyword represents the crossfade A function of a sequence.

Crossfade A gradually fades out dimmer attributes of a current cue in a sequence in accordance with the position of the fader.

For more information see **Executors**.

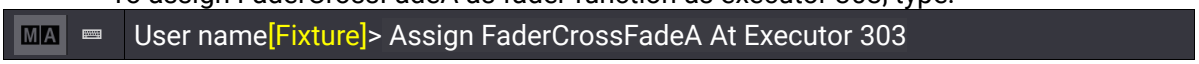
Syntax

[Function] FaderCrossFadeA At [Object] ["Object_Name" or Object_Number]

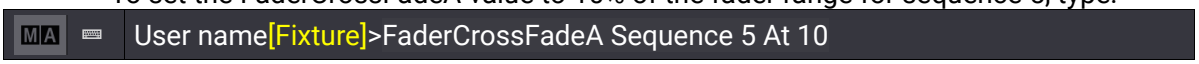
FaderCrossFadeA [Object] ["Object_Name" or Object_Number] At [Value]

Example

- To assign FaderCrossFadeA as fader function as executor 303, type:



- To set the FaderCrossFadeA value to 10% of the fader range for sequence 5, type:



1.9.3.114. FaderCrossFadeB

To enter the FaderCrossFadeB keyword in the command line:

- Type **FaderCrossFadeB**
- Type the shortcut **FaderXB**

Description

The FaderCrossFadeB keyword represents the crossfade B function of a sequence.

Crossfade B gradually fades in dimmer attributes of the next cue in a sequence in accordance with the position of the fader.

For more information see **Executors**.

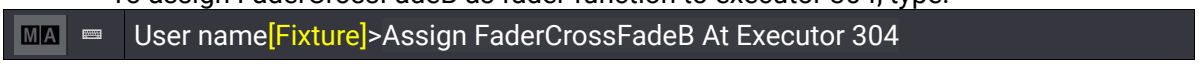
Syntax

[Function] FaderCrossFadeB At ["Object_Name" or Object_Number]

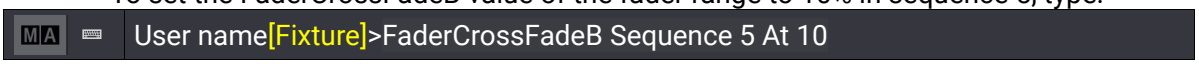
FaderCrossFadeB [Object] ["Object_Name" or Object_Number] At [Value]

Example

- To assign FaderCrossFadeB as fader function to executor 304, type:



- To set the FaderCrossFadeB value of the fader range to 10% in sequence 5, type:



1.9.3.115. FaderMaster

To enter the FaderMaster keyword in the command line, use one of the options:

- Type **FaderMaster**
- Type the shortcut **FaderM**
- Press MA + X16 | Exec + X16 | Exec + X16 | Exec

Description

The FaderMaster keyword applies the Master function to a number of objects such as executors, sequences, groups, masters, and other.

The master function controls the intensity of the assigned object.

For more information see **Executors**, **Cues and Sequences**, **Groups**, or **Masters**.

Syntax

Assign FaderMaster At [Object] ["Object_Name" or Object_Number]

FaderMaster [Object] ["Object_Name" or Object_Number] At [Value] (Fade [Time])

Only sequences and presets can move the master using a fade time. Other objects snap to the value.

Examples

- To assign FaderMaster to executor 204, type:

```
User name[Fixture]>Assign FaderMaster At Executor 204
```

- To move FaderMaster 205 at position 50% in a 5 seconds fade:

```
User name[Fixture]>FaderMaster 205 At 50 Fade 5
```

1.9.3.116. FaderRate

To enter the FaderRate keyword in the command line, use one of the options:

- Type **FaderRate**
- Type the shortcut **FaderR**

Description

The FaderRate keyword applies the Rate function to a number of objects such as executors, sequences, gropus, masters, presets, and other.

Rate divides or multiplies the fade and delay time in a sequence by the value of the fader. If Speed from Rate is enabled, it is then also valid for the speed stored in cues.

For more information see **Executors**, **Cues and Sequences**, **Presets**, **Groups**, or **Masters**.

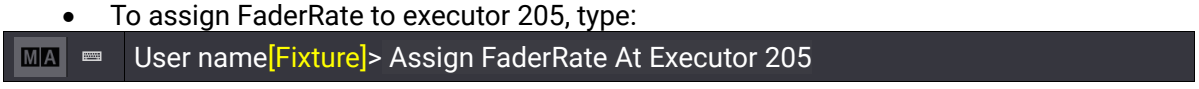
Syntax

Assign FaderRate At [Object] ["Object_Name " or Object_Number]

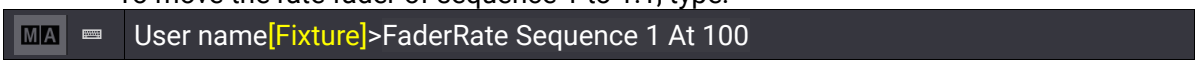
FaderRate [Object] ["Object_Name" or Object_Number] At [Value]

Examples

- To assign FaderRate to executor 205, type:



- To move the rate fader of sequence 1 to 1:1, type:



1.9.3.117. FaderSpeed

To enter the FaderSpeed keyword in the command line, use one of the options:

- Type **FaderSpeed**
- Type the shortcut **FaderS**

Description

The FaderSpeed keyword applies the Speed function to a number of objects such as executors, sequences, gropus, masters, and other.

It controls the speed of a phaser in a cue.

For more information see **Executors**, **Cues and Sequences**, **Groups**, or **Masters**.

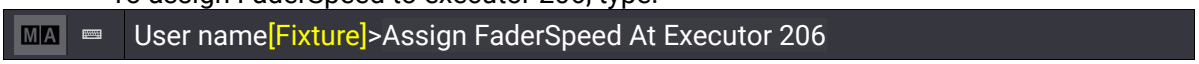
Syntax

Assign FaderSpeed At [Object] ["Object_Name" or Object_Number]

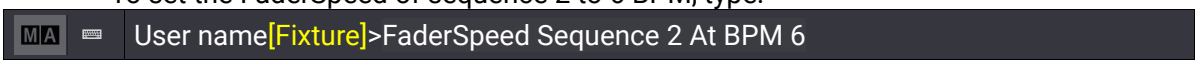
FaderSpeed [Object] ["Object_Name" or Object_Number] At [Speed_Readout] [Speed_Value]

Examples

- To assign FaderSpeed to executor 206, type:



- To set the FaderSpeed of sequence 2 to 6 BPM, type:



1.9.3.118. FaderTime

To enter the FaderTime keyword in the command line, use one of the options:

- Type **FaderTime**
- Type the shortcut **FaderTi**

Description

The FaderTime keyword applies the Time function to a number of objects such as executors, sequences, gropus, masters, and other. It is used to overwrite the stored cue part times by setting a time value and activating the time function.

For more information about cue time overwriting see **Cue Timing**.

For more information on how to activate and deactivate the function see **Time**.

Syntax

Assign FaderTime At [Object] ["Object_Name" or Object_Number]

FaderTime [Object] ["Object_Name" or Object_Number] At [Value]

Examples

- To assign FaderTime function to executor 207 on the selected page, type:

```
User name[Fixture]>Assign FaderTime At Executor 207
```

- To set the FaderTime value to 50% of the time range for the selected sequence, type:

```
User name[Fixture]> FaderTime At 50
```

- To set the FaderTime value to 10% of the time range for sequence 5, type:

```
User name[Fixture]>FaderTime Sequence 5 At 10
```

1.9.3.119. FaderTemp

To enter the FaderTemp keyword in the command line, use one of the options:

- Type **FaderTemp**
- Type the shortcut **FaderT**

Description

The FaderTemp keyword applies the Temp function to a number of objects such as executors, sequences, gropus, masters, and other.

Temp crossfades the first cue on when pulled up, and off when pulled down.

For more information see **Executors**, **Cues and Sequences**, **Groups**, or **Masters**.

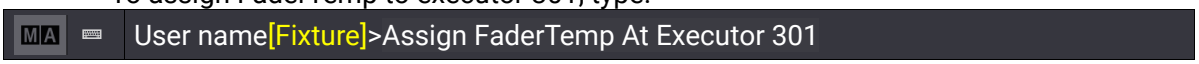
Syntax

Assign FaderTemp At [Object] ["Object_Name" or Object_Number]

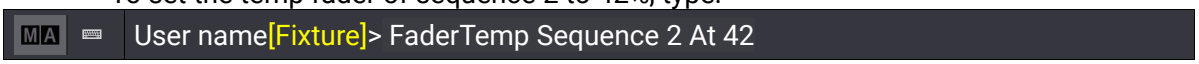
FaderTemp [Object] ["Object_Name" or Object_Number] At [Value]

Examples

- To assign FaderTemp to executor 301, type:



- To set the temp fader of sequence 2 to 42%, type:



1.9.3.120. FeatureGroup

To enter the FeatureGroup keyword in the command line, use one of the options:

- Press MA + Preset
- Type **FeatureGroup**
- Type the shortcut **Fg**

Description

The FeatureGroup keyword is used to address feature groups.

Syntax

[Function] FeatureGroup ["FeatureGroup_Name" or FeatureGroup_Number]

Example

- To set the value Remove in all attributes of the FeatureGroup "Dimmer" in the current selection, type:

1.9.3.121. Filter

To enter the Filter keyword in the command line, use one of the options:

- Press Group Group Group
- Type **Filter**
- Type the shortcut **Fil**

Description

The Filter keyword is used to call a filter. It represents an attribute and a layer filter.

For more information see **Worlds and Filters**.

Syntax

[Function] Filter ["Filter_Name" or Filter_Number] (/Option)

Option Keywords

The Filter keyword uses the following option keywords:

- **/Overwrite**

Example

- To call filter 4, type:

1.9.3.122. Fix

To enter the Fix keyword in the command line, use one of the options:

- Press MA + Pause
- Type **Fix**

Description

Fix is a function keyword which is used to latch executors on a page. The executors that are latched will always be displayed, even if you change the page.

Fix is a toggle function. This means that using Fix without any helping keyword toggles the fixing of the executors between enable and disable.

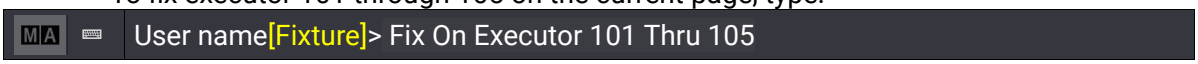
Syntax

Fix (On or Off) Executor [Executor_Number]

Fix (On or Off) Page ["Page_Name" or Page_Number].[Executor_Number]

Examples

- To fix executor 101 through 105 on the current page, type:



- To toggle executor 103 between changing page, or not changing page, type:

1.9.3.123. Fixture

To enter the Fixture keyword in the command line, use one of the options:

- Press Fixture
- Type **Fixture**
- Type the shortcut **F**

Description

The Fixture keyword is used as an object keyword to access fixtures that have a fixture ID.

Syntax

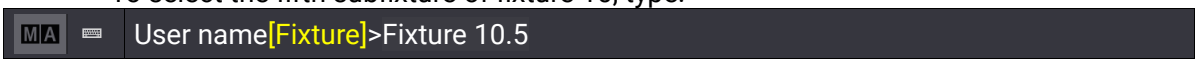
Fixture ["Fixture_Name" or Fixture_Number]

Fixture ["Fixture_Name" or Fixture_Number].["SubFixture_Name" or SubFixture_Number]

Examples

- To select fixture 2, type:

- To select the fifth subfixture of fixture 10, type:



1.9.3.124. FixtureClass

To enter the FixtureClass keyword in the command line, use one of the options:

- Press MA + Fixture + Fixture + Fixture
- Type **FixtureClass**
- Type the shortcuts **FC**

Description

FixtureClass is an object keyword which addresses the fixture classes of a show file.

Syntax

[Function] FixtureClass ["FixtureClass_Name" or FixtureClass_Number]

Option Keywords

The FixtureClass keyword uses the following option keywords:

- **/All**
- **/Single**

Examples

Requirement: Create the class "Spots" in the patch and link fixtures to it.

- To create a group in the group pool object 301 that contains all patched fixtures that are set to class "Spots" in the patch, type:

```
User name[Fixture]>AutoCreate FixtureClass "Spots" At Group 301
```

- To select all fixtures that are set to the class "Spots" in the patch, type:

```
User name[Fixture]>SelectFixture FixtureClass "Spots"
```

1.9.3.125. FixtureLayer

To enter the FixtureLayer keyword in the command line, use one of the options:

- Press MA + Fixture + Fixture
- Type **FixtureLayer**
- Type the shortcuts **FL**

Description

FixtureLayer is an object keyword which addresses the layers of fixtures in a show file.

Syntax

[Function] FixtureLayer ["FixtureLayer_Name" or FixtureLayer_Number]

Option Keywords

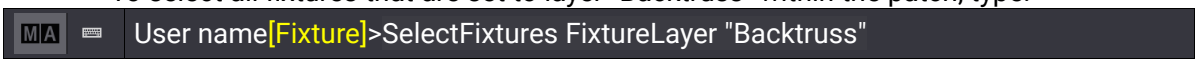
The FixtureLayer keyword uses the following option keywords:

- **/All**
- **/Single**

Example

Requirement: Create the layer "Backtruss" in the patch and link fixtures to it.

- To select all fixtures that are set to layer "Backtruss" within the patch, type:



1.9.3.126. FixtureType

To enter the FixtureType keyword in the command line, use one of the options:

- Press MA + Fixture
- Type **FixtureType**
- Type the shortcuts **FT** or **FixtureT**

Description

FixtureType is an object keyword which addresses the fixture types of a show file.

	Important:
	Most edits and command line actions with the keyword FixtureType has to be done while in the Edit Setup mode. For more information, see **ChangeDestination keyword**.

Syntax

[Function] FixtureType ["FixtureType_Name" or FixtureType_Number]

Option Keywords

The FixtureType keyword uses the following option keywords:

- **/All**
- **/Single**

Examples

Requirement:

- Enter the Patch menu first.
  For more information see **Patch and Fixture Setup**.

- To assign fixture type 2 to fixtures 1 through 4, type:

```
User name@ShowData/Patch/Stages/Stage 1> Assign FixtureType 2 At 1 Thru 4
```

- To select all patched fixtures of fixture type 3, type:

```
User name[Fixture]>SelectFixtures FixtureType 3
```

1.9.3.127. Flash

To enter the Flash keyword in the command line, use one of the options:

- Press MA + >>>
- Type **Flash**
- Type the shortcut **Fla**

Description

The Flash keyword is a playback keyword which is used to temporarily overwrite master level in order to set it to full on executing objects without using times.

If the executor is disabled and Flash is applied, the executor is temporarily activated using zero timing.

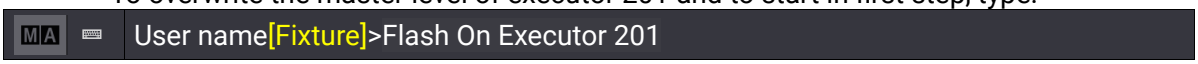
For more information on how to assign executors see **Assign Objects to an Executor**.

Syntax

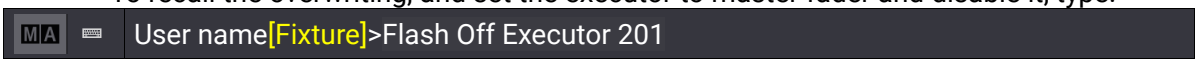
Flash (On/Off) [Object] ["Object_Name" or Object_Number]

Examples

- To overwrite the master level of executor 201 and to start in first step, type:



- To recall the overwriting, and set the executor to master fader and disable it, type:



1.9.3.128. Flip

To enter the Flip keyword in the command line, use one of the options:

- Type **Flip**
- Type the shortcut **Fli**

Description

The Flip keyword is used to access the different pan/tilt combinations that direct a moving head in the same direction.

Flip adds 180 degrees to the pan value of the fixtures and inverts the tilt angle. If the fixtures reach their physical breakpoint, the pan and tilt values will be set to the smallest possible value.
That is, Flip directs the fixture in the same direction using a different pan/tilt combinations.

	Hint:
	-If no selection list is entered, Flip is applied to the fixture selection. -If no number is entered, the function toggles through the different possible combinations. -The number of combinations depends on the possible degree value the fixture can pan in.

Syntax

Flip ([Flip_Number] [Object] ["Object_Name" or Object_Number])

Examples

- To set the pan and tilt of the fixture selection to the next pan/tilt combination, type:

User name[Fixture]>Flip

- To set the pan and tilt of group 7 to the second pan/tilt combination that directs the fixtures in the same direction, type:

User name[Fixture]>Flip 2 Group 7

1.9.3.129. Fog

To enter the **Fog** keyword in the command line, use one of the options:

- Press Channel until Fog appears in the command line
- Type **F + 6**
- Type **Fog**
- Type the shortcut **Fo**

	**Hint:**
	To use the Fog keyword, make sure you assign the Fog ID type and a custom ID to a fixture first.

Description

The Fog keyword is an object keyword which is used to call fixtures of the ID type Fog in the programmer.

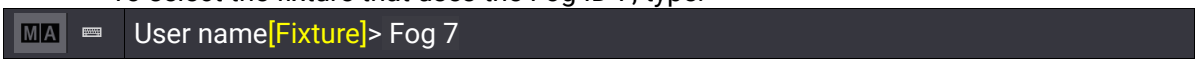
The name of this keyword can change because it is a custom ID type. For more information see the **Custom ID Type**.

Syntax

Fog ["IDType_Name" or IDType_Number]

Example

- To select the fixture that uses the Fog ID 7, type:



1.9.3.130. Freeze

	Important:
	If Freeze is on, programmer values have a higher priority as playbacks. Only executor priority **Super** has a higher priority than **Freeze**.

To enter the Freeze keyword in the command line, use one of the options:

- Press Freeze
- Type **Freeze**
- Type the shortcut **Fr**

Description

Freeze is a function keyword which is used to change the priority of the programmer.

Freeze is a toggle function. This means that entering Freeze without any helping keyword toggles the Freeze mode on/off.

Syntax

Freeze (On/Off)

Example

- To turn on the freeze mode, type:

MA	User name[Fixture]>Freeze On

1.9.3.131. Full

	Important:
	Pressing Full instantly executes the keyword.

To enter the **Full** keyword in the command line, use one of the options:

- Type **Full**
- Type the shortcut **Fu**

Description

Full is a function keyword which is used to set the dimmer values to 100 %.

The default function is At. This means that entering Full sets the dimmer attributes of the current selection to 100 %.

Syntax

([Object] ["Object_Name" or Object_Number]) Full

Examples

- To select fixture 1 thru 10 and set the dimmer to 100 %, type:

User name[Fixture]>Fixture 1 Thru 10 Full

- To select channel 53 and set the dimmer to 100 %, type:

User name[Fixture]>Channel 53 Full

1.9.3.132. Gel

To enter the **Gel** keyword in the command line, use one of the options:

- Press Preset Preset Preset
- Type **Gel**

Description

The keyword Gel provides the opportunity to edit or apply swatch book colors using the command line.

Syntax

[Function] Gel ["Swatch_Name"."Gel_Name"]

[Function] Gel [Swatch_Number].[Gel_Number]

[Function] Gel [Swatch_Number]

[Function] Gel ["Swatch_Name"]

Examples

- To set the color of the selected fixtures to Lee's color Mauve, type:

```
User name[Fixture]>At Gel "Lee"."Mauve"
```

- To set the color of the selected fixtures to the 44th color of the eighth swatch book, type:

```
User name[Fixture]>At Gel 8.44
```

- To export the "Lee" swatch book, type:

```
User name[Fixture]>Export Gel 8
```

1.9.3.133. Generator

To enter the **Generator** keyword in the command line, use one of the options:

- Type **Generator**
- Type the shortcut **Gen**

Description

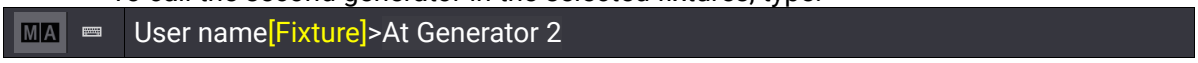
The Generator keyword is an object keyword which is used to address the Generator pool objects.

Syntax

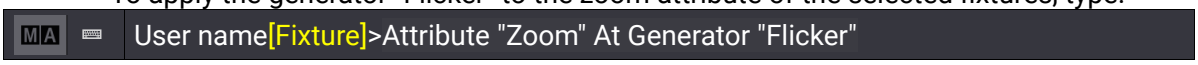
(Attribute ["Attribute_Name" or Attribute_Number]) At Generator ["Generator_Name" or Generator_Number]

Examples

- To call the second generator in the selected fixtures, type:



- To apply the generator "Flicker" to the zoom attribute of the selected fixtures, type:



1.9.3.134. GetGlobalVariable

To enter the GetGlobalVariable keyword in the command line, use one of the options:

- Type **GetGlobalVariable**
- Type the shortcut **Getg**

Description

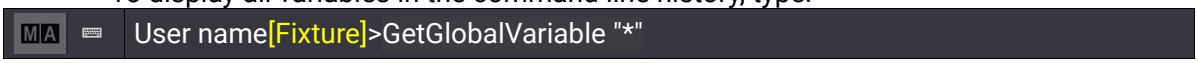
The GetGlobalVariable keyword is used to display global variables along with their values in the command line history.

Syntax

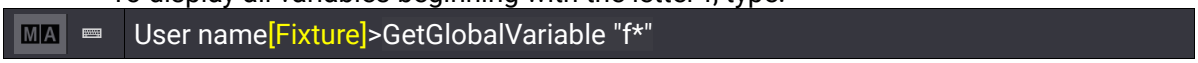
GetGlobalVariable ["Variable _Name"]

Examples

- To display all variables in the command line history, type:



- To display all variables beginning with the letter f, type:



1.9.3.135. GetUserVariable

To enter the GetUserVariable keyword in the command line, use one of the options:

- Type **GetUserVariable**
- Type the shortcut **GetU**

Description

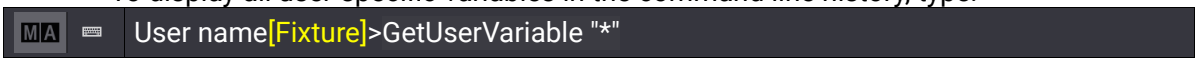
The GetUserVariable keyword is used to display user-specific variables along with their values in the command line history.

Syntax

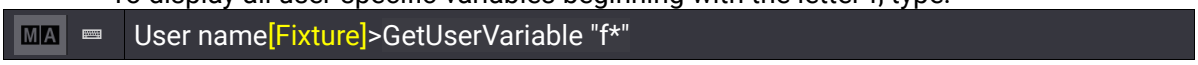
GetUserVariable ["Variable_Name"]

Examples

- To display all user-specific variables in the command line history, type:



- To display all user-specific variables beginning with the letter f, type:



1.9.3.136. Go+

To enter the Go+ keyword in the command line, use one of the options:

- Press Go+ | Temp
- Type **Go+**
- Type the shortcut **Go**

Description

The Go+ keyword is a playback keyword and it operates as a start signal for commands to be executed.

If the target object has cues, it will proceed to the next cue.

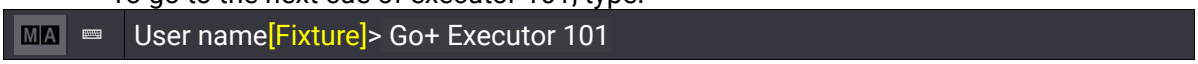
For more information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Go [Object] ["Object_Name" or Object_Number]

Examples

- To go to the next cue of executor 101, type:



- To start macro 2, type:

1.9.3.137. Go-

To enter the Go- keyword in the command line, use one of the options:

- Press Go- | Top
- Type **Go-**

Description

The Go- keyword is a playback keyword that is used to activate the previous cue.

If the target object has cues, it will go back to the previous cue.

For more information on how to assign executors see **Assign Objects to an Executor**.

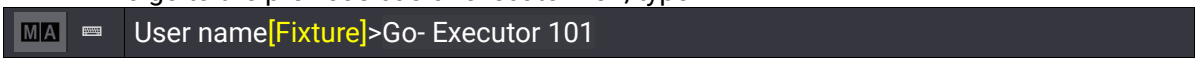
	**Hint:**
	Instead of Goto it is also possible to use Go- in order to go to a specified cue.

Syntax

Go- [Object] ["Object_Name" or Object_Number]

Example

- To go to the previous cue of executor 101, type:



1.9.3.138. Goto

To enter the **Goto** keyword in the command line, use one of the options:

- Press Goto
- Type **Goto**
- Type the shortcut **Got**

Description

Goto is a function keyword that is used to jump in a list using the original cue timing of the cue where the values were initially stored. Setting fade times using the Goto keyword overrides the original cue timing. For more information see **Relevant Playback Commands**.

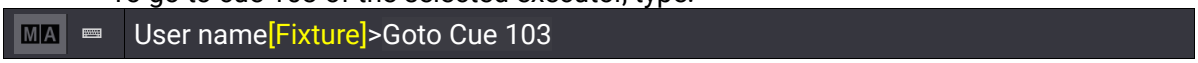
	**Important:**
	A set fade time overrides the original cue timing.

Syntax

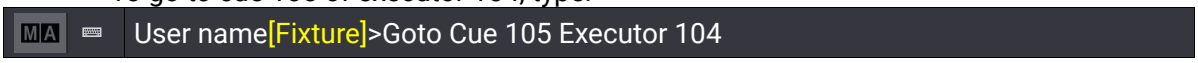
Goto [Object] ["Object_Name" or Object_Number] (Fade [Fade_Time])

Examples

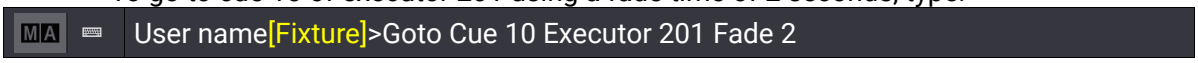
- To go to cue 103 of the selected executor, type:



- To go to cue 105 of executor 104, type:



- To go to cue 10 of executor 201 using a fade time of 2 seconds, type:



1.9.3.139. Grid

To enter the Grid keyword in the command line, use one of these options:

- Press MA + X3
- Type **Grid**
- Type the shortcut **Gri**

Description

The Grid keyword is used to set the grid cursor or span a grid area in the selection grid window. After the cursor is set, it is possible to arrange fixtures in the selection grid window. To position fixtures with a z-axis in the selection window, use the Grid keyword.

To set the grid cursor using the user interface see **Selection Grid Window**.

The grid command supports a space or a / (slash) as delimiters for command line input.

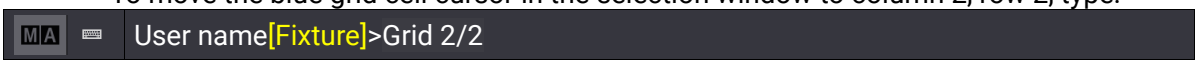
Syntax

Grid [x-axis]/[y-axis]/[z-axis]

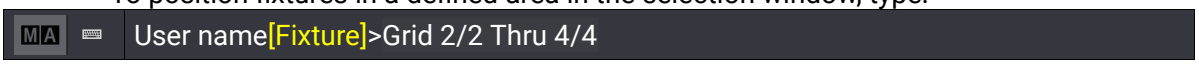
Grid [x-axis] [y-axis] [z-axis]

Examples

- To move the blue grid cell cursor in the selection window to column 2, row 2, type:



- To position fixtures in a defined area in the selection window, type:



1.9.3.140. GridPosition

To enter the GridPosition keyword in the command line, use one of the options:

- Type **GridPosition**
- Type the shortcut **GridP**

Description

GridPosition is a layer keyword which represents the xyz grid coordinates of fixtures at a time as values were set in attributes. It is used to change the selected layer to GridPosition. This layer can show information in, for instance, the fixture sheet.

Syntax

GridPosition

Example

- To select the layer GridPosition, type:

1.9.3.141. GridStore

To enter the GridStore keyword in the command line, use one of the options:

- Type **GridStore**
- Type the shortcut **Grids** or **GS**

Description

GridStore transfers grid positions of subfixtures to fixture types.

Syntax

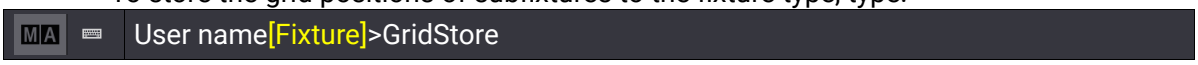
GridStore

Example

Requirement:

1. Patch fixtures containing subfixtures, for example a Robe Spiider and select Mode4.
   For more information on patching see **Add Fixtures to the Show**.
2. Select one of the fixtures you patched and select subfixtures. To do so, press Down in the command area of your device or the command section of the onPC, or type the **Down keyword** in the command line.
3. Arrange the subfixtures in the selection grid.
   For more information see **Selection Grid**.

- To store the grid positions of subfixtures to the fixture type, type:



1.9.3.142. GoboImage

To enter the GoboImage keyword in the command line, use one of the options:

- Type **GoboImage**
- Type the shortcut **Gob**

Description

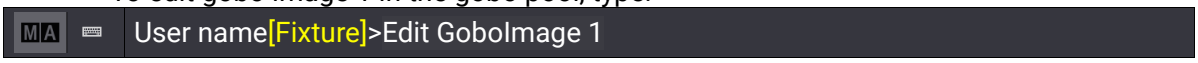
The keyword GoboImage is an object keyword. It is used to manage images of gobos using the command line.

Syntax

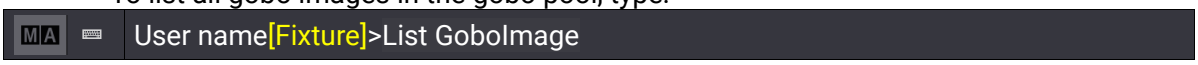
[Function] GoboImage ["GoboImage_Name" or GoboImage_Number]

Examples

- To edit gobo image 1 in the gobo pool, type:



- To list all gobo images in the gobo pool, type:



1.9.3.143. Group

To enter the Group keyword in the command line, use one of the options:

- Press Group
- Type **Group**
- Type the shortcut **G**

Description

The Group keyword is an object type that contains a certain selection of fixtures in a specific order along with xyz-coordinates used in the **Selection window**.

The default function of the Group keyword is SelFix. This means that calling groups without specifying any function selects the fixtures within the group.

For more information see the **SelFix keyword**.

Syntax

([Function]) Group ["Group_Name" or Group_Number]

Option Keywords

The Group keyword uses the following option keywords:

- **/GridMergeMode**
- **/Merge**
- **/Overwrite**
- **/Remove**

Examples

- To select the fixtures stored in group 3, type:

```
User name[Fixture]>Group 3
```

- To list all stored groups of the group pool in the command line history, type:

```
User name[Fixture]>List Group
```

1.9.3.144. HalfSpeed

To enter the HalfSpeed keyword in the command line, use one of the options:

- Type **HalfSpeed**
- Type the shortcut **HS** or **Hal**

Description

The HalfSpeed keyword is a playback keyword which is used to change the speed of the speed masters, presets, or sequences to half the speed.

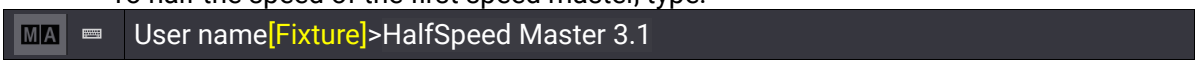
For more information on how to assign executors see **Assign Objects to an Executor**.

Syntax

HalfSpeed [Object] ["Object_Name" or Object_Number]

Example

- To half the speed of the first speed master, type:



1.9.3.145. Help

To enter the Help keyword in the command line, use one of the options:

- Press Help
- Type **Help**
- Type the shortcut **H**

	Hint:
	For more information on how to open the help see **Open the Help in the Console**.

Description

Help is a function keyword which is used to open the Help pop-up for you to search for topics or to use the context sensitive help.

Syntax

Help ["Search_Term"]

Example

- To list all keywords starting with an f, type:

```
User name[Fixture]>Help f*
```

1.9.3.146. HelpLua

To enter the HelpLua keyword in the command line, use one of the options:

- Type **HelpLua**
- Type the shortcut **HelpL**

Description

HelpLua is a function keyword which is used to export a list of all grandMA Lua functions.

To learn more about Lua functions, read the **Plugins** topic.

Syntax

HelpLua

Example

- To list all specific grandMA3 Lua functions, type:

The Lua functions are exported into the file "grandMA3_lua_functions.txt" in the **grandMA3 library folder**.

1.9.3.147. Hex8

To enter the Hex8 keyword in the command line, use one of the options:

- Type **Hex8**
- Type the shortcut **Hex**

Description

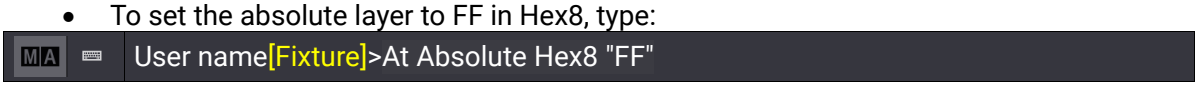
The Hex8 keyword is used to set the values of a fixture selection using the 8bit hexadecimal notation.

Syntax

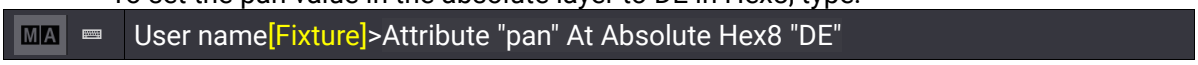
At [Layer] Hex8 ["Value"]

Examples

- To set the absolute layer to FF in Hex8, type:



- To set the pan value in the absolute layer to DE in Hex8, type:



1.9.3.148. Hex16

To enter the Hex16 keyword in the command line, use one of the options:

- Type **Hex16**
- Type the shortcut **Hex1**

Description

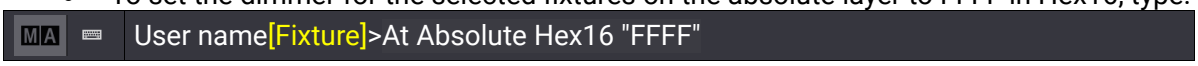
The Hex16 keyword is used to set the values of a fixture selection using the 16bit hexadecimal notation.

Syntax

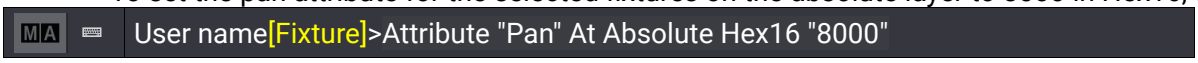
At [Layer] Hex16 ["Value"]

Examples

- To set the dimmer for the selected fixtures on the absolute layer to FFFF in Hex16, type:



- To set the pan attribute for the selected fixtures on the absolute layer to 8000 in Hex16, type:



1.9.3.149. HelpKeyword

To enter the HelpKeyword keyword in the command line, use one of the options:

- Type **HelpKeyword**
- Type the shortcut **Helpk**

Description

HelpKeyword is a function keyword which is used to list all grandMA3 keywords.

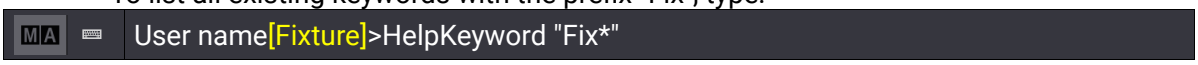
Syntax

HelpKeyword ["Search_Term"]

Examples

- To list all grandMA3 keywords in the command line history, type:

- To list all existing keywords with the prefix "Fix", type:



1.9.3.150. Hex24

To enter the Hex24 keyword in the command line, use one of the options:

- Type **Hex24**
- Type the shortcut **Hex2**

Description

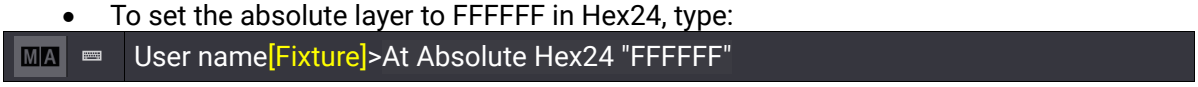
The Hex24 keyword is used to set the values of a fixture selection using the 24bit hexadecimal notation.

Syntax

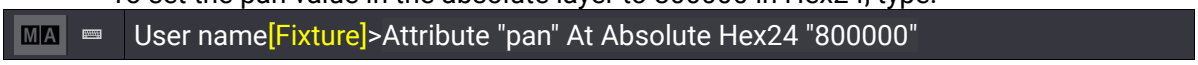
At [Layer] Hex24 ["Value"]

Examples

- To set the absolute layer to FFFFFF in Hex24, type:



- To set the pan value in the absolute layer to 800000 in Hex24, type:



1.9.3.151. Highlight

To enter the Highlight keyword in the command line, use one of the options:

- Press HighIt
- Type **Highlight**
- Type the shortcut **Hi**

Description

The Highlight keyword is a playback keyword that is used to apply the defined highlight values on fixtures that are selected.

It is a toggle function. This means, entering Highlight without a helping keyword enables or disables the Highlight mode.

	**Hint:**
	To manually fade from the current value to the highlight value and back, use the Highlight fader. For more information see **Master Controls**.

For information on location see the **HighIt Key**.

For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

([Function]) Highlight (On/Off)

Examples

- To toggle the highlight mode, type:

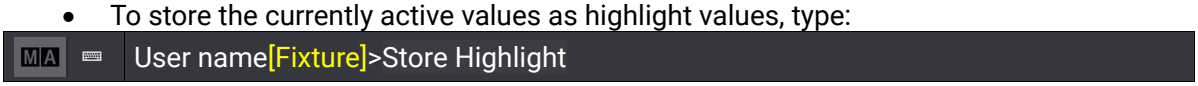
Example: Special Values

Requirement:

Activate values in the programmer:

-Call a preset or turn the attribute encoders.

- To store the currently active values as highlight values, type:



For more information on the special values see **Parameter List**.

- To load the highlight values of the selected fixtures into the programmer, type:

1.9.3.152. Hold

To enter the Hold keyword in the command line, use one of the options:

- Press MA + 0
- Type **Hold**
- Type the shortcut **Ho**

Description

The Hold keyword is a special value that prevents an MIB action while the dimmer is closed. Hold can only be applied to a dimmer attribute and is equivalent to a value of 0.

To prevent all fixtures (for example, scrollers) from performing an MIB action in the same cue, Hold can be used to distribute the movement across multiple cues.

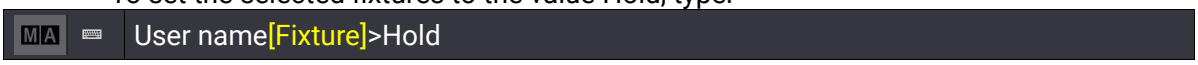
To expressly perform a movement between two cues using an open dimmer, Hold can be used to enhance individual fixtures.

Syntax

(At) Hold

Example

- To set the selected fixtures to the value Hold, type:



1.9.3.153. Houselights

To enter the **Houselights** keyword in the command line, use one of the options:

- Press Channel until Houselights appears in the command line
- Type **F + 3**
- Type **Houselights**
- Type the shortcut **Ho**

	Hint:
	To use the Houselights keyword, make sure you assign the Houselights ID type and a custom ID to a fixture first.

Description

The Houselights keyword is an object keyword which is used to call fixtures of the ID type Houselights in the programmer.

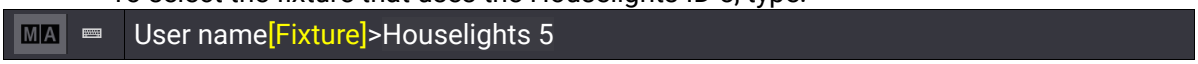
The name of this keyword can change because it is a custom ID type. For more information see the **Custom ID Type**.

Syntax

Houselights ["IDType_Name" or IDType_Number]

Example

- To select the fixture that uses the Houselights ID 5, type:



1.9.3.154. Hz

To enter the Hz (Hertz) keyword in the command line, type: **Hz**

Description

The Hz (Hertz) keyword is used to set the speed of a fixture selection or a speed master using the unit hertz.

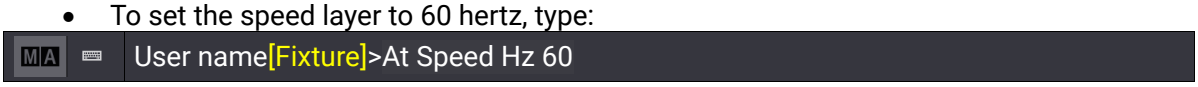
Syntax

At Speed Hz [Value]

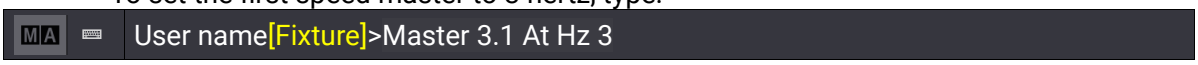
Master [Master_Number] At Hz [Value]

Examples

- To set the speed layer to 60 hertz, type:



- To set the first speed master to 3 hertz, type:



1.9.3.155. If

	Important:
	To end an if statement in the middle of a longer syntax, use the EndIf keyword. For more information see **EndIf Keyword**.

To enter the If keyword in the command line, use one of the options:

- Press If
- Type If

Description

The **If** keyword is a function keyword which is used to deselect fixtures in a selection.

As a helping keyword If sets a filter for the operation.

As a helping keyword for the Clone function If sets the scope of cloning.

For more information see the **Clone Keyword**.

Syntax

([Function] [Object] ["Object_Name" or Object_Number]) If [Object] ["Object_Name" or Object_Number]

Clone [Object] ["Object_Name" or Object_Number] At [Object] ["Object_Name" or Object_Number] If [Object] ["Object_Name" or Object_Number]

Examples

- To deselect fixtures which are not in group 5, type:

```
User name[Fixture]>If Group 5
```

- To deselect fixtures which are not in group 3 and group 5, type:

```
User name[Fixture]>Group 3 If Group 5
```

- To delete channel 4 in cue 3, type:

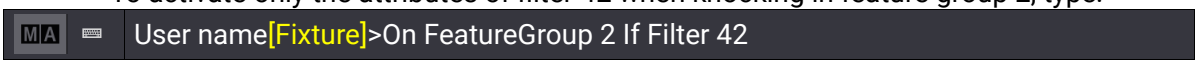
```
User name[Fixture]>Delete Cue 3 If Channel 4
```

- To delete attibute "Pan" of fixture 4 in cue 3, type:

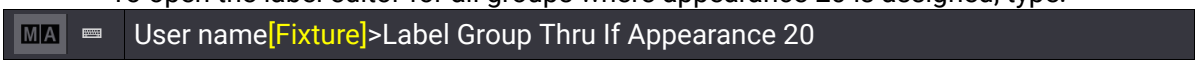
```
User name[Fixture]>Delete Cue 3 If Fixture 4 Attribute "Pan"
```

- To clone fixture 1 to fixture 2 only in sequence 1, type:

- To activate only the attributes of filter 42 when knocking in feature group 2, type:



- To open the label editor for all groups where appearance 20 is assigned, type:



1.9.3.156. IfActive

To enter the IfActive keyword in the command line, use one of the options:

- Press If If
- Type **IfActive**
- Type the shortcut **Ifa**

Description

IfActive is a function keyword that selects fixtures with active values in the programmer.

	**Important:**
	Executed on its own, the IfActive keyword only selects fixtures of the current command line default.
	**Important:**
	IfActive only works with dimmer values.

If no filter is entered, IfActive will select all fixtures with active values.

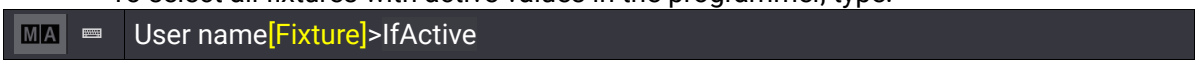
If a filter is entered, IfActive will select all fixtures which have this filter and also fixtures that have active values in the programmer.

Syntax

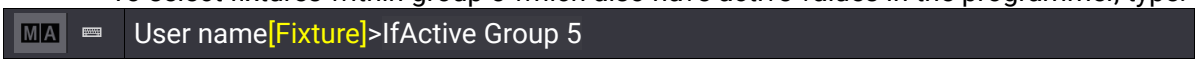
IfActive ([Object] ["Object_Name" or Object_Number])

Examples

- To select all fixtures with active values in the programmer, type:



- To select fixtures within group 5 which also have active values in the programmer, type:



1.9.3.157. IfProgrammer

To enter the IfProgrammer keyword in the command line, use one of the options:

- Press If If If
- Type **IfProgrammer**
- Type the shortcut **Ifp**

Description

IfProgrammer is a function keyword that selects fixtures that contain values in the programmer.

	Important:
	Executed on its own, the IfProgrammer keyword only selects fixtures of the current command line default.
	Important:
	IfProgrammer only works with dimmer values.

If no filter is entered, IfProgrammer selects all fixtures containing values in the programmer.

If a filter is entered, IfProgrammer selects the fixtures which are in the filter and also which contain values in the programmer.

Syntax

IfProgrammer ([Object] ["Object_Name" or Object_Number])

Examples

- To select all fixtures containing values in the programmer, type:

```
User name[Fixture]>IfProgrammer
```

- To only select the fixtures of group 5 which contain values in the programmer, type:

```
User name[Fixture]>IfProgrammer Group 5
```

1.9.3.158. IfOutput

To enter the IfOutput keyword in the command line, use one of the options:

- Press If
- Type **IfOutput**
- Type the shortcut **Ifo**

Description

IfOutput is a function keyword that selects fixtures based on their current output. This function works with presets and sequences only.

	Important:
	Executed on its own, the IfOutput keyword only selects fixtures of the current command line default.

Syntax

IfOutput ([Object] ["Object_Name" or Object_Number])

Examples

- To select all fixtures of the current command line default which have output on stage, type:

```
User name[Fixture]>IfOutput
```

- To select all fixtures that output the values stored in sequence 1, type:

```
User name[Fixture]>IfOutput Sequence 1
```

- To select all fixtures that use the color preset "Green", type:

```
User name[Fixture]>IfOutput Preset "Color"."Green"
```

1.9.3.159. Image

To enter the Image keyword in the command line, use one of the options:

- Type **Image**
- Type the shortcut **Ima**

Description

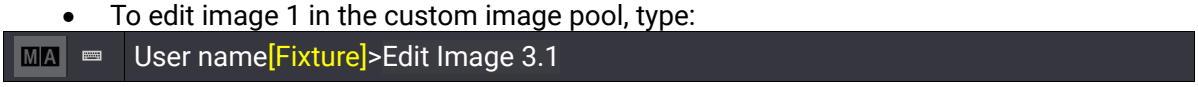
The keyword Image is an object keyword. It is used to manage images using the command line.

Syntax

[Function] Image [ImagePool_Number].[Image_Number]

Examples

- To edit image 1 in the custom image pool, type:



- To delete image 2 in the custom image pool, type:

1.9.3.160. Import

To enter the Import keyword in the command line, use one of the options:

- Type **Import**
- Type the shortcut **I**

Description

Import is a function keyword that incorporates small portions of exported show file data, available as XML files, into the current show file.

The Import command loads data into the specified destination occupying the first free pool object when no certain pool object is given.

By default, files will be imported from the relevant folder within the library folder structure, either on the local drive of the console or onPC station, or on a selected USB flash drive. For more information on this folder structure see **Folder Structure**.

Syntax

Import [Object] Library "File Name.xml" (If Drive [Drive_Number]) (At ["Object_Name" or Object_Number]) (/Option) ("Option_Value")

Option Keywords

The Import keyword uses the following option keywords:

- **/Path**
- **/Gaps**
- **/NoDependencies**
- **/NoRefresh**
- **/Type**

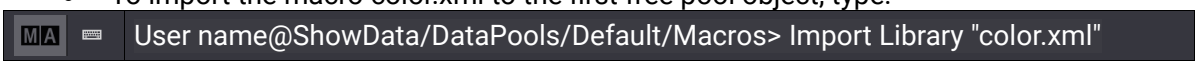
Examples

Requirements: Destination is changed to fixture types.

- To import a MAC Aura XB as a new fixture type in the show file, type:

Requirements: Destination is changed to macros.

- To import the macro color.xml to the first free pool object, type:



- To import the same macro to macro 42 in the macro pool without changing the command line destination, type:

Requirements: Destination is changed to macros.

- To import all macros from the library into the show file, type:

1.9.3.161. Index

To enter the Index keyword in the command line, use one of the options:

- Type **Index**
- Type the shortcut **Ind**

Description

Index keyword is a command keyword that addresses values where a property can have more than one value. For example, a fixture with at least two DMX breaks. The patch property has 2 values for this fixture.

	**Hint:**
	The values of the Index keyword are zero nominated.

Syntax

Set [Object] ["Object_Name" or Object_Number] ["Property"] Index [Index_Number] ["Value"]

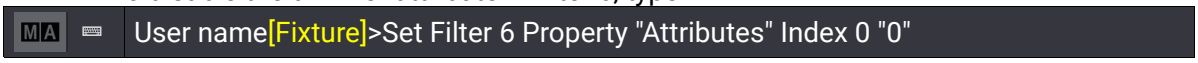
Examples

Requirement:

- Fixture 1 has to have at least two DMX breaks.
  For example, the fixture type LED Wall 20x20

- To set the patch address of the second break to universe 2 address 1 without affecting the first address, type:

- To disable the dimmer attribute in filter 6, type:



1.9.3.162. Insert

To enter the **Insert** keyword in the command line, use one of the options:

- Press Copy Copy Copy
- Type **Insert**
- Type the shortcut **Ins**

Description

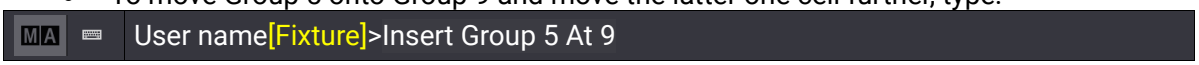
The Insert keyword is a function keyword which is used to insert pool objects between two other and already occupied pool objects. The following pool objects will be moved to the next empty destination.

Syntax

Insert [Object] ["Object_Name" or Object_Number] At ["Object_Name" or Object_Number]

Example

- To move Group 5 onto Group 9 and move the latter one cell further, type:



1.9.3.163. Integrate

To enter the Integrate keyword in the command line, use one of the options:

- Press MA + At
- Type **Integrate**
- Type the shortcut **Int**

Description

The Integrate keyword integrates step 1 of a preset into other steps of the programmer.

These steps can be stored into presets and cues.

Syntax

Integrate Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] Step [Step_Number]

Examples

Requirement: Example 1

- There are already active values in step 1 in the programmer

- To integrate step 1 of preset 1.1 into step 2 of the programmer, type:

```
User name[Fixture]>Integrate Preset 1.1 Step 2
```

Requirement: Example 2

- Select fixtures and create preset 4.5
- Follow steps 1 to 4

1. To call preset 4.5 into the programmer, type:

```
User name[Fixture]>At Preset 4.5
```

2. To create the next step, type:

```
User name[Fixture]>Next Step
```

3. To integrate step 1 of preset 4.2, type:

4. To store the changes in the preset, type:

1.9.3.164. Interface

To enter the Interface keyword in the command line, use one of the options:

- Type **Interface**
- Type the shortcut **Inter**

Description

Interface is an object keyword which is used to set the network interface using the command line.

For more information see **Interfaces and IP**.

Syntax

[Function] Interface

Interface [Number]

Examples

- To list all network interfaces, enter the Interface folder in the data structure first:

```
User name[Fixture]>ChangeDestination Interface
```

- To list the interfaces, type:

```
User name[Fixture]> List
```

1.9.3.165. Invert

To enter the **Invert** keyword in the command line, use one of the options:

- Type **Invert**
- Type the shortcut **Inve**

Description

Invert is a function to invert the selection status of fixtures.

If a fixture is selected, using this keyword deselects the fixtures. If a fixture is not selected, using the Invert keyword selects the fixtures.

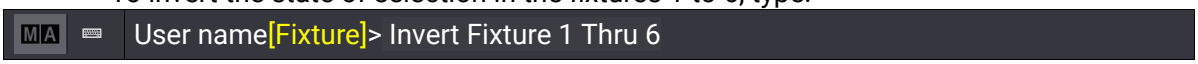
Syntax

Invert Fixture ["Fixture_Name" or Fixture_Number]

Example

Requirement: Fixtures 1, 3 and 5 are selected

- To invert the state of selection in the fixtures 1 to 6, type:



1.9.3.166. Invite

To enter the Invite keyword in the command line, use one of the options:

- Type **Invite**
- Type the shortcut **Inv**

Description

Invite is a function keyword which is used to invite other stations into the session.

Syntax

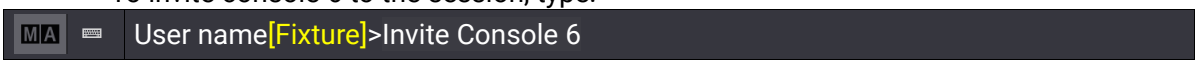
Invite IP [Device_IP]

Invite [Device_Type] ["Device_Name" or Device_Number]

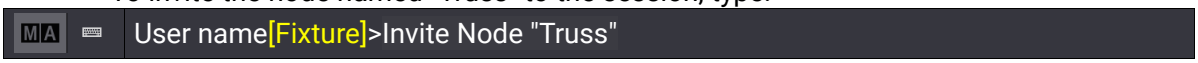
	**Hint:**
	The device types are: **Console**, **Node**, **onPC**, **ProcessingUnit**, **Station**, **Extension**.

Examples

- To invite console 6 to the session, type:



- To invite the node named "Truss" to the session, type:



- To invite the station with the IP address 192.168.0.10, type:

1.9.3.167. IP

To enter the IP keyword in the command line, type **IP**.

Description

IP is an object keyword which is used to manage devices with IP addresses.

It interacts with other function keywords and the IP address of the corresponding device.

Syntax

[Function] IP [IP Address]

Examples

For information on IP examples see:

- **Dismiss keyword**
- **Reboot keyword**
- **RemoteCommand keyword**
- **ShutDown keyword**

1.9.3.168. JoinSession

To enter the JoinSession keyword in the command line, use one of the options:

- Type **JoinSession**
- Type the shortcut **J**

Description

JoinSession is a function keyword which is used to join a session.

Syntax

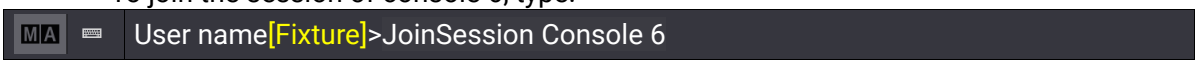
JoinSession IP [Device_IP]

JoinSession [DeviceType] ["Device_Name" or Device_Number]

	**Hint:**
	The device types are: **Console**, **Node**, **onPC**, **ProcessingUnit**, **Station**, **Extension**.

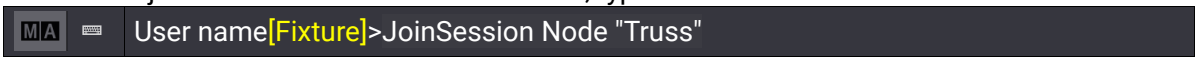
Examples

- To join the session of console 6, type:



- To join the session on the station with the IP address 192.168.10.21, type:

- To join the session of the node "Truss", type:



1.9.3.169. Key

To enter the Key keyword in the command line, use one of the options:

- Type **Key**
- Type the shortcut **K**

Description

The Key keyword is used to address the network keys.

For more information on what network keys are see **Create a Custom Key**.

Syntax

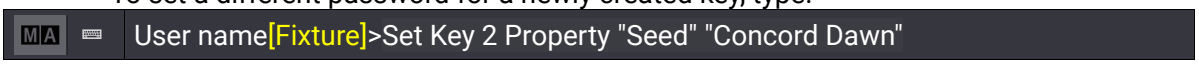
[Function] Key ["Key_Name" or Key_Number] (Property ["Property_Name"] ["Value"])

Examples

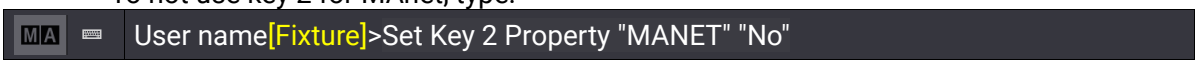
- To store a new key, type:

- To list all keys, type:

- To set a different password for a newly created key, type:



- To not use key 2 for MAnet, type:



1.9.3.170. Keyboard

To enter the Keyboard keyword in the command line, use one of the options:

- Type **Keyboard**
- Type the shortcut **Keyb**

Description

Keyboard is an object keyword which is used to change the language of the on-screen keyboard.

Syntax

[Function] Keyboard

Keyboard ["Keyboard_Language" or Keyboard_Number]

Examples

- To list the keyboard languages, type:

- To set the keyboard to German, type:

- To set the keyboard to English, type:

1.9.3.171. KeyboardShortcut

To enter the KeyboardShortcuts keyword in the command line, use one of the options:

- Type **KeyboardShortcut**
- Type the shortcut **KeyboardS**

Description

The KeyboardShortcut keyword is an object keyword that is used to work with keyboard shortcuts using the command line.

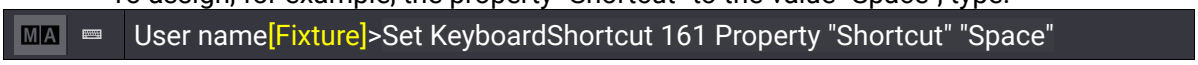
Syntax

[Function] KeyboardShortcut [KeyboardShortcut_Number] (Property ["Property_Name"] ["Value"])

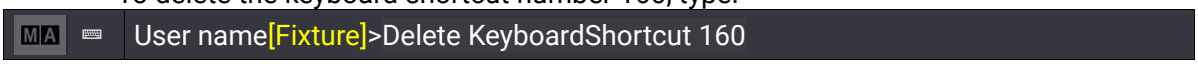
Examples

- To list all keyboard shortcuts, type:

- To assign, for example, the property "Shortcut" to the value "Space", type:



- To delete the keyboard shortcut number 160, type:



1.9.3.172. Kill

To enter the **Kill** keyword in the command line, use one of the options:

- Press MA + Go- + Go-
- Type **Kill**
- Type the shortcut **Ki**

Description

The Kill keyword is a function keyword that executes the Go+ function on sequences and presets and disables all other played back sequences and presets.

If the sequences and presets are not protected against kill, they will be disabled by the Kill keyword.

For more information on Kill Protect see **Sequence Settings**.

Syntax

Kill [Object] ["Object_Name" or Object_Number]

Example

- To jump to the next cue on executor 102 and disable all other playbacks, type:

1.9.3.173. KnockIn

To enter the KnockIn keyword in the command line, use one of the options:

- Type **KnockIn**
- Type the shortcut **Kn**

Description

KnockIn is used to take values that are located for example in sequences or presets actively into the programmer.

This is useful to the effect that you can actively take values that have already been stored into the programmer.

Syntax

Fixture ["Fixture_Name" or Fixture_Number] KnockIn ["Attribute_Name" or Attribute_Number]/["FeatureGroup_Name" or FeatureGroup_Number]

KnockIn Fixture ["Fixture_Name" or Fixture_Number]

Attribute ["Attribute_Name" or Attribute_Number] At KnockIn

Examples

- To knock in the attribute Color RGB_R in fixture 1, type:

```
User name[Fixture]>Fixture 1 KnockIn Attribute "ColorRGB_R"
```

- To knock in all attributes of fixture 2, type:

```
User name[Fixture]>KnockIn Fixture 2
```

- To knock in only the dimmer values of fixtures 11 through 15, type:

```
User name[Fixture]> Fixture 11 Thru 15 At KnockIn
```

1.9.3.174. KnockOut

To enter the Off keyword in the command line, use one of the options:

- Type **KnockOut**
- Type the shortcut **Knocko**

	**Hint:**
	The KnockOut keyword is used as a synonym of the Off keyword.

Description

The KnockOut keyword is used as a function and value keyword to:

- Knock out selections in the programmer
- Knock out active attributes in the programmer

Syntax

Fixture ["Fixture_Name" or Fixture_Number] KnockOut ["Attribute_Name" or Attribute_Number]/["FeatureGroup_Name" or FeatureGroup_Number]

KnockOut Fixture ["Fixture_Name" or Fixture_Number]

Attribute ["Attribute_Name" or Attribute_Number] At KnockOut

Examples

- To knock out the parameters of fixture 2 and 4 in the programmer, type:

```
User name[Fixture]>KnockOut Fixture 2 + 4
```

- To knock out the attribute Color RGB_R in the selected fixtures, type:

```
User name[Fixture]>Attribute "ColorRGB_R"  At KnockOut
```

- To knock out the dimmer attribute in fixture 5, type:

```
User name[Fixture]>Fixture 5 At KnockOut
```

1.9.3.175. Label

To enter the Label keyword in the command line, use one of the options:

- Press Assign Assign
- Type **Label**
- Type the shortcut **L**

Description

The Label keyword is used to give objects a name.

If multiple objects are labeled, and the name contains a free-standing number, the number will be enumerated for each object.

If you do not label an object, a pop-up appears.

If no object is specified when using the Label keyword, the cue in the selected sequence will be addressed. If the selected sequence is disabled, the sequence will be addressed.

Syntax

Label [Object] ["Object_Name" or Object_Number] ["Name"]

	Important:
	The name must not contain the following characters: \ " $ & * ? , . ; ^ { | } ~

Examples

- To label group 3 "Higgs Boson", type:

```
User name[Fixture]>Label Group 3 "Higgs Boson"
```

- To label fixtures 1 to 10 as "Mac700 1", "Mac700 2" and so on, type:

```
User name[Fixture]>Label Fixture 1 Thru 10 "Mac700 1"
```

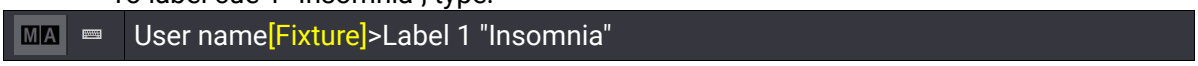
- To rename the color preset "Red" to "Dark Red", type:

```
User name[Fixture]>Label Preset "Color"."Red" "Dark Red"
```

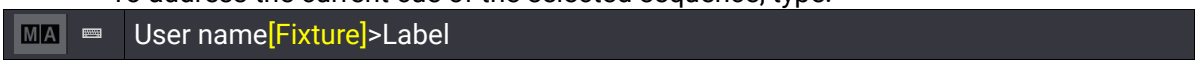
- To label group 1 using the name of the selected attribute, type:

```
User name[Fixture]>Label Group 1 At Attribute
```

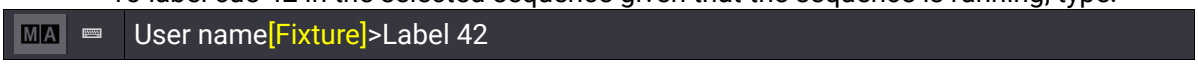
- To label cue 1 "Insomnia", type:



- To address the current cue of the selected sequence, type:



- To label cue 42 in the selected sequence given that the sequence is running, type:



- To label sequence 42 given that the selected sequence is disabled, type:

1.9.3.176. Layout

To enter the Layout Keyword in the command line, use one of the options:

- Press MA + X4
- Type **Layout**
- Type the shortcut **Lay**

Description

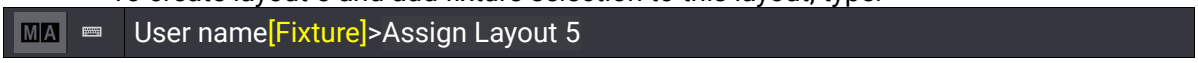
Layout is an object keyword representing a layout of fixtures and other objects.

Syntax

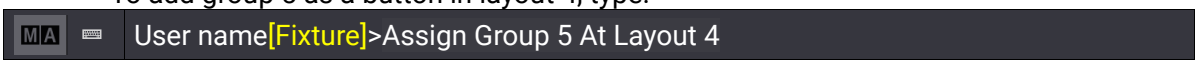
[Function] Layout ["Layout_Name" or Layout_Number]

Examples

- To create layout 5 and add fixture selection to this layout, type:



- To add group 5 as a button in layout 4, type:



1.9.3.177. LearnSpeed

To enter the LearnSpeed keyword in the command line, use one of the options:

- Press Learn | Rate1
- Type **LearnSpeed**
- Type the shortcut **Lear**

Description

The LearnSpeed keyword is a playback keyword that is used to set the speed in phasers.

For more information on how to assign executors see **Assign Objects to an Executor**.

Syntax

LearnSpeed [Object] ["Object_Name" or Object_Number]

Example

To set the speed in a phaser:

- Press and hold Learn | Rate1 and press the executor key several times using the speed you would like to have your phaser at.

The phaser learns the speed you press the executor key at.

1.9.3.178. LeaveSession

To enter the LeaveSession keyword in the command line, use one of the options:

- Type **LeaveSession**
- Type **Leave**
- Type the shortcut **Le**

Description

LeaveSession is a function keyword which is used to leave the session you are currently in.

Syntax

LeaveSession

Example

- To leave the current session, type:

1.9.3.179. Library

To enter the **Library** keyword in the command line, use one of the options:

- Type **Library**
- Type the shortcut **Lib**

Description

Library is an object keyword that is used to access the corresponding library folder down to the root on the hard drive.

Syntax

[Function] Library

Examples

Requirements: Change destination to macros.

For more information see **ChangeDestination Keyword**.

- To list all macros stored in the macro library, type:

```
Admin@Root/ShowData/Pools/Global/Macros> List Library
```

- To import the first macro library, type:

```
Admin@Root/ShowData/Pools/Global/Macros> Import Library 1
```

Requirements: Change destination to fixture types.

For more information see **ChangeDestination Keyword**.

- To list all fixture types stored in the fixture types library, type:

```
Admin@Root/ShowData/LivePatch/FixtureTypes> List Library
```

1.9.3.180. List

To enter the List keyword in the command line, use one of these options:

- Press List
- Type **List**
- Type the shortcut **Li**

Description

The List keyword displays show data, including objects and their properties, in the command line history.

It is possible to list any object or any property of:

- Cues of the selected executor
- Groups
- Presets

If the List command does not specify any type of object, the data of the current command line destination is displayed.

The List command can also display the contents of library folders on the console, onPC station, or connected USB drives.

	**Hint:**
	The /Path option instructs the software to access folders other than the defaults. For more information on the default folder structure, see the **Folder Structure** topic.

The List keyword is a function keyword.

Syntax

List [Object] ["Object_Name" or Object_Number]

List [Object] Library (If Drive [Drive_Number]) (/Path [FolderPath]) (/Type [FileType])

	**Important:**
	When specifying a type, the word "System" or "User" must be capitalized. For more information see the option keyword **/Type**.

Examples

- To list the first ten fixtures of the Fixture Edit Setup, type:

- To list all existing attributes in the show file, type:

User name[Fixture]>List Attribute

- To list the first 5 groups of the group pool, type:

User name[Fixture]>List Group Thru 5

- To list the macros available to import from the default macro library folder on the first connected USB drive, type:

User name[Fixture]>List Macro Library If Drive 2

Example - List Properties of a Parent and Their Children's Properties

	**Important:**
	In case the children of an object should have various properties, they will be listed using names and numbers when executing List within the parent object. To display the properties of a child directly, you have to specifically list the child. However, the steps described here can also be used in other constellations consisting of a parent and their children.

In the following example we use StationSettings as the parent and DeskLightsCollect as one of their children.

1. Change the destination of the command line first:

User name[Fixture]>ChangeDestination StationSettings

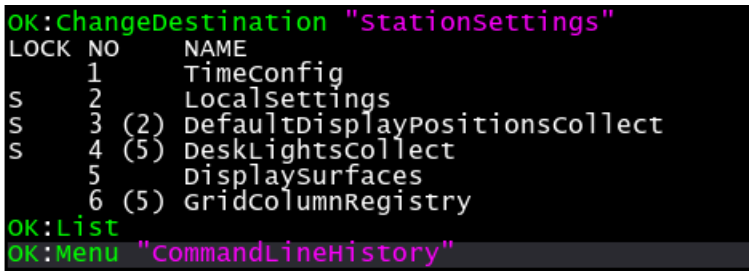
Result:

User name@StationSettings>

2. To list the properties of the station settings, type:

User name@StationSettings> List

The children are now displayed in the command line history:

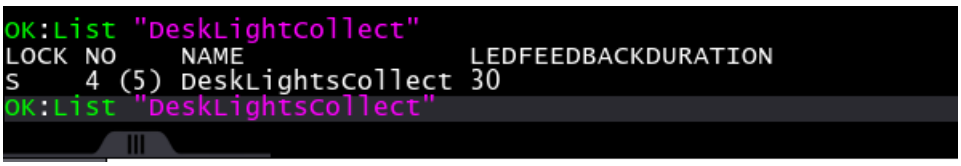


The children of the Station Settings are listed

3. To list the properties of one child, for example DeskLightsCollect, type:

```
User name@StationSettings> List DeskLightsCollect
```

The properties of the child DeskLightsCollect are now displayed in the command line history:



The properties of DeskLightsCollect are listed

4. To access the folder DeskLightsCollect within Station Settings, type:

```
User name@StationSettings> ChangeDestination DeskLightsCollect
```

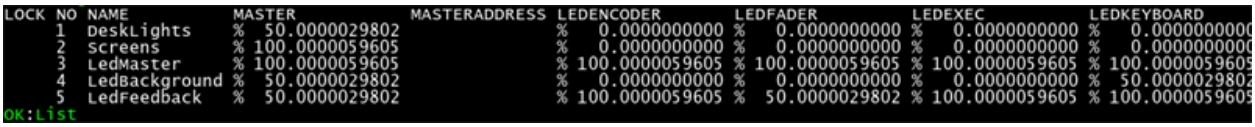
Result:

```
User name@StationSettings/DeskLightsCollect>
```

5. To list the properties of the children of DeskLightsCollect, type:

```
User name@StationSettings/DeskLightsCollect> List
```

The children's properties are now displayed.



Properties of DeskLightsCollect listed in the command line history

1.9.3.181. ListOwnership

To enter the ListOwnership keyword in the command line, use one of the options:

- Type **ListOwnership**
- Type the shortcut **Listo**

Description

ListOwnership is used as a troubleshooting keyword in case of a multi-user access conflict.

If a multi-user access conflict occurs, use the ListOwnership keyword. It lists the objects that are currently locked for all users in the session, hence causing the conflict.

Syntax

ListOwnership ([Object] ["Object_Name" or Object_Number])

Examples

- To display all objects that are locked for all users and that might cause a multi-user conflict, type:

```
User name[Fixture]>ListOwnership
```

- To display the ownership in macro 1 line 1, type:

```
User name[Fixture]>ListOwnership Macro 1
```

or type:

```
User name[Fixture]>ListOwnership Macro 1.1
```

or:

```
User name[Fixture]>ListOwnership Macro 1.*
```

1.9.3.182. ListReference

To enter the ListReference keyword in the command line, use one of the options:

- Press MA + List
- Type **ListReference**
- Type the shortcut **Listr**

Description

ListReference is used to display references and/or dependencies in your show file.

Examples of references between objects, only to mention a few:

- Layout to group, to macro, or to fixture
- Preset to cue, to preset (embedded preset)

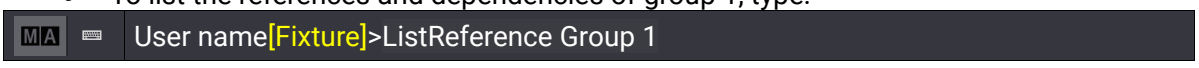
These references and/or dependencies depend on the use of the object in the show file.

Syntax

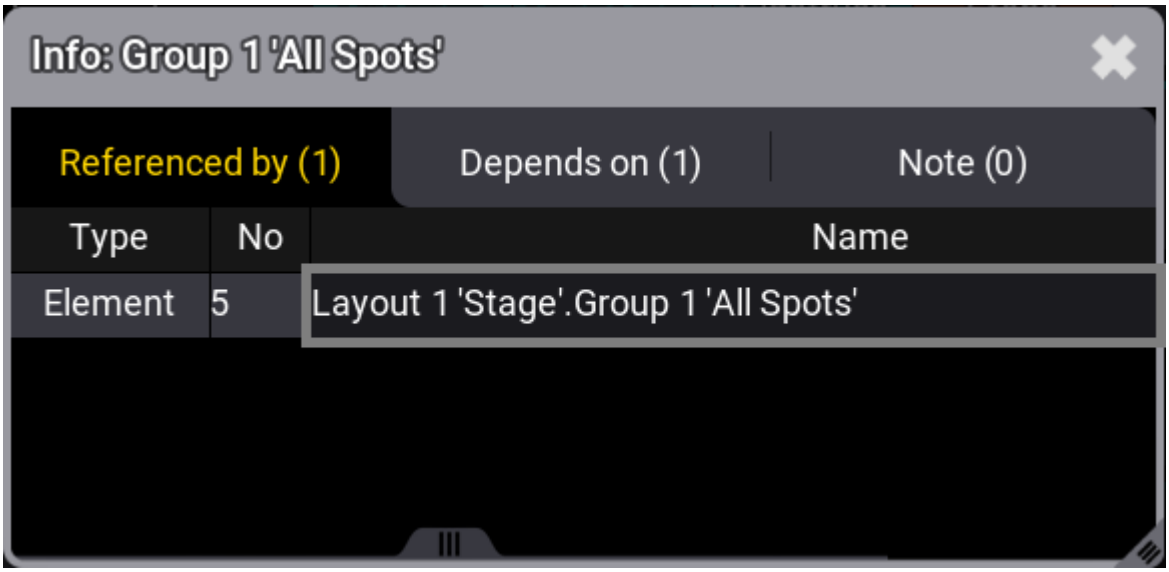
ListReference [Object] ["Object_Name" or Object_Number]

Example

- To list the references and dependencies of group 1, type:



The info pop-up opens listing references and/or dependencies:



Current references, dependencies, or notes

For more information see **Info Window**.

1.9.3.183. Load

To enter the Load keyword in the command line, use one of the options:

- Press Goto Goto
- Type **Load**

Description

The Load keyword is a playback keyword which is used to prepare an executor to jump to another cue rather than jumping to the next cue when a **Go+** is performed on the executor.

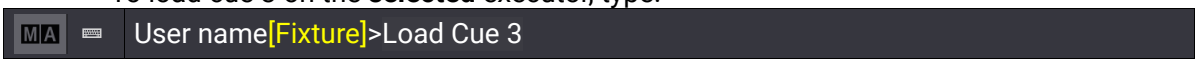
For more information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Load [Object] ["Object_Name" or Object_Number]

Examples

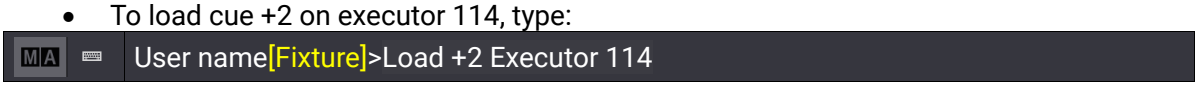
- To load cue 3 on the **selected** executor, type:



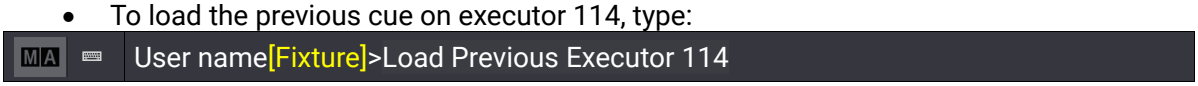
Cue 3 is loaded. To indicate that a cue is loaded, the display toggles between Cue 3 and Cue "Name".

- To load cue 5 on executor 114, type:

- To load cue +2 on executor 114, type:



- To load the previous cue on executor 114, type:



1.9.3.184. Loaded

	Important:
	MA + Go+ [large] immediately executes the **Go+ Loaded** command.

To enter the Loaded keyword in the command line, use one of the options:

- Press MA + Goto Goto
- Type **Loaded**

Description

The Loaded keyword is a function keyword which is used to simultaneously handle several cues in a sequence that are in the Load status.

For more information on Load see the **Load Keyword**.

Syntax

[Function] Loaded

Examples

Requirement: At least one cue has to be in the Load status.

- To start several loaded cues, type:

MA | User name[Fixture]>Go+ Loaded

- To disable sequences that are loaded, type:

MA | User name[Fixture]>Off Loaded

1.9.3.185. ListCrashLog

To enter the ListCrashLog keyword in the command line, use one of the options:

- Type **ListCrashLog**
- Type the shortcuts **ListC**

Description

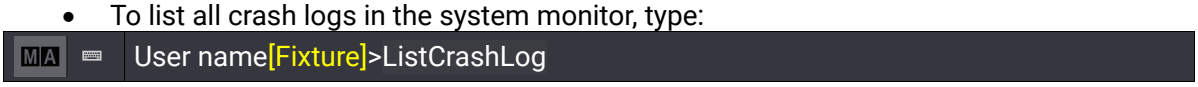
ListCrashLog is a function keyword that is used to list all crash logs in the system monitor.

Syntax

ListCrashLog

Example

- To list all crash logs in the system monitor, type:



1.9.3.186. LoadShow

To enter the LoadShow keyword in the command line, use one of the options:

- Type **LoadShow**
- Type the shortcut **Loa**

Description

The LoadShow keyword is a function keyword that loads a show from the folders: shows, demo shows, or backup**.**

For more information on the folder structure of shows, demo shows, and backups, see **Show File Handling** and **Folder Structure**.

Syntax

LoadShow ["Show_Name"] (/Option)

Option Keywords

The LoadShow keyword uses the following option keywords:

- **/All**
- **/DMXProtocols**
- **/LocalSettings**
- **/NoSave**
- **/NoShowData**
- **/OutputStations**
- **/Path**
- **/Save**
- **/Type**

Examples

- To load the show file with the file name "MacBeth", type:

```
User name[Fixture]>LoadShow "MacBeth"
```

- To create a new show file with the name "Henrietta", type:

```
User name[Fixture]>LoadShow "Henrietta"
```

Important:

	To load the new show again, save the new show file after loading it. For more information see **SaveShow keyword**.

1.9.3.187. Lock

To enter the Lock keyword in the command line, use one of the options:

- Type **Lock**
- Type the shortcut **Loc**

Description

The Lock keyword is a function keyword that locks objects.

Objects that are locked are indicated by a padlock.

Syntax

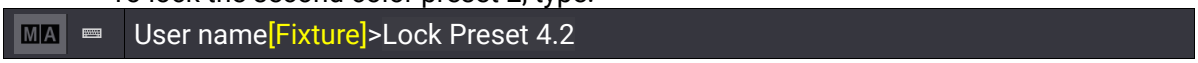
Lock [Object] ["Object_Name" or Object_Number]

Examples

- To lock macro 1, type:

To unlock objects see the **Unlock Keyword**.

- To lock the second color preset 2, type:



- To lock sequence 3, type:

1.9.3.188. LogIn

To enter the LogIn keyword in the command line, use one of the options:

- Type **LogIn**
- Type the shortcut **Log**

Description

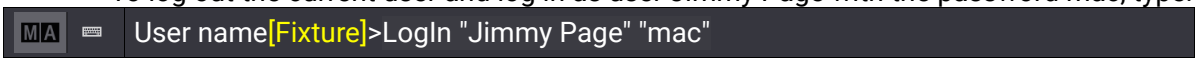
The LogIn keyword is a function keyword to log in another user.

Syntax

LogIn ["User_Name"] ["Password"]

Example

- To log out the current user and log in as user Jimmy Page with the password mac, type:



1.9.3.189. LogOut

To enter the LogOut keyword in the command line, use one of the options:

- Type **LogOut**
- Type the shortcut **Logo**

Description

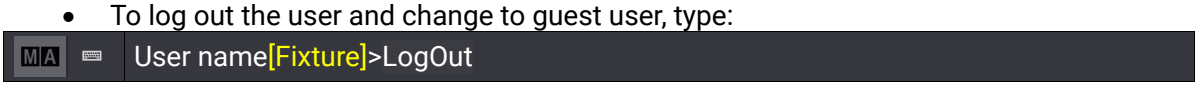
The LogOut keyword is a function keyword which is used to log out the user and change to guest user.

Syntax

LogOut

Example

- To log out the user and change to guest user, type:



1.9.3.190. Lowlight

To enter the Lowlight keyword in the command line, use one of the options:

- Press MA + Highlt
- Type **Lowlight**
- Type the shortcut **Low**

Description

The Lowlight keyword is a playback keyword that is used to apply the defined lowlight values in fixtures that are subselected when highlight is enabled.

Lowlight is also a toggle function meaning that entering Lowlight without using a helping keyword enables or disables the Lowlight mode.

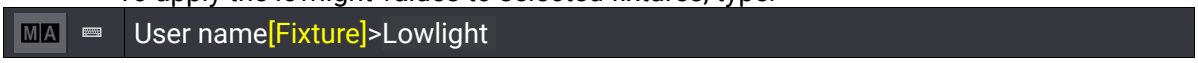
For more information on how to assign executors see **Assign Objects to an Executor**.

Syntax

([Function]) Lowlight (On/Off)

Example

- To apply the lowlight values to selected fixtures, type:



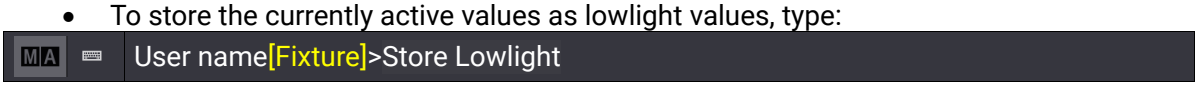
Example: Special Values

Requirement:

Activate values in the programmer:

-Call a preset or turn the attribute encoders.

- To store the currently active values as lowlight values, type:



For more information on the special values see **Parameter List**.

- To load the lowlight values of the selected fixtures into the programmer, type:

MA ⌨	User name[Fixture]>At Lowlight

1.9.3.191. Lua

To enter the Lua keyword in the command line, type **Lua**.

Description

The Lua keyword is used to execute commands in the script language Lua.

After entering Lua into the command line, type in the script language Lua version 5.4. The grandMA3 will directly execute the commands written in Lua.

To learn more about the grandMA3 specific Lua functions, see **Plugins** .

Syntax

Lua ["LuaCode"]

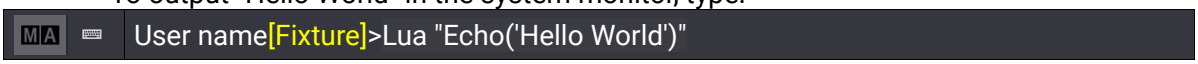
Examples

- To output "Hello World" in the Command Line History, type:

Result:

Hello World

- To output "Hello World" in the system monitor, type:



Result:

Hello World

1.9.3.192. LuaFile

To enter the LuaFile keyword in the command line, use one of the options:

- Type **LuaFile**
- Type the shortcut **LuaF**

Description

The LuaFile keyword is used to execute a Lua file directly without the need to import it into the grandMA3 show file.

Syntax

LuaFile ["AbsolutePathToLuaFile/FileName.lua"]

Example

- To execute the Lua file "execute_example.lua", type:

```
User name[Fixture]>LuaFile
"C:\ProgramData\MALightingTechnology\gma3_1.6.3\shared\resource\lib_plugins\examples\
execute_example\execute_example.lua"
```

1.9.3.193. Macro

To enter the **Macro** keyword in the command line, use one of the options:

- Press MA + X14 | Macro
- Type **Macro**
- Type the shortcut **M**

Description

The Macro keyword is an object keyword which is used to access Macros.

The default function for macro is Go+. This means calling macros without specifying any function starts the macro.

For more information see **Macros**.

Syntax

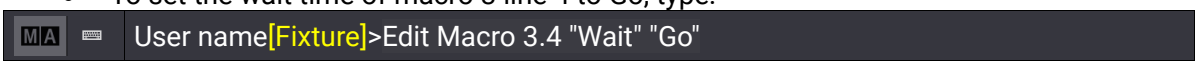
[Function] Macro ["Macro_Name" or Macro_Number]

[Function] DataPool ["DataPool_Name" or DataPool_Number] Macro ["Macro_Name" or Macro_Number]

Examples

- To start macro 5, type:

- To set the wait time of macro 3 line 4 to Go, type:



- To store a new and empty macro 2, type:

1.9.3.194. MArker

To enter the MArker keyword in the command line, use one of the options:

- Press Channel until MArker appears in the command line
- Type **MArker**
- Type the shortcut **MAr**

	**Hint:**
	To use the MArker keyword, make sure you assign the MArker ID type and a custom ID to a fixture first.

Description

The MArker keyword is an object keyword that is used to select fixtures of the ID type MArker.

Syntax

MArker ["MArker_Name" or MArker_Number]

Example

- To select MArker 30, type:

1.9.3.195. Master

To enter the **Master** keyword in the command line, use one of the options:

- Type **Master**
- Type the shortcut **Mas**

Description

The Master keyword is an object keyword that is used to address the different master functions, e.g., for the selected sequence, grand masters, speed masters and playback masters.

Syntax

[Function] Master ["MasterCategory_Name" or MasterCategory_Number].["Master_Name" or Master_Number]

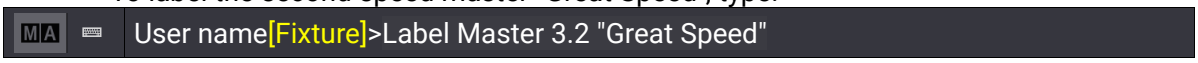
Master [MasterCategory_Number].["Master_Name" or Master_Number] At [SpeedReadout] [SpeedReadout_Value]

Examples

- To assign the master to the selected sequence on executor 206, type:

- To assign the grand master to executor 207, type:

- To label the second speed master "Great Speed", type:



- To assign the playback master 3 to executor 209 on page 6, type:

For more information on how to assign objects to executors see **Assign Object to an Executor**.

- To set the first speed master to a speed of 42 BPM, type:

User name[Fixture]>Master 3.1 At BPM 42

- To set the first speed master to a speed of 2 hertz, type:

User name[Fixture]>Master 3."Speed1" At Hz 2

- To set the first speed master to a speed of 0.69 seconds, type:

User name[Fixture]>Master 3."Speed1" At Seconds 0.69

Hint:

You can also set the speed readout of other objects such as sequences and presets.

1.9.3.196. MAtricks

To enter the MAtricks keyword in the command line, use one of the following options:

- Type **MAtricks**
- Type the shortcut **MAt**

Description

The MAtricks keyword behaves as an object type.

Used with an ID, MAtricks represents MAtricks objects stored in the MAtricks pool.

With the helping keywords On, Off, and Toggle, the MAtricks of the two selections may temporarily be enabled or disabled.

Furthermore, you can set the values of the MAtricks of the two selections.

Syntax

Set Selection (Selection_Number) MAtricks (Property) ["Property_Name" or Property_Value]

[Function] MAtricks ["MAtricks_Name" or MAtricks_Number]

Properties

X	Y	Z
XBlock	YBlock	ZBlock
XGroup	YGroup	ZGroup
XWings	YWings	ZWings
XWidth	YWidth	ZWidth
XShuffle	YShuffle	ZShuffle
XShift	YShift	ZShift

Examples

- To set the MAtricks X to 2 in the active selection, type:

```
User name[Fixture]>Set Selection MAtricks "X" 2
```

- To set the MAtricks XBlock to 4 in selection 2, type:

```
User name[Fixture]>Set Selection 2 MAtricks "XBlock" 4
```

- To disable MAtricks in the active selection, type:

User name[Fixture]>Off Selection MAtricks

- Toggle to activate MAtricks in the active selection, press Set or type:

User name[Fixture]>Toggle Selection MAtricks

- To reset the MAtricks in the first selection, type:

User name[Fixture]>Reset Selection 1 MAtricks

- To call the first MAtricks object in the MAtricks pool, type:

User name[Fixture]>Call MAtricks 1

- To label MAtricks 2 "Great", type:

User name[Fixture]>Label MAtricks 2 "Great"

- To assign the fourth MAtricks object of the pool to the first recipe in cue 1 part 0 of the selected sequence, type:

User name[Fixture]>Assign MAtricks 4 At Cue 1 Part 0.1

- To apply the speed value of 10 Hz to the speed of the X property of the MAtricks selection, type:

User name[Fixture]>Set Selection MAtricks "SpeedFromX" "Hz 10"

1.9.3.197. Measure

To enter the Measure keyword in the command line, use one of the options:

- Type **Measure**
- Type the shortcut **Mea**

Description

The Measure keyword is used together with the phaser speed (speed layer) to define the length of time of a phaser.

For more information see **Phasers**.

	**Hint:**
	The measure layer also affects how the width of each step is calculated.
	**Important:**
	When using the measure layer, multiple width value combinations can produce identical results. To ensure predictable timing, we recommended you consider the width value of a step as the percentage of a beat and ensure that the total width of all steps in the phaser equals the number of beats specified in the measure layer.

Syntax

Measure

[At] Measure [Value]

Examples

Requirement:

To set values in the measure layer, create at least 2 steps in the programmer.

- To set the selected layer to measure, type:

```
User name[Fixture]>Measure
```

- To set the measure of the selected feature group to 4 beats, type:

```
User name[Fixture]>At Measure 4
```

Result:

The phaser cycle now lasts 4 beats.

1.9.3.198. Media

To enter the **Media** keyword in the command line, use one of the options:

- Type **F** + **5**
- Type **Media**
- Type the shortcut **Med**

	**Hint:**
	To use the Media keyword, make sure you assign the Media ID type and a custom ID to a fixture first.

Description

The Media keyword is an object keyword which is used to call fixtures of the ID type Media in the programmer.

The name of this keyword can change because it is a custom ID type. For more information see the **Custom ID Type**.

Syntax

Media ["IDType_Name" or IDType_Number]

Example

- To select the fixture that uses the Media ID 2, type:

1.9.3.199. MemoryInfo

To enter the **MemoryInfo** keyword in the command line, use one of the options:

- Type **MemoryInfo**
- Type the shortcut **Mem**

Description

The MemoryInfo keyword is a command keyword that is used to indicate how much memory is occupied.

Syntax

MemoryInfo

Example

- To check how much memory is occupied, type:

1.9.3.200. Menu

To enter the **Menu** keyword in the command line, use one of the options:

- Type **Menu**
- Type the shortcut **Men**

Description

The Menu keyword is an object keyword which is used to address menus.

Syntax

([Function]) Menu ["Menu_Name" or Menu_Number]

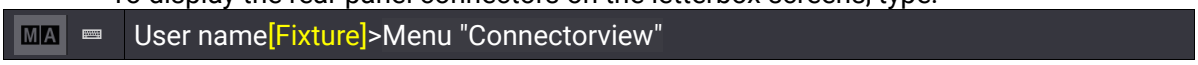
Examples

- To open the menu Network, type:

- To list all menus, type:

The command line feedback window lists all menus you can open using this keyword.

- To display the rear panel connectors on the letterbox screens, type:



1.9.3.201. Mesh

To enter the **Mesh** keyword in the command line, use one of the options:

- Type **Mesh**

Description

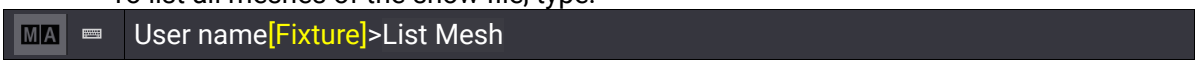
The Mesh keyword is an object keyword which is used to address meshes.

Syntax

[Function] Mesh ["Mesh_Name" or Mesh_Number]

Example

- To list all meshes of the show file, type:



1.9.3.202. Move

To enter the Move keyword in the command line, use one of the options:

- Press Move
- Type **Move**
- Type the shortcut **Mo**

Description

The Move keyword is a function keyword which is used to move objects to another location giving them a new object ID.

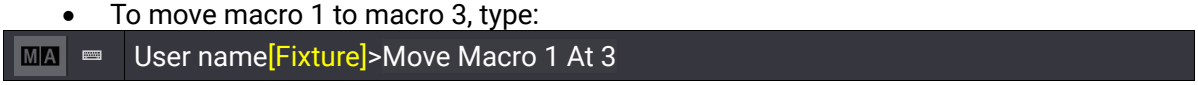
If the destination is already occupied, the object will be moved to the target object. The previous target object will be moved to the next object until an empty object will be occupied.

Syntax

Move [Object] ["Object_Name" or Object_Number] At ["Object_Name" or Object_Number]

Example

- To move macro 1 to macro 3, type:



1.9.3.203. Multipatch

To enter the Multipatch keyword in the command line, use one of the options:

- Press Channel until Multipatch appears in the command line
- Type **Multipatch**
- Type the shortcut **MU**

	Hint:
	To use the Multipatch keyword, make sure you create a Multipatch in an existing fixture first.

Description

The Multipatch keyword is a keyword which is used to address the multipatch fixtures of a fixture.

Syntax

Multipatch [Absolute_Multipatch_ID]

Fixture ["Fixture_Name" or Fixture_Number] Multipatch [Multipatch_ID]

Examples

- To select the second multipatch fixture of fixture 4, type:

```
User name[Fixture]>Fixture 4 Multipatch 2
```

- To patch the third multipatch fixture of fixture 2 to DMX address 6 in DMX universe 42, type:

```
User name[Fixture]>Patch Fixture 2 Multipatch 3 42.6
```

1.9.3.204. MyRunningPreset

To enter the MyRunningPreset keyword in the command line, use one of the options:

- Type **MyRunningPreset**
- Type the shortcut **MyRunningP**

Description

The MyRunningPreset keyword is used when there are several users in a session.

It allows you to disable your own running presets in a show file.

	Important:
	The numbers of the running presets are not equal to the numbers in the preset pool.

Syntax

[Function] MyRunningPreset ["Preset_Name" or Preset_Number]

Examples

- To disable your second preset of the gobo preset pool that is running, type:

```
User name[Fixture]>Off MyRunningPreset 3.2
```

- To list all your running presets, change destination to the running presets first and type:

```
User name[Fixture]>ChangeDestination MyRunningPreset
```

Result:

```
User name@Temp/RunningPlaybacksCollect/MyRunningPresets>
```

- To list all your running presets, type:

```
User name@Temp/RunningPlaybacksCollect/MyRunningPresets> List
```

1.9.3.205. MyRunningMacro

To enter the MyRunningMacro keyword in the command line, use one of the options:

- Type **MyRunningMacro**
- Type the shortcut **MyRunningM**

Description

The MyRunningMacro keyword is used when there are several users in a session.

It allows you to list or disable your own running macros in a show file.

	Important:
	The numbers of the running macros are not equal to the numbers of the macro in the macros data pool.

Syntax

[Function] MyRunningMacro ["Macro_Name" or Macro_Number]

Examples

- To disable the first macro that is running and which you started, type:

```
User name[Fixture]>Off MyRunningMacro 1
```

- To list all your running macros, change destination to the running macros first:

```
User name[Fixture]>ChangeDestination MyRunningMacro
```

Result:

```
User name@Temp/RunningPlaybacksCollect/MyRunningMacros>
```

- To list all your running macros, type:

```
User name@Temp/RunningPlaybacksCollect/MyRunningMacros> List
```

1.9.3.206. MyRunningSequence

To enter the MyRunningSequence keyword in the command line, use one of the options:

- Type **MyRunningSequence**
- Type the shortcut **My**

Description

The MyRunningSequence keyword is used when there are several users in a session.

It allows you to disable your own running sequences in a show file.

	Important:
	The numbers of the running sequences are not equal to the numbers of sequences in the sequences data pool.

Syntax

[Function] MyRunningSequence [Number]

Examples

- To disable the first sequence that is running and which you started, type:

```
User name[Fixture]>Off MyRunningSequence 1
```

- To list all your running sequences, change destination to the running sequences first:

```
User name[Fixture]>ChangeDestination MyRunningSequence
```

Result:

```
User name@Temp/RunningPlaybacksCollect/MyRunningSequences>
```

- To list all your running sequences, type:

```
User name@Temp/RunningPlaybacksCollect/MyRunningSequences>List
```

1.9.3.207. MyRunningSoundFile

To enter the MyRunningSoundFile keyword in the command line, use one of the options:

- Type **MyRunningSoundFile**
- Type the shortcut **MyRunningSo**

Description

The MyRunningSoundFile keyword is used when there are several users in a session.

It allows you to disable your own running sound files in a show file.

	**Important:**
	The numbers of the running sound files are not equal to the numbers of sound files in the sound pool.

Syntax

[Function] MyRunningSoundFile [Number]

Examples

- To disable the first sound file that is running and which you started, type:

```
User name[Fixture]>Off MyRunningSoundFile 1
```

- To list all your running sound files, change destination to the running sound files first:

```
User name[Fixture]>ChangeDestination MyRunningSoundFile
```

Result:

```
User name@Temp/RunningPlaybacksCollect/MyRunningSoundFiles>
```

- To list all your running sound files, type:

```
User name@Temp/RunningPlaybacksCollect/MyRunningSoundFiles> List
```

1.9.3.208. MyRunningTimecode

To enter the MyRunningTimecode keyword in the command line, use one of the options:

- Type **MyRunningTimecode**
- Type the shortcut **MyRunningT**

Description

The MyRunningTimecode keyword is used when there are several users in a session.

It allows you to disable your own running timecodes in a show file.

	Important:
	The numbers of the running timecodes are not equal to the numbers of timecodes in the timecode data pool.

Syntax

[Function] MyRunningTimecode [Number]

Examples

- To disable the first timecode that is running and which you started, type:

```
User name[Fixture]>Off MyRunningTimecode 1
```

- To list all your running timecodes, change destination to the running timecodes first:

```
User name[Fixture]>ChangeDestination MyRunningTimecode
```

Result:

```
User name@Temp/RunningPlaybacksCollect/MyRunningTimecodes>
```

- To list all your running timecodes, type:

```
User name@Temp/RunningPlaybacksCollect/MyRunningTimecodes>List
```

1.9.3.209. MyRunningTimer

To enter the MyRunningTimer keyword in the command line, use one of the options:

- Type **MyRunningTimer**
- Type the shortcut **MyRunningT**

Description

The MyRunningTimer keyword is used when there are several users in a session.

It allows you to list or disable your own running timers in a show file.

	**Important:**
	The numbers of the running timers are not equal to the numbers of timers in the timers data pool.

Syntax

[Function] MyRunningTimer ["Timer_Name" or Timer_Number]

Examples

- To disable the first timer that is running and which you started, type:

```
User name[Fixture]>Off MyRunningTimer 1
```

- To list all your running timers, change destination to the running timers first:

```
User name[Fixture]>ChangeDestination MyRunningTimer
```

Result:

```
User name@Temp/RunningPlaybacksCollect/MyRunningTimers>
```

- To list all your running timers, type:

```
User name@Temp/RunningPlaybacksCollect/MyRunningTimers>List
```

1.9.3.210. Natural

To enter the Natural keyword in the command line, use one of the options:

- Type **Natural**
- Type the shortcut **Na**

Description

The Natural keyword is used to set the values of a fixture selection using the notation that is defined as natural in the different attributes.

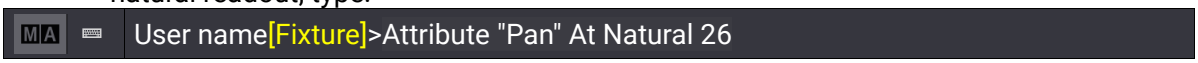
For information on how to define which readout is used as natural readout in each attribute in the user interface, see **Attribute Definitions**.

Syntax

Attribute ["Attribute_Name"] At Natural [Value]

Example

- To set the pan attribute of the selected fixtures to value 26 using the readout that is defined as natural readout, type:



1.9.3.211. NDI

To enter the NDI keyword in the command line, use one of the options:

- Type **NDI**
- Type the shortcut **ND**

Description

The keyword NDI is an object keyword that addresses NDI streams within the NDI folder of the video pool.

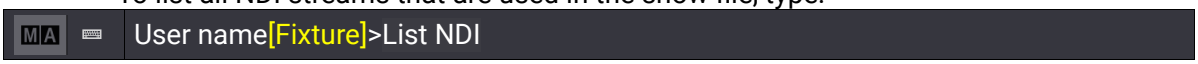
For more information on NDI and how to use it see **Video - Use an NDI Source**.

Syntax

[Function] NDI

Example

- To list all NDI streams that are used in the show file, type:



1.9.3.212. NewShow

To enter the NewShow keyword in the command line, use one of the options:

- Type **NewShow**
- Type the shortcut **New**

Description

The NewShow keyword is a function keyword that is used to create new shows.

Syntax

NewShow ["Show_Name"] (/Option)

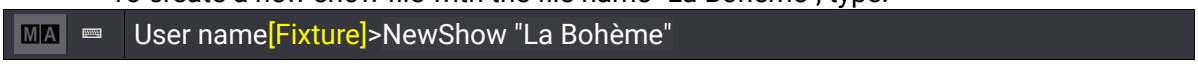
Option Keywords

The NewShow keyword uses the following option keywords:

- **/All**
- **/DMXProtocols**
- **/LocalSettings**
- **/NoShowData**
- **/OutputStations**
- **/Path**

Example

- To create a new show file with the file name "La Bohème", type:



1.9.3.213. Next

To enter the **Next** keyword in the command line, use one of the options:

- Press Next
- Type **Next**
- Type the shortcut **N**

	**Important:**
	To execute the command directly, press Next.

Description

If no fixtures are selected and the default keyword is Fixture, the Next keyword selects the fixture with the lowest fixture ID.

If only one fixture is selected and the default keyword is Fixture, the Next keyword selects the following fixture.

If multiple fixtures are selected, the Next keyword selects the following fixture one after the other within the selected block of fixtures.

For more information see **MAtricks and Shuffle**.

Syntax

Next

Option Keywords

The Next keyword uses the following option keywords:

- **/Wrap**

Example

- To step through single fixtures in the selected block of fixtures, press Next.

Name	FID	IDType	CID	Dimmer	PanTilt	
				Dim	P	T
AS QWO 1	1	Fixture		60	50	50
AS QWO 2	2	Fixture		60	50	50
AS QWO 3	3	Fixture		60	50	50
AS QWO 4	4	Fixture		60	50	50
AS QWO 5	5	Fixture		60	50	50
AS QWO 6	6	Fixture		60	50	50
AS QWO 7	7	Fixture		60	50	50
AS QWO 8	8	Fixture		60	50	50
AS QWO 9	9	Fixture		60	50	50
AS QWO 10	10	Fixture		60	50	50
AS QWO 11	11	Fixture		0	50	50
AS QWO 12	12	Fixture		0	50	50
AS QWO 13	13	Fixture		0	50	50

Using Next in a selected block of fixtures

1.9.3.214. NextY

To enter the **NextY** keyword in the command line, use one of the options:

- Type **NextY**

Description

The NextY keyword is a command keyword that is used to move the main selection from fixture to fixture on the y-axis. This can be observed in the selection grid for example.

Syntax

NextY

Examples:

Requirement:

- Select more than one fixture on the y-axis:

MA	User name[Fixture]>Grid 0/0 Thru 4/4
MA	User name[Fixture]>Fixture 1 Thru 25

- To move this selection downward, that is, on the y-axis, type:

MA	User name[Fixture]>NextY

1.9.3.215. NextZ

To enter the **NextZ** keyword in the command line, use one of the options:

- Type **NextZ**

Description

The NextZ keyword is a command keyword that is used to move the main selection from fixture to fixture on the z-axis. This can be observed in the selection grid for example.

Syntax

NextZ

Examples

Requirement:

- Select more than one fixture on the z-axis:

```
User name[Fixture]>Grid 0/0/0 Thru 4/0/4
User name[Fixture]>Fixture 1 Thru 25
```

- To move this selection upward, that is, on the z-axis, type:

```
User name[Fixture]>NextZ
```

1.9.3.216. NetworkNode

To enter the NetworkNode keyword in the command line, use one of the options:

- Type **NetworkNode**
- Type the shortcuts **Net** or **Node**

Description

The NetworkNode keyword is an object keyword which is used to display the grandMA3 nodes in the network.

Syntax

[Function] NetworkNode ["NetworkNode_Name" or NetworkNode_Number]

Examples

- To list all nodes that are currently in the same network, type:

```
User name[Fixture]>List NetworkNode
```

- To invite node "FOH3" into the session, type:

```
User name[Fixture]>Invite NetworkNode "FOH3"
```

1.9.3.217. NonDim

To enter the **NonDim** keyword in the command line, use one of the options:

- Press Channel until NonDim appears in the command line
- Type **F + 4**
- Type **NonDim**
- Type the shortcut **Non**

	**Hint:**
	To use the NonDim keyword, make sure you assign the NonDim ID type and a custom ID to a fixture first.

Description

The NonDim keyword is an object keyword which is used to call fixtures of the ID type NonDim in the programmer.

The name of this keyword can change because it is a custom ID type. For more information see the **Custom ID Type Topic**.

Syntax

NonDim ["NonDim_Name" or NonDim_Number]

Example

- To select the fixture that uses the NonDim ID 1, type:

1.9.3.218. Normal

	Important:
	Pressing At twice executes the Normal keyword.

To enter the Normal keyword in the command line, use one of the options:

- Press At At
- Type **Normal**
- Type the shortcut **No**

Description

Normal is an object keyword wich is used to set the dimmer of the fixtures to the defined normal value.

For information on how to change the normal value see **User Settings**.

Syntax

Normal

Example

- To set the dimmer of the selected fixtures to the normal value, type:

```
User name[Fixture]>Normal
```

1.9.3.219. Note

To enter the Note keyword in the command line, use one of the options:

- Type **Note**
- Type the shortcut **Not**

Description

The Note keyword is used to add or edit a short note in an object.

For more information see **Notes**.

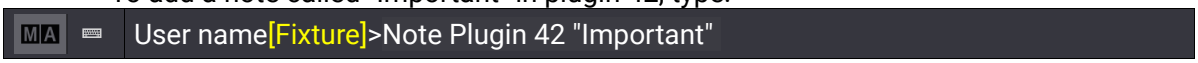
Syntax

Note [Object] ["Object_Name" or Object_Number] (["Text"])

Examples

- To add a note in macro 5, type:

- To add a note called "Important" in plugin 42, type:



1.9.3.220. Off

To enter the Off keyword in the command line, use one of the options:

- Press Off
- Type **Off**
- Type the shortcut **Of**

Description

The Off keyword is used as a function keyword to:

- Stop an executor
- Knock out parameters in the programmer
- Knock out selections in the programmer
- Knock out active attributes in the programmer

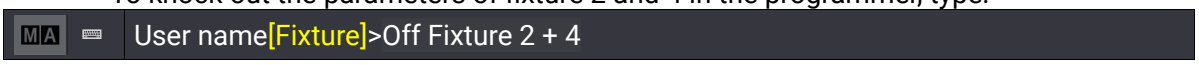
For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

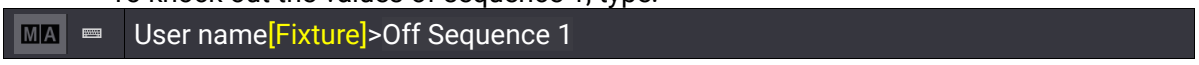
Off [Object] ["Object_Name" or Object_Number]

Examples

- To knock out the parameters of fixture 2 and 4 in the programmer, type:



- To knock out the values of sequence 1, type:



- To turn off all sequences containing loaded cues, type:

1.9.3.221. Offset

	Hint:
	Store Offset is more precise when the resulting offsets are as small as possible. It is also possible to use relative values in the programmer.

To enter the Offset keyword in the command line, use one of the options:

- Press MA + Update
- Type **Offset**
- Type the shortcut **Offs**

	Important:
	If you use the key combination MA + Update to enter the Offset keyword in the command line, make sure you enter the Store keyword in the command line first.

Description

The Offset keyword is a helping keyword that helps calculate the pan and tilt offset in fixtures.

Syntax

[Function] Offset

Example

Requirement:

1. Call a preset into the programmer or play a cue in the fixtures where you want to calculate the offset;
2. The position of the fixtures in 3D have to be aligned with the fixtures on real stage;
3. Adjust the pan and tilt values in the programmer so that the fixtures are in the same position on real stage.
   - To calculate and use offset in the selected fixtures, type:

1.9.3.222. On

To enter the On keyword in the command line, use one of the options:

- Press On
- Type **On**

Description

The On keyword is used as a playback or as a helping keyword.

Use the On keyword as a playback keyword to:

- Start or restart an executor;
- Activate selection in the programmer;
- Activate attributes in the programmer.

Use the On keyword as a helping keyword to:

- Indicate the start of a temporary function;
- Enable the state of a toggle function.

For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Function keyword:

On [Object] ["Object_Name" or Object_Number]

Helping keyword:

[Function] On

Examples

- To activate the attributes of fixture 2 and 4 in the programmer, type:

- To activate sequence 3, type:

1.9.3.223. NetworkSpeedTest

To enter the NetworkSpeedTest keyword in the command line, use one of the options:

- Type **NetworkSpeedTest**
- Type the shortcut **Net**

Description

The NetworkSpeedTest keyword is used to excute a test of the network connection in devices defined in the command.

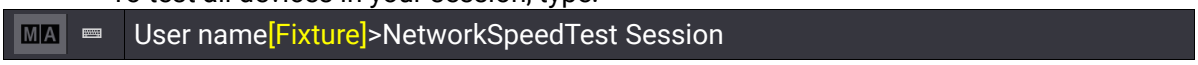
Syntax

NetworkSpeedTest [DeviceType] ["Device_Name" or Device_Number]

Examples

- To execute a network speed test in all consoles that are available in the network, type:

- To test all devices in your session, type:



1.9.3.224. onPC

To enter the onPC keyword in the command line, use one of the options:

- Type **onPC**
- Type the shortcut **onP**

Description

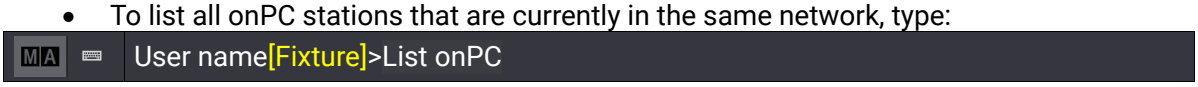
The onPC keyword is an object keyword which is used to display all onPC stations in the network.

Syntax

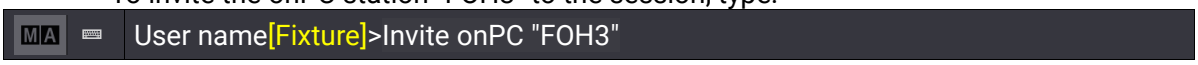
[Function] onPC ["onPC_Name" or onPC_Number]

Examples

- To list all onPC stations that are currently in the same network, type:



- To invite the onPC station "FOH3" to the session, type:



1.9.3.225. OSC

To enter the **OSC** keyword in the command line, use one of the options:

- Type **OSC**
- Type the shortcut **Os**

Description

The OSC keyword represents the open sound control data.

OSC keyword lists, exports or imports open sound control data in Remote Inputs.

Syntax

[Function] OSC

Examples

1. List all OSC data in an interface

1. Change destination to OSC:

```
User name[Fixture]>ChangeDestination OSC
```

Result:

```
User name@ShowData/OSCBase>
```

2. To list OSC data in the interface, type:

```
User name@ShowData/OSCBase> List
```

2. Import OSC data in Remote Input

Requirement: Change destination to OSC.

For more information see **Example 1**.

- To import OSC data, type:

User name@ShowData/OSCBase> Import

If the target already exists, the import pop-up will appear asking you to choose an option:

Target is occupied

- Choose an option.

The OSC is imported.

1.9.3.226. Oops

	Hint:
	Use **Undo** as a synonym of the Oops keyword.

To enter the Oops keyword in the command line, use one of the options:

- Press Oops
- Type **Oops**
- Type the shortcut **O**

Description

The Oops keyword is a function keyword which is used to cancel:

- The last action made in the command line;
- The last fixture selection;
- The last action made in the programmer.

Syntax

Oops

Option Keywords

The Oops keyword uses the following option keywords:

- **/NoConfirmation**

Example

- To undo the last operation, type:

MA	User name[Fixture]>Oops

1.9.3.227. OutputLayer

To enter the OutputLayer keyword in the command line, use one of the options:

- Type **OutputLayer**
- Type the shortcut **OU**

Description

OutputLayer is a layer keyword. It is used to change the selected layer to the OutputLayer. This layer can show information in, for instance, the fixture sheet.

Syntax

OutputLayer

Example

- To select the OutputLayer, type:

1.9.3.228. Page

To enter the Page keyword in the command line, use one of the options:

- Press MA + X15 | Page
- Type **Page**
- Type the shortcut **P**

	**Important:**
	Before calling a page, create a page in the page pool first.

Description

The Page keyword is an object keyword which is used to access pages.

Syntax

([Function]) Page ["Page_Name" or Page_Number] ([/Option] ["Option_Value"])

Option Keywords

The Page keyword uses the following option keywords:

- **/Tab**

Examples

- To change to page 2, type:

```
User name[Fixture]>Page 2
```

- To label page 2 "Ray", type:

```
User name[Fixture]>Label Page 2 "Ray"
```

- To call the page with the name "Ray", type:

```
User name[Fixture]>Page "Ray"
```

- To change to the next page, type:

```
User name[Fixture]>Next Page
```

1.9.3.229. Part

To enter the Part keyword in the command line, use one of the options:

- Press Cue Cue
- Type **Part**
- Type the shortcut **Par**

Description

The Part keyword is an object keyword which is used to segment cues in parts.

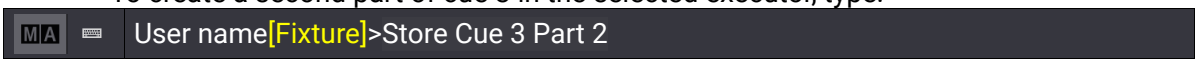
Parts are useful for the assignment and the edit of different timings of groups of fixture parameters.

Syntax

[Function] Part ["Part_Name" or Part_Number]

Examples

- To create a second part of cue 3 in the selected executor, type:



- To delete part 3 of cue 1, type:

1.9.3.230. Park

To enter the Park keyword in the command line, use one of the options:

- Press Pause Pause
- Type **Park**

Description

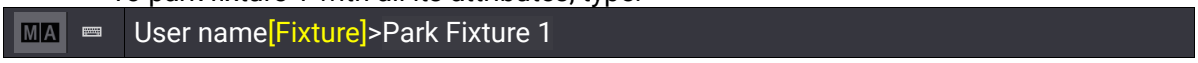
The Park keyword is a command keyword which is used to prevent DMX channels of fixtures to change their value.

Syntax

Park [Object] ["Object_Name" or Object_Number]

Examples

- To park fixture 1 with all its attributes, type:



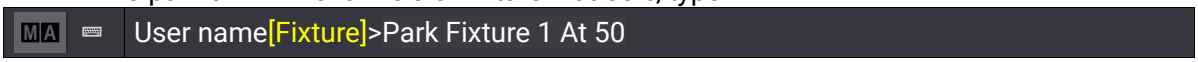
- To park the current selection, type:

It is also possible to enter Park into the command line and tap a cell in the fixture sheet to park a certain attribute.

To unpark fixtures and/or attributes, see the **Unpark Keyword**.

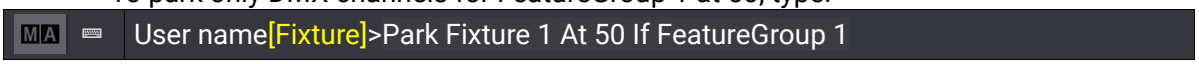
	Important:
	When parking a fixture and/or attribute it will park the corresponding DMX channel.

- To park all DMX channels of fixture 1 at 50%, type:



	Hint:
	The command **Park Fixture At** will park all DMX channels of the fixture to the value that is set.

- To park only DMX channels for FeatureGroup 1 at 50, type:

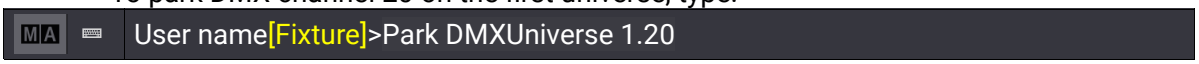


- To park DMX universe 2, type:

To specify the universe:

1. Enter the Park keyword in the command line.
2. Tap the universe in the universe pool.

- To park DMX channel 20 on the first universe, type:



To specify the DMX channel:

1. Enter the Park keyword in the command line.
2. Tap the channel in the DMX sheet.

	Hint:
	If there are parked channels in a universe, they will be indicated by a blue **P** icon in the universe pool.

1.9.3.231. Paste

To enter the Paste keyword in the command line, use one of the options:

- Press Copy Copy
- Type **Paste**
- Type the shortcut **Pas**

Description

The Paste keyword pastes or moves previously cut or copied objects.

For more information see the **Cut keyword** and the **Copy keyword**.

Syntax

Paste Object ["Object_Name" or Object_Number] (/Option)

Option Keywords

The Paste keyword uses the following option keywords:

- **/Date**

Examples

- To cut and then paste group 1 to group 5, type:

```
User name[Fixture]>Cut Group 1
User name[Fixture]>Paste Group 5
```

- To copy and then paste cue 5 to cue 15, type:

```
User name[Fixture]>Copy Cue 5
User name[Fixture]>Paste Cue 15
```

1.9.3.232. Patch

To enter the **Patch** keyword in the command line, use one of the options:

- Type **Patch**
- Type the shortcut **Pat**

Description

The Patch keyword is used to edit the patch address of single fixtures or entire fixture selections.

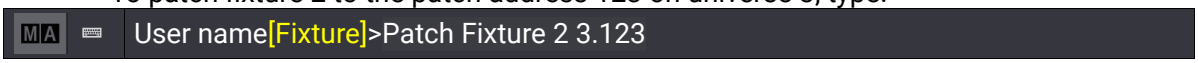
Syntax

Patch Fixture ["Fixture_Name" or Fixture_Number] [Universe.Address]

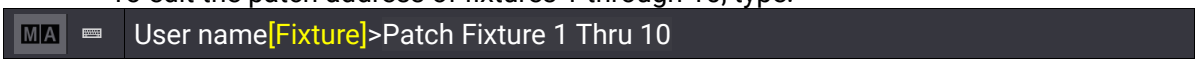
Patch Fixture ["Fixture_Name" or Fixture_Number] Thru ["Fixture_Name" or Fixture_Number]

Examples

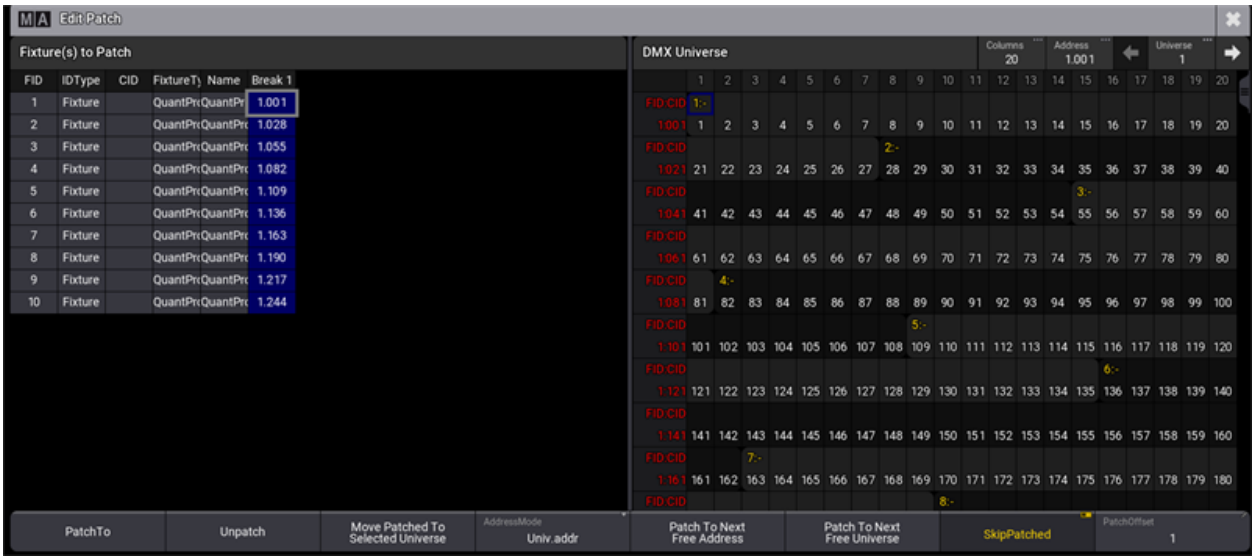
- To patch fixture 2 to the patch address 123 on universe 3, type:



- To edit the patch address of fixtures 1 through 10, type:



The Edit Patch opens:



Edit patched fixtures

You can now patch the fixtures to a new patch address or unpatch them using the user interface.

1.9.3.233. Pause

To enter the Pause keyword in the command line, use one of the options:

- Press Pause
- Type **Pause**
- Type the shortcut **Pau**

Description

The Pause keyword is a playback keyword which is used to pause:

- Crossfades between steps and cues;
- Timecode shows;
- Macros.

	**Hint:**
	Pause is a toggle function between **Pause On** and **Pause Off**.

For information on how to assign executors see **Assign Objects to an Executor**.

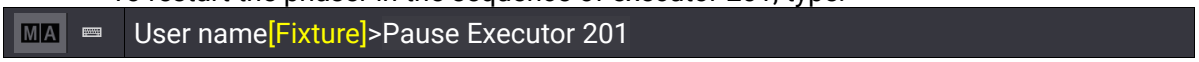
Syntax

Pause (On/Off) [Object] ["Object_Name" or Object_Number]

Examples

- To pause an active phaser in the sequence on executor 201, type:

- To restart the phaser in the sequence of executor 201, type:



1.9.3.234. Percent

To enter the Percent keyword in the command line, use one of the options:

- Type **Percent**
- Type the shortcut **Pe**

Description

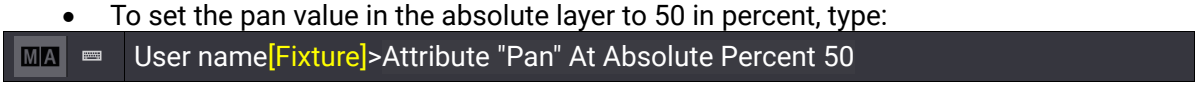
The Percent keyword is used to set the values of a fixture selection using the percent (%) notation.

Syntax

(Attribute ["Attribute_Name" or Attribute_Number]) At ([Layer]) Percent [Value]

Example

- To set the pan value in the absolute layer to 50 in percent, type:



1.9.3.235. PercentFine

To enter the PercentFine keyword in the command line, use one of the options:

- Type **PercentFine**
- Type the shortcut **Percentf**

Description

The PercentFine keyword is used to set the values of a fixture selection taking into consideration two decimal places.

Syntax

(Attribute ["Attribute_Name" or Attribute_Number]) At ([Layer]) PercentFine [Value]

Examples

- To set the dimmer on the absolute layer to 50.35% in PercentFine, type:

```
User name[Fixture]>At Absolute PercentFine 50.35
```

- To set the pan value in the absolute layer to 75.50%, type:

```
User name[Fixture]>Attribute "Pan" At Absolute PercentFine 75.50
```

1.9.3.236. Phase

To enter the Phase keyword in the command line, use one of the options:

- Type **Phase**
- Type the shortcut **Ph**

Description

Phase is an object keyword that is used to set the Phase layer.

For more information see **Phasers**.

Syntax

Phase

At Phase [Value] (Thru [Value])

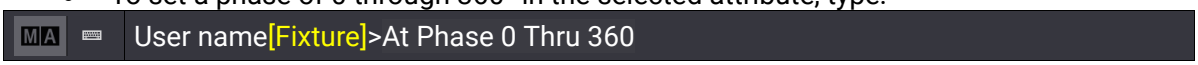
Examples

- To change to the phase layer, type:

Requirement:

In this example, at least 2 steps have to be active in the programmer.

- To set a phase of 0 through 360° in the selected attribute, type:



1.9.3.237. Physical

To enter the Physical keyword in the command line, use one of the options:

- Type **Physical**
- Type the shortcut **Phy**

Description

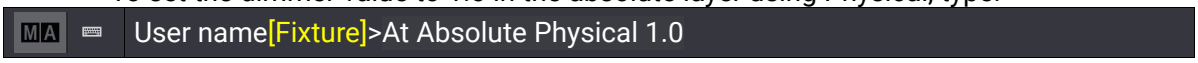
The Physical keyword is used to set the physical values of a fixture selection using the Physical notation. It comprises RPM (rounds per minute), Hz (Hertz), degrees, or intensity.

Syntax

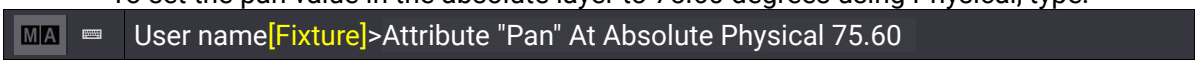
(Attribute ["Attribute_Name" or Attribute_Number]) At ([Layer]) Physical [Value]

Examples

- To set the dimmer value to 1.0 in the absolute layer using Physical, type:



- To set the pan value in the absolute layer to 75.60 degrees using Physical, type:



1.9.3.238. Plugin

To enter the Plugin keyword in the command line, use one of the options:

- Press MA + X14 | Macro + X14 | Macro
- Type **Plugin**
- Type the shortcut **Pl**

Description

The Plugin keyword is an object keyword which is used to access plugins.

The default function of the Plugin keyword is **Go+**.

Syntax

([Function]) Plugin ["Plugin_Name" or Plugin_Number](.["LuaComponent_Name" or LuaComponent_Number]) ("Argument_Value")

Examples

- To edit plugin 2, type:

- To label plugin 1 "Weakon," type:

Call a Plugin and Specify an Argument

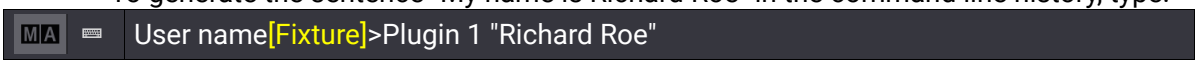
Requirement:

- Create a plugin that uses an argument when calling a function.

In this example our plugin is plugin 1 of the plugin pool and the argument is called name in the definition of the function.

```
Lua
return function(display_handle, name)
    Printf("My name is "..name)
end
```

- To generate the sentence "My name is Richard Roe" in the command line history, type:



Result:

Response in the command line history

"My name is Richard Roe" is now displayed in the command line history.

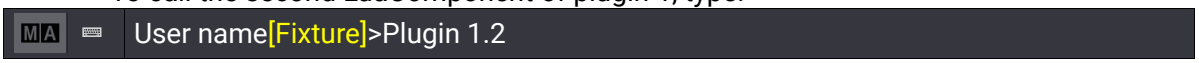
Call a Dedicated LuaComponent

Requirement:

- Create at least two lua components in the plugin.

For more information on how to create lua components see **Plugins**.

- To call the second LuaComponent of plugin 1, type:



1.9.3.239. Preset

To enter the Preset keyword in the command line, use one of the options:

- Press Preset
- Type **Preset**
- Type the shortcut **Pres**

Description

With the Preset keyword, it is possible to:

- Select the fixtures stored in a preset;
- Apply the At function on the preset within the fixture or channel selection;
- Set a property in a preset.

A command containing only the Preset keyword and the preset ID performs the default function SelectFixtures. For more information see **SelectFixtures keyword**.

To give the fixture attributes a link to the preset, the At keyword needs to be added in front of the Preset keyword and the preset ID. For more information see **At Keyword**.

Syntax

Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number]

[Function] Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] ([Setting] ["Setting_Value"] [/OptionKeyword])

Assign [Object] ["Object_Name" or Object_Number] At Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number]

Settings

Some settings need an object assigned. These can be assigned using the **Assign Keyword**.

Other settings contain a text option or a value. The **Set keyword** is used for these settings.

The following table displays the settings that need an object:

Setting	Object	Description
Appearance	"Appearance 1"	Assigns the appearance to the pool object.
InputFilter	"Filter 12"	Assigns a filter or world as an input filter to the pool object.
Scribble	"Scribble 1"	Assigns the scribble to the preset pool object.

The following table displays the settings that need an option or value:

Setting	Option/Value	Description
Name	"Preset Name"	Sets the name of the preset pool object.
MoveGridCursor	"Yes" or "No"	This defines if the grid cursor is moved after calling the preset.
CuePart	"Default" or a specific part number	This is the programmer cue part the preset will be called into.
MAgic	"Yes" or "No"	This defines if the preset is a magic preset or not.
PresetMode	Read only	This is information only about the preset mode.
StoredData	Read only	This is information only about the stored data.

Presets can also contain settings for MAtricks and recipe values. These can be changed using the **Set keyword**.

Option Keywords

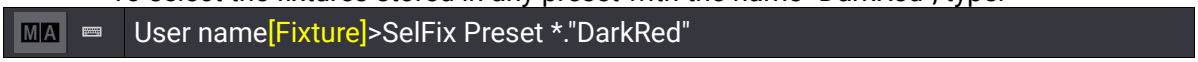
The Preset keyword uses the following option keywords:

- **/Active**
- **/ActiveForSelected**
- **/AddNewContent**
- **/All**
- **/AllForSelected**
- **/Ask**
- **/Auto**
- **/DMX**
- **/Embed**
- **/ForceGlobal**
- **/Global**
- **/GridMergeMode**
- **/InputFilter**
- **/KeepActivation**
- **/Look**
- **/MAtricks**
- **/Output**
- **/Overwrite**
- **/PhaserData**
- **/Programmer**
- **/Selective**
- **/Universal**

Examples

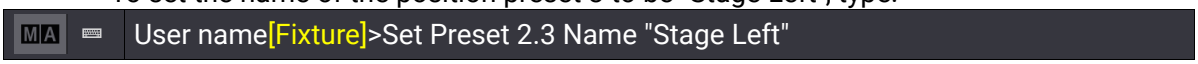
- To select the fixtures that can use preset 5 of the dimmer feature group, type:

- To select the fixtures stored in any preset with the name "DarkRed", type:

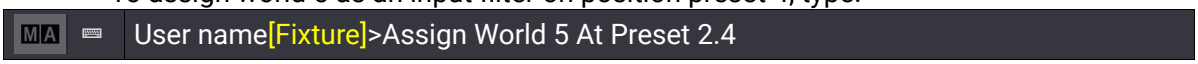


- To call a reference to preset 21.45 ("All 1" preset number 45) to the attributes of the selected fixtures, type:

- To set the name of the position preset 3 to be "Stage Left", type:



- To assign world 5 as an input filter on position preset 4, type:



1.9.3.240. PresetUpdate

To enter the PresetUpdate keyword in the command line, use one of these options:

- Type **PresetUpdate**
- Type the shortcut **PresetU**

Description

The PresetUpdate keyword is an object keyword that contains all presets that can be currently updated.

To show the preset list, use the **List keyword**.

Syntax

[Function] PresetUpdate (["PresetUpdate_Name" or PresetUpdate_Number])

Examples

- To list all the presets that can be updated in the command line feedback, type:

The same list is displayed in the **Update Menu**.

- To update the second preset that is part of the PresetUpdate List , type:

1.9.3.241. Press

To enter the Press keyword in the command line, type **Press**.

Description

The Press keyword is used to simulate the pressed state of a key.

It is possible to assign this state to macros.

For more information see **Macros**.

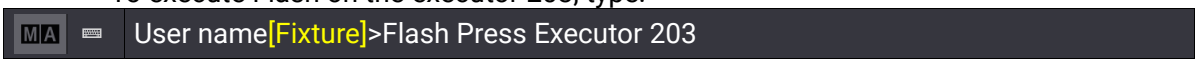
For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

[Function] Press [Object] ["Object_Name" or Object_Number]

Example

- To execute Flash on the executor 203, type:



To unpress a key, see **Unpress Keyword**.

1.9.3.242. Preview

To enter the Preview keyword in the command line, use one of the options:

- Press Preview
- Type **Preview**
- Type the shortcut **Previe**

Description

The Preview keyword is used to toggle the preview mode. For more information on preview see **What is the Programmer**.

Syntax

Preview (On/Off)

Example

- To toggle the preview mode, type:

The title bar and the frame around the windows that display the preview mode turn red.

1.9.3.243. Previous

To enter the Previous keyword in the command line, use one of the options:

- Press Prev
- Type **Previous**
- Type the shortcut **Prev**

Description

The Previous keyword has several functions:

- If no fixture is selected, the fixture with the highest ID number will be selected.
- If one fixture is selected, the fixture before this fixture will be selected.
- If MAtricks are active, the Previous keyword has special functions depending on the selected MAtricks options.

For more information on MAtricks see **MAtricks and Shuffle**.

Syntax

([Function]) Previous ([Object])

Option Keywords

The Previous keyword uses the following option keywords:

- **/Wrap**

Examples

- To select the last fixture of the selected fixtures in the selection order, type:

User name[Fixture]>Previous

- To go to the previous page in the page pool, type:

User name[Fixture]>Previous Page

- To load the previous cue in the selected sequence, type:

User name[Fixture]>Load Previous

1.9.3.244. PreviousY

To enter the **PreviousY** keyword in the command line, use one of the options:

- Type **PreviousY**

Description

The PreviousY keyword is a command keyword that is used to move the main selection from fixture to fixture back to the previous position on the y-axis. This can be observed in the selection grid for example.

Syntax

PreviousY

Example

Requirement: Select more than one fixture on the y-axis

MA	User name[Fixture]>Grid 0/0 Thru 4/4
MA	User name[Fixture]>Fixture 1 Thru 25

- To move this selection to the previous position on the y-axis, type:

MA	User name[Fixture]>PreviousY

1.9.3.245. PreviousZ

To enter the **PreviousZ** keyword in the command line, use one of the options:

- Type **PreviousZ**

Description

The PreviousZ keyword is a command keyword that is used to move the main selection from fixture to fixture back to the previous position on the z-axis. This can be observed in the selection grid for example.

Syntax

PreviousZ

Example

Requirement: Select more than one fixture on the z-axis

MA ⌨	User name[Fixture]>Grid 0/0/0 Thru 4/0/4
MA ⌨	User name[Fixture]>Fixture 1 Thru 25

- To move this selection to the previous position on the z-axis, type:

MA ⌨	User name[Fixture]>PreviousZ

1.9.3.246. Programmer

To enter the Programmer keyword in the command line, use one of the options:

- Press MA + Cue
- Type **Programmer**
- Type the shortcut **Prog**

Description

The Programmer keyword represents the programmer playback. It is used to enable, disable, or pause values in the programmer.

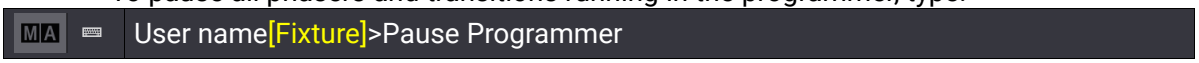
Syntax

[Function] Programmer ([ProgrammerPart_Number])

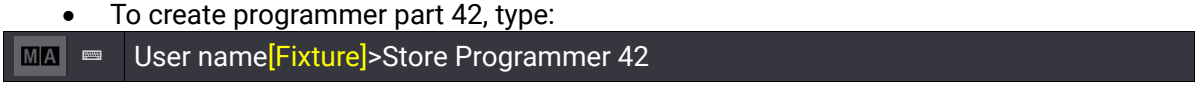
Examples

Requirement: Phasers are running

- To pause all phasers and transitions running in the programmer, type:



- To create programmer part 42, type:



1.9.3.247. Property

To enter the Property keyword in the command line, use one of the options:

- Type **Property**
- Type the shortcut **Prop**

Description

The Property keyword is an object keyword which is used to communicate with the console for you to set a specific property.

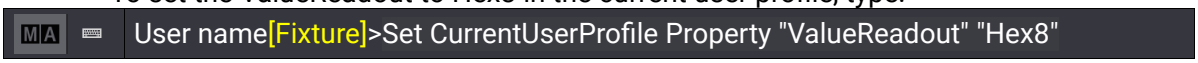
It is used in conjunction with the **Set keyword**, the **SetUserVariable keyword**, or the **SetGlobalVariable keyword**.

Syntax

[Function] [Object] Property ["Property_Name"] ["Value"]

Example

- To set the ValueReadout to Hex8 in the current user profile, type:



1.9.3.248. ProcessingUnit

To enter the ProcessingUnit keyword in the command line, use one of the options:

- Type **ProcessingUnit**
- Type the shortcut **PU** or **Proc**

Description

The ProcessingUnit keyword is an object keyword which is used to address the grandMA3 processing units in the network.

Syntax

[Function] ProcessingUnit ["ProcessingUnit_Name" or ProcessingUnit_Number]

Examples

To list all processing units that are currently in the same network, type:

To invite processing unit "FOH3" into the session, type:

1.9.3.249. PSR

To enter the PSR keyword in the command line, use one of the options:

- Press Channel until PSR appears in the command line
- Type **F + 11**
- Type **PSR**
- Type the shortcut **Ps**

	Hint:
	To use the PSR keyword, make sure you assign the PSR ID type and a custom ID to a fixture first.

Description

The PSR keyword is an object keyword which is used to select fixtures of the ID type PSR in the programmer.

	Hint:
	The PSR ID type will be automatically allocated when using PSR to ensure that all fixtures have at least a CID of the ID type PSR. For more information see **Partial Show Read**.

Syntax

PSR ["PSR_Name" or PSR_Number]

Example

- To select the the fixture that uses the PSR ID 2, type:

MA | User name[Fixture]>PSR 2

1.9.3.250. Pyro

To enter the Pyro keyword in the command line, use one of the options:

- Press Channel until Pyro appears in the command line
- Type **F + 8**
- Type **Pyro**
- Type the shortcut **Py**

	**Hint:**
	To use the Pyro keyword, make sure you assign the Pyro ID type and a custom ID to a fixture first.

Description

The Pyro keyword is an object keyword which is used to call fixtures of the ID type Pyro in the programmer.

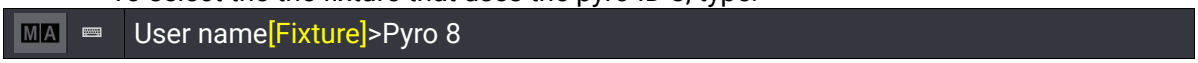
The name of this keyword can change because it is a custom ID type. For more information see the **Custom ID Type Topic**.

Syntax

Pyro ["Pyro_Name" or Pyro_Number]

Example

- To select the the fixture that uses the pyro ID 8, type:



1.9.3.251. Quickey

To enter the Quickey keyword in the command line, use one of the options:

- Press MA + X14 | Macro + X14 | Macro + X14 | Macro
- Type **Quickey**
- Type the shortcut **Q**

Description

The Quickey keyword is an object keyword that is used to address quickeys in the pool.

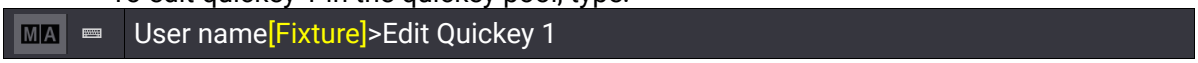
For more information see **Quickeys**.

Syntax

[Function] Quickey ["Quickey_Name" or Quickey_Number]

Example

- To edit quickey 1 in the quickey pool, type:



1.9.3.252. Rate1

To enter the Rate1 keyword in the command line, use one of the options:

- Press MA + Learn
- Type **Rate1**
- Type the shortcut **Ra**

Description

The Rate1 keyword is a playback keyword that is used to reset the rate of a phaser to 1:1.

For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Rate1 [Object] ["Object_Name" or Object_Number]

Example

- To reset the rate of executor 105 back to 1:1, type:

1.9.3.253. RDM

To enter the RDM keyword in the command line, use one of the options:

- Type **RDM**
- Type the shortcut **RD**

Description

The keyword RDM is an object keyword that addresses RDM objects within the RDM folder.

For more information on RDM and how to use it see **RDM**.

Syntax

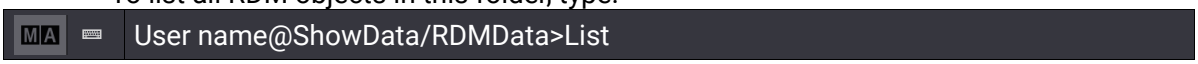
[Function] RDM

Examples

- Access the RDM folder first, type:

Result:

- To list all RDM objects in this folder, type:



1.9.3.254. Readout

	**Important:**
	Changing the Readout is a user profile setting. That is, the Readout will be automatically set throughout the entire user profile. For more information see **User Settings**.
	**Hint:**
	-The Readout values are: Decimal8, Decimal24, Hex8, Hex24, Percent, PercentFine, Physical. -The names of the readout are case-sensitive, e.g., "**H**ex8", "**P**ercent".

To enter the Readout keyword in the command line, use one of the options:

- Type **Readout**
- Type the shortcut **Rea**

Description

The Readout keyword is used to set the readout of fixtures in the usual notation.

Syntax

Readout ["Readout_Notation"]

Example

- To set the Readout to Hex8, type:

1.9.3.255. Reboot

To enter the Reboot keyword in the command line, use one of the following options:

- Type **Reboot**
- Type the shortcut **R**

Description

The Reboot keyword is a function keyword that is used to shut down the station in use and boot it up again.

A confirmation pop-up opens on the station in use.

	**Hint:**
	The device types are: **Console**, **NetworkNode**, **onPC**, **ProcessingUnit**, **Session**, **Station**, and **Extension**.

Syntax

Reboot (/Option)

Reboot IP [IP_Address] (/Option)

Reboot [Device_Type] ["Device_Name" or Device_Number] (/Option)

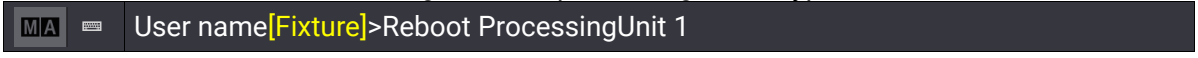
Option Keywords

The Reboot keyword uses the following option keywords:

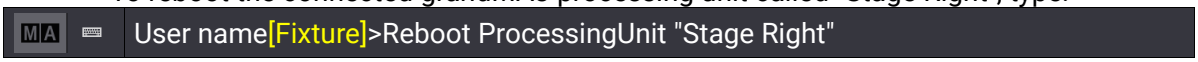
- **/NoConfirmation**
- **/NoSave**
- **/Save**
- **/Wait**

Examples

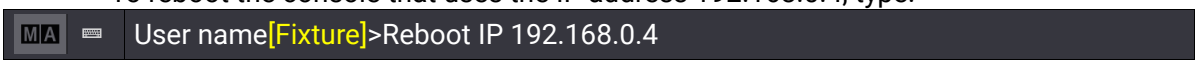
- To reboot the connected grandMA3 processing unit 1, type:



- To reboot the connected grandMA3 processing unit called "Stage Right", type:



- To reboot the console that uses the IP address 192.168.0.4, type:



1.9.3.256. Recast

To enter the Recast keyword in the command line, use one of the options:

- Press MA + X1 | Clone + X1 | Clone
- Type **Recast**
- Type the shortcut **Reca**

Description

The Recast keyword is a command keyword which is used to update attributes that were added or removed in the presets. It will add or remove these values in cues where the preset is used.

Furthermore, recast can be used when configuring executors. When an executor configuration is used on several executors, and the assignment of handle for one of these executors changes, the changes will not automatically be transmitted to other executors using this configuration. When storing the changes into the executor configuration, it is possible to recast the executor configuration. All other executors using this configuration will then get the new handle assignmen. For more information on executor configurations see the **Executor Configurations**.

	Known Limitation:
	Recast will only recast presets to cues where a preset link exists in the absolute layer.

Syntax

Recast Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number]

Recast Configuration ["ExecutorConfiguration_Name" or ExecutorConfiguration_Number]

Example

- The dimmer is open and the color is red in ten fixtures in the All preset 21.1. This preset is used in sequence 1. We now add position to the preset. To recast preset 21.1, type:

MA	User name[Fixture]>Recast Preset 21.1

1.9.3.257. Reconnect

To enter the Reconnect keyword in the command line, use one of the options:

- Type **Reconnect**
- Type the shortcut **Recon**

Description

The Reconnect keyword takes the station that was previously disconnected back to session.

	**Hint:**
	Reconnect only works from station to station where session credentials are the same (same name of session and location).

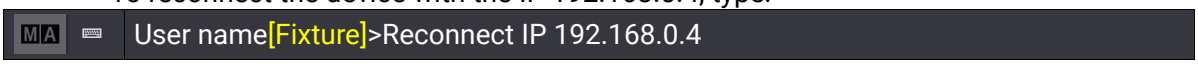
Syntax

Reconnect IP [IP_Number]

Reconnect ["StationType_Name" or StationType_Number].["Station_Name" or Station_Number]

Example

- To reconnect the device with the IP 192.168.0.4, type:



1.9.3.258. Record

To enter the Record keyword in the command line, use one of the options:

- Press MA + Store
- Type **Record**
- Type the shortcut **Rec**

Description

The Record keyword is used to record a timecode show using the command line.

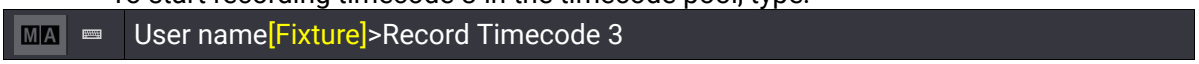
For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Record (On/Off) Timecode ["Timecode_Name" or Timecode_Number]

Example

- To start recording timecode 3 in the timecode pool, type:



1.9.3.259. Relative

To enter the Relative keyword in the command line, use one of the options:

- Press MA + Time
- Type **Relative**

Description

The Relative keyword is an object keyword which is used to set the relative layer.

Syntax

Relative

Attribute ["Attribute_Name"] At Relative [Value]

Examples

- To set the layer to relative, type:

- To set the color red to 10 in the programmer, type:

1.9.3.260. Release

To enter the **Release** keyword in the command line, use one of the options:

- Press Delete Delete Delete
- Type **Release**
- Type the shortcut **Rel**

Description

The Release keyword enters release values in the programmer depending on attributes specified and the fixtures that were selected.

Release values that are stored using the merge function release previously tracked values in the tracking list. The fixtures use their default values.

If you release an object, the release is applied to the corresponding layer.

Syntax

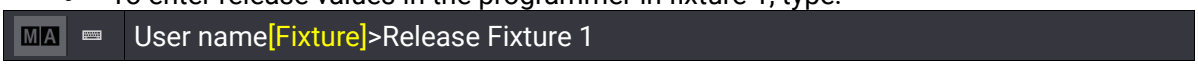
Release [Object] ["Object_Name" or Object_Number]

Examples

- To enter release values in the programmer in all attributes of the fixture selection in the corresponding layer, type:

- To enter release values in the programmer in the attribute "Pan" of the fixture selection in the corresponding layer, type:

- To enter release values in the programmer in fixture 1, type:



1.9.3.261. ReloadAllPlugins

To enter the ReloadAllPlugins keyword in the command line, use one of the options:

- Type **ReloadAllPlugins**
- Type the shortcut **ReloadP** or **RP**

Description

The ReloadAllPlugins keyword is a function keyword that is used to reload the content of the external Lua files.

It is necesssary to reload the external Lua files after you edited them, as the edits can influence how Lua behaves.

You may want to test the integrity of the Lua system to make sure that it behaves as expected next time you load the show. This is important as the show file saved does not contain a snapshot of the Lua memory. It only contains the integrated functions and the code in the defined plugins.

When the show file is loaded, the external Lua files and the code of the plugins are reloaded. That is, the code of the Lua file on the harddrive is reread and loaded into the show file again. This then may result in a different state than that after you powered down the console or saved the show file.

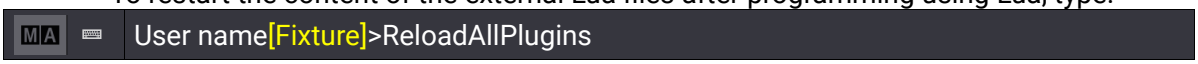
	**Hint:**
	Double-check the executed command in the system monitor.

Syntax

ReloadAllPlugins

Example

- To restart the content of the external Lua files after programming using Lua, type:



1.9.3.262. ReloadUI

To enter the ReloadUI keyword in the command line, use one of the following options:

- Type **ReloadUI**
- Type **Relo**
- Type the shortcut **RU**

Description

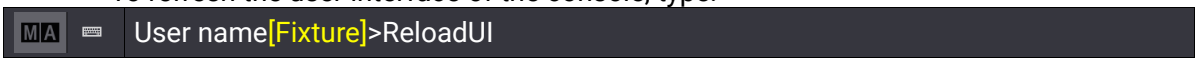
The ReloadUI keyword is a function keyword that is used to refresh the user interface of the console.

Syntax

ReloadUI

Example

- To refresh the user interface of the console, type:



1.9.3.263. Remote

To enter the Remote keyword in the command line, use one of the options:

- Type **Remote**
- Type the shortcut **Rem**

Description

The Remote keyword is an object keyword that is used to access the remote input types.

You can store or delete remote input types and set parameters.

Syntax

[Function] Remote ["RemoteInputType_Name" or RemoteInputType_Number].["Remote_Name" or Remote_Number] (Property ["Property_Name"] ["Property_Value"])

Assign [Object] ["Object_Name" or Object_Number] At Remote ["RemoteInputType_Name" or RemoteInputType_Number].["Remote_Name" or Remote_Number] (Property ["Property_Name"] ["Property_Value"])

The following table displays the available remote input types and their remote input type IDs.

Remote Input Type	Remote Input Type ID
DC Remote	1
MIDI Remote	2
DMX Remote	3

The IDs in the input type have to be in an order and have to start with 1.

Properties

The following table displays the properties you can set using the command line with the help of the **Set Keyword**.

Hint:

If an option or any other part of the keyword command requires two quotation marks, the outer quotation marks are "+" and the inner quotation marks are '+'.

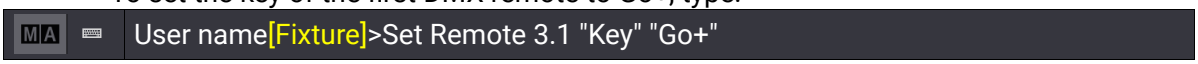
Property	Property Value	Description
Lock	"Yes", "No"	Sets the lock status.
Name	"This is the name of the remote"	Sets the name of the remote.
Target	"World", "Sequence", "Macro", "Group", "Plugin", "View", "Master"	Sets the target of the action when the contact is active.
Fader	"Master", "X", "Temp", and all	Sets the fader the console should

Property	Property Value	Description
	the fader functions.	activate.
Key	"Fix", "Select", "SelFix", and all the key functions.	Sets the key the console should activate.
TriggerOn	"0%...100%"	Sets the value at which the trigger will be set to on.
TriggerOff	"0%...100%"	Sets the value at wich the trigger will be set to off.
InFrom	"0%...100%"	Sets the starting point of the range of the incoming signal in use.
InTo	"0%...100%"	Sets the end point of the range of the incoming signal in use.
OutFrom	"0%...100%"	Sets the starting point of the range of the outgoing signal in use.
OutTo	"0%...100%"	Sets the end point of the range of the outgoing signal in use.
Enabled	"Yes", "No"	Sets the status to enabled or not enabled.
Only for DMX remotes: Address	1.001...1024.512 [universe].[dmx address]	Sets the DMX universe and address.
Only for DMX remotes: Resolution	"8bit", "16bit", "24bit"	Sets the DMX resolution. For 16 bit and 24 bit, the DMX channels have to be consecutive.
Only for MIDI remotes: MIDIChannel	"1, 2, 3, ..., 16"	Sets the MIDI channel.
Only for MIDI remotes: MIDIIndex	"1, 2, 3, ..., 128"	Sets the MIDI index.
Only for MIDI remotes: MIDIType	"Note", "NoteAttack", "NoteAttackDecay", "Control"	Sets the MIDI type. Note = MIDI note only NoteAttack = MIDI note and uses the velocity to regulate the master except note off NoteAttackDecay = MIDI note and uses the velocity to regulate the master with note off Control = Control change (CC) messages.
Only for DC remotes: DC start signal	"1, 2, 3, ..., 64"	Sets the DC start signal.

The start signal and the MIDI offset of the desired input console for **DC Remotes** and **MIDI Remotes** have to be set in the **Output Configuration Menu**.

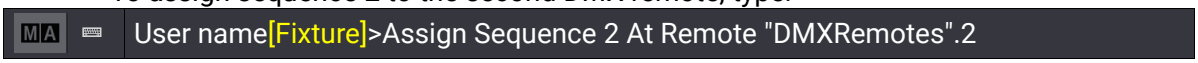
Examples

- To set the key of the first DMX remote to Go+, type:



- To store a new MIDI remote, type:

- To assign sequence 2 to the second DMX remote, type:



1.9.3.264. RemoteHID

To enter the RemoteHID keyword, use one of the options:

- Type **RemoteHID**

Description

The RemoteHID keyword connects a mouse and/or an external keyboard with different stations.

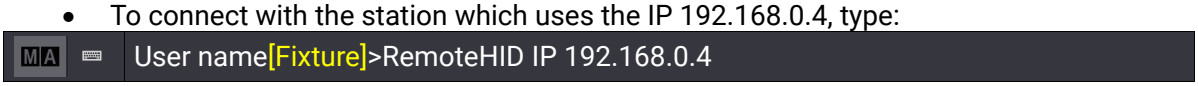
Syntax

RemoteHID IP [IP Address]

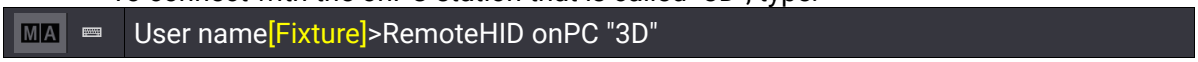
RemoteHID onPC ["onPC_Name" or onPC_Number]

Examples

- To connect with the station which uses the IP 192.168.0.4, type:



- To connect with the onPC station that is called "3D", type:



1.9.3.265. RemoteCommand

To enter the RemoteCommand keyword, use one of the options:

- Type **RemoteCommand**
- Type **RemoteC**

Description

The RemoteCommand keyword remotely sends commands to other stations.

	**Hint:**
	The device types are: **Console**, **NetworkNode**, **onPC**, **ProcessingUnit**, **Session**, **Station**, and **Extension**.

Syntax

RemoteCommand IP [IP] ["Command to be Executed"]

RemoteCommand [Device_Type] ["Device_Name" or Device_Number] ["Command to be Executed"]

Example

- To remotely execute the command "Call ViewButton 2.1" on the station with the IP address 192.168.0.10, type:

```
User name[Fixture]>RemoteCommand IP 192.168.0.10 "Call ViewButton 2.1"
```

- To remotely lock the desk on the station with the IP address 192.168.0.10, type:

```
User name[Fixture]>RemoteCommand IP 192.168.0.10 'Menu "DeskLock" '
```

Alternatively type:

```
User name[Fixture]>RemoteCommand IP 192.168.0.10 "Menu 'DeskLock' "
```

For more information on the usage of quotation marks see **General Syntax Rules**.

For more mutual examples see the **Station Keyword**.

1.9.3.266. Remove

To enter the **Remove** keyword in the command line, use one of the options:

- Press Delete Delete
- Type **Remove**
- Type the shortcut **Remov**

Description

The Remove keyword enters remove values in the programmer depending on the attributes specified and the fixtures that were selected.

Remove values that are stored using the merge function remove previously stored values.

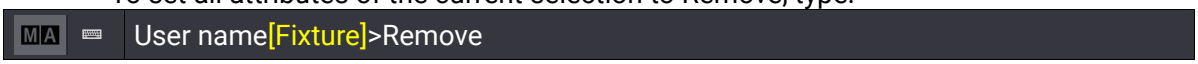
If a stored value is removed, values from the previous cue will be tracked again.

Syntax

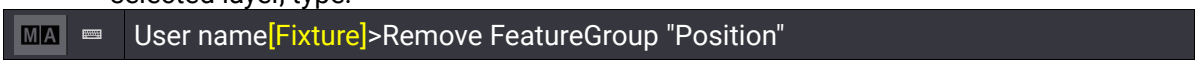
Remove Object ["Object_Name" or Object_Number]

Examples

- To set all attributes of the current selection to Remove, type:



- To set the Remove value in all attributes of the feature group Position, however in only the selected layer, type:



1.9.3.267. RenderQuality

To enter the RenderQuality keyword in the command line, use one of the options:

- Type **RenderQuality**
- Or type the shortcut **Ren**

Description

The RenderQuality keyword is an object keyword that is used to address the objects within the render qualities pool.

	**Hint:**
	The first eight render quality objects are locked by default.

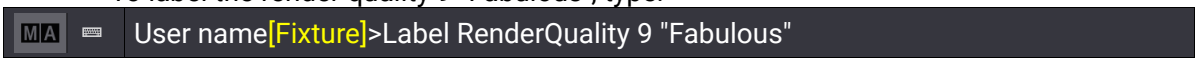
Syntax

[Function] RenderQaulity ["RenderQuality_Name" or RenderQuality_Number]

Examples

- To list all render qualities, type:

- To label the render quality 9 "Fabulous", type:



1.9.3.268. Reset

To enter the Reset keyword in the command line, use one of the options:

- Type **Reset**
- Type the shortcut **Rese**

Description

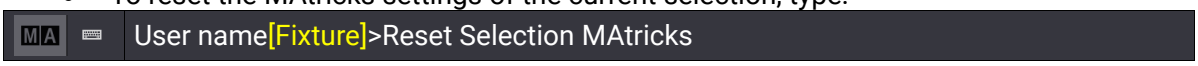
The Reset keyword is a function keyword used to clear the settings of objects, for example the MAtricks.

Syntax

Reset [Object]

Example

- To reset the MAtricks settings of the current selection, type:



1.9.3.269. Restart

To enter the Restart keyword in the command line, use one of the options:

- Type **Restart**
- Type the shortcut **Res**

Description

The Restart keyword is a function keyword that is used to restart the application. Restart behaves the same as closing the program and reopening it without shutting down the console.

	**Hint:**
	The device types are: **Console**, **NetworkNode**, **onPC**, **ProcessingUnit**, **Session**, **Station**, and **Extension**.

Syntax

Restart (/Option)

Restart [Device_Type] ["Device_Name" or Device_Number] (/Option)

Restart IP [IP_Address] (/Option)

Option Keywords

The Restart keyword uses the following option keywords:

- **/NoConfirmation**
- **/NoSave**
- **/Save**
- **/Wait**

Examples

- To restart the application of the console, type:

- To restart the application of the station using the IP address 192.168.0.32, type:

1.9.3.270. Root

	Important:
	We recommend using the index name of the root. If you use root index numbers, double-check after every release if the root index number is still valid. If the root index number has changed, adjust your macros.

To enter the Root keyword in the command line, use one of the options:

- Type **Root**
- Type the shortcut **Ro**

Description

The Root keyword is an object keyword which is used to access the root in the object tree.

Syntax

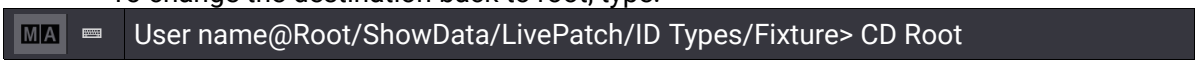
Root [root-index]

Root [root-index].[sub-index]

Root [root-index].[sub-index].[sub-sub-index]

Example

- To change the destination back to root, type:



1.9.3.271. RealtimeChannel

To enter the RealtimeChannel keyword in the command line, use one of the options:

- Type **RealtimeChannel**
- Type the shortcut **Real**

Description

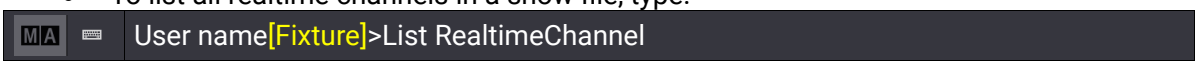
The RealtimeChannel keyword is used to address realtime channels.

Syntax

[Function] RealtimeChannel

Example

- To list all realtime channels in a show file, type:



Realtime channels are also displayed in the parameter list of the patch.

For more information see **Patch and Fixture Setup**.

1.9.3.272. RunningMacro

To enter the RunningMacro keyword in the command line, use one of the options:

- Type **RunningMacro**
- Type the shortcut **RunningM**

Description

The RunningMacro keyword is useful when there are several users in a session.

It allows you to list or disable all running macros in a show file.

	Important:
	The numbers of the running macros are not equal to the numbers of macros in the macros data pool.

Syntax

[Function] RunningMacro [Number]

Example

- To disable the first macro that is currently running, type:

```
User name[Fixture]>Off RunningMacro 1
```

- To list all running macros, change destination to the running macros first:

```
User name[Fixture]>ChangeDestination RunningMacro
```

Result:

```
User name@Temp/RunningPlaybacksCollect/RunningMacros>
```

- To list all running macros, type:

```
User name@Temp/RunningPlaybacksCollect/RunningMacros>List
```

1.9.3.273. RunningPreset

To enter the RunningPreset keyword in the command line, use one of the options:

- Type **RunningPreset**
- Type the shortcut **RunningP**

Description

The RunningPreset keyword is useful when there are several users in a session.

It allows you to list or disable all running presets in a show file.

	Important:
	The numbers of the running presets are not equal to the numbers in the preset pool.

Syntax

[Function] RunningPreset ["Preset_Name" or Preset_Number]

Example

- To disable the first color preset that is currently running, type:

```
User name[Fixture]>Off RunningPreset 4.1
```

- To list all running presets, change destination to the running presets first:

```
User name[Fixture]>ChangeDestination RunningPreset
```

Result:

```
User name@Temp/RunningPlaybacksCollect/RunningPresets>
```

- To list all running presets, type:

```
User name@Temp/RunningPlaybacksCollect/RunningPresets>List
```

1.9.3.274. RunningSoundFile

To enter the RunningSoundFile keyword in the command line, use one of the options:

- Type **RunningSoundFile**
- Type the shortcut **RunningSo**

Description

The RunningSoundFile keyword is used when there are several users in a session.

It allows you to disable all running sound files in a show file.

	**Important:**
	The numbers of the running sound files are not equal to the numbers of sound files in the sound pool.

Syntax

[Function] RunningSoundFile [Number]

Examples

- To disable the first sound file that is currently running, type:

```
User name[Fixture]>Off RunningSoundFile 1
```

- To list all running sound files, change destination to the running sound files first:

```
User name[Fixture]>ChangeDestination RunningSoundFile
```

Result:

```
User name@Temp/RunningPlaybacksCollect/RunningSoundFiles>
```

- To list all running sound files, type:

```
User name@Temp/RunningPlaybacksCollect/RunningSoundFiles> List
```

1.9.3.275. RunningSequence

To enter the RunningSequence keyword in the command line, use one of the options:

- Type **RunningSequence**
- Type the shortcut **Run**

Description

The RunningSequence keyword is useful when there are several users in a session.

It allows you to list or disable all running sequences in a show file.

	**Important:**
	The numbers of the running sequences are not equal to the numbers of sequences in the sequences data pool.

Syntax

[Function] RunningSequence [Number]

Example

- To disable the first sequence that is currently running, type:

```
User name[Fixture]>Off RunningSequence 1
```

- To list all running sequences, change destination to the running sequences first:

```
User name[Fixture]>ChangeDestination RunningSequence
```

Result:

```
User name@Temp/RunningPlaybacksCollect/RunningSequences>
```

- To list all running sequences, type:

```
User name@Temp/RunningPlaybacksCollect/RunningSequences> List
```

1.9.3.276. RunningTimecode

To enter the RunningTimecode keyword in the command line, use one of the options:

- Type **RunningTimecode**
- Type the shortcut **RunningTimec**

Description

The RunningTimecode keyword addresses the running timecodes in the show file.

	**Important:**
	The numbers of the running timcodes are not equal to the numbers of timecodes in the timecodes data pool.

Syntax

[Function] RunningTimcode ["RunningTimecode_Name" or RunningTimecode_Number]

Examples

- To disable the first timecode that is currently running, type:

```
User name[Fixture]>Off RunningTimecode 1
```

- To list all running timecodes, change destination to the running timecodes first:

```
User name[Fixture]>ChangeDestination RunningTimecodes
```

Result:

```
User name@Temp/RunningPlaybacksCollect/RunningTimecodes>
```

- To list all running timecodes, type:

```
User name@Temp/RunningPlaybacksCollect/RunningTimecodes> List
```

1.9.3.277. RunningTimer

To enter the RunningTimer keyword in the command line, use one of the options:

- Type **RunningTimer**
- Type the shortcut **RunningT**

Description

The RunningTimer keyword grants access to all running timers.

It allows you to list or disable all running timers in a show file.

	Important:
	The numbers of the running timers are not equal to the numbers of timers in the timers data pool.

Syntax

[Function] RunningTimer [Number]

Examples

- To disable the first timer that is currently running, type:

```
User name[Fixture]>Off RunningTimer 1
```

- To list all running timers, change destination to the running timers first:

```
User name[Fixture]>ChangeDestination RunningTimer
```

Result:

```
User name@Temp/RunningPlaybacksCollect/RunningTimers>
```

- To list all running timers, type:

```
User name@Temp/RunningPlaybacksCollect/RunningTimers> List
```

1.9.3.278. SaveShow

To enter the SaveShow keyword in the command line, use one of the options:

- Type **SaveShow**
- Type the shortcut **Sa**

Description

The SaveShow keyword is a function keyword which is used to save the current show file.

If you do not enter a new show name, the show will be saved using the name of the current show.

	**Important:**
	If there is already a show file with the same show name, the console overwrites the existing show file.

For more information on the folder structure of shows, demo shows, and backups, see **Show File Handling**.

Syntax

SaveShow ("Show_Name") (/Option)

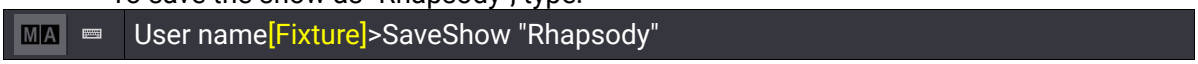
Option Keywords

The SaveShow keyword uses the following option keywords:

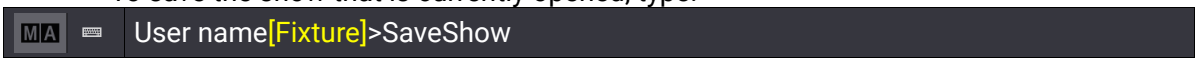
- **/Enumerate**

Examples

- To save the show as "Rhapsody", type:



- To save the show that is currently opened, type:



1.9.3.279. ScreenConfiguration

	Important:
	It is possible to create screen configurations for single user profiles. However, every user of the same user profile can use a different screen configuration. By default, every user profile has 2 screen configurations: Default and 3D

To enter the ScreenConfiguration keyword in the command line, use one of the options:

- Type **ScreenConfiguration**
- Type the shortcut **ScreenConf**

Description

The ScreeConfiguration keyword is an object keyword that addresses the different screen configurations of a user profile. For example, one setup for the operation of a console, and a second setup for the operation of a dedicated onPC station.

Syntax

[Function] ScreenConfiguration ["ScreenConfiguration_Name" or ScreenConfiguration_Number]

Examples

- To store a new screen configuration called "Average Joe", type:

```
User name[Fixture]>Store ScreenConfiguration 3 "Average Joe"
```

- To enter screen configuration 2, type:

```
User name[Fixture]>ScreenConfiguration 2
```

- To assign the screen configuration "Average Joe" to ViewButton 1.6, type:

```
User name[Fixture]>Assign ScreenConfiguration "Average Joe" At ViewButton 1.6.
```

For more information on ViewButtons see **Store and Recall Views**.

1.9.3.280. ScreenContent

To enter the ScreenContent keyword in the command line, use one of the options:

- Type **ScreenContent**
- Type the shortcut **Scre**

Description

The ScreeContent keyword is used to represent the windows of a display.

Syntax

[Function] ScreenContent

[Function] ScreenContent [ID].[ID] ["Property"] ["Property_Value"]

Examples

- To delete all windows on all screens, type:

```
User name[Fixture]>Delete ScreenContent *.*
```

- To delete all windows on screen 1, type:

```
User name[Fixture]>Delete ScreenContent 1.*
```

- To set the width of the first window you created to 12 half columns in screen 1, type:

```
User name[Fixture]>Set ScreenContent 1.1 "W" "12"
```

1.9.3.281. Scribble

To enter the Scribble keyword into the command line, use one of the options:

- Press MA + X4 + X4 + X4
- Type **Scribble**
- Type the shortcut **Scr**

Description

Scribbles are little drawings that are used as visualizations instead of labels.

The scribble keyword is an object keyword that is used to access scribbles with a scribble ID.

It is also possible to label scribbles and address them by their label.

For more information about scribbles see **Scribbles**.

Syntax

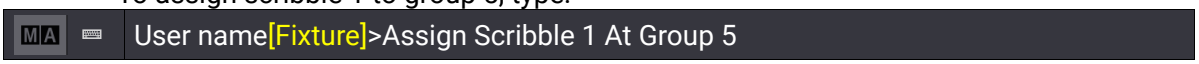
[Function] Scribble ["Scribble_Name" or Scribble_Number] (At ["Object_Name" or Object_Number])

Examples

- To edit scribble 1, type:

The edit scribble pop-up opens.

- To assign scribble 1 to group 5, type:



- To delete scribble 1, type:

1.9.3.282. Seconds

To enter the Seconds keyword in the command line, use one of the options:

- Type **Seconds**
- Type the shortcut **Sec**

Description

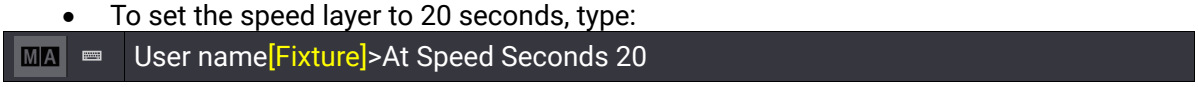
The Seconds keyword is used to set the speed of a fixture selection using the unit seconds.

Syntax

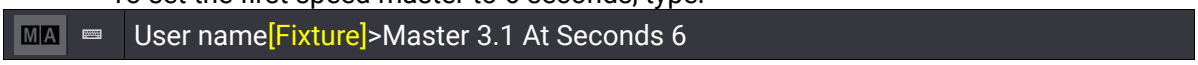
([Object] ["Object_Name" or Object_Number]) At Speed Seconds [Value]

Examples

- To set the speed layer to 20 seconds, type:



- To set the first speed master to 6 seconds, type:



1.9.3.283. Select

To enter the Select keyword in the command line, use one of the options:

- Press Select
- Type **Select**
- Type the shortcut **Sel**

Description

The Select keyword selects objects as default objects.

Every selected object is indicated by a yellow frame.

- The **selected sequence** is the target for all sequence-related commands, e.g., **Store Cue 4**.
- The **selected sequence** is displayed in the Sequence Sheet, if the **Link Type** is set to **Selected**.
- The **selected camera** is displayed in the 3D View, if the **Camera Selection** is set to **Link Selected**.

For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Select [Object] ["Object_Name" or Object_Number]

Example

- To select the sequence assigned on executor 105, type:

- To select page 2, type:

- To select world 1, type:

- To select data pool 3, type:

1.9.3.284. Selection

To enter the Selection keyword in the command line, use one of the options:

- Press Fixture Fixture
- Type **Selection**
- Type the shortcut **Selecti**

Description

The Selection keyword is an object keyword that is used to represent the current selection of fixtures in the programmer.

Syntax

[Function] Selection

Example

- To disable all programmer values of the current fixture selection, type:

1.9.3.285. SelectFixtures

To enter the **SelectFixtures** keyword in the command line, use one of the options:

- Press SelFix
- Type **SelectFixtures**
- Type the shortcut **SelF**

Description

The SelectFixtures keyword is a function keyword that is used to create selections of fixtures in the programmer.

If only fixtures are selected, the SelectFixtures keyword adds additional fixtures to the selection.

If fixtures are selected and activated in the programmer, the SelectFixtures keyword replaces the selection by the SelectFixtures selection.

If the exact same SelectFixtures command is successively used:

- using it the second time activates all attributes of the selected fixtures in the programmer
- using it the third time deactivates all attributes of the selected fixtures in the programmer

SelectFixtures is the default function of most objects, for example, fixture or group or preset.

To clear the selection, press Clear.

Syntax

SelectFixtures [Object] ["Object_Name" or Object_Number]

Examples

- To select all fixtures or channels stored in a sequence of the executor 101, type:

```
User name[Fixture]>SelectFixtures Executor 101
```

- To select all fixtures stored in dimmer preset 1.1, type:

```
User name[Fixture]>SelectFixtures Preset 1.1
```

1.9.3.286. Sequence

To enter the Sequence keyword in the command line, use one of the options:

- Press Sequ
- Type **Sequence**
- Type **Se**

Description

The sequence keyword is an object keyword that is used to select sequences in the sequence pool.

Syntax

[Function] Sequence ["Sequence_Name" or Sequence_Number]

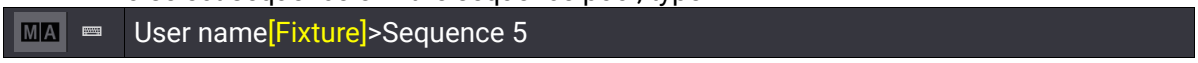
Option Keywords

The Sequence keyword uses the following option keywords:

- **/CreateSecondCue**
- **/DMX**
- **/Merge**
- **/Output**
- **/Overwrite**
- **/Programmer**
- **/Release**
- **/Remove**
- **/Selection**

Examples

- To select sequence 5 in the sequence pool, type:



- To block all cues in sequence 5, type:

1.9.3.287. SendMIDI

	Important:
	If you do not define a channel value, the MIDI messages will be automatically addressed to channel 1.

To enter the SendMIDI keyword in the command line, use one of the options:

- Type **SendMIDI**
- Type **Sendm**

Description

The SendMIDI keyword is a command keyword that is used to output MIDI notes, MIDI control change messages, and MIDI program change messages.

Syntax

SendMIDI "Note" (Channel/)Note (Velocity) (Status)

SendMIDI "Program" (Channel/)Value

SendMIDI "Control" (Channel/)Controller (Value)

Examples

- To send out the MIDI note 42 on MIDI channel 2 with a velocity of 99, type:

```
User name[Fixture]>SendMIDI "Note" 2/42 99
```

- To send out the MIDI note 37 on MIDI channel 1, type:

```
User name[Fixture]>SendMIDI "Note" 37
```

- To send out the MIDI control value 64 to MIDI channel 1 controller 8, type:

```
User name[Fixture]>SendMIDI "Control" 1/8 64
```

- To send out a MIDI program change of 12, type:

```
User name[Fixture]>SendMIDI "Program" 12
```

1.9.3.288. SendOSC

To enter the **SendOSC** keyword in the command line, use one of the options:

- Type **SendOSC**
- Type the shortcut **Sen**

Description

The SendOSC keyword is a command keyword that is used to send an OSC command.

For more information see **Remote In and Out**.

Syntax

SendOSC [ID] "/[OSCAddress],[OSC Type],[Value]"

The supported types are:

- Int(i)
- Float(f)
- Blob(b)
- String(s)
- True(T)
- False(F)
- Null(N)
- Impulse(I)
- Timetag(t)

It is not necessary to set a value (Payload) for:

- True
- False
- Null
- Impulse
- Timetag

	**Hint:**
	When using the OSC types True, False, Nil/Null, Impulse and Timetag it is not necessary to enter a value.
	**Hint:**
	Several values can be sent at once when separated by commas.
	**Important:**
	When addressing an executor, a page must be specified as well.

Instead of using page and executor numbers, it is also possible to address them by name.

When addressing executor keys, a value of 0 will be interpreted as not pressed. Values greater 0 will be interpreted as button press.

If a prefix is specified for an OSCData entry, then this very prefix will be added to the sent string when using the OSCSend command.

	Hint:
	The supported OSC types to control faders, executor knobs, and buttons are: Integer32, Float32, True, False and Nil/Null. A True will be interpreted as 1, while a False will be interpreted as 0.
	Hint:
	The addresses defined for Page, Fader, ExecutorKnob, and Key are case-sensitive.

Examples

- To send an OSC command using the first configuration in the OSC menu with integer value 50 to fader 201 on page 1, type:

```
User name[Fixture]>SendOSC 1 "/Page1/Fader201,i,50"
```

- To send an OSC command using the first configuration in the OSC menu with integer value 100 to fader 201 on page 1 and a fade time of 5s, type:

```
User name[Fixture]>SendOSC 1 "/Page1/Fader201,ii,100,5"
```

- To send commands via OSC to the second grandMA3 station, the OSC address /cmd can be used. To store cue 1 via OSC, type:

```
User name[Fixture]>SendOSC 1 "/cmd,s,Store Cue 1"
```

1.9.3.289. SendMVR

To enter the SendMVR keyword in the command line, use one of the options:

- Type **SendMVR**
- Type the shortcut **SendMV**

Description

The keyword SendMVR can be used to commit and request MVR files, or to join and leave the connection to other devices during MVR-xchange.

Syntax

SendMVR ["Connection_Type"] ["Number]

SendMVR "Commit" ["Path_to_Folder/Name"] ["Name"]

Examples

- To establish a connection to the third service device, type:

```
User name[Fixture]>SendMVR "Join" "3"
```

- To end connection to the the first service device, type:

```
User name[Fixture]>SendMVR "Leave" "1"
```

- To commit the MVR file "BestShow.mvr" (which is located on C:\ProgamData\MA Lighting Technology\gma3_library\mvr) to the MVR-xchange group, type:

```
User name[Fixture]>SendMVR "Commit" "C:\ProgramData\MA Lighting Technology\gma3_library\mvr\BestShow" "BestShow"
```

- To request the second file, type:

```
User name[Fixture]>SendMVR "Request" "2"
```

1.9.3.290. Set

To enter the Set keyword in the command line, use one of the options:

- Press MA + Assign
- Type **Set**

Description

The Set keyword sets values to properties of objects. It is also used to transfer properties of objects to the same value as another object.

It is used in conjunction with the **Property keyword** or the **= [Equal] keyword.**

Syntax

Set [Object_Type] ["Object_Name" or Object_Number] Property ["Property_Name"] ["Property_Value"]

Set [Object_Type] ["Target_Object_Name" or Target_Object_Number] Property ["Property_Name"] At [Object_Type] (["Source_Object_Name" or Source_Object_Number]) (Property ["Property_Name"])

Option Keywords

The Set keyword uses the following option keywords:

- **/PatchOffset**
- **/Look**

Examples

- To set sequence 8 to priority HTP, type:

```
User name[Fixture]>Set Sequence 8 "Priority" 3
```

- To transfer the value of the Priority setting of sequence 1 to sequence 42, type:

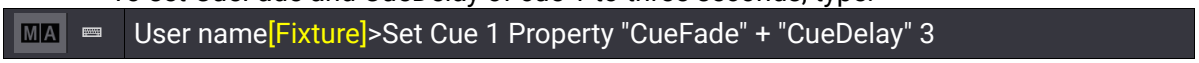
```
User name[Fixture]>Set Sequence 42 Property "Priority" At Sequence 1
```

- To transfer the name of the selected sequence to the name of group 5, type:

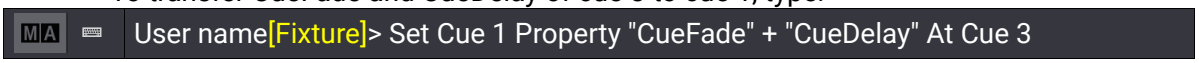
```
User name[Fixture]>Set Group 5 Property "Name" At Sequence Property "Name"
```

- To transfer the value of the CueFade property of cue 9 to the CueDelay property of cue 6, type:

- To set CueFade and CueDelay of cue 1 to three seconds, type:



- To transfer CueFade and CueDelay of cue 3 to cue 1, type:



1.9.3.291. SetGlobalVariable

To enter the SetGlobalVariable keyword in the command line, use one of the options:

- Type **SetGlobalVariable**
- Type the shortcut **Setg**

Description

The SetGlobalVariable keyword is used to set global variables in a show. It also supports the use of values in properties of other objects as a value of the variable.

Syntax

SetGlobalVariable ["Name of Variable"] [Numeric Value]

SetGlobalVariable ["Name of Variable"] ["Text_Value"]

SetGlobalVariable ["Name of Variable"] At [Object] ["Object Name" or Object Number] Property ["Property_Name"] (/Look)

Examples

- To set the global variable "Urban Blues" to the value of 3, type:

```
User name[Fixture]>SetGlobalVariable "Urban Blues" "3"
```

- To set the global variable "Hook" to the CueFade value of cue 2 of the selected sequence, type:

```
User name[Fixture]>SetGlobalVariable "Hook" At Cue 2 Property "CueFade"
```

- To set the global variable "PositionX" to the 3D X coordinate of fixture 1, type:

```
User name[Fixture]>SetGlobalVariable "PositionX" At Fixture 1 Property "PosX"
```

- To set the global variable "myShow" to the name of the show file that was loaded, type:

```
User name[Fixture]>SetGlobalVariable myShow At Root "MANetSocket" Property "ShowFile"
```

1.9.3.292. SetUserVariable

To enter the SetUserVariable keyword in the command line, use one of the options:

- Type **SetUserVariable**
- Type the shortcut **Setu**

Description

The SetUserVariable keyword is used to set user-specific variables. It also supports the use of values in properties of other objects as a value of the variable.

Syntax

SetUserVariable ["Name of Variable"] [Numeric Value]

SetUserVariable ["Name of Variable"] ["Text_Value"]

SetUserVariable ["Name of Variable"] At [Object] ["Object_Name" or Object Number] Property ["Property_Name"] (/Look)

Option Keywords

The SetUserVariable keyword uses the following option keywords:

- **/Look**

Examples

- To set the user variable "Green" to the value 5, type:

```
User name[Fixture]>SetUserVariable "Green" 5
```

- To set the user variable "MyVar" to the CueFade value of cue 1 of the selected sequence, type:

```
User name[Fixture]>SetUserVariable "MyVar" At Cue 1 Property "CueFade"
```

- To set the user variable "mySeqPrioIdx" to the Priority index value of the selected sequence, type:

```
User name[Fixture]>SetUserVariable "mySeqPrioIdx" At Sequence Property "Priority"
```

- To set the user variable myEncoderBar to the name of the selected encoder bar, type:

```
User name[Fixture]>SetUserVariable myEncoderBar At EncoderBar Property "Name"
```

1.9.3.293. Shuffle

To enter the Shuffle keyword in the command line, use one of the options:

- Press SelFix SelFix
- Type **Shuffle**
- Type the shortcut **Shuf**

Description

Shuffle is a command keyword which is used to shuffle the order of the fixture selection. Shuffle is part of the MAtricks toolset.

For more information, see **MAtricks and Shuffle**.

Syntax

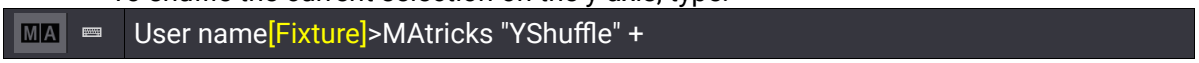
MAtricks [Axis] [Value] text

MAtricks [Axis] +

	**Hint:**
	The plus is a replacement of the value. You can either use plus/minus or a value.

Examples

- To shuffle the current selection on the y-axis, type:



- To shuffle the current selection on the z-axis, type:

- It is also possible to set a certain value to any of the three shuffle settings. To set the shuffle to 4 for the x-axis, type:

	**Hint:**
	When deactivating or resetting the MAtricks, the original selection order will be restored.

1.9.3.294. ShutDown

To enter the ShutDown keyword in the command line, use one of these options:

- Type **ShutDown**
- Type the shortcut **Sh**

Description

The ShutDown keyword powers down the grandMA3 console or closes the grandMA3 onPC.

It requires a confirmation in the local station and can be canceled within 10 seconds using a remote station.

	**Hint:**
	The device types are: **Console**, **NetworkNode**, **onPC**, **ProcessingUnit**, **Session**, **Station**, and **Extension**.

Syntax

ShutDown (/Option)

ShutDown [Device_Type] ["Device_Name" or Device_Number] (/Option)

ShutDown IP [IP_Address] (/Option)

Option Keywords

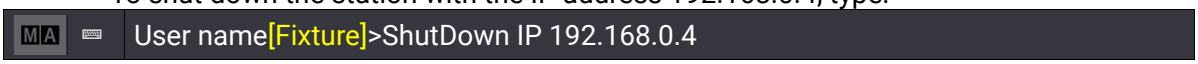
The ShutDown keyword uses the following option keywords:

- **/NoAutoClose**
- **/NoConfirmation**
- **/NoSave**
- **/Save**
- **/Wait**

Examples

- To shut down the current station and provoke a countdown pop-up, type:

- To shut down the station with the IP address 192.168.0.4, type:



1.9.3.295. SnapDelay

To enter the SnapDelay keyword in the command line, use one of the options:

- Type **SnapDelay**
- Type the shortcut **Sn**

Description

The SnapDelay keyword sets a snap time.

The snap time is a delay time for attributes that do not fade, for example a gobo wheel or a color wheel.

Syntax

SnapDelay [Value]

Examples

- To set the snap time of the current cue of the selected sequence to 4 seconds, type:

1.9.3.296. SoftwareImport

To enter the SoftwareImport keyword in the command line, use one of the options:

- Type **SoftwareImport**
- Type the shortcut **SoftwareI**

Description

The SoftwareImport keyword is a function keyword which is used to import installation packages of a grandMA3 installer permanently to the hard drive of your grandMA3 console or the grandMA3 onPC using a given path.

Syntax

SoftwareImport "release_type_vx.x.x.x.xml;Path/to/the/location/of/the/installer"

Example

- To import the installation packages of the grandMA3 stick v1.6.3.7 on the USB drive "Software" that is recognized as drive D on a Windows® computer, and where the files are located within the ma folder, type:

```
User name[Fixture]>SoftwareImport "release_stick_v1.6.3.7.xml;D:/Software/ma"
```

1.9.3.297. Session

To enter the Session keyword in the command line, use one of the options:

- Type **Session**
- Type the shortcut **Ses**

Description

The Session keyword is an object keyword which is used to address all sessions in the network. If the name and number of session is not specified, your own session will be addressed.

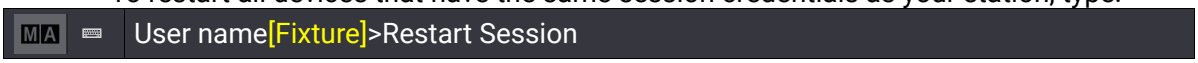
The name of the session consists of the network properties "Session" and "Location" which are connected by @ – "Session@Location".

Syntax

[Function] Session ["Session_Name" or Session_Number]

Examples

- To restart all devices that have the same session credentials as your station, type:



- To shut down all devices that use the session name "Athena" with location "Caledonia", type:

1.9.3.298. SoftwareUpdate

To enter the SoftwareUpdate keyword in the command line, use one of the options:

- Type **SoftwareUpdate**
- Type the shortcut **SoftwareU**

Description

The SoftwareUpdate keyword is a function keyword which is used to update the software of every MA device or program in the network.

For more information on how to update the software and requirements see **Update grandMA3 Consoles**.

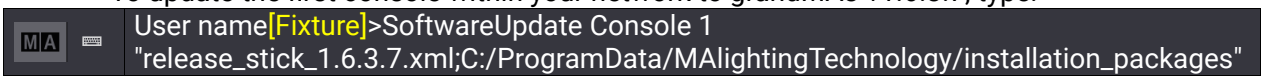
Syntax

SoftwareUpdate [StationType] [ID/"Name"]
"release_type_x.y.z.a.xml;/Path/to/MALightingTechnology/installation_packages"

Example

Requirement:

1. The grandMA3 onPC runs on Windows®
2. Copy the files of the ma folder of the grandMA3_stick_v1.6.3.7.zip file to C:\ProgramData\MALightingTechnology\installation_packages
   - To update the first console within your network to grandMA3 v1.6.3.7, type:



1.9.3.299. SoundChannel

To enter the SoundChannel keyword in the command line, use one of the options:

- Type **SoundChannel**
- Type the shortcut **SoundC**

Description

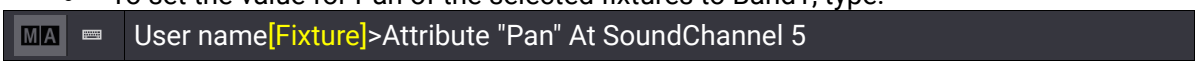
The SoundChannel keyword represents sound codes in the attribute calculators.

Syntax

[Attribute List] [At] SoundChannel [ID/"Name"]

Example

- To set the value for Pan of the selected fixtures to Band1, type:



For more information about using sound input, see **Sound Window**.

1.9.3.300. Solo

To enter the Solo keyword in the command line, use one of the options:

- Press Solo
- Type **Solo**
- Type **So**

Description

The Solo keyword is a function keyword which is used to set the intensity values of fixtures - that are not selected - to zero.

Only the fixtures that are selected generate visible output on stage.

If Solo is used standalone, it toggles between Solo on and Solo off.

To manually fade from the current value to solo and vice versa, use the Grand Solo.

For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Solo (On/Off) [Object] ["Object_Name" or Object_Number]

Example

- To enable Solo, type:

1.9.3.301. SpecialExecutor

To enter the SpecialExecutor keyword in the command line, use one of the options:

- Press MA + X16 | Exec + X16 | Exec
- Type **SpecialExecutor**
- Or type the shortcut **SE**

Description

The SpecialExecutor keyword addresses special executors.

For more information on the configuration see **Special Executors** and **Executor Configurations**.

Syntax

[Function] SpecialExecutor ["SpecialExecutor_Name" or SpecialExecutor_Number]

Examples

- To open the menu **Special Executor Configuration**, type:

```
User name[Fixture]>Assign SpecialExecutor 7
```

- To assign the functionality of grand master to SpecialExecutor 2, type:

```
User name[Fixture]>Assign Master 2.1 At SpecialExecutor 2
```

1.9.3.302. Speed

To enter the Speed keyword in the command line, type **Speed.**

Description

The Speed keyword is used to set the speed of phasers.

For more information see **Phasers**.

Requirement:
To set values to speed, there must be at least 2 steps in the programmer.

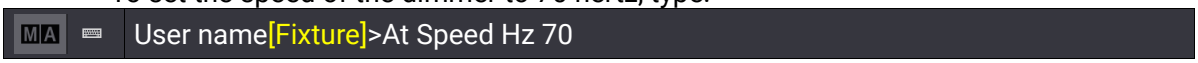
Syntax

[At] Speed [Value]

Examples

- To set the selected layer to speed, type:

- To set the speed of the dimmer to 70 hertz, type:



1.9.3.303. Speed1

To enter the Speed1 keyword in the command line, use one of the options:

- Type **Speed1**
- Type the shortcut **Spee**

Description

The Speed1 keyword is used to reset the speed to 60 BPM.

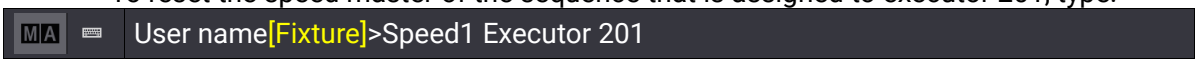
For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Speed1 [Object] ["Object_Name" or Object_Number]

Example

- To reset the speed master of the sequence that is assigned to executor 201, type:



1.9.3.304. SpeedMaster

To enter the SpeedMaster keyword in the command line, use one of the options:

- Type **SpeedMaster**
- Type the shortcut **Speedm**

Description

The SpeedMaster is a layer keyword. It defines which speed master the phaser is to adapt its speed to in each attribute.

	**Hint:**
	You can set speed masters using the speed readouts BPM, Hertz, or Seconds. For more informations see the **Master keyword**.

Syntax

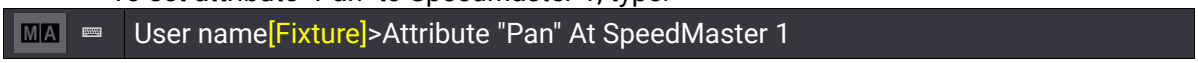
SpeedMaster

(Attribute ["Attribute_Name" or Attribute_Number]) At SpeedMaster [SpeedMaster_Number]

Examples

- To set the layer to speed master, type:

- To set attribute "Pan" to SpeedMaster 1, type:



1.9.3.305. Stage

To enter the Stage keyword in the command line, use one of the options:

- Type **Stage**
- Type the shortcut **Sta**

Description

The Stage keyword addresses stages in the 3D space.

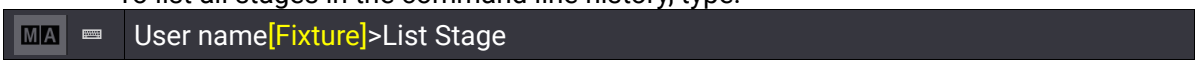
Syntax

[Function] Stage [ID/"Name"] ["Property"] ["Value"]

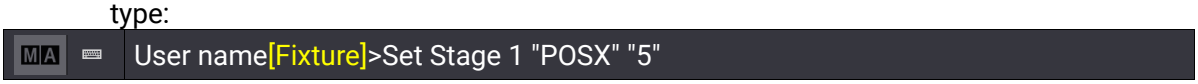
Examples

- To store a new stage, type:

- To list all stages in the command line history, type:



- To set the center of stage 1 to 5 meters in the x direction of the Cartesian coordinate system, type:



1.9.3.306. Station

To enter the Station keyword in the command line, use one of the options:

- Type **Station**
- Type the shortcut **Stat**

Description

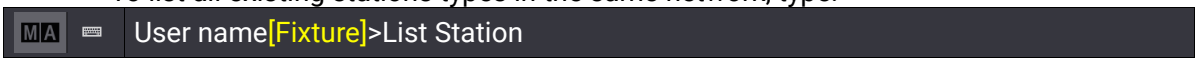
The Station keyword is an object keyword which is used to address all stations in the network. You can also **invite** stations to your session or **dismiss** them.

Syntax

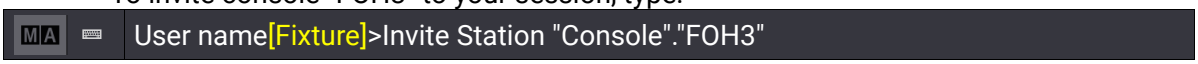
[Function] Station ["DeviceType_Name" or DeviceType_Number].["Device_Name" or Device_Number]

Examples

- To list all existing stations types in the same network, type:



- To invite console "FOH3" to your session, type:



1.9.3.307. StationSettings

To enter the StationSettings keyword in the command line, use one of the options:

- Type **StationSettings**
- Type the shortcut **Stations**

Description

StationSettings keyword is used to address settings of your station.

Syntax

[Function] StationSettings (["StationSettings_Name" or StationSettings_Number] Property ["Property_Name"] [Value])

Examples

- To list all station settings, type:

```
User name[Fixture]>List StationSettings
```

- To set your desk light to 25%, type:

```
User name[Fixture]>Set StationSettings "DesklightsCollect"."DeskLights" Property "Master" 25
```

1.9.3.308. Step

To enter the Step keyword in the command line, use one of the options:

- Press MA + X5 | Step
- Type **Step**

Description

The Step keyword is a command keyword which is used to define the step of the At command.

Syntax

[Function] Step [Step_Number]

Option Keywords

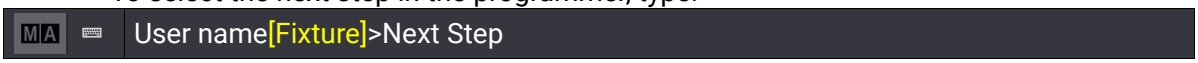
The Step keyword uses the following option keywords:

- **/Wrap**

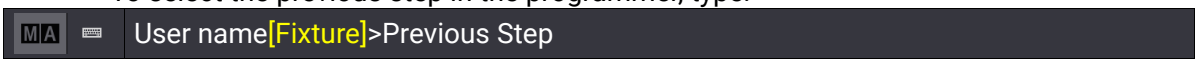
Example

Requirement: Step 1 has to contain information

- To select the next step in the programmer, type:



- To select the previous step in the programmer, type:



1.9.3.309. Stomp

To enter the Stomp keyword in the command line, use one of the options:

- Press Stomp
- Type **Stomp**
- Type the shortcut **Stom**

Description

The Stomp keyword is a command keyword which is used to stop a running phaser.

Stomp knocks in the single-step value which was lastly used in the absolute layer of the attribute. In the relative layer, stomp knocks in a value of 0.
This can be a value which is being used in the programmer or a playback. If there is no single-step value, Stomp will use the default value.

Syntax

Stomp ([Object] ["Object_Name" or Object_Number])

Examples

- To stomp the phasers in the selected fixtures, type:

```
User name[Fixture]>Stomp
```

- To stomp the phasers of the feature group "Position" in the selected fixtures, type:

```
User name[Fixture]>Stomp FeatureGroup "Position"
```

- To stomp the running phasers in fixtures 1 to 10, type:

```
User name[Fixture]>Stomp Fixture 1 Thru 10
```

- To stomp the phasers in all fixtures that are stored in the group "All Spots", type:

```
User name[Fixture]>Stomp Group "All Spots"
```

1.9.3.310. Store

To enter the **Store** keyword in the command line, use one of the options:

- Press Store
- Type **Store**
- Type the shortcut **S**

Description

The Store keyword is a function keyword which is used to store objects in the show file.

If no object type or destination is given, the object type **Cue** will be used in the selected sequence.

If you do not specify a target during storing, the new object will occupy the first free spot in the pool.

Syntax

Store [Object] ["Object_Name" or Object_Number] (/Option)

Option Keywords

The Store keyword uses the following option keywords:

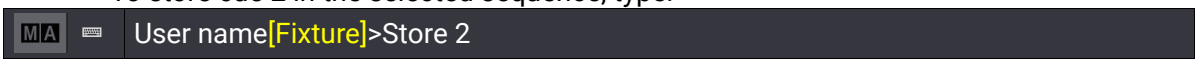
- **/Active**
- **/ActiveForSelected**
- **/All**
- **/AllForSelected**
- **/Ask**
- **/Auto**
- **/CreateSecondCue**
- **/CueOnly**
- **/Embed**
- **/ForceGlobal**
- **/Global**
- **/InputFilter**
- **/KeepActivation**
- **/Look**
- **/MAtricks**
- **/Merge**
- **/NoConfirmation**
- **/Overwrite**
- **/PhaserData**
- **/Remove**
- **/Screen**
- **/ScreenOnly**
- **/Selective**
- **/Universal**

- **/Wait**

For more information see **Store Options and Store Preferences**.

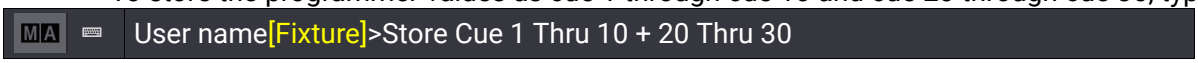
Examples

- To store cue 2 in the selected sequence, type:

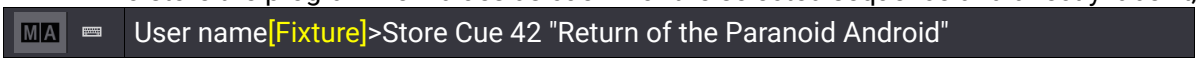


For more information see **Store Cues**.

- To store the programmer values as cue 1 through cue 10 and cue 20 through cue 30, type:



- To store the programmer values as cue 42 of the selected sequence and directly label it, type:



- To store a new group to the first free spot in the groups pool, type:

1.9.3.311. SwitchTograndMA2Software

To enter the SwitchTograndMA2Software keyword in the command line, use one of the options:

- Type **SwitchTograndMA2Software**
- Type the shortcut **SwgMA2**

Description

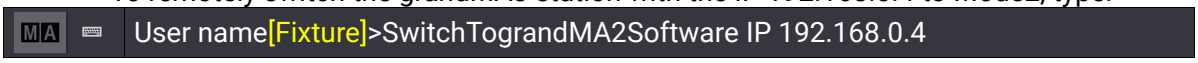
The SwitchTograndMA2Software keyword is a command keyword that is used to remotely switch to Mode2 on a grandMA3 station.

Syntax

SwitchTograndMA2Software IP [IP_Address]

Example

- To remotely switch the grandMA3 station with the IP 192.168.0.4 to Mode2, type:



1.9.3.312. SwitchTograndMA3Software

To enter the SwitchTograndMA3Software keyword in the command line, use one of the options:

- Type **SwitchTograndMA3Software**
- Type the shortcut **SwgMA3**

Description

The SwitchTograndMA3Software keyword is a command keyword that is used to remotely switch to grandMA3 software on a grandMA3 station that is currently running in Mode2.

Syntax

SwitchTograndMA3Software IP [IP_Address]

Example

- To remotely switch a grandMA3 station with the IP 192.168.0.4 from Mode2 to grandMA3 software, type:

1.9.3.313. Swap

To enter the Swap keyword in the command line, use one of the options:

- Type **Swap**
- Type the shortcut **Swa**

Description

The Swap keyword is a playback keyword that starts the playback of a sequence and pulls down the dimmer of all other fixtures that are not part of the sequence or that are protected against Swap.

For more information on how to assign executors see **Assign Objects to an Executor**.

- To disable the executor, release the executor key.

Pressing the executor button plays back the sequence using the swap functionality.
If you swap sequences that have the same playback master, one will go to zero should it not be swap-protected.

For more information on Swap Protect see **Sequence Settings**.

Syntax

Swap (On/Off) [Object] ["Object_Name" or Object_Number]

Example

- To playback sequence 5 and pull down the dimmers of all other running playbacks to 0, type:

1.9.3.314. Temp

To enter the Temp keyword in the command line, use one of the options:

- Press MA + Go+
- Type **Temp**
- Type the shortcut **Te**

Description

The Temp keyword is a function keyword that enables an executor as long as you hold the executor key.

The Temp keyword follows cue timing, off timing, and the position of the master fader on the executor.

- To disable the executor, release the executor key

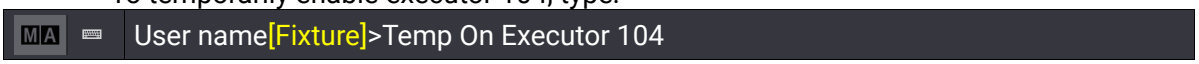
For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

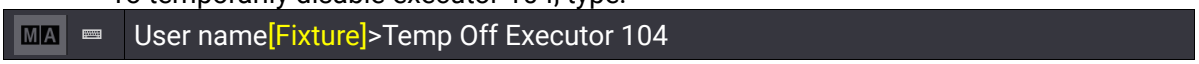
Temp (On/Off) [Object] ["Object_Name" or Object_Number]

Examples

- To temporarily enable executor 104, type:



- To temporarily disable executor 104, type:



1.9.3.315. Thru

To enter the Thru keyword in the command line, use one of the options:

- Press Thru
- Type **Thru**
- Type the shortcut **T**

Description

The Thru keyword indicates the range of objects or values.

If the beginning or the end of the range is not defined, the objects or values that were used last are used again.

Syntax

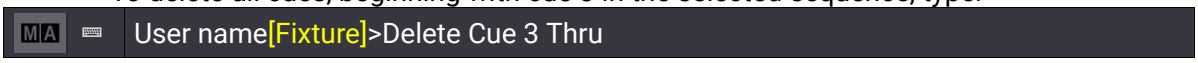
[Beginning of Range] Thru [End of Range]

Examples

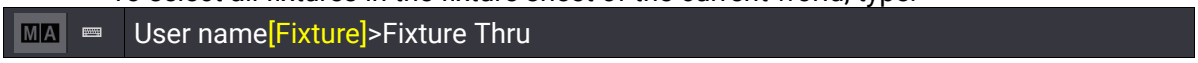
- To select fixture 3 through 6, type:

- To select all fixtures starting with the first fixture and ending with fixture 10, type:

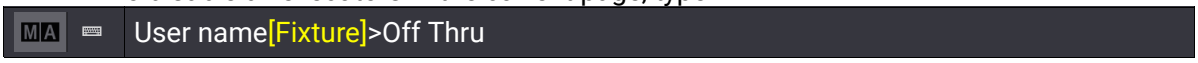
- To delete all cues, beginning with cue 3 in the selected sequence, type:



- To select all fixtures in the fixture sheet of the current world, type:



- To disable all executors in the current page, type:



1.9.3.316. Time

To enter the Time keyword in the command line, type **Time.**

Description

The Time keyword is used to toggle the time overwrite function on sequences.

For more information see **Cue Timing topic**.

For more information on how to set time see **FaderTime keyword**.

For information on how to assign executors see **Assign Objects to an Executor**.

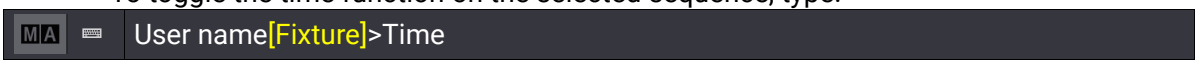
Syntax

Time (On/Off) [Object] ["Object_Name" or Object_Number]

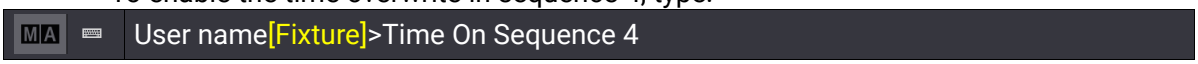
	**Hint:**
	If the sequence ID is not defined, the sequence that is selected will be affected.

Examples

- To toggle the time function on the selected sequence, type:



- To enable the time overwrite in sequence 4, type:



1.9.3.317. Timecode

To enter the Timecode keyword in the command line, use one of the options:

- Press MA + X6
- Type **Timecode**
- Type the shortcut **Ti** or **TC**

Description

The Timecode keyword is an object keyword used to select the timecode by default.

Using the Timecode keyword you can:

- store
- play (go)
- record
- edit
- label
- set properties of
- rewind (top)

timecode shows.

For more information see **Timecode**.

Syntax

[Function] Timecode ["Timecode_Name" or Timecode_Number] ["Property_Name" or "Property_Value"] text

Property

The following table displays the properties that can be assigned using the **Set Keyword**.

Property	Property Value	Description
Name	"Timecode show name"	Set the name of timecode show.
Time		Set position of the time cursor.
Duration	0s to 255h59m58.96s	Set the entire length of timecode show.
Offset	0s to 255h59m58.96s	To move the entire timecode show forward, set an offset in the timecode show.
Loop Mode	Loop Pause Off	Basic settings for Loop and how to pause or stop.
Loop Count	"Endless Repeat" (0), "No Repeat" (1), 2..1000	**Only for timecode shows syncing to the internal clock.** Set if the timecode show runs: Endless (Endless Repeat)

Property	Property Value	Description
		Once and stop (No Repeat) A specific number of times (number between 2 and 1000)
TimeMarkers		Time markers are used to select the track. They will be stored.
TCSlot	"Internal" (-2), "Link Selected" (-1), "Slot1" (0), "Slot2" (1), "Slot3" (2), etc.	Set the timecode show to a timecode slot.
AutoStart (only available with external timecode slot)	"Yes", "No"	**Only available when syncing to an external source.** If a timecode signal is received, the timecode show switches from the off mode to the play mode.
AutoStop	"Yes", "No"	**Only available when syncing to an external source.** If a timecode signal is received, the timecode show switches from the play mode to the off mode.
User Bits	0 .. FFFFFFFF, 0 .. 4294967296	**Only available when syncing to an external source.** To transmit several kinds of information, for example, a second Timecode Stream, set user bits in hex or decimal. So several incoming Timecode Streams can be discerned.

Examples

- To set timecode show "Intro" to a duration of 55 seconds, type:

```
User name[Fixture]>Set Timecode "Intro" "Duration" "55"
```

- To store a new timecode show called "Napalm Skies" in the timecode pool, type:

```
User name[Fixture]>Store Timecode "Napalm Skies"
```

- To rename timecode show "Intro" "Prelude", type:

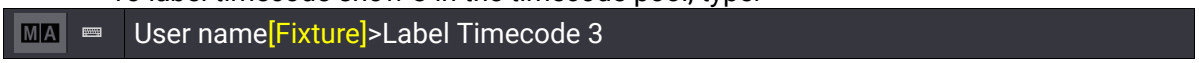
```
User name[Fixture]>Set Timecode "Intro" "Name" "Prelude"
```

- To start the timecode show "Prelude", type:

```
User name[Fixture]>Go Timecode "Prelude"
```

- To rewind the timecode show "Prelude", type:

- To label timecode show 3 in the timecode pool, type:



The label pop-up opens and you can now label the timecode show.

1.9.3.318. TimecodeSlot

To enter the TimecodeSlot keyword in the command line, use one of the options:

- Press MA + X6 + X6
- Type **TimecodeSlot**
- Type **TimecodeS**

Description

The TimecodeSlot keyword is an object keyword which is used to address the timecode slots.

For more information see **What are Timecode Slots.**

Syntax

[Function] TimecodeSlot ["TimecodeSlot_Name" or TimecodeSlot_Number]

Examples

- To select timecode slot 4, type:

```
User name[Fixture]>Select TimecodeSlot 4
```

- To start the timecode generator of TimecodeSlot 2, type:

```
User name[Fixture]>Go+ TimecodeSlot 2
```

- To stop the timecode generator of timecode slot 2, type:

```
User name[Fixture]>Off TimecodeSlot 2
```

1.9.3.319. Timer

To enter the Timer keyword in the command line, use one of the options:

- Press MA + X6 | TC + X6 | TC + X6 | TC
- Type **Timer**

Description

Timer is an object keyword which is used to access the timer objects.

Timers are stored in the Timer pool. There are two types of timers:

- **Stopwatch** - counting up from zero.
- **Countdown** - counting down to zero.

They can be controlled using the Timer pool or the command line.

For more information on timers see **Timers topic**.

Syntax

[Function] Timer ["Timer_Name" or Timer_Number] (Property ["Setting"] ["Setting_Value"])

Settings

In some settings you have to assign an object. To do so use the **Assign Keyword**.

Other settings contain a text option or a value. Use the **Set keyword** for these settings.

The following table displays the settings that have to have an object:

Setting	Object	Description
Appearance	"Appearance 1"	Assigns the appearance to the pool object.
Scribble	"Scribble 1"	Assigns the scribble to the preset pool object.

The following table displays the settings that have to have an option or value:

Setting	Option/Value	Description
Name	"Timer Name"	Sets the name of the timer pool object.
Note	"This is a note"	This is the note to the timer pool object.
TimerMode	"Stopwatch" / "Countdown"	Modes of the timer.
TimeReadout	One of the four readout options	Readout options in the timer.
RestartOption	"Continue on Go+" / "Reset on Go+"	Sets if the timer should restart or continue with a new Go+.
TimerLinkType	One of the five link options	Timers can be linked to sequences and can be activated or toggled.
CountdownAlertType	One of the four alert type	Defines what should happen once the

Setting	Option/Value	Description
	options	countdown has elapsed.
CountdownAlertRange	"Local" / "All Stations"	Defines where an alert pop-up should appear.
CountdownDuration	Time value	Sets the duration of the countdown.
AlertCommand	"Command Line Input"	The command that can be executed once the countdown has elapsed.
AlertText	"Text that appears in a pop-up"	The text is displayed in an alert pop-up.
FrameFormat	One of the five frame format options	A setting that defines which of five formats is used in the timer.
AlertDuration	Time value	Displays the duration of the alert pop-up.

For more information on the settings of the Timer see **Timer**.

Examples

- To run timer 1, type:

User name[Fixture]>Go Timer 1

- To change the timer mode in timer 2, type:

User name[Fixture]>Set Timer 2 "TimerMode" "Countdown"

- To set timer 2 to a countdown time of 25 seconds, type:

User name[Fixture]>Set Timer 2 "CountdownDuration" 25

1.9.3.320. Toggle

To enter the Toggle keyword in the command line, use one of the options:

- Press and hold MA and press Go+ Go+
- Type **Toggle**
- Type the shortcut **To**

Description

The Toggle keyword is a playback and a helping keyword that behaves as an On or Off keyword, depending on the on/off status of an object, function, or mode it is applied to.

If an object, function, or mode is enabled, Toggle disables it. If an object, function, or mode is disabled, Toggle enables it.

For more information see the **On** and **Off Keywords**.

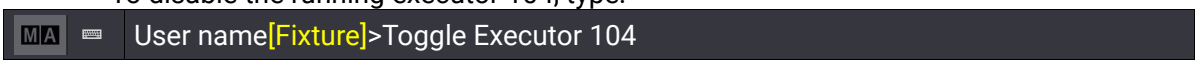
For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Toggle [Object] ["Object_Name" or Object_Number]xt

Examples

- To disable the running executor 104, type:



1.9.3.321. Top

To enter the Top keyword in the command line, use one of the options:

- Press MA + Go-
- Type **Top**

Description

The Top keyword is a playback keyword which is used to jump to the beginning of a cue list or to set a timecode marker at the beginning of a timecode show.

For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

Top [Object] ["Object_Name" or Object_Number]

Examples

- To jump to the top of the cue list on executor 105, type:

```
User name[Fixture]>Top Executor 105
```

- To jump to the top of the cue list on executor 105 in 3 seconds, type:

```
User name[Fixture]>Top Executor 105 Fade 3
```

- To jump to the top of the timecode show 1, type:

```
User name[Fixture]>Top Timecode 1
```

1.9.3.322. Tag

To enter the Tag keyword in the command line, use one of the options:

- Press MA + Group
- Type **Tag**
- Type the shortcut **Ta**

Description

The Tag keyword is a function keyword used to address tags.

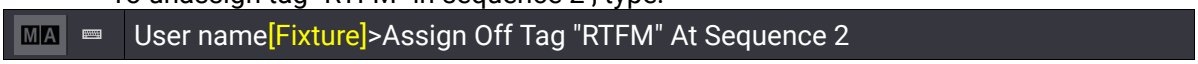
Syntax

[Function] Tag ["Tag_Name" or Tag_Number] (At [Object] ["Object_Name" or Object_Number])

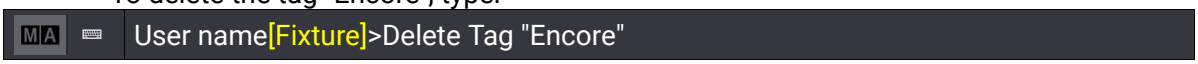
Examples

- To assign tag 1 to sequence 1, type:

- To unassign tag "RTFM" in sequence 2 , type:



- To delete the tag "Encore", type:



1.9.3.323. TopUp

To enter the TopUp keyword in the command line, use one of the options:

- Press Set + Up
- Type **TopUp**
- Type the shortcut **Topu**

Description

The TopUp keyword is a function keyword which is used to select the highest level in the hierarchy of multi-instance fixtures whenever subfixtures are selected.

Syntax

TopUp

Example

- To jump to the main instance of the fixture within which subfixtures are selected, type:

1.9.3.324. Transition

To enter the Transition keyword in the command line, use one of the options:

- Type **Transition**
- Type the shortcut **Tr**

Description

The Transition keyword is used to determine the length of the period in a step.

For more information see **Phasers**.

Syntax

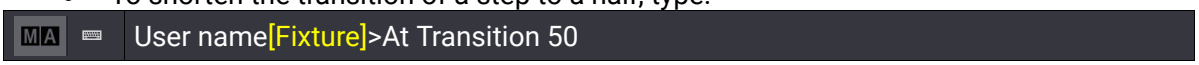
Transition

[At] Transition [Value]

Examples

- To set the layer to transition, type:

- To shorten the transition of a step to a half, type:



1.9.3.325. Type

To enter the Type keyword in the command line, use one of the options:

- Type **Type**
- Type the shortcut **Ty**

Description

The Type keyword is used to address children of an object.

Syntax

[Function] [Object] [Number/"Name"] Type "Name of Type"

Example

- To store cue part 1 in a newly created sequence without specifying the cue number, type:

MA	User name[Fixture]>Store Part 1 Type "Part"

	**Hint:**
	To circumvent the pop-up "Select type of object", press an executor immediately after typing this command.

1.9.3.326. UIChannel

To enter the UIChannel keyword in the command line, use one of the options:

- Type **UIChannel**
- Type the shortcut **Ui**

Description

The UIChannel is an object keyword which represents an attribute in the fixture sheet.

	Important:
	Depending on the fixtures that are patched in the show file, the attributes that are represented in a UIChannel may differ.

Syntax

[List] UIChannel

UIChannel [UIChannel_Number] At [Value]

Examples

- To list all UIChannels in the command line, type:

```
User name[Fixture]>List UIChannel
```

- To set the UIChannel 9 to a value of 42, type:

```
User name[Fixture]>UIChannel 9 At 42
```

1.9.3.327. UIGridSelection

To enter the UIGridSelection keyword in the command line, use one of the options:

- Type **UIGridSelection**
- Type the shortcut **UIG**

Description

UIGridSelection keyword is used by the system to hold information about which objects are selected in a grid window. Grid windows include sheets. It **does not include** the Selection Grid window.

The keyword is primarily used internally by the system, but the selection can be used for normal operations that are executed on the selected objects.

	Important:
	The grid with the selection must have a focus so the UIGridSelection command can actually work.

Syntax

[Function] UIGridSelection

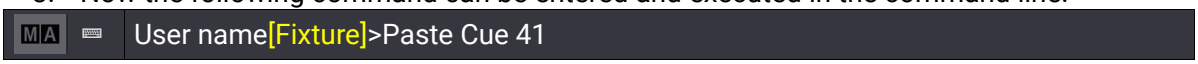
Example

Requirement:

- Several cues in a sequence.
- A macro containing a **Copy UIGridSelection** command. This is important so you can execute the command without moving the focus.

- To copy a selection of the cues to cue 41 using the UIGridCommand, follow these steps:

1. Select the desired cues in the sequence sheet.
2. Run the macro using the command keys (Important to use the keys to keep the focus in the cue selection).
3. Now the following command can be entered and executed in the command line:



1.9.3.328. Unblock

	Important:
	To block parameters, use the **Block keyword**.

To enter the Unblock keyword in the command line, use one of the options:

- Type **Unblock**
- Type the shortcut **UB** or **Unb**

Description

Unblock is a function that converts blocked values into tracking values in cues.

If the object list does not contain any references to any cues, the unblock function is applied to the selected sequence.

If unblock does not contain any selection list filters, all fixtures will be used.

If unblock does not contain any attribute list filters, all attributes will be used.

Syntax

Unblock (Sequence ["Sequence_Name" or Sequence_Number]) Cue ["Cue_Name" or Cue_Number] (If Fixture ["Fixture_Name" or Fixture_Number] FeatureGroup ["FeatureGroup_Name" or FeatureGroup_Number] or Attribute ["Attribute_Name" or Attribute_Number] EndIf)

Examples

- To unblock all parameters of the selected sequence, type:

```
User name[Fixture]>Unblock
```

- To unblock pan and tilt of fixture 4 in cue 5 of the selected sequence, type:

```
User name[Fixture]>Unblock Cue 5 If Fixture 4 Attribute "Position" EndIf
```

1.9.3.329. Universal

To enter the **Universal** keyword in the command line, use one of the options:

- Press Channel Channel
- Type **Universal**
- Type the shortcut **Uni**
- Press **F + 2** on a keyboard

Description

The Universal keyword is an object keyword wich is used to call fixtures of the ID type Universal in the programmer.

Syntax

Universal ["IDType_Name" or IDType_Number]

Example

- To select the fixture that uses the Universal ID 1, type:

1.9.3.330. Unlock

	Hint:
	A padlock indicates a locked object.
	Hint:
	To lock objects, use the **Lock Keyword**.

To enter the Unlock keyword in the command line, use one of the options:

- Type **Unlock**
- Type the shortcut **Ul** or **Unl**

Description

The Unlock keyword is a function keyword which is used to unlock objects that were previously locked.

Syntax

Unlock [Object] ["Object_Name" or Object_Number]

Examples

- To unlock cue 3 to edit the cue, type:

```
User name[Fixture]>Unlock Cue 3
```

- To unlock world 3 to edit the world, type:

```
User name[Fixture]>Unlock World 3
```

- To unlock the macro line 2 of the macro 1, type:

```
User name[Fixture]>Unlock Macro 1.2
```

1.9.3.331. Unpark

To enter the Unpark keyword in the command line, use one of the options:

- Press Go+ Go+
- Type **Unpark**
- Type **Unp**

Description

The Unpark keyword is a command keyword which is used to release a parked DMX channel or universe.

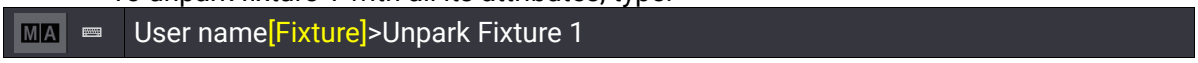
	**Hint:**
	If you unpark a universe, the blue **P** icon disappears in the universe pool.

Syntax

Unpark [Object] ["Object_Name" or Object_Number]

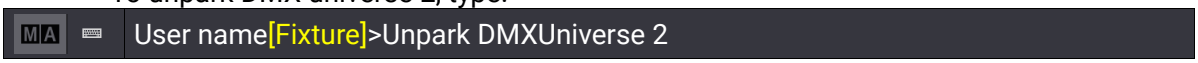
Examples

- To unpark fixture 1 with all its attributes, type:



- To unpark the current selection, type:

- To unpark DMX universe 2, type:



1.9.3.332. Unpress

To enter the Unpress keyword in the command line, use one of the options:

- Type **Unpress**
- Type the shortcut **Unpr**

Description

The Unpress keyword is used to simulate the unpressed state of a key.

It is possible to assign this state to macros.

For more information see **Macros**.

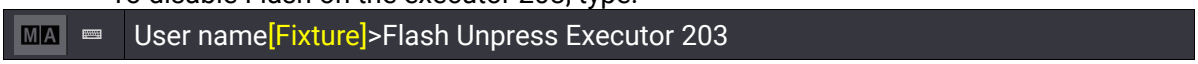
For information on how to assign executors see **Assign Objects to an Executor**.

Syntax

[Function] Unpress [Object] ["Object_Name" or Object_Number]

Example

- To disable Flash on the executor 203, type:



1.9.3.333. Up

To enter the Up keyword in the command line, use one of the options:

- Press Up
- Type **Up**

Description

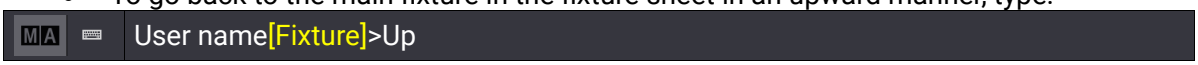
The Up keyword is a function keyword which is used to navigate upward in the fixture structure.

Syntax

Up

Example

- To go back to the main fixture in the fixture sheet in an upward manner, type:



1.9.3.334. Update

To enter the Update keyword in the command line, use one of the options:

- Press Update
- Type **Update**
- Type the shortcut **U**

Description

The Update keyword is a function keyword which is used to update values in their source objects.

Syntax

Update [Object List]

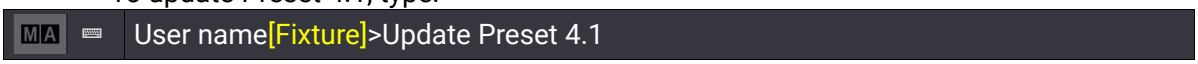
Option Keywords

The Update Keyword uses the following option keywords:

- **/AddNewContent**
- **/ForceGlobal**
- **/Global**
- **/InputFilter**
- **/OriginalContentOnly**
- **/Selective**
- **/Universal**

Example

- To update Preset 4.1, type:



1.9.3.335. UpdateContent

To enter the UpdateContent keyword in the command line, use one of the options:

- Type **UpdateContent**
- Type the shortcut **UC** or **UpdateC**

Description

The UpdateContent keyword is used to scan a media pool, for example images, and create XML files for media files.

Syntax

UpdateContent [Object] ["Object_Name" or Object_Number]

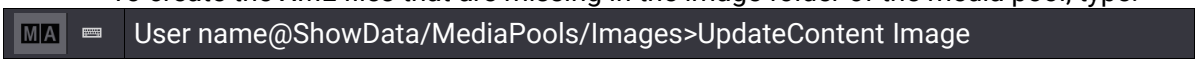
Example

The following example is explained using images.

Requirement:

1. A media file, for example an image, was added to the corresponding media folder, for example in grandMA3_lib/media/images, without an XML file.
2. Enter the media pool folder and then the image folder.
   For more information see **ChangeDestination keyword**.

- To create the XML files that are missing in the image folder of the media pool, type:



1.9.3.336. User

To enter the User keyword in the command line, use one of the options:

- Type **User**
- Type the shortcut **Us**

Description

The User keyword is used to log in or to change the user settings.

For more information on users and different settings see **User Setting**.

Syntax

User ["User_Name" or User_Number]

[Function] User ["User_Name or User_Number] ([Setting] [Setting_Value])

Settings

The User keyword uses a number of settings. Change the settings using the **Set Keyword**. If a setting has to have a pool object, use the **Assign keyword** to assign an object to the setting.

Here are the settings:

Setting	Object/Option/Value	Description
Name	Text	The name of the user. This can be used as user ID.
Scribble	Scribble pool object	The Scribble assigned to the User object.
Appearance	Appearance pool object	The Appearance assigned to the User object.
Password	Text	The password is not shown in clear text.
Profile	UserProfile object	The assigned User Profile.
ScreenConfig	Screen configuration object	The active screen configuration.
Rights	"Admin", "Setup", "Program", "Preset", "Playback", "View", or "None"	The access right assigned to the user.
Language	"de", "en", "ru", or "dk"	The language assigned to the user.
Keyboard	"German", "English", "Russian", or "Danish"	The keyboard layout assigned to the user's onscreen keyboard.

Example

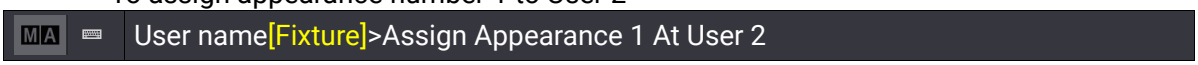
- To list the details of all users, type:

```
User name[Fixture]>List User
```

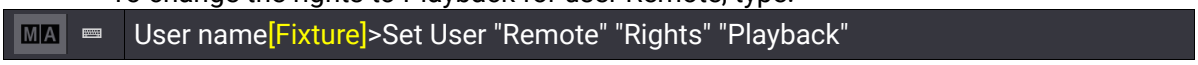
- To log in as "Admin", type:

Alternatively see the **Login Keyword**.

- To assign appearance number 1 to User 2



- To change the rights to Playback for user Remote, type:



1.9.3.337. User1

	**Important:**
	The **User1** keyword can only be used on the grandMA3 compact, the grandMA3 compact XT, and the grandMA3 onPC command wing where the user key is located on the right of Special Executor 1 and 2.
	**Hint:**
	The **User1** keyword must be used as one word.

To enter the **User1** keyword in the command line, use one of the options:

- Press U1
- Type **User1**

Description

The **User1** keyword is a function keyword that toggles between the encoder bar and the Xkeys bar.

Syntax

User1

Example

- To toggle the encoder bar to the Xkeys bar, type:

1.9.3.338. User2

	**Important:**
	The **User2** keyword can only be used on the grandMA3 compact, the grandMA3 compact XT, and the grandMA3 onPC command wing where the user key is located on the right of Special Executor 1 and 2.
	**Hint:**
	The **User2** keyword must be used as one word.

To enter the **User2** keyword in the command line, use one of the options:

- Press U2
- Type **User2**

Description

The **User2** keyword is a function keyword that toggles between the encoder bar and the executor bar.

Syntax

User2

Example

- To toggle the encoder bar to the executor bar, type:

1.9.3.339. UserProfile

To enter the UserProfile keyword in the command line, use one of the options:

- Type **UserProfile**
- Type **Userp**
- Type the shortcut **UPR**

Description

The UserProfile keyword is an object keyword. It can be used to execute functions in the UserProfile such as create, delete, or change a setting.

For more information on user profiles see **Create User**. For information on different settings see **User Settings**.

Syntax

[Function] UserProfile ["UserProfile_Name" or UserProfile_Number] ([Setting] [Setting_Value])

Settings

The UserProfile keyword has several settings. The settings can be changed using the **Set Keyword**.

Here are the settings:

Setting	Object/Option/Value	Description
Name	Text	The name of the user profile. This can be used as the user profile ID.
DMXReadout	"Percent", "Dec8", "Dec16", "Dec24", "Hex8", "Hex16", or "Hex24"	This is the default DMX readout.
NormalValue	A percent value	This is the value assigned to the intensity attribute if the **Normal keyword** is used.
WheelResolution	"Coarse", "Normal", or "Fine"	The resolution of the **level wheel** on the consoles.
WheelMode	"Additive", "Incremental", "Prop.+", or "Prop.-"	This defines the level wheel mode.
PreciseEdit	"Yes" or "No"	This turns the precise edit mode On or Off.
ScreenEncoder	"Yes" or "No"	This enables or disables the screen encoder on the fifth **dual encoder**.
TimeKeyTarget	"Cue" or "Fixture"	This defines how the **Time key** functions.
TCSlot	"TCSlot 1" .. "TCSlot 8"	This is the user profile's selected timecode slot.
ValueReadout	"Natural", "Percent", "PercentFine", "Physical", "Decimal8",	This is the default value readout.

Setting	Object/Option/Value	Description
	"Decimal16", "Decimal24", "Hex8", "Hex16", or "Hex24"	
SpeedReadout	"Hertz", "BPM", or "Seconds"	This is the default speed readout.
PresetReadout	"ID", "ID+Name", "ID+Name+Value", "Name", "Name+Value", or "Value"	This is the default readout for preset in sheets.
OverlayFade	Time value in milliseconds	This is the time used by pop-ups and overlays when fading into visibility.
TimeReadout	"10d11h23m45", "251h23m45", "10.11:23:45", or "251:23:45"	This is the default time readout.
FrameReadout	"Seconds", "24 fps", "25 fps", "30 fps", or "60 fps"	This is the default time frame readout.
UndoProgrammer	"Yes" or "No"	This turns On or Off if programmer actions can be oopsed.
UndoGeneral	"Yes" or "No"	This turns On or Off if general actions can be oopsed.
AutoRemoveGaps	"Yes" or "No"	This turns On or Off the auto-remove gaps function in the selection grid.
MirrorSpecialExecutorPages	"Yes" or "No"	This turns On or Off the mirror function on extension wings.
ShowAppearanceInCueInput	"Yes" or "No"	This turns On or Off the cue appearance in cue selection pop-ups.
ShowSettingsInEditors	"Yes" or "No"	This turns On or Off if settings are visible in editors as a default.
ExecConfigSequence	Executor Configuration Object	Default executor configuration for sequences.
ExecConfigMacro	Executor Configuration Object	Default executor configuration for macros.
ExecConfigView	Executor Configuration Object	Default executor configuration for views.
ExecConfigWorld	Executor Configuration Object	Default executor configuration for worlds.
ExecConfigGroup	Executor Configuration Object	Default executor configuration for groups.
ExecConfigPreset	Executor Configuration Object	Default executor configuration for presets.
ExecConfigPlugin	Executor Configuration Object	Default executor configuration for plugins.
ExecConfigUser	Executor Configuration Object	Default executor configuration for users.
ExecConfigSound	Executor Configuration Object	Default executor configuration for sounds.
ExecConfigScreenConfig	Executor Configuration Object	Default executor configuration for screen configurations.
ExecConfigMaster	Executor Configuration Object	Default executor configuration for masters.
ExecConfigSpeedMaster	Executor Configuration Object	Default executor configuration for speed masters.
ExecConfigPlaybackMaster	Executor Configuration Object	Default executor configuration for playback masters.
ShowConnectors	"Yes" or "No"	This turns On or Off if the connector overlay is visible when the "Output Configuration" menu is visible.

Setting	Object/Option/Value	Description
FixtureLibShowMA	"Yes" or "No"	This turns On or Off if MA fixtures are shown in the "Insert New Fixture" pop-up.
FixtureLibShowUser	"Yes" or "No"	This turns On or Off if User fixtures are shown in the "Insert New Fixture" pop-up.
FixtureLibShowShare	"Yes" or "No"	This turns On or Off if Share fixtures are shown in the "Insert New Fixture" pop-up.

For detailed expanation see **User Settings**.

Examples

- To list all available user profiles, type:

```
User name[Fixture]>List UserProfile
```

- To turn off the screen encoder in the default user profile, type:

```
User name[Fixture]>Set UserProfile "Default" "ScreenEncoder" "No"
```

1.9.3.340. Video

To enter the Video keyword in the command line, use one of the options:

- Type **Video**
- Type the shortcut **Vid**

Description

The Video keyword is an object keyword. It is used to address videos using the command line.

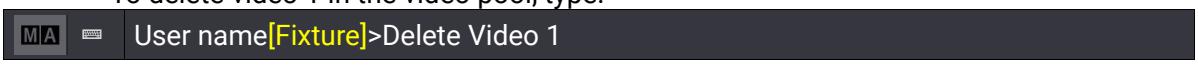
Syntax

[Function] Video ["Video_Name" or Video_Number]

Examples

- To edit video 1 in the video pool, type:

- To delete video 1 in the video pool, type:



1.9.3.341. Version

To enter the Version keyword in the command line, use one of the options:

- Type **Version**
- Type the shortcut **Ve**

Description

The Version keyword is a function keyword which is used to grant access to details of the version in the console or in the onPC software.

It displays the following details in the command line history:

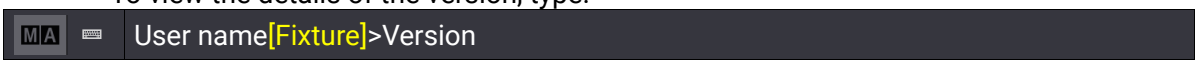
- Version number
- Release type
- Build type
- Code type
- Host type
- Host sub type
- Station serial number
- Date of compilation
- Repository branch
- Repository hash
- Version number of Lua
- Version number of Lua sockets
- Version number of Lua file system

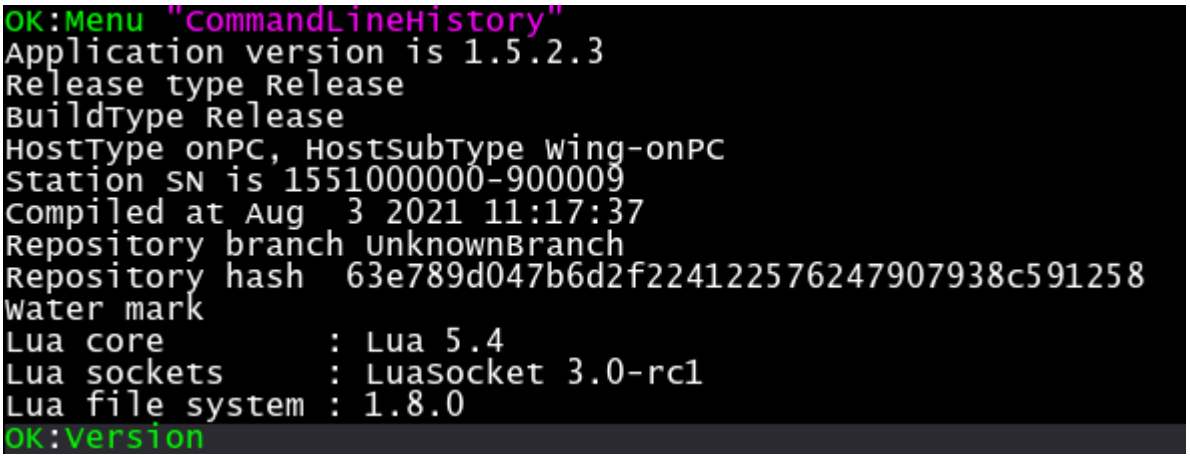
Syntax

Version

Example

- To view the details of the version, type:





Version details in the command line history window

The version details are displayed in the command line history window.

1.9.3.342. View

To enter the View keyword in the command line, use one of the options:

- Press MA + X7 | View
- Type **View**
- Type the shortcut **V**

Description

The View keyword calls or stores views on a screen.

Syntax

View ["View_Name" or View_Number] (/Option ["Option_Value"])

[Function] View ["View_Name" or View_Number] (/Option ["Option_Value"])

Settings

The following table displays the properties that can be assigned using the **Set keyword**.

Setting	Option/Value	Description
Name	View name, for instance, "Stage External"	The name of the view.
Scribble	Scribble pool object	The assigned scribble object.
Appearance	Appearance pool object	The assigned appearance object.
ScreenContentMask	1 to 127	An internal number that indicates the screens used when it was saved.
RequestedW	1 to 327	The width of the view (grid width).
RequestedH	1 to 327	The height of the view (grid height).

Option Keywords

The View keyword uses the following option keywords:

- **/Screen**

Example

- To call view 2 in the view pool on the screen where it is stored, type:

1.9.3.343. ViewButton

To enter the ViewButton keyword in the command line, use one of the options:

- Press MA + X7|View + X7|View
- Type **ViewButton**
- Or type the shortcut **ViewB**

Description

The ViewButton keyword is an object keyword which is used to call or store the object assigned on the view button.

Calling a view button only works if the object assigned to it supports it.

For more information see the **Call Keyword**.

Syntax

ViewButton [Screen_Number].["ViewButton_Name" or ViewButton_Number] (/Option "[Option_Value]")

[Function] ViewButton [Screen_Number].["ViewButton_Name" or ViewButton_Number] (/Option "[Option_Value]")

Option Keywords

The ViewButton keyword uses the following option keywords:

- **/Screen**

Examples

- To call the view assigned to ViewButton 4 on screen 2, type:

MA | User name[Fixture]>ViewButton 2.4

Important:
If you do not specify the screen location using the /Screen option keyword, the view will be called to the screen that currently has focus.

- To assign the user pool object "Guest" to ViewButton 1.1, type:

MA | User name[Fixture]>Assign User "Guest" At ViewButton 1.1

- To remove the object assigned to ViewButton 4, screen 1, type:

MA | User name[Fixture]>Delete ViewButton 1.4

- To label the view that is assigned to ViewButton 5 on screen 2 "Layout", type:

```
User name[Fixture]>Label ViewButton 2.5 "Layout"
```

1.9.3.344. Width

To enter the Width keyword in the command line, use one of the options:

- Type **Width**
- Type the shortcut **Wi**

Description

The Width keyword is used to determine the width incorporating transition of the entire step.

For more information see **Phasers**.

Syntax

Width

[At] Width [Value]

Examples

- To set the layer to width, type:

- To create a PWM effect, type:

1.9.3.345. World

To enter the World keyword in the command line, use one of the options:

- Press Group Group
- Type **World**
- Or type the shortcut **W**

Description

The World keyword is an object keyword which is used to call worlds along with their filters, and limit the access to the parameters in the world.

By default, World 1 is always locked. That is, it neither can be edited nor deleted. World 1 includes all parameters (fixtures and attributes) of the show.

Syntax

[Function] World ["World_Name" or World_Number]

Option Keywords

The World keyword uses the following option keywords:

- **/GridMergeMode**
- **/Merge**
- **/Overwrite**
- **/Remove**

Examples

- To call world 3, type:

```
User name[Fixture]>World 3
```

- To label the world 3 "All Fixtures", type:

```
User name[Fixture]>Label World 3 "All Fixtures"
```

1.9.3.346. Zero

To enter the Zero keyword in the command line, use one of the options:

- Press .
- Type **Zero**
- Type the shortcut **Z**

Description

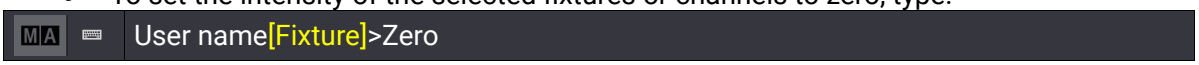
The keyword Zero is a helping keyword which is used to set the intensity of the dimmer to zero if fixtures or channels are selected.

Syntax

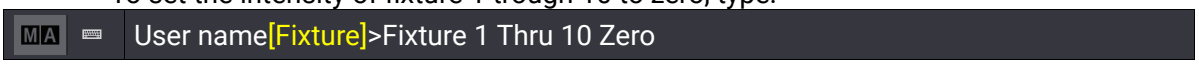
[Object List] Zero

Examples

- To set the intensity of the selected fixtures or channels to zero, type:



- To set the intensity of fixture 1 trough 10 to zero, type:



1.9.4. Option Keywords

Option keywords can be seen as temporary filters in commands. You can use them to define your command in an even more detailed manner.

The result is always a single event – that is, it will not be set permanently as it is the case with settings for example.

Subtopics

- **/Active**
- **/ActiveForSelected**
- **/AddNewContent**
- **/All**
- **/AllForSelected**
- **/Ask**
- **/Async**
- **/Auto**
- **/ChannelSet**
- **/CopyCueDestination**
- **/CopyCueSource**
- **/CreateReferenceObject**
- **/CreateSecondCue**
- **/CueOnly**
- **/Date**
- **/Default**
- **/DiscardChanges**
- **/DMX**
- **/DMXProtocols**
- **/Embed**
- **/Enumerate**
- **/File**
- **/ForceGlobal**
- **/GDTF**
- **/Gaps**
- **/Global**
- **/GridMergeMode**
- **/Indirect**
- **/InputFilter**
- **/KeepActivation**
- **/Limit**
- **/LocalSettings**
- **/Look**
- **/MAtricks**
- **/Merge**
- **/MergeHighPriority**
- **/MergeLowPriority**
- **/MoveValues**
- **/NoAutoClose**

- **/NoConfirmation**
- **/NDI**
- **/NoDependencies**
- **/NoOops**
- **/NoRefresh**
- **/NoSave**
- **/NoShowData**
- **/NoSubfolders**
- **/OriginalContentOnly**
- **/Output**
- **/OutputStations**
- **/Overwrite**
- **/PatchOffset**
- **/Path**
- **/PhaserData**
- **/Programmer**
- **/Release**
- **/Recipe**
- **/Remove**
- **/Save**
- **/Screen**
- **/ScreenOnly**
- **/Selection**
- **/Selective**
- **/Single**
- **/Sort**
- **/SubTab**
- **/Tab**
- **/TrackingShield**
- **/Type**
- **/Universal**
- **/Wait**
- **/Wrap**
- **/XResolution**
- **/YResolution**

1.9.4.1. /Active

To enter the **/Active** option keyword in the command line, use one of the options:

- Type **/Active**
- Type the shortcut **/Ac**

Description

The /Active option keyword stores all active values of the programmer no matter the selection.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /Active

General Keywords

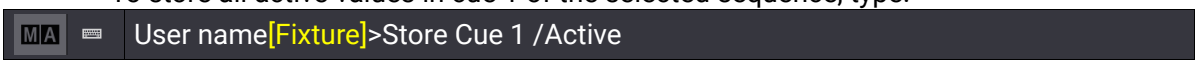
General keywords that use the /Active option keyword:

- **Cue keyword**
- **Preset keyword**
- **Store keyword**

Example

Requirement: Active values in the programmer

- To store all active values in cue 1 of the selected sequence, type:



1.9.4.2. /ActiveForSelected

To enter the **/ActiveForSelected** option keyword in the command line, use one of the options:

- Type **/ActiveForSelected**
- Type the shortcut **/ActiveF**

Description

The /ActiveForSelected option keyword stores all active attributes of the selected fixtures in the programmer.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /ActiveForSelected

General Keywords

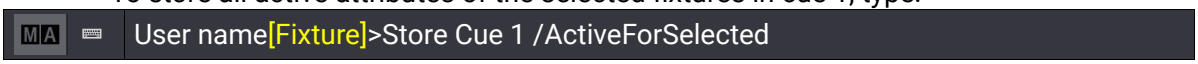
General keywords that use the /ActiveForSelected option keyword:

- **Cue keyword**
- **Preset keyword**
- **Store keyword**

Example

Requirement: Selected fixtures with active values in the programmer

- To store all active attributes of the selected fixtures in cue 1, type:



1.9.4.3. /AddNewContent

To enter the **/AddNewContent** option keyword in the command line, use one of the options:

- Type **/AddNewContent**
- Type the shortcut **/Ad**

Description

The /AddNewContent option keyword updates existing values of attributes and adds new values in presets or cues.

Syntax

Update [Object] ["Object_Name" or Object_Number] /AddNewContent

General Keywords

General keywords that use the /AddNewContent option keyword:

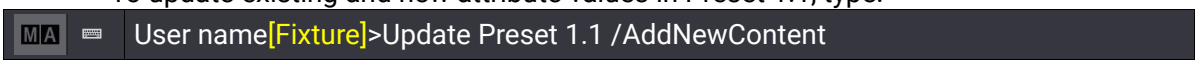
- **Cue keyword**
- **Preset keyword**
- **Update keyword**

Example

Requirement:

- Create presets or cues
- Playback presets or cues or activate presets in the programmer
- At least one attribute must have active values in the programmer

- To update existing and new attribute values in Preset 1.1, type:



1.9.4.4. /All

To enter the **/All** option keyword in the command line, use one of the options:

- Type **/All**
- Type the shortcut **/Al**

Description

The /All option keyword stores the values of all attributes when using Data Source Output or DMX.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /All

General Keywords

General keywords that use the /All option keyword:

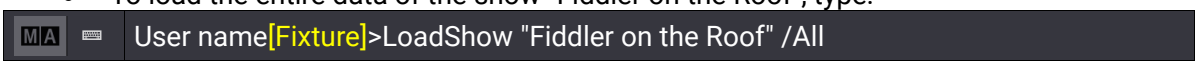
- **AutoCreate keyword**
- **Cue keyword**
- **FixtureClass keyword**
- **FixtureLayer keyword**
- **FixtureType keyword**
- **LoadShow keyword**
- **NewShow keyword**
- **Preset keyword**
- **Store keyword**

	**Hint:**
	The /All option keyword also works in conjunction with the **/Look option keyword**.

Examples

- To store all attributes in cue 1, type:

- To load the entire data of the show "Fiddler on the Roof", type:



- To create a new show file called "Rocky Horror Picture Show" and clear all previous settings, type:

User name[Fixture]>NewShow "Rocky Horror Picture Show" /All

Requirement:

1. Select eight fixtures.
2. Set MAtricks XGroup to 2.
3. Set MAtricks X to 0.
   - To create group 21 in the group pool containing all main fixtures of the selection (odd fixtures of the current selection), type:

User name[Fixture]>AutoCreate Selection At Group 21 /All

Requirement: Create layers and classes and set fixtures to these.

For more information and how to use fixture classes and fixture layers see Patch and **Fixture Setup - Classes and Layers**.

Hint:

The demo show runs using this setting. For more information see **Backup, Demo, and Template Show Files**.

- To create a group at group pool object 301 with all fixtures that are set to class "Spots" within the patch, type:

User name[Fixture]>AutoCreate FixtureClass "Spots" At Group 301 /All

- To create a group at group pool object 201 with all fixtures that are set to layer "Backtruss" within the patch, type:

User name[Fixture]>AutoCreate FixtureLayer "Backtruss" At Group 201 /All

- To create a group at group pool object 42 with all patched fixtures of fixture type 9, type:

User name[Fixture]>AutoCreate FixtureType 10 At Group 42 /All

1.9.4.5. /AllForSelected

To enter the **/AllForSelected** option keyword in the command line, use one of the options:

- Type **/AllForSelected**
- Type the shortcut **/Allf**

Description

The /AllForSelected option keyword stores all attributes of the selected fixtures in the programmer.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /AllForSelected

General Keywords

General keywords that use the /AllForSelected option keyword:

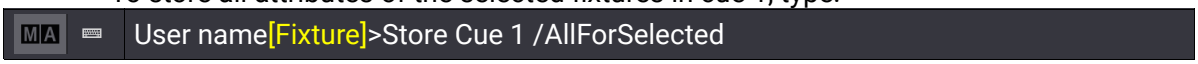
- **Cue keyword**
- **Preset keyword**
- **Store keyword**

	**Hint:**
	The /AllForSelected option keyword also works in conjunction with the **/Look option keyword**.

Example

Requirement: Selected fixtures with values in the programmer

- To store all attributes of the selected fixtures in cue 1, type:



1.9.4.6. /Ask

To enter the **/Ask** option keyword in the command line, use one of the options:

- Type **/Ask**
- Type the shortcut **/A**

Description

The /Ask option keyword is a default setting which is used to define the store option in an object that already has values. The /Ask option keyword triggers a pop-up where it is possible to set the store options.

Syntax

Store [Object] ["Object_Name" or Object_ID] /Ask

General Keywords

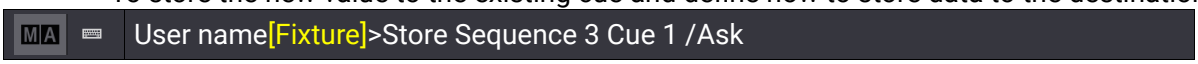
General keywords that use the /Ask option keyword:

- **Cue keyword**
- **Filter keyword**
- **Group keyword**
- **MAtricks keyword**
- **Preset keyword**
- **Sequence keyword**
- **Store keyword**
- **World keyword**

Example

Requirement: An existing cue

- To store the new value to the existing cue and define how to store data to the destination, type:



1.9.4.7. /Auto

To enter the **/Auto** option keyword in the command line, use one of the options:

- Type **/Auto**
- Type the shortcut **/Au**

Description

The /Auto option keyword is used to store and update presets. Executing the /Auto option keyword sets the preset mode to **Auto**. For more information see **Presets**.

Syntax

[Function] Preset ["FeatureGroup_Name" or FeatureGroup_ID].["Preset_Name" or Preset_ID] /Auto

General Keywords

General keywords that use the /Auto option keyword:

- **Preset keyword**
- **Store keyword**
- **Update keyword**

Example

- To store the active values into color preset 3 using the Preset Mode "Auto", type:

1.9.4.8. /Async

To enter the **/Async** option keyword in the command line, use one of the options:

- Type **/Async**
- Type the shortcut **/Asy**

Description

The /Async option keyword is used to asynchronously execute remote commands.

Syntax

RemoteCommand IP [IP_Address] ["Command to be Executed"] /Async

RemoteCommand [DeviceType] ["Device_Name" or Device_Number] ["Command to be Executed"] /Async

Example

- To asynchronously execute the command "Delete Macro 1" on the console named "DimmerBeach", type:

```
User name[Fixture]>RemoteCommand Console "DimmerBeach" "Delete Macro 1 /NoConfirmation" /Async
```

1.9.4.9. /ChannelSet

To enter the **/ChannelSet** option keyword in the command line, use one of the options:

- Type **/ChannelSet**
- Type the shortcut **/Ch**

Description

The /ChannelSet option keyword is used in conjuction with the AutoCreate keyword – whereby the AutoCreate command creates objects out of channel sets.

Syntax

AutoCreate [Source_Object] ["Source_Object_Name" or Source_Object_Number] At [Destination_Object] ["Destination_Object_Name" or Destination_Object_Number] /ChannelSet

General Keywords

General keywords that use the /ChannelSet option keyword:

- **AutoCreate keyword**

Example

- To create global beam presets out of channel sets of fixture type 9, type:

```
User name[Fixture]>AutoCreate FixtureType 9 At Preset * If FeatureGroup "Beam" /ChannelSet
```

1.9.4.10. /CopyCueDestination

To enter the **/CopyCueDestination** option keyword in the command line, use one of the options:

- Type **/CopyCueDestination**
- Type the shortcut **/CopyCueD** or **/CCD**

Description

The /CopyCueDestination option keyword defines how the copied data is used in the destination cue when copying cues.

For more information see **Copy Cues**.

Syntax

Copy (Sequence ["Source_Sequence_Name" or Source_Sequence_ID]) Cue ["Source_Cue_Name" or Source_Cue_ID] At (Sequence ["Destination_Sequence_Name" or Destination_Sequence_ID]) Cue ["Destination_Cue_Name" or Destination_Cue_ID] /CopyCueDestination "Value"

General Keywords

General keywords that use the /CopyCueDestination option keyword:

- **Cue keyword**
- **Copy keyword**

Values

The /CopyCueDestination option keyword uses these values:

- Merge
- Overwrite

Example

Requirement: Create cues 1 and 2

- To copy cue 1 to cue 2 and merge the data of cue 1 with cue 2, type:

1.9.4.11. /CopyCueSource

To enter the **/CopyCueSource** option keyword in the command line, use one of the options:

- Type **/CopyCueSource**
- Type the shortcut **/Cop, /CC** or **/CCS**

Description

The /CopyCueSource option keyword defines how data is extracted in the source cue when copying cues.

For more information see **Copy Cues**.

Syntax

Copy (Sequence ["Source_Sequence_Name" or Source_Sequence_ID]) Cue ["Source_Cue_Name" or Source_Cue_ID] At (Sequence ["Destination_Sequence_Name" or Destination_Sequence_ID]) Cue ["Destination_Cue_Name" or Destination_Cue_ID] /CopyCueSource "Value"

General Keywords

General keywords that use the /CopyCueSource option keyword:

- **Cue keyword**
- **Copy keyword**

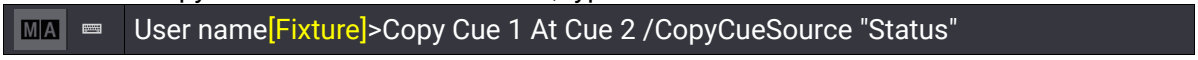
Values

The /CopyCueSource option keyword uses these values:

- Content
- Status
- Look

Example

- To copy the status of cue 1 to cue 2, type:



1.9.4.12. /CreateReferenceObject

To enter the **/CreateReferenceObject** option keyword in the command line, use one of the options:

- Type **/CreateReferenceObject**
- Type the shortcut **/CreateR**

Description

The /CreateReferenceObject option keyword creates a referenced object when an object is imported. For example, an appearance which uses the imported image.

For more information on importing references see **Import/Export Menu**.

Syntax

Import [Object Type] Library "File Name.file_type" At [Object Type] ([Object_Number] [Object_ID]) /CreateReferenceObject

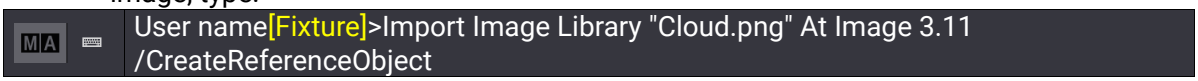
General Keywords

General keywords that use the /CreateReferenceObject option keyword:

- **Import keyword**
- **Image keyword**

Example

- To import an image and automatically create an appearance which references to the imported image, type:



1.9.4.13. /CreateSecondCue

To enter the **/CreateSecondCue** option keyword in the command line, use one of the options:

- Type **/CreateSecondCue**
- Type the shortcut **/C**

Description

The /CreateSecondCue option keyword stores a cue using the next whole number provided the sequence has only one cue.

Syntax

[Function] (Sequence ["Sequence_Name" or "Sequence_ID"]) /CreateSecondCue

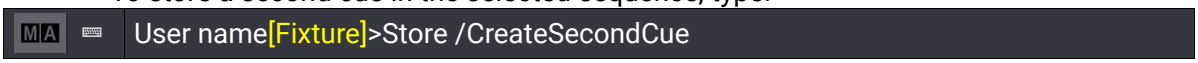
General Keywords

General keywords that use the /CreateSecondCue option keyword:

- **Cue keyword**
- **Store keyword**

Example

- To store a second cue in the selected sequence, type:



1.9.4.14. /CueOnly

To enter the **/CueOnly** option keyword in the command line, use one of the options:

- Type **/CueOnly**
- Type the shortcuts **/Co** or **/Cu**

Description

The /CueOnly option keyword blocks tracked values in the next cue or cue part to preserve the previous look on stage.

For more information see **Store Cues** and **Store Settings and Store Preferences**.

Syntax

[Function] (Sequence ["Sequence_Name" or Sequence_Number]) Cue ["Cue_Name" or Cue_Number] /CueOnly ["Value"]

General Keywords

General keywords that use the /CueOnly option keyword:

- **Cue keyword**
- **Delete keyword**
- **Store keyword**
- **Update keyword**

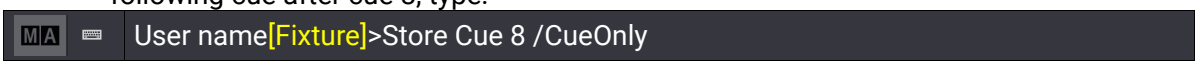
Values

The /CueOnly option keyword uses these values:

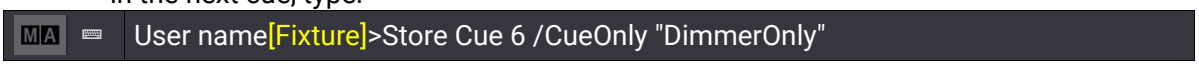
- DimmerOnly – uses the Dimmer Cue Only and releases the recently stored dimmer attributes in the next cue.
- DimmerOnlyDefaultNew – uses the Dimmer Cue Only and sets recently stored dimmer attributes to the default value in the next cue.
- Off – does not use CueOnly
- On – uses CueOnly
- OnDefaultNew – uses Cue Only and sets new attributes within the sequence to the default value in the next cue.

Examples

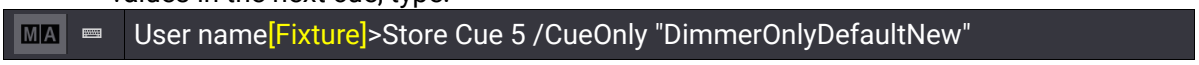
- To store the current programmer values in cue 8 and to preserve the previous look in the following cue after cue 8, type:



- To store the current programmer values in cue 6 and release the recently stored dimmer values in the next cue, type:



- To store the current programmer values in cue 5 and set the dimmer attributes to default values in the next cue, type:



1.9.4.15. /Date

To enter the **/Date** option keyword in the command line, use one of the options:

- Type **/Date**
- Type the shortcut **/Da**

Description

The /Date option keyword changes the start date of an agenda event that is to be pasted from clipboard. All other settings of this agenda event will not be changed and stay the same.

Syntax

Paste Agenda [Agenda_Number] /Date "DD.MM.YYYY"

General Keywords

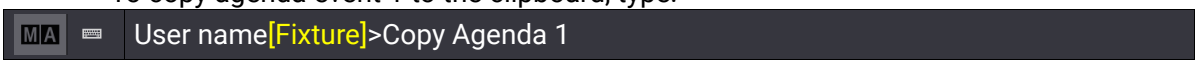
General keywords that use the /Date option keyword:

- **Agenda keyword**
- **Paste keyword**

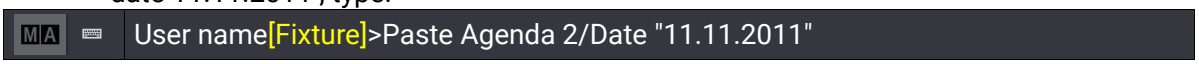
Example

Requirement: An existing agenda event was copied to the clipboard.

- To copy agenda event 1 to the clipboard, type:



- To paste agenda event 1 back from the clipboard to create an agenda event 2 with a new start date 11.11.2011 , type:



1.9.4.16. /Default

To enter the **/Default** option keyword in the command line, use one of the options:

- Type **/Default**
- Type the shortcut **/D**

Description

The /Default option keyword is used when copying cues. It replaces non-existing data in the destination cue by default values.

Syntax

Copy Cue ["Cue_Name" or Cue_Number] /Default

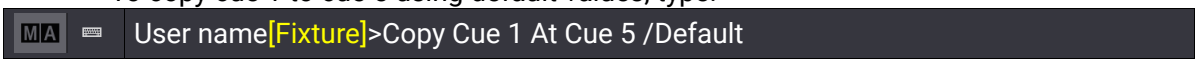
General Keywords

General keywords that use the /Default option keyword:

- **Copy keyword**
- **Cue keyword**

Example

- To copy cue 1 to cue 5 using default values, type:



1.9.4.17. /DiscardChanges

To enter the **/DiscardChanges** option keyword in the command line, use one of the options:

- Type **/DiscardChanges**
- Type the shortcut **/DC** or **/Di**

Description

The /DiscardChanges option keyword is used to leave the Patch without storing the changes.

Syntax

ChangeDestination [Object] (["Object_Name" or Object_Number]) /DiscardChanges

General Keywords

General keywords that use the /DiscardChanges option keyword:

- **ChangeDestination keyword**

Example

Requirement: You are in the Patch and you made edits.

- To leave the patch without storing your recent edits in Stage1, type:

```
User name@ShowData/Patch/Stages/Stage1/Fixtures>ChangeDestination Root /DiscardChanges
```

1.9.4.18. /DMX

To enter the **/DMX** option keyword in the command line, use one of the options:

- Type **/DMX**
- Type the shortcut **/DM**

Description

The /DMX option keyword is used to store incoming DMX data to cues or presets.

For more information on data sources see **Store Settings and Preferences**.

Syntax

Store [Object] ["Object_Name" or Object_Number] /DMX [/AllForSelected or /All]

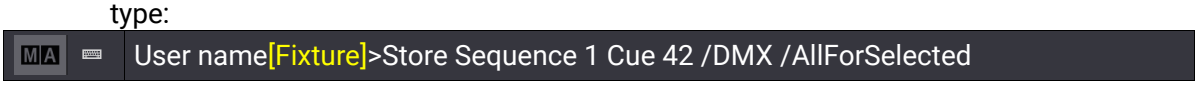
General Keywords

General keywords that use the /DMX option keyword:

- **Cue keyword**
- **Preset keyword**
- **Sequence keyword**

Example

- To store the incoming DMX data to sequence 1 cue 42 in all attributes of the selected fixtures, type:



1.9.4.19. /DMXProtocols

To enter the **/DMXProtocols** option keyword in the command line, use one of the options:

- Type **/DMXProtocols**
- Type the shortcut **/DMXP**

Description

The /DMXProtocols option keyword is used to load the DMX protocol settings (Art-Net and sACN) of the show file.

Syntax

[Function] ["Show_Name"] /DMXProtocols

General Keywords

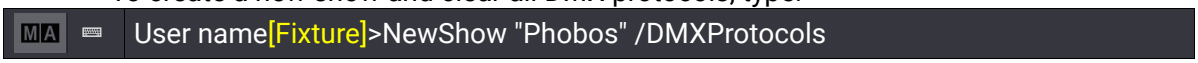
General keywords that use the /DMXProtocols option keyword:

- **LoadShow**
- **NewShow**

Examples

- To load the DMX protocols of the show file "A Midsummer Night's Dream" and its show data, type:

- To create a new show and clear all DMX protocols, type:



1.9.4.20. /Embed

To enter the **/Embed** option keyword in the command line, use one of the options:

- Type **/Embed**
- Type the shortcut **/E** or **/EB**

Description

The /Embed option keyword is used to store presets that are active in the programmer into a preset. The values in the new preset reference the preset that was active in the programmer.

Syntax

Store Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /Embed

General Keywords

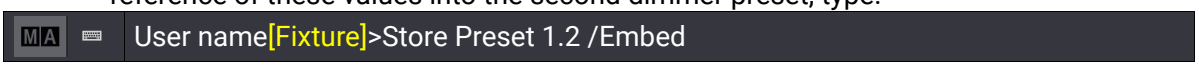
General keywords that use the /Embed option keyword:

- **Preset keyword**
- **Store keyword**

Example

Requirement: Presets are activated in the programmer.

- To store the preset values that are currently active in the programmer together with the reference of these values into the second dimmer preset, type:



1.9.4.21. /Enumerate

To enter the **/Enumerate** option keyword in the command line, use one of the options:

- Type **/Enumerate**
- Type the shortcut **/En**

Description

The /Enumerate option keyword is used to add a count to the name of a show file.

	Important:
	The number is a three-digit number. If the name of the show file along with the enumeration reaches the limit of 31 characters, the excess characters will be cut and replaced by the enumeration instead.

Syntax

SaveShow "Show_Name" /Enumerate

General Keywords

General keywords that use the /Enumerate option keyword:

- **SaveShow keyword**

Example

- To enumerate a show file type:

```
User name[Fixture]>SaveShow /Enumerate
```

1.9.4.22. /File

To enter the **/File** option keyword in the command line, use one of the options:

- Type **/File**
- Type the shortcut **/Fi**

Description

The /File option keyword is used to specify the file name of objects during import or export.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] (If Drive [Drive_Number]) /File ["File_Name"]

General Keywords

General keywords that use the /File option keyword:

- **Export keyword**
- **Import keyword**

Example

- To export the preset file Endor, type:

1.9.4.23. /ForceGlobal

To enter the **/ForceGlobal** option keyword in the command line, use one of the options:

- Type **/ForceGlobal**
- Type the shortcut **/ForceG**

Description

The /ForceGlobal option keyword adds global data to a preset already containing selective data. Selective data which is not used will be removed in the preset of fixtures of the same fixture type. /ForceGlobal removes selective data when updating or storing using the merge option.

Syntax

[Function] Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /ForceGlobal

General Keywords

General keywords that use the /ForceGlobal option keyword:

- **Preset keyword**
- **Store keyword**
- **Update keyword**

Example

- To replace the selective data by global data in all fixtures of the same fixture type in the second dimmer preset, type:

1.9.4.24. /GDTF

To enter the **/GDTF** option keyword in the command line, use one of the options:

- Type **/GDTF**
- Type the shortcut **/GD**

Description

The /GDTF option keyword is used during export of a fixture type that uses the GDTF format.

Syntax

Export FixtureType ["FixtureType_Name" or FixtureType_Number] "filename.gdtf" /GDTF

General Keywords

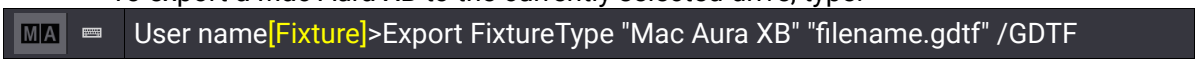
General keywords that use the /GDTF option keyword:

- **Export keyword**

Example

Requirement: Select a drive you would like to export your fixture to.

- To export a Mac Aura XB to the currently selected drive, type:



1.9.4.25. /Gaps

To enter the **/Gaps** option keyword in the command line, use one of the options:

- Type **/Gaps**
- Type the shortcut **/Ga**

Description

The /Gaps option keyword retains or suppresses empty spaces when importing or exporting a range of pool objects.

	**Hint:**
	/Gaps or **/Gaps "Yes"** retains empty spaces when importing or exporting a range of pool objects. **/Gaps "No"** suppresses empty spaces when importing or exporting a range of pool objects.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /Gaps ("Yes" or "No")

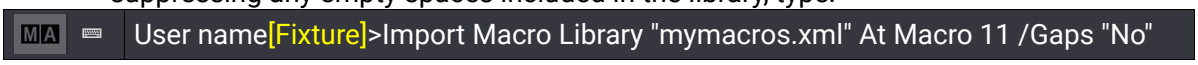
General Keywords

General keywords that use the /Gaps option keyword:

- **Import keyword**
- **Export keyword**

Example

- To import all macros from the library "mymacros.xml," starting with macro 11, while suppressing any empty spaces included in the library, type:



1.9.4.26. /Global

To enter the **/Global** option keyword in the command line, use one of the options:

- Type **/Global**
- Type the shortcut **/G**

Description

The /Global option keyword is used to store or update global data into presets regardless their mode.

Syntax

[Function] Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /Global

General Keywords

General keywords that use the /Global option keyword:

- **Preset keyword**
- **Store keyword**
- **Update keyword**

Example

- To store the current programmer content as global data in the second dimmer preset, type:

1.9.4.27. /GridMergeMode

To enter the **/GridMergeMode** option keyword in the command line, use one of the options:

- Type **/GridMergeMode**
- Type the shortcut **/Gr** or **GMM**

Description

The /GridMergeMode option keyword is used when storing fixtures with their selection grid data into objects using the merge mode.

For more information see **Store Settings and Preferences**.

Syntax

Store [Object] ["Object_Name" or Object_Number] /Merge /GridMergeMode "Value"

General Keywords

General keywords that use the /GridMergeMode option keyword:

- **Cue keyword**
- **Group keyword**
- **Preset keyword**
- **World keyword**

Values

The /GridMergeMode option keyword uses these values:

- AppendX
- Off

Examples

- To merge the currently selected fixtures in group 27 and append the new fixtures in the selection grid on the x-axis following the fixtures that were alrady stored, type:

```
User name[Fixture]>Store Group 27 /GridMergeMode "AppendX"
```

- To merge the currently selected fixtures in group 28 and place them on their original position in the selection grid, type:

```
User name[Fixture]>Store Group 28 /GridMergeMode "Off"
```

1.9.4.28. /Indirect

To enter the **/Indirect** option keyword in the command line, use one of the options:

- Type **/Indirect**
- Type the shortcut **/Ind**

Description

	**Hint:**
	The /Indirect Option keyword can be used with all object types to which you can assign other objects.

The /Indirect option keyword is used to directly edit an assigned object. Executing a command containing this option keyword opens the editor for the object to be modified.

Syntax

Edit [Object] ["Object_Name" or Object_Number] Property ["Property_Name"] /Indirect

General Keywords

General keywords that use the /Indirect option keyword:

- **Edit keyword**
- **Property keyword**

Examples

- To open the appearance editor and edit the appearance that is assigned to group 8, type:

```
User name[Fixture]>Edit Group 8 Property "Appearance" /Indirect
```

- To edit the object that is triggered by the first agenda entry, type:

```
User name[Fixture]>Edit Agenda 1 Property "Object" /Indirect
```

1.9.4.29. /InputFilter

To enter the **/InputFilter** option keyword in the command line, use one of the options:

- Type **/InputFilter**
- Type the shortcut **/I**

Description

The /InputFilter option keyword is used to enable or disable the filter of a feature group when storing or updating presets.

Syntax

[Function] Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /InputFilter ["Option_Value"]

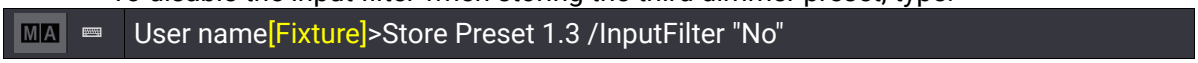
General Keywords

General keywords that use the /InputFilter option keyword:

- **Preset keyword**
- **Store keyword**
- **Update keyword**

Example

- To disable the input filter when storing the third dimmer preset, type:



1.9.4.30. /KeepActivation

To enter the **/KeepActivation** option keyword in the command line, use one of the options:

- Type **/KeepActivation**
- Type the shortcut **/KA** or **/K**

Description

The /KeepActivation option keyword is used to keep the programmer content still active after storing the preset.

Syntax

Store Peset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /KeepActivation

General Keywords

General keywords that use the /KeepActivation option keyword:

- **Preset keyword**
- **Store keyword**

Example

- To store the current programmer content in the third dimmer preset and keep this data active in the programmer, type:

1.9.4.31. /Limit

To enter the **/Limit** option keyword in the command line, use one of the options:

- Type **/Limit**
- Type the shortcut **/Li**

Description

The /Limit option keyword defines how many entries will be displayed in the command line history.

	**Hint:**
	You can use **/Limit** together with the **/Sort option keyword**.

Syntax

MemoryInfo /Limit [Limit_Number]

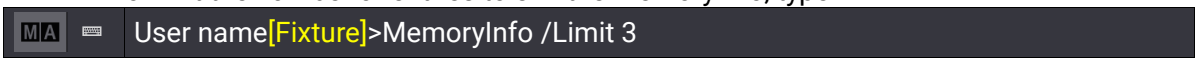
General Keywords

General keywords that use the /Limit option keyword:

- **DumpLog keyword**
- **MemoryInfo keyword**

Examples

- To limit the number of entries to 3 in the MemoryInfo, type:



- To store the last 10 lines of the log file, shown in the system monitor, onto a connected USB drive, type:

1.9.4.32. /LocalSettings

To enter the **/LocalSettings** option keyword in the command line, use one of the options:

- Type **/LocalSettings**
- Type the shortcut **/Loc**

Description

The /LocalSettings option keyword is used to load the local settings of a show file.

Syntax

[Function] ["Show_Name"] /LocalSettings

General Keywords

General keywords that use the /LocalSettings option keyword:

- **LoadShow**
- **NewShow**

Example

- To load local settings together with the show data of the show file "A Midsummer Night's Dream", type:

1.9.4.33. /Look

To enter the **/Look** option keyword in the command line, use one of the options:

- Type **/Look**
- Type the shortcut **/Loo**

Description

The /Look option keyword stores all dimmer values of all fixtures in the show.

If the dimmer value of a fixture is zero, the /Look option keyword only stores dimmer values of the fixture as there is no visible output onstage.

If the dimmer value of a fixture is above zero, the /Look option keyword stores the values of all attributes of the fixture.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /Look

General Keywords

General keywords that use the /Look option keyword:

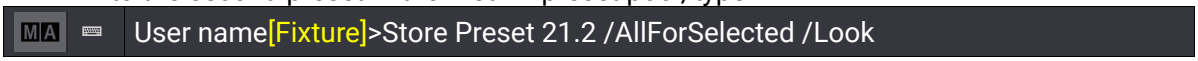
- **Cue keyword**
- **Preset keyword**
- **Store keyword**
- **SetUserVariable keyword**
- **SetGlobalVariable keyword**

	**Hint:**
	The /Look option keyword can be used together with the **/All option keyword** or the **/AllForSelected option keyword**.

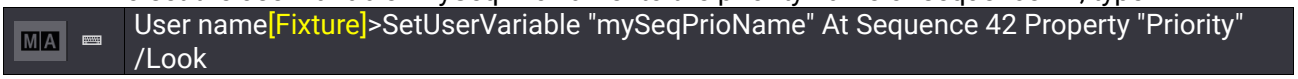
Examples

- To store all dimmer values and all attributes in fixtures, with dimmer open in cue 1, type:

- To store the dimmer values and all active attributes of the selected fixtures in the programmer to the second preset in the first All preset pool, type:



- To set the user variable "mySeqPrioName" to the priority name of sequence 42, type:



1.9.4.34. /MAtricks

To enter the **/MAtricks** option keyword in the command line, use one of the options:

- Type **/MAtricks**
- Type the shortcut **/MA**

Description

The /MAtricks option keyword is used to store the MAtricks settings in the preset whenever the MAtricks are active.

Syntax

Store Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /MAtricks

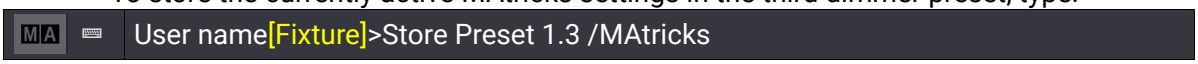
General Keywords

General keywords that use the /MAtricks option keyword:

- **Preset keyword**
- **Store keyword**

Example

- To store the currently active MAtricks settings in the third dimmer preset, type:



1.9.4.35. /Merge

To enter the **/Merge** option keyword in the command line, use one of these options:

- Type /**Merge**
- Type the shortcut **/m**

Description

The /Merge option keyword merges new values with existing values.

The most recent values have a higher priority than the preexisting and will effectively overwrite these.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /Merge

General Keywords

General keywords that use the /Merge option keyword:

- **Assign keyword**
- **AutoCreate keyword**
- **AutoStore keyword**
- **Cook keyword**
- **Copy keyword**
- **Cue keyword**
- **Group keyword**
- **Layout keyword**
- **MAtricks keyword**
- **Preset keyword**
- **Store keyword**
- **World keyword**

Example

- To merge the values of cue 1, type:

1.9.4.36. /MergeHighPriority

To enter the **/MergeHighPriority** option keyword in the command line, use one of these options:

- Type /**MergeHighPriority**
- Type the shortcut **/MergeH**

Description

The /MergeHighPriority option keyword adds all data that is in the source to destination. Data originating in the source overwrite the entire data at destination.

Syntax

[Function] [Source_Object] ["Object_Name" or Object_Number] At [Destination_Object] ["Object_Name" or Object_Number] /MergeHighPriority

General Keywords

General keywords that use the /MergeHighPriority option keyword:

- **Clone keyword**

Example

- To clone fixture 1 to fixture 2 overwriting the entire content of fixture 2 in the programmer, type:

1.9.4.37. /MergeLowPriority

To enter the **/MergeLowPriority** option keyword in the command line, use one of these options:

- Type /**MergeLowPriority**
- Type the shortcut **/MergeL**

Description

The /MergeLowPriority option keyword adds data that is in the source to the destination only where it does not contain data. This option keyword preserves existing data at destination and is the least extreme form of merging.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /MergeLowPriority

General Keywords

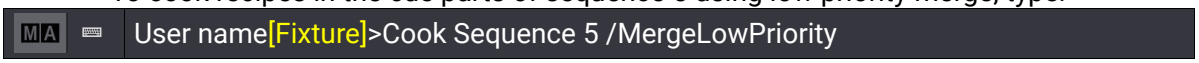
General keywords that use the /MergeLowPriority option keyword:

- **Clone keyword**
- **Cook keyword**

Examples

- To clone data from fixture 1 to fixture 2 as a low priority merge in the programmer, type:

- To cook recipes in the cue parts of sequence 5 using low priority merge, type:



1.9.4.38. /MoveValues

To enter the **/MoveValues** option keyword in the command line, use one of the options:

- Type **/MoveValues**
- Type the shortcut **/MV** or **/Mo**

Description

The /MoveValues option keyword is used to move values to part 0 when deleting cue parts.

Syntax

Delete Cue ["Cue_Name" or Cue_Number] Part ["Part_Name" or Part_Number] /MoveValues

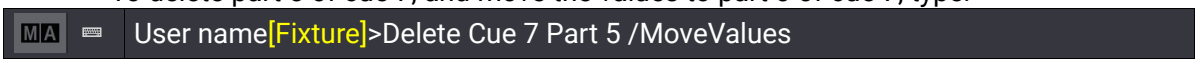
General Keywords

General keywords that use the /MoveValues option keyword:

- **Cue keyword**
- **Delete keyword**
- **Part keyword**

Example

- To delete part 5 of cue 7, and move the values to part 0 of cue 7, type:



1.9.4.39. /NoAutoClose

To enter the **/NoAutoClose** option keyword in the command line, use one of the options:

- Type **/NoAutoClose**
- Type the shortcuts **/NoA**

Description

The /NoAutoClose option keyword is used to suppress the countdown in pop-ups.

Syntax

[Function] (["Object_Name" or Object_Number]) /NoAutoClose

General Keywords

General keywords that use the /NoAutoClose option keyword:

- **ShutDown keyword**

Example

- To shut down the station you are working with at the moment suppressing the countdown in the shutdown pop-up, type:

1.9.4.40. /NoConfirmation

To enter the **/NoConfirmation** option keyword in the command line, use one of the options:

- Type **/NoConfirmation**
- Type the shortcuts **/N** or **/NC**

Description

The /NoConfirmation option keyword is used to suppress the confirmation pop-ups when storing objects or shutting down the station.

Syntax

[Function] ["Object_Name" or Object_Number] /NoConfirmation

General Keywords

General keywords that use the /NoConfirmation option keyword:

- **AutoStore keyword**
- **Delete keyword**
- **Oops keyword**
- **Reboot keyword**
- **Restart keyword**
- **ShutDown keyword**
- **Store keyword**

Examples

- To shut down the station you are working with without provoking a confirmation pop-up, type:

```
User name[Fixture]>Shutdown /NoConfirmation
```

- To store the programmer values as cue 2 without displaying the confirmation pop-up, type:

```
User name[Fixture]>Store Cue 2 /NoConfirmation
```

1.9.4.41. /NDI

To enter the **/NDI** option keyword in the command line, use one of the options:

- Type **/NDI**
- Type the shortcut **/ND**

Description

The /NDI option keyword stores screenshots of NDI streams directly as images.

Syntax

Store Image ["MediaPool_Name" or MediaPool_Number].["Image_Name" or Image_Number] /NDI=[NDI_Number]

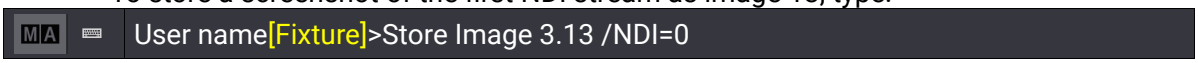
General Keywords

General keywords that use the /NDI option keyword:

- **Store keyword**
- **Image keyword**

Example

- To store a screenshot of the first NDI stream as image 13, type:



1.9.4.42. /NoDependencies

To enter the **/NoDependencies** option keyword in the command line, use one of the options:

- Type **/NoDependencies**
- Type the shortcuts **/NoD**

Description

The /NoDependencies option keyword is used to import or export data without the dependent objects.

Syntax

[Function] ["Object_Name" or Object_Number] /NoDependencies

General Keywords

General keywords that use the /NoDependencies option keyword:

- **Clone keyword**
- **Export keyword**
- **Import keyword**

Examples

- To import the sequence in the file "Mimas.xml" to sequence 4 without dependent objects, type:

```
User name[Fixture]>Import Sequence Library "Mimas.xml" At Sequence 4 /NoDependencies
```

Requirement: Fixture 1, which uses a selective preset, is stored in a cue of sequence 1. Only fixture 1 is part of this preset.

- To clone the data of fixture 1 to fixture 2 and use hard values in sequence 1 in fixture 2 where fixture 1 uses the selective preset, type:

```
User name[Fixture]>Clone Fixture 1 At Fixture 2 If Sequence 1 /NoDependencies
```

1.9.4.43. /NoOops

To enter the **/NoOops** option keyword in the command line, use one of the options:

- Type **/NoOops**
- Type the shortcut **/NU**

Description

The /NoOops option keyword does not generate oops events when executing commands.

Syntax

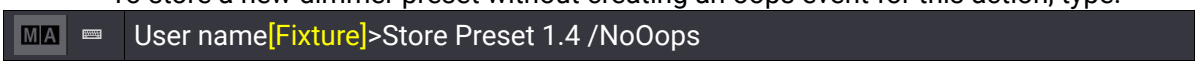
[Command] /NoOops

General Keywords

Almost all of the keywords in the grandMA3 software can be combined with the /NoOops option keyword.

Example

- To store a new dimmer preset without creating an oops event for this action, type:



1.9.4.44. /NoRefresh

To enter the **/NoRefresh** option keyword in the command line, use one of the options:

- Type **/NoRefresh**
- Type the shortcut **/NoR**

Description

The /NoRefresh option keyword is used to suppress the refresh of libraries during import or export of objects.

	**Hint:**
	When listing the library files of a certain type and/or path, it may take a while to type the path and the options into the command line. When a file is to be imported after a type and/or path has been specified during a previous command, the type and/or path will typically need to be entered again as part of the import command. Using the **/NoRefresh** option, it is not necessary to reenter **/Type** and/or **/Path** as part of the import command.

Syntax

[Function] Object ["Object_Name" or Object_Number] (If Drive [Drive_Number]) (/Option) ("/Option_Value")

General Keywords

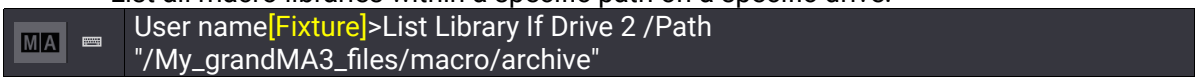
General keywords that use the /NoRefresh option keyword:

- **Export keyword**
- **Import keyword**

Examples

To use the /NoRefresh option to avoid reentering drive and path specifications:

- List all macro libraries within a specific path on a specific drive:



- To import the second library from the list without reentering the drive and path, type:

1.9.4.45. /NoSave

To enter the **/NoSave** option keyword in the command line, use one of the options:

- Type **/NoSave**
- Type the shortcut **/NS**

Description

The /NoSave option keyword defines for the show file not to be saved when loading a show, or shuttig down, restarting, or rebooting the console or your onPC.

	**Important:**
	The /NoSave option keyword only works in the conjunction with the **/NoConfirmation option keyword**.

Syntax

[Function] ([Object] ["Object_Name" or Object_Number]) /NoConfirmation /NoSave

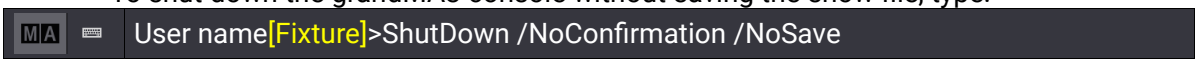
General Keywords

General keywords that use the /NoSave option keyword:

- **LoadShow keyword**
- **Reboot keyword**
- **Restart keyword**
- **ShutDown keyword**

Example

- To shut down the grandMA3 console without saving the show file, type:



1.9.4.46. /NoShowData

To enter the **/NoShowData** option keyword in the command line, use one of the options:

- Type **/NoShowData**
- Type the shortcut **/NoSh**

Description

The /NoShowData option keyword is used to keep current show data when loading shows.

Syntax

[Function] ["Show_Name"] /NoShowData

General Keywords

General keywords that use the /NoShowData option keyword:

- **LoadShow**
- **NewShow**

Example

- To load all data, except the show data of the show file "A Midsummer Night's Dream" and keep the current show data, type:

1.9.4.47. /NoSubfolders

To enter the **/NoSubfolders** option keyword in the command line, use one of the options:

- Type **/NoSubfolders**
- Type the shortcut **/NoS**

Description

The /NoSubfolders option keyword is used to exclude subfolders when importing library files or listing libraries.

	**Hint:**
	Subfolders are created by the users.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /NoSubfolders

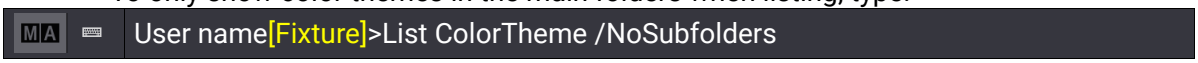
General Keywords

General keywords that use the /NoSubfolders option keyword:

- **List keyword**
- **Import keyword**

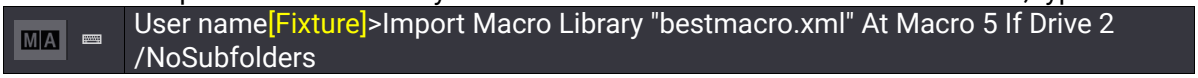
Examples

- To only show color themes in the main folders when listing, type:



It might as well happen that there are two files with the same name located in the main folder and a subfolder.

- To import the macro library file "bestmacro.xml" located in the main folder, type:



1.9.4.48. /OriginalContentOnly

To enter the **/OriginalContentOnly** option keyword in the command line, use one of the options:

- Type **/OriginalContentOnly**
- Type the shortcut **/Or**

Description

The /OriginalContentOnly option keyword updates already existing values of attributes in their original location in presets or cues.

Syntax

Update [Object] ["Object_Name" or Object_Number] /OriginalContentOnly

General Keywords

General keywords that use the /OriginalContentOnly option keyword:

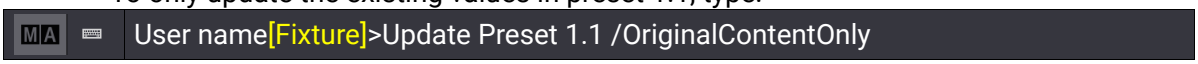
- **Update keyword**

Example

Requirement:

- Create presets or cues
- Playback presets or cues or activate presets in the programmer
- The different attributes must have active values in the programmer

- To only update the existing values in preset 1.1, type:



1.9.4.49. /Output

To enter the **/Output** option keyword in the command line, use one of the options:

- Type **/Output**
- Type the shortcut **/Ou**

Description

The /Output option keyword is used to store incoming output data to cues or presets.

For more information on data sources see **Store Settings and Preferences**.

Syntax

Store [Object] ["Object_Name" or Object_Number] /Output [/AllForSelected or /All]

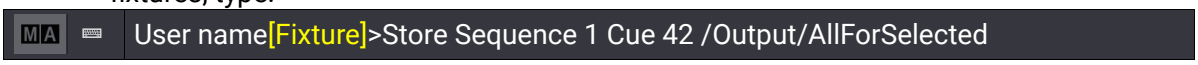
General Keywords

General keywords that use the /Output option keyword:

- **Cue keyword**
- **Preset keyword**
- **Sequence keyword**

Example

- To store the incoming output data to sequence 1 cue 42 in all attributes of the selected fixtures, type:



1.9.4.50. /OutputStations

To enter the **/OutputStations** option keyword in the command line, use one of the options:

- Type **/OutputStations**
- Type the shortcut **/OutputS**

Description

The /OutputStations option keyword is used to overwrite the current output stations with the output stations of the show file that will be loaded.

Syntax

[Function] ["Show_Name"] /OutputStations

General Keywords

General keywords that use the /OutputStations option keyword:

- **LoadShow**
- **NewShow**

Example

- To load the output stations of the show file "A Midsummer Night's Dream" together with its show data, type:

1.9.4.51. /Overwrite

To enter the **/Overwrite** option keyword in the command line, use one of the options:

- Type **/Overwrite**
- Type the shortcut **/O**

Description

The /Overwrite option keyword is used to overwrite existing values.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /Overwrite

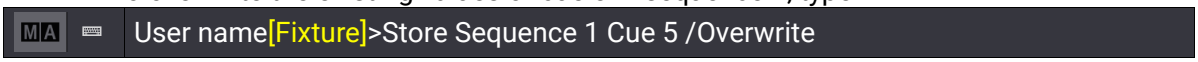
General Keywords

General keywords that use the /Overwrite option keyword:

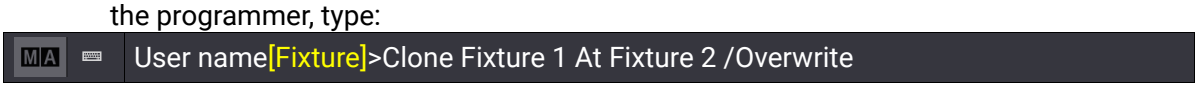
- **AutoStore keyword**
- **Cook keyword**
- **Clone keyword**
- **Cue keyword**
- **Filter keyword**
- **Group keyword**
- **MAtricks keyword**
- **Preset keyword**
- **Sequence keyword**
- **Store keyword**
- **World keyword**

Examples

- To overwrite the existing values of cue 5 in sequence 1, type:



- To clone the programmer data of fixture 1 to fixture 2 and overwrite all the data of fixture 2 in the programmer, type:



1.9.4.52. /PatchOffset

To enter the **/PatchOffset** option keyword in the command line, use one of the options:

- Type **/PatchOffset**
- Type the shortcut **/Patc**

Description

When patching the /PatchOffset option keyword sets the offset of DMX addresses in fixtures. The DMX address has to be specified using breaks of the fixture type.

Syntax

Set (Object) [Object_Number] Property "Break[Number]" ["DMXAddress"] /PatchOffset [PatchOffset_Value]

General Keywords

General keywords that use the /PatchOffset option keyword:

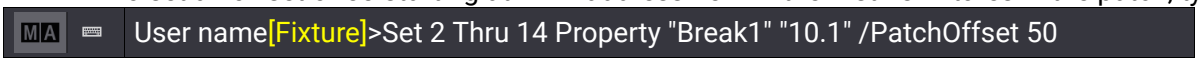
- **Set keyword**

Example

Requirement: Go to the Live Patch

For information on the Live Patch and how to use it in the grandMA3, see **Live Patch**.

- To set an offset of 50 starting at DMX address 10.1 in the first 13 fixtures in the patch, type:



1.9.4.53. /Path

To enter the **/Path** option keyword in the command line, use one of the options:

- Type **/Path**
- Type the shortcut **/Pa**

Description

The /Path option keyword defines the folder path where an imported or exported file is saved.

	Hint:
	-Enter a path beginning with a letter or number if the path is incorporated with the default folder structure. -Enter a path beginning with the forward-slash (/) character if the path begins at the root of the device. -Entering a path that does not already exist creates the necessary folders.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] (If Drive [Drive_Number]) /Path "The/Path/To/My/Files"

General Keywords

General keywords that use the /Path option keyword:

- **Export keyword**
- **Import keyword**

Example

- To export macro 1, with the name "test," to a folder labeled "myfavorites" at the root of the first connected USB drive, type:

```
User name[Fixture]>Export Macro 1 "test" If Drive 2 /Path "/myfavorites"
```

1.9.4.54. /PhaserData

To enter the **/PhaserData** option keyword in the command line, use one of the options:

- Type **/PhaserData**
- Type the shortcut **/Ph**

Description

The /PhaserData option keyword is used to define if attribute data will be stored in presets.

Syntax

Store Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /PhaserData ["Option_Value"]

General Keywords

General keywords that use the /PhaserData option keyword:

- **Preset keyword**
- **Store keyword**

Example

Requirement:

1. Select 10 fixtures.
2. Set XWings to 2.
3. Type in the command line:

- To store all data, except the active programmer data, into the first preset of the third All preset pool, type:

1.9.4.55. /Programmer

To enter the **/Programmer** option keyword in the command line, use one of the options:

- Type **/Programmer**
- Type the shortcut **/P**

Description

The /Programmer option keyword is used to store active programmer data to cues or presets.

For more information on data sources see **Store Settings and Preferences**.

Syntax

Store [Object] ["Object_Name" or Object_Number] /Programmer

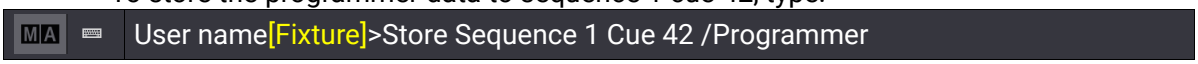
General Keywords

General keywords that use the /Programmer option keyword:

- **Cue keyword**
- **Preset keyword**
- **Sequence keyword**

Example

- To store the programmer data to sequence 1 cue 42, type:



1.9.4.56. /Release

To enter the **/Release** option keyword in the command line, use one of the options:

- Type **/Release**
- Type the shortcut **/Rel**

Description

The /Release option keyword is used when copying cues. It replaces non-existing data in the destination cue by released values.

Syntax

Copy Cue ["Cue_Name" or Cue_Number] /Release

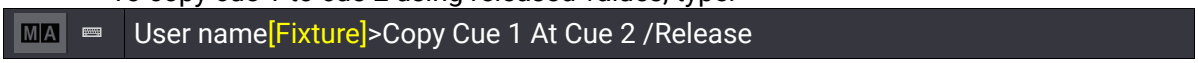
General Keywords

General keywords that use the /Release option keyword:

- **Copy keyword**
- **Cue keyword**

Example

- To copy cue 1 to cue 2 using released values, type:



1.9.4.57. /Remove

To enter the **/Remove** option keyword in the command line, use one of the options:

- Type **/Remove**
- Type the shortcut **/R**

Description

The /Remove option keyword is used to remove values that are stored in attributes and active programmer values.

Syntax

[Function] ["Object_Name" or Object_Number] /Remove

General Keywords

General keywords that use the /Remove option keyword:

- **Assign keyword**
- **Cook keyword**
- **CopyCrashLog**
- **Cue keyword**
- **Group keyword**
- **Layout keyword**
- **MAtricks keyword**
- **Preset keyword**
- **Store keyword**
- **Update keyword**
- **World keyword**

Examples

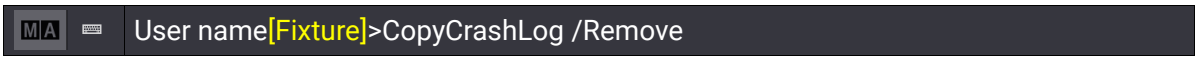
- To remove the values that are active in the programmer and that are stored in cue 5 of sequence 1, type:

```
User name[Fixture]>Store Sequence 1 Cue 5 /Remove
```

- To remove all cooked data from sequence 1, type:

```
User name[Fixture]>Cook Sequence 1 /Remove
```

- To delete crash logs on your device after copying them to a USB drive, type:



1.9.4.58. /Recipe

To enter the **/Recipe** option keyword in the command line, use one of the options:

- Type **/Recipe**
- Type the shortcut **/Rec**

Description

The /Recipe option keyword can only be used together with the **CleanUp keyword** and only if the **/Type option keyword** is set to **Recipe**.

Using /Recipe allows you to define more precisely which recipes are to be deleted.

Syntax

CleanUp [Object] ["Object_Name" or Object_Number] /Type "Recipe" (/Recipe "Recipe_Value")

General Keywords and Option Keywords

General keywords and option keywords that use the /Recipe option keyword.

- **CleanUp keyword**
- **/Type option keyword**

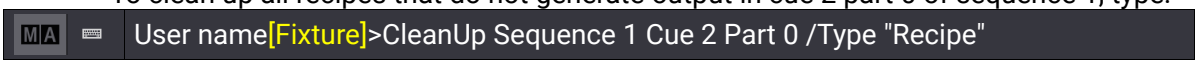
Values

The /Recipe option keyword uses these values:

- **NoOutput:**
  Recipes that do not generate output are deleted. This combines the following values – NotCooked and CookedButOverwritten. When specifying /Type "Recipe" and not using /Recipe in addition, NoOutput will be applied.
- **NotCooked:**
  If the assigned preset cannot be used by the selection or if the assigned group is empty, **NotCooked** will remove such non-functional recipes.
- **CookedButOverwritten:**
  If a later recipe uses a preset with the values of the same attributes in the same selection, **CookedButOverwritten** will delete all recipes that could have been cooked successfully, but which do not generate output.

Examples

- To clean up all recipes that do not generate output in cue 2 part 0 of sequence 1, type:



or:

```
User name[Fixture]>CleanUp Sequence 1 Cue 2 Part 0 /Type "Recipe" /Recipe "NoOutput"
```

- To clean up the recipes in the second position preset that could not be cooked, type:

```
User name[Fixture]>CleanUp Preset 2.2 /Type "Recipe" /Recipe "NotCooked"
```

1.9.4.59. /Save

To enter the **/Save** option keyword in the command line, use one of the options:

- Type **/Save**
- Type the shortcut **/S**

Description

The /Save option keyword defines for the show file to be saved when loading a show, shuttig down, restarting, or rebooting the console or your onPC.

	**Important:**
	The /Save option keyword only works in the conjunction with the **/NoConfirmation option keyword**.

Syntax

[Function] ([Object] ["Object_Name" or Object_Number]) /NoConfirmation /Save

General Keywords

General keywords that use the /Save option keyword:

- **LoadShow keyword**
- **Reboot keyword**
- **Restart keyword**
- **ShutDown keyword**

Example

- To shut down the grandMA3 console and saving the current show file suppressing the confirmation pop-up, type:

1.9.4.60. /Screen

To enter the **/Screen** option keyword in the command line, use one of the options:

- Type **/Screen**
- Type the shortcut **/S**

Description

The /Screen option keyword addresses screens when storing or calling views. It can be used in conjunction with the **/ScreenOnly option keyword**.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /Screen

General Keywords

General keywords that use the /Screen option keyword:

- **Call keyword**
- **Store keyword**
- **View keyword**
- **ViewButton keyword**

Examples

- To store screen 1 on the view button 1, type:

```
User name[Fixture]>Store View 1 /Screen "1"
```

- To call view 5 on screen 2, type:

```
User name[Fixture]>View 5 /Screen "2"
```

- To call the view which is assigned to view button 1, screen 2 on screen 3, type:

```
User name[Fixture]>ViewButton 2.1 /Screen "3"
```

1.9.4.61. /ScreenOnly

To enter the **/ScreenOnly** option keyword in the command line, use one of the options:

- Type **/Screen**
- Type the shortcut **/ScreenO**

Description

The /ScreenOny option keyword defines which parts of the screen will be used when taking screenshots. It is used in conjunction with the **/Screen option keyword**, the **/XResolution option keyword** and the **/YResolution option keyword**.

Syntax

Store Image ["MediaPool_Name" or MediaPool_Number].["Image_Name" or Image_Number] /ScreenOnly ["Value"]

General Keywords

General keywords that use the /ScreenOnly option keyword:

- **Image keyword**
- **Store keyword**

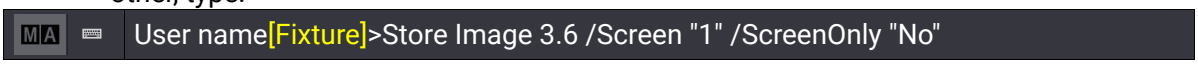
Values

The /ScreenOnly option keyword uses these values:

- Yes - This is the default if /ScreenOnly is not defined. The screenshot only includes the area that can be defined by the user.
- No - The entire screen will be used, including view bar, encoder bar and other.

Example

- To store a screenshot of the entire screen 1 as image 6, including view bar, encoder bar, and other, type:



1.9.4.62. /Selection

To enter the **/Selection** option keyword in the command line, use one of the options:

- Type **/Selection**
- Type the shortcut **/Selectio**

Description

The /Selection option keyword defines if the current selection will be stored in a recipe.

For more informtion on how recipes work see **Recipes**.

Syntax

Store Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number].["Recipe_Name" or Recipe_Number] /Selection ["Option_Value"]

Store Sequence ["Sequence_Name" or Sequence_Number] Cue ["Cue_Name" or Cue_Number] Part ["Part_Name" or Part_Number].["Recipe_Name" or Recipe_Number] /Selection ["Option_Value"]

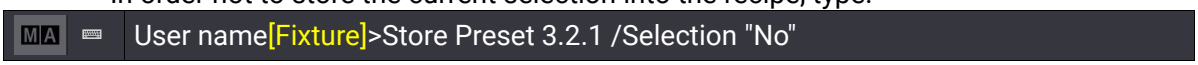
General Keywords

General keywords that use the /Selection option keyword:

- **Preset keyword**
- **Sequence keyword**
- **Store keyword**

Example

- In order not to store the current selection into the recipe, type:



1.9.4.63. /Selective

To enter the **/Selective** option keyword in the command line, use one of the options:

- Type **/Selective**
- Type the shortcut **/Se**

Description

The /Selective option keyword is used to store or update selective data into presets regardless their mode.

Syntax

[Function] Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /Selective

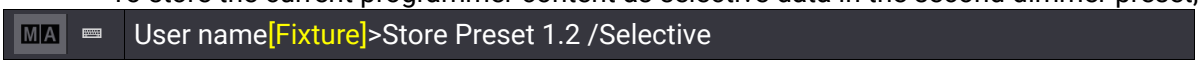
General Keywords

General keywords that use the /Selective option keyword:

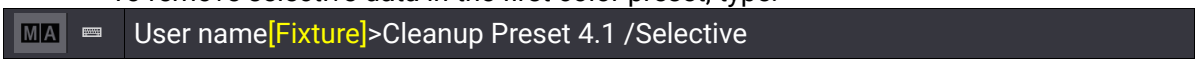
- **CleanUp keyword**
- **Preset keyword**
- **Store keyword**
- **Update keyword**

Examples

- To store the current programmer content as selective data in the second dimmer preset, type:



- To remove selective data in the first color preset, type:



1.9.4.64. /Single

To enter the **/Single** option keyword in the command line, use one of the options:

- Type **/Single**
- Type the shortcut **/Single**

Description

The /Single option keyword is used when extracting presets or when autocreating single fixture groups.

When extracting embedded presets or phaser presets where presets are integrated in the steps, Extract will call directly the values of the source presets.

Using the **/Single** option keyword together with the Extract keyword makes it possible to extract one level down in the hierarchy of presets.

	Important:
	The combination of the Extract keyword together with the /Single option keyword currently works with presets that are active in the programmer.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] /Single

General Keywords

General keywords that use the /Single option keyword:

- **AutoCreate keyword**
- **Extract keyword**
- **FixtureClass keyword**
- **FixtureLayer keyword**
- **FixtureType keyword**

Examples

Extract Example

Requirement:

1. Create 2 color presets.
2. Embed these color presets into a different color preset:

   Embed the second color preset into the All preset 21.1.
3. Select fixtures and apply the All preset on them.
   - To call the embedded color preset into the programmer, type once:

User name[Fixture]>Extract Selection /Single

- To activate hard values, type a second time:

User name[Fixture]>Extract Selection /Single

or:

User name[Fixture]>Extract Selection

AutoCreate and FixtureType Example

Requirement:

- Insert at least ten fixture types in the show file and patch at least two fixtures of every fixture type.

- To create single fixture groups starting with group pool object 42 that contain all patched fixtures of the fixture type 9, type:

User name[Fixture]>AutoCreate FixtureType 9 At Group 42 /Single

- To create single fixture groups that contain all patched fixtures of fixture types 9 and 10 starting with group pool object 101, type:

User name[Fixture]>AutoCreate FixtureType 9 + 10 At Group 101 /Single

AutoCreate FixtureLayer and FixtureClass Example

Requirement:

1. Create layers and classes within the patch.
2. Set fixtures to these layers and classes.
   - To create single fixture groups starting with group pool object 201 using all patched fixtures that are set to the layer "Backtruss" within the patch, type:

User name[Fixture]>AutoCreate FixtureLayer "Backtruss" At Group 201 /Single

- To create single fixture groups starting at pool object 301 with all fixtures that are set to class "Spots" within the patch, type:

User name[Fixture]>AutoCreate FixtureClass "Spots" At Group 301/Single

1.9.4.65. /Sort

To enter the **/Sort** option keyword in the command line, use one of the options:

- Type **/Sort**
- Type the shortcut **/So**

Description

The /Sort option keyword defines how to sort results in the command line history.

	**Hint:**
	You can use **/Sort** together with the **/Limit option keyword**.

Use the following sort names to define the order you are using /Sort with:

Sort Name	Description
Asc	Sorts results in an ascending order.
Desc	Sorts results in an descending order.
None	Sorts results in the order of destination.

Syntax

MemoryInfo /Sort ["Sort_Name"]

General Keywords

General keywords that use the /Sort option keyword:

- **MemoryInfo keyword**

Example

- To display the 5 objects that are occupying the most of memory, type:

MA	User name[Fixture]>MemoryInfo /Sort "Desc" /Limit 5

1.9.4.66. /SubTab

To enter the **/SubTab** option keyword in the command line, use one of the options:

- Type **/SubTab**
- Type the shortcut **/Su**

Description

The /SubTab option keyword is used to switch between tabs within tab bars in the PSR menu.

	**Important:**
	Currently the /SubTab option keyword is solely used by the PSR menu itself. As there is a defined order how to navigate in the PSR menu, only the PSR menu can use this option keyword and you cannot execute the command using the command line.

1.9.4.67. /Tab

To enter the **/Tab** option keyword in the command line, use one of the options:

- Type **/Tab**
- Type the shortcut **/Ta**

Description

The /Tab option keyword is used to define which tab will open in the assign menu.

Syntax

[Function] Page ["Page_Name" or Page_Number].["Executor_Name" or Executor_Number] /Tab ["Option_Value"]

General Keywords

General keywords that use the /Tab option keyword:

- **Assign keyword**
- **Edit keyword**
- **EditSetting keyword**
- **Page keyword**

Values

The /Tab option keyword uses these values:

- Edit
- EditSetting
- Handle
- Object

Examples

- To open the edit tab of executor 201 on page 1 using the assign command, type:

```
User name[Fixture]>Assign Page 1.201 /Tab "Edit"
```

- To open the assign menu and display the handle tab using the edit command, type:

```
User name[Fixture]>Edit Page 1.201 /Tab "Handle"
```

1.9.4.68. /TrackingShield

To enter the **/TrackingShield** option keyword in the command line, use one of the options:

- Type **/TrackingShield**
- Type the shortcut **/T** or **/TS**

Description

The /TrackingShield option keyword is used to protect tracked attributes when storing values.

For more information see **What is Tracking** and **Tracking Shield**.

Syntax

[Function] Cue ["Cue_Name" or Cue_Number] /TrackingShield ["TrackingShield_Value"]

General Keywords

General keywords that use the /TrackingShield option keyword:

- **Cue keyword**
- **Store keyword**
- **Update keyword**

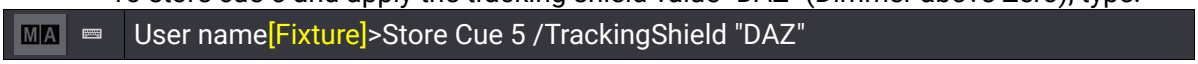
Values

The /TrackingShield option keyword uses these values:

- DRZ or DimmerRisingFromZero
- DAZ or DimmerAboveZero

Example

- To store cue 5 and apply the tracking shield value "DAZ" (Dimmer above Zero), type:



1.9.4.69. /Type

To enter the **/Type** option keyword in the command line, use one of the options:

- Type **/Type**
- Type the shortcut **/Ty**

Description

The /Type option keyword can have different values depending on the keyword it is combined with.

The Import keyword and the Export keyword can be both used with the /Type "User" and "System".
The LoadShow keyword can be used with the /Type "Demo".
In a nutshell - the Import and the Export keyword both use library files and the LoadShow keyword uses show files.

When importing the desired file may be either a "User" file, which has been previously exported, or a "System" file, which is predefined and included with the system software. If this option is not defined within the import syntax, the console will first search the user library for the specified file name. If the file does not exist within the user library, the console will then search within the system files.

- **/Type "User"** restricts the console to only search within the user library of the selected drive.
- **/Type "System"** restricts the console to only search within the system files, ignoring the user library.
- **/Type "Demo"** restricts the console to only search within demo shows folder.
- **/Type "Template"** restricts the console to only search within the template shows folder.
- **/Type "NoReference"** deletes all objects that do not have any reference in the specified range. For example, "NoReference" will delete presets that are not used in cues or recipes.
- **/Type "Recipe"** deletes recipes in the specified object that do not generate output.

	**Important:**
	When no type is specified, the type "User" has priority.

Syntax

[Function] [Object] ["Object_Name" or Object_Number] (If Drive [Drive_Number]) /Type "Value"

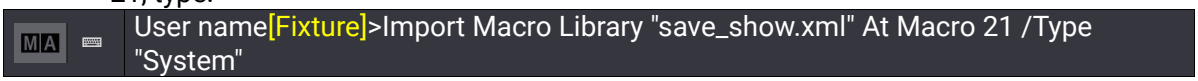
General Keywords and Option Keywords

General keywords and option keywords that use the /Type option keyword:

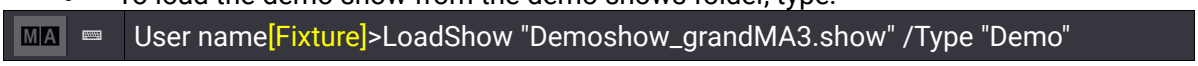
- **CleanUp keyword**
- **Export keyword**
- **Import keyword**
- **LoadShow keyword**
- **/Recipe option keyword**

Examples

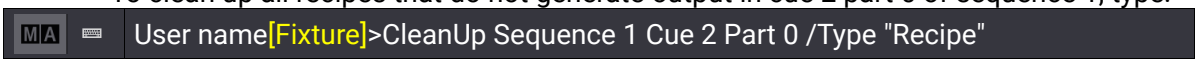
- To import the save_show macro from the system library instead of the user library to macro 21, type:



- To load the demo show from the demo shows folder, type:



- To clean up all recipes that do not generate output in cue 2 part 0 of sequence 1, type:



- To clean up all presets that are not used in any other object, type:

or type:

1.9.4.70. /Universal

To enter the **/Universal** option keyword in the command line, use one of the options:

- Type **/Universal**
- Type the shortcut **/U**

Description

The /Universal option keyword is used to store or update universal data into presets regardless their mode.

Syntax

[Function] Preset ["FeatureGroup_Name" or FeatureGroup_Number].["Preset_Name" or Preset_Number] /Universal

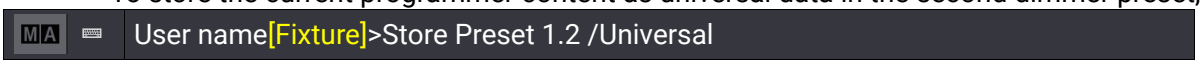
General Keywords

General keywords that use the /Universal option keyword:

- **Preset keyword**
- **Store keyword**
- **Update keyword**

Example

- To store the current programmer content as universal data in the second dimmer preset, type:



1.9.4.71. /Wait

To enter the **/Wait** option keyword in the command line, use one of the options:

- Type **/Wait**
- Type the shortcut **/Wa**

Description

In conjunction with the Store keyword /Wait option keyword defines the latency time in milli seconds after which your command will be executed. When used with remote calls such as reboot, restart or shut down, the /Wait option keyword defines the time in the countdown of the the pop-up that consequently appears after executing the command.

Syntax

[Function] /Wait [LatencyTime]

General Keywords

General keywords that use the /Wait option keyword:

- **Store keyword**
- **Reboot keyword**
- **Restart keyword**
- **ShutDown keyword**

	**Hint:**
	/Wait works with all function keywords.

Examples

- To delay the Replace this syntax text **Store Cue 5** command by 3 seconds, type:

- To set the countdown of the ShutDown pop-up to 20 seconds, type:

1.9.4.72. /Wrap

To enter the **/Wrap** option keyword in the command line, use one of the options:

- Type **/Wrap**
- Type the shortcut **/W**

Description

The /Wrap option keyword loops through all steps and selects the first step after reaching the last step.

This option keyword works in both directions using Next and Previous.

Syntax

[Function] Step /Wrap

General Keywords

General keywords that use the /Wrap option keyword:

- **Next keyword**
- **Previous keyword**
- **Step keyword**

Example

- To loop through all steps in the selected fixtures and subsequently select the first step in the selection, type:

1.9.4.73. /XResolution

To enter the **/XResolution** option keyword in the command line, use one of the options:

- Type **/XResolution**
- Type the shortcut **/X**

Description

The /XResolution option keyword defines the resolution of images on the x-axis during storing.

Syntax

Store Image ["MediaPool_Name" or MediaPool_Number].["Image_Name" or Image_Number] /XResolution ["XResolution_Value"]

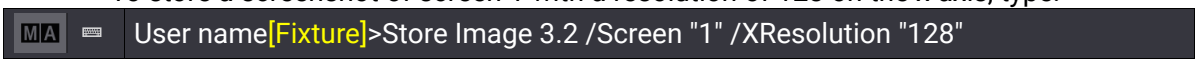
General Keywords

General keywords that use the /XResolution option keyword:

- **Store keyword**

Example

- To store a screenshot of screen 1 with a resolution of 128 on the x-axis, type:



1.9.4.74. /YResolution

To enter the **/YResolution** option keyword in the command line, use one of the options:

- Type **/YResolution**
- Type the shortcut **/Y**

Description

The /YResolution option keyword defines the resolution of images on the y-axis during storing.

Syntax

Store Image ["MediaPool_Name" or MediaPool_Number].["Image_Name" or Image_Number] /YResolution ["YResolution_Value"]

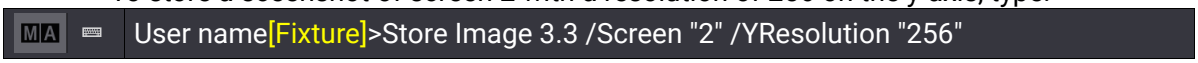
General Keywords

General keywords that use the /YResolution option keyword:

- **Store keyword**

Example

- To store a sceenshot of screen 2 with a resolution of 256 on the y-axis, type:



1.9.5. Internal Keywords

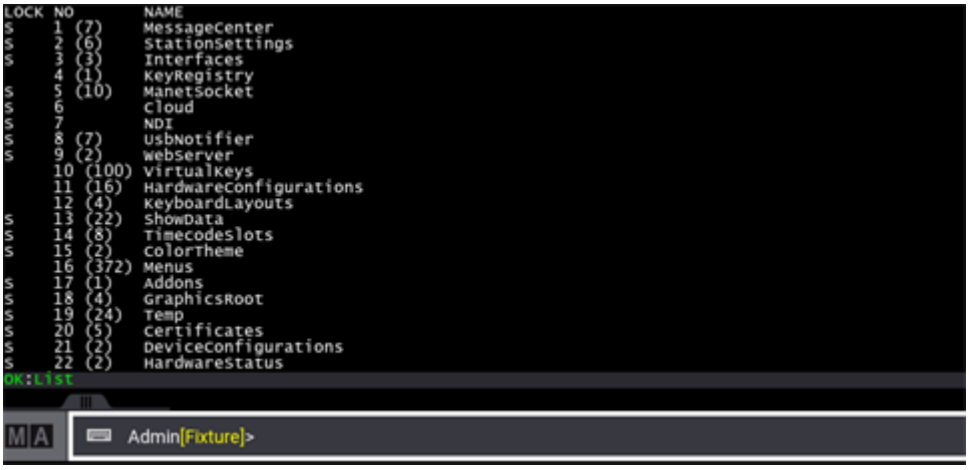
Internal keywords are keywords that you may encounter in the command line history. However, they are designed for internal purposes only and are not intended for user interaction.

1.9.6. Extended Command Line Syntax Options

To control the input in the command line use the **List keyword** any time.

For more information about the command line and the Command Line History, read the **command line topic**.

To list all available subfolders, type:



List view in Command Line History

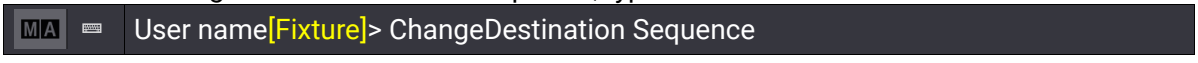
	Hint:
	When the List command returns multiple objects of different types, the settings for those objects are not displayed. To view the settings of a specific object, include the name or number of the object with the List keyword. For more information, see **List keyword** topic.

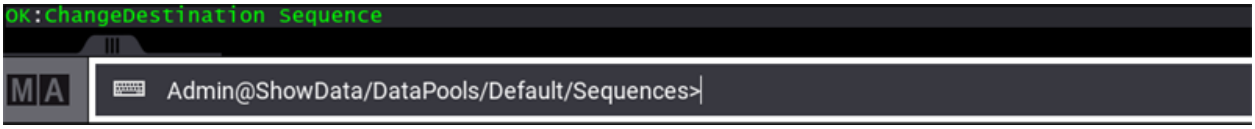
Examples

Change the Value of a Sequence Property

The included data is categorized in directories. In order to change a directory, use the command "Change Destination".

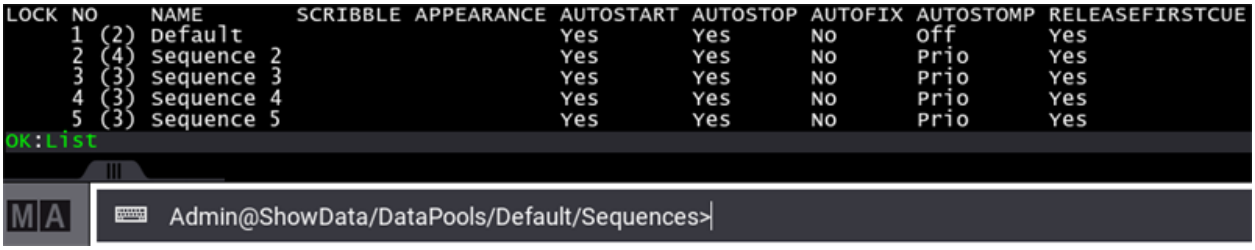
- To change the destination to Sequence, type:





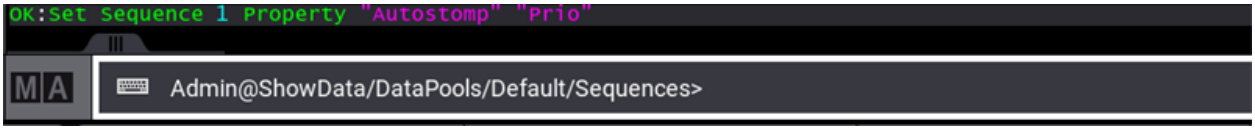
- To display the options in the command line feedback, type:

User name[Fixture]> List



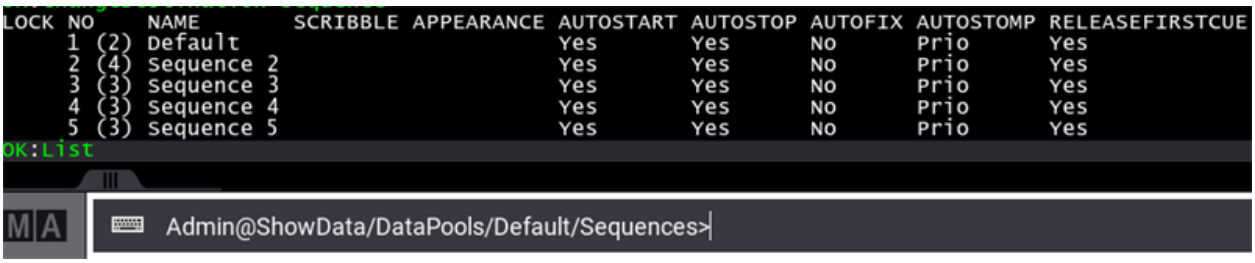
- To change Autostomp from Off to Prio, type:

User name[Fixture]> Set Sequence 1 Property "Autostomp" "Prio"



- To doublecheck the options in the command line feedback, type:

User name[Fixture]> List



Autostomp is now set to Prio.

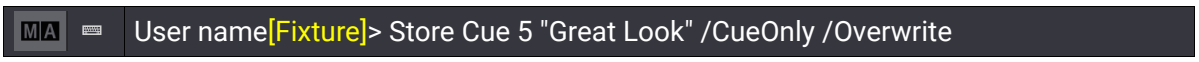
- To return to the destination root, type:

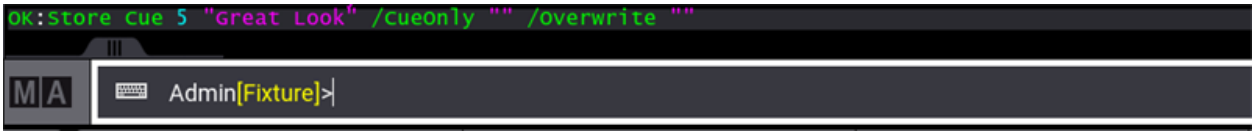
User name[Fixture]> ChangeDestination Root

Store a Cue

Using the command line enables to combine a keyword with different options.

To overwrite cue 5 with the current programmer content and rename the cue to "Great Look" and storing the data as cue only, type:





1.10. Windows, Views, and Menus

Windows are created on the **Screens**. The screens are the monitors. The different sizes of grandMA3 hardware have different amounts of screens. The grandMA3 onPC has quick access to open 5 displays with a large screen area and 2 displays with a small screen area. This offers access to the same primary screens as a grandMA3 full-size. For more information, see **Screen Allocation**.

There are several possible **Screen Configurations** for each user profile. A user always has a screen configuration assigned. Each screen configuration stores a set of view button assignments and the current window set up on the screens. The screen configurations can be selected in the **User menu**.

Most screens are empty in a new show. An empty screen is a blank space where each user can create their own arrangement of windows. This arrangement can be stored as a **View**.

On the right side or top of each screen, there can be a number of **View Buttons**. Views can be assigned to these buttons. This allows for fast access to recall a stored view or update a view by storing it again.

A new show has six factory-made views assigned to the first six view buttons on each of the big screens. The two smaller screens have four and two views assigned. These can be changed or deleted.

A **Menu** is a big pop-up that covers most of the screen. The software has several menus that give access to setting up the console, system, fixtures, and much more. For more information, see **Menus**.

Subtopics

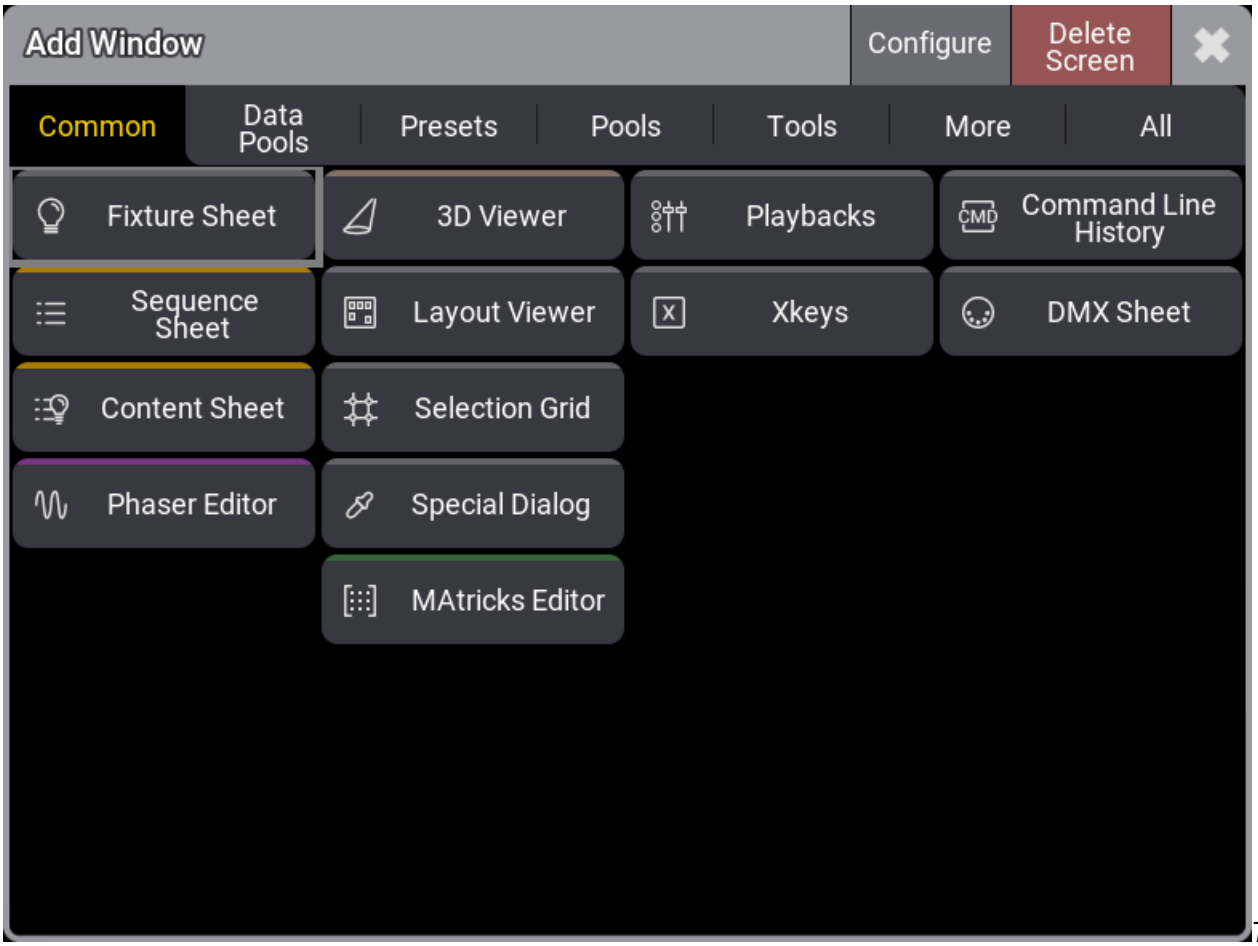
1.10.1. Add Window

Windows are added to the **user definable areas** on the screens.

The best way to add a window is by tapping inside an empty screen area. The window is added with the new window's upper left corner, where the screen is tapped.

Another option is to tap and draw a square on the empty screen to define the window size.

Tap the screen to open the **Add Window** pop-up:



Tap a button with the desired window to create it

Windows are divided into tabs: **Common**, **Data Pools, Presets, Pools, Tools**, **More**, and **All**. Tapping the different tabs reveals buttons with windows for each section.

Tap the desired button to create the window. This creates a new window. By default, the first stored preferences of a window are loaded. This applies to all windows except pool windows.

The pop-up's title bar has two buttons. The red Delete Screen button removes all windows from the screen.

The Configure button opens the **Configure Display** pop-up. This can be used to customize some settings regarding tools on the side of the display and change the size of the user area. Read more in the **Configurations of Displays topic**.

List of current windows that can be created using the **Add Window** pop-up:

Common

A fixed selection of commonly used windows. (These windows are also represented in other tabs):

- Fixture Sheet
- Sequence Sheet
- Content Sheet
- Phaser Editor
- 3D Viewer
- Layout Viewer
- Selection Grid
- Special Dialog
- MAtricks Editor
- Playbacks
- Xkeys
- Command Line History
- DMX Sheet

Data Pools

All object pools that are part of a data pool:

- **Bitmaps**
- **Executor Configurations**
- **Filters**
- **Generators**
- **Groups**
- **Layouts**
- **Macros**
- **MAtricks**
- **Pages**
- **Plugins**
- **Quickeys**
- **Sequences**
- **Timecodes**
- **Timers**
- **Worlds**

Presets

All preset pools. (**One link to all about presets**):

- Dynamic
- All 1
- All 2
- All 3
- All 4
- All 5
- Dimmer
- Position
- Gobo
- Color
- Beam
- Focus
- Control
- Shapers
- Video

Pools

All pools that are part of the show data or the user profiles:

Show Data:

- **Appearances**
- **Data Pools**
- **Gels**
- **Scribbles**
- **Tags**
- **Timecode Slots**
- **Universes**
- **Users**

Media:

- **Gobos**
- **Images**
- **Meshes**
- **Sounds**
- **Symbols**
- **Videos**

User Profile:

- **Cameras**
- **Encoder Bars**
- **Render Qualities**
- **Views**

Tools

General windows for displaying show data:

Sheets:

- **Content Sheet**
- **DMX Sheet**
- **Fixture Sheet**
- **Sequence Sheet**

Programmer Tools:

- **At Filter**
- **Recipe Editor**
- **Selection Grid**
- **sMArt**
- **Special Dialog**

Viewers and Editors:

- **3D Viewer**
- **Agenda Viewer**
- **Clock Viewer**
- **Layout Viewer**
- **MAtricks Editor**
- **Phaser Editor**
- **RDM Devices Viewer**
- **Sound Viewer**
- **Timecode Viewer**

More

All different bars, playback windows, or info and system-related windows:

Bars:

- **Align Bar**
- **command wing Bar**
- **Encoder Bar**
- **Selection Bar**
- **Step Bar**

Playback:

- **Custom Master Section**
- **Running Playbacks**
- **Playback**
- **Xkeys**

Info and System:

- **Command Line History**
- **Help**
- **Info**
- **Message Center**

- **System Info**
- **System Monitor**
- **Trackpad**

All

The **All** tab contains all windows in an alphabetical list that can be filtered and sorted.

1.10.2. Rearrange Windows

Tapping a free space on a screen opens the **Add Window** pop-up. The tapped grid field is used as the upper left corner of the created window. From here, the window takes all the available space it can and becomes as big as possible, trying to fill all available space on the screen.

Tap, hold, and drag on an empty screen to draw a square. Releasing the screen opens the **Add Window** pop-up, and the selected window fills the drawn square.

The window size can often be made smaller.

In the lower right corner of each window, there is a small resize corner: 

Tap or click, and holding this corner allows it to be moved. Moving it resizes the window.

Windows can also be resized by tapping and holding in the title bar while tapping another location on the screen. The window will fill the available space.

Double tap the title bar to resize the window to the largest possible size on the screen. In pool windows double tap on the title field below the MA logo.

Windows can also be moved or resized by moving the entire window. Tap or click and hold the title bar while moving it on the screen. It will show the grid, and the window can be dragged and dropped to a new position.

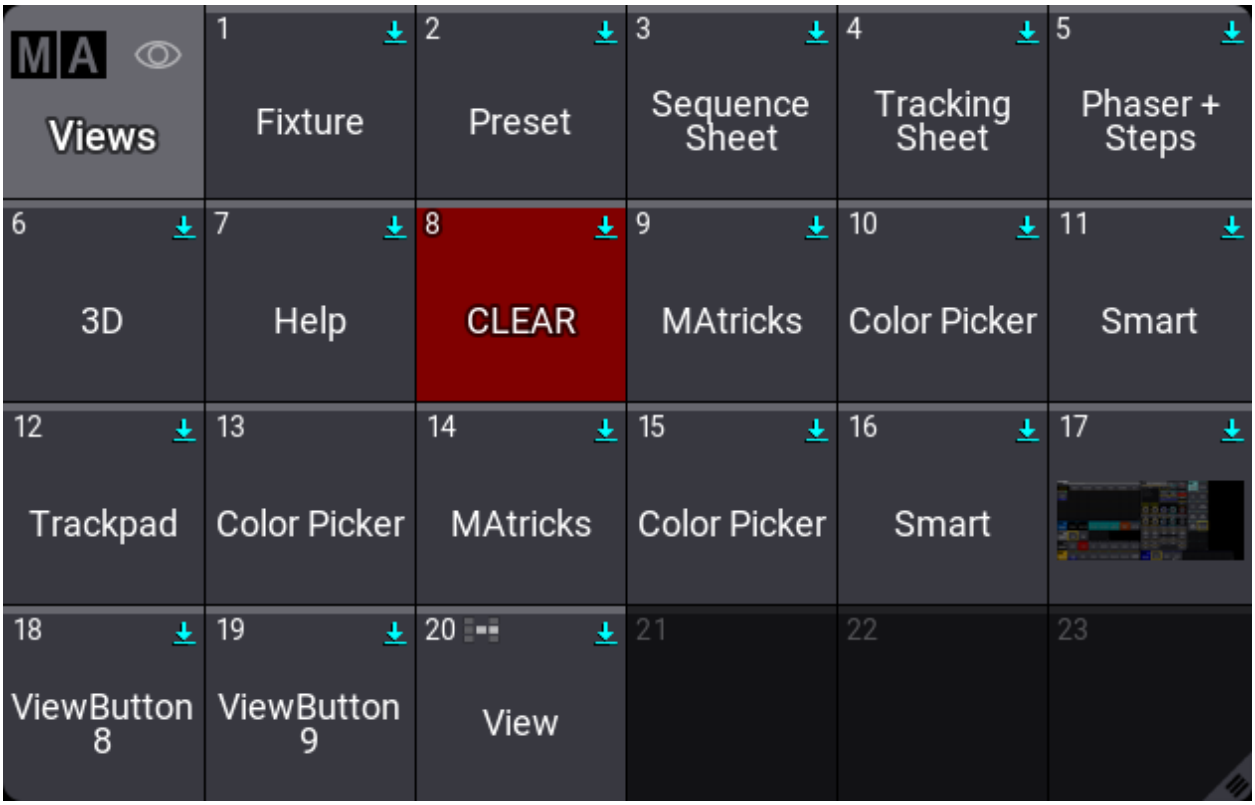
Windows cannot overlap each other, so they can only be as big as the free space allows. If a window is moved up against another window, it will resize.

The windows snap to a dotted grid. This grid is used when windows are created, resized, and moved. There are half-size grid dots, but pools and preset windows are limited to a full grid size when resized.

Some windows have a minimum size limit. For instance, a Fixture Sheet window needs to be a minimum of 3 x 2 grid fields.

1.10.3. Store and Recall Views

The arrangement of windows is called a **View**. Views are stored in a **View Pool**. The pool can be created as a window.



View pool window

Views can be assigned to **View Buttons** and **Executors**.

ViewButtons are located in the View Bar. The view bar is usually located on the right side of screens. View bars can be shown or hidden and they can be on the right side or the top of screens. Learn more in the **View Bar and View Button Topic**.

Executors are handles used to control other objects. They are often used to control sequences, but they can control other objects. For instance, calling different views. Learn more in the **Executor topic**.

Store Views

A view is stored using the standard store syntax and the **View keyword**. This means it is possible to use the keys, keys in combination with buttons on the screens, or the command line.

Storing a view opens the **Store View Options** pop-up.

Select what screens to store in a view.

This is used to directly label the view and select which screens should be stored in the view. Each screen has a button that can be turned On or Off. The last touched screen is selected as a default. There are two buttons at the bottom that give fast access to select All or None of the screens. Tap OK to store the window arrangement of the selected screens or press Please to confirm the options.

It is also possible to store a screenshot of the view if a single screen is selected. This can automatically be created by toggling the screenshot button ( ). Activating the screenshot function automatically creates an image, which is assigned to an appearance, which in turn is assigned to the view pool object. It also removes the text label. A label can be added again. This will be on top of the image. View number 17, in the image above, has a screenshot.

If a view has stored information about two or more screens, then there is an icon on the view pool object indicating the relevant screens in a brighter color than the others in a grid matching the screen grid in the Store View Options pop-up. View number 20, in the image above, has stored windows for Internal 1 and 2.

If the command line is used, the screens can be specified using the **/Screen option keyword**. For more information, please read the **Store keyword** and **View keyword** topics.

Store Using Keys

1. Press Store.

2. Press and hold MA while pressing X7 | View.
   This puts the **View keyword** in the command line.
3. Use the numeric keys to type the view number.
4. Execute the command by pressing Please.
5. Select the desired screens in the pop-up and press Please to confirm the options.

If point 3 is skipped (not adding a number), the first available view in the pool is stored.

Store Using a Combination of Keys and Touch Screens

Requirement:

- A visible **View Pool** on one of the screens.

To store a new view:

1. Press Store.
2. Tap an available pool object in the view pool to create a new view or one of the existing pool objects to overwrite the existing view.
3. Select the desired screens in the pop-up and tap OK to confirm the options.

The last touched object (the object with the white frame) can be labeled if you start typing on the keyboard. Any of the labeling methods described in the **Label Pool Objects** topic can also be used.

Store Using Command Line

Storing using the command line is very simple. The two keywords needed are **Store** and **View.**

For example, storing the window arrangement on screens 1 and 2 as view 15 with the "layout" label:

See more in the **View keyword** topic.

Store a View Directly on a View Button

A new view can be stored on a **View Button**. This creates a view in the view pool and immediately assigns it to the view button.

1. Press Store.
2. Tap a view button.
3. Select the desired screens in the pop-up and tap OK to confirm the options.

- OR -

1. Press and hold a view button until the pop-up appears.
2. Select the desired screens in the pop-up and tap OK to confirm the options.

Now, there is a new view in the pool, and it is assigned to the view button.

Any of the **described label functions** can be used with the view buttons.

Assign Existing View to a View Button

The above method is for creating a new view and having it available on a view button. Existing views can also be assigned to the view buttons.

Again, there are three primary ways to do it: Keys, keys and screens, and command line.

Assign View Using Keys

1. Press Assign.
2. Press and hold MA while pressing X7 | View once.
3. Use the numeric keys to type the view number.
4. Press At.
5. Press and hold MA while pressing X7 | View twice (this gives the **ViewButton** keyword).
6. Use the numeric keys to type the view button number.
7. Execute the command by pressing Please.

Example:

To assign view four at view button seven on screen number one, the following key presses are needed:

Assign MA + X7 | View 4 At MA + X7 | View MA + X7 | View 1 . 7 Please

This is the command result:

Assign View Using Keys and Screens

Requirement:

- A visible **View Pool** on one of the screens and visible view buttons.

To assign a view:

1. Press Assign.
2. Tap the desired view in the view pool.
3. Tap the desired view button.

Assign View Using the Command Line

The **Assign**, **View**, and **ViewButton** keywords are needed for this command.

Assign View ["View_Name" or View_Number] At ViewButton ["Display_Name" or Display_Number].[ViewButton_Number]

Assign Existing Views to an Executor

Existing views can be assigned to executors.

Again, there are three primary ways to do it: Keys, keys and screens, and the command line.

Assign View Using Keys

When objects are assigned to executors on a specific page using the keys and command line, then the object needs to be assigned to the child of the page using the **Page** keyword. The executors are children of the page.

If the object needs to be assigned to executors on the active page, it can be assigned to the executor using the **Executor** keyword.

1. Press Assign.
2. Press and hold MA while pressing X7 | View once.
3. Use the numeric keys to type the view number.
4. Press At.
5. Press and hold MA while pressing X16 | Exec once.
6. Use the numeric keys to type the executor number (executor on the active page).
7. Execute the command by pressing Please.

Example:

To assign view 9 at executor X1 | Clone (executor 291) on executor page 3, the following key presses are needed:

Assign MA + X7 | View 9 At MA + X15 | Page 3 . 2 9 1 Please

This is the command result:

Assign View Using Keys and Screens

Requirement:

- A visible **View Pool** on one of the screens and access to executors.

To assign a view:

1. Press Assign.
2. Tap the desired view in the view pool.

3. Tap the desired executor.

Assign View Using the Command Line

The **Assign**, **View**, and **Page** or **Executor** keywords are needed for this command.

Assign View ["View_Name" or View_Number] At Executor ["Executor_Name" or Executor_Number]

Assign View ["View_Name" or View_Number] At Page ["Page_Name" or Page_Number].[Executor_Number]

Recall Views

Stored views are recalled to load the view.

If a view is recalled without specifying a destination (which screen should the view be recalled to), then it, by default, opens on the screen where it was stored. The destination can be set using the /Screen preference. See the example below and learn more in the **View keyword** topic. If the view is stored with two or more screens, then the views are always recalled on the stored screens.

When the view is assigned to a view button, then pressing the view button recalls the view on the same screen as the view button.

Tapping a view in the pool recalls the view on the same screen as the tapped view pool.

Pressing an executor to recall a view uses the default behavior described above.

If a view is recalled using the command line without specifying a destination, it is recalled on the screen where the command line has focus (the command line can be on several screens). Specifying a destination in the command line overrides this default behavior.

(Call) View ["View_Name" or View_Number]

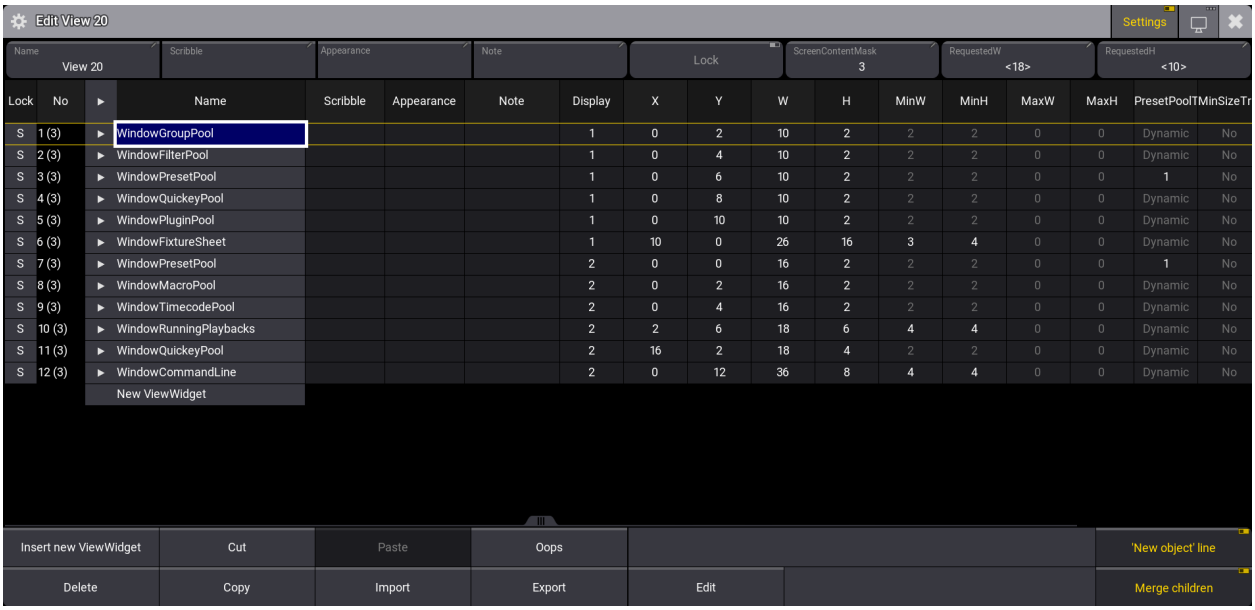
(Call) View ["View_Name" or View_Number] /Screen "[Screen_Number]"

Update Views

A view is updated simply by storing it again and overwriting the existing view. This can be done directly in the view pool or on the view button using the methods described above.

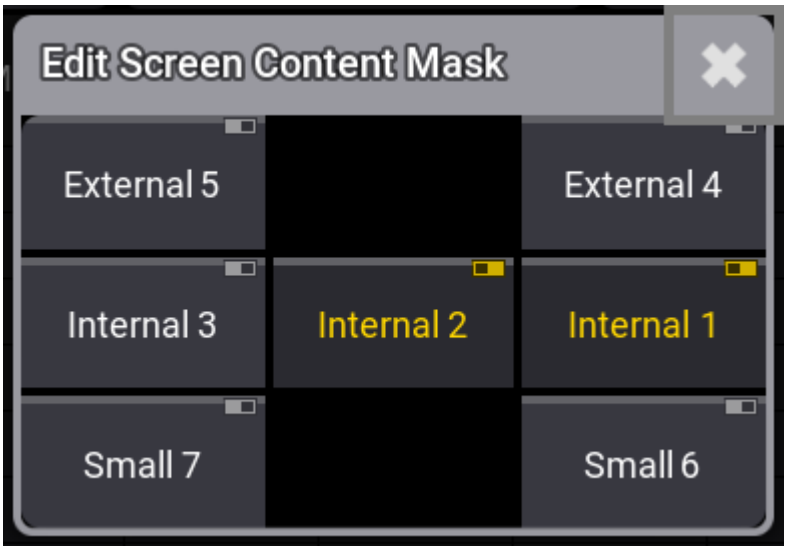
Edit a View Object

A view object can be edited. This opens the View Editor.



This editor has a table view with each window as a row and all the different settings for the windows in columns. The window objects (called "ViewWidget") can be unfolded by tapping the white arrow. This exposes all the settings for the window.

At the top of this editor, the settings can be visible (like in the example above). This has the standard settings for Name, Scribble, Appearance, Note, and Lock. There are also three settings specific to the views. The **ScreenContentMask** is a numeric representation of which screens are stored in the view. Tapping this opens a small **Edit Screen Content Mask** pop-up.



This is similar to the screen grid from the **Store View Options** pop-up.

The **RequestedW** and **RequstedH** settings define the size this view would like the screen to have.

Selecting a window in the list and tapping Edit in the bottom bar opens the relevant window settings pop-up.

1.10.4. Remove Windows from a Screen

There are three different ways to remove or delete a window from a screen: A single window can be deleted, all windows on a screen can be deleted, or all windows can be deleted on all screens.

This can be done using the command line or the GUI. This topic is about the GUI method. For information about the command method, please see the **Delete keyword** and the **ScreenContent keyword**.

Delete a Single Window on a Screen

Requirement:
A window needs to be on the screen.

1. Tap the MA icon in the windows title bar.
   - The settings pop-up appears.
2. Tap the Delete Window button in the title bar of the settings pop-up.

The window is now deleted from the screen.

Delete All Windows from a Screen

There are several methods to do this. Use the **Add Window pop-up**, the **Display pop-up**, or tap an empty View Button.

Add Window Method

Requirement:
An empty area on the screen.

1. Tap the empty area on the screen.
2. Tap the Delete Screen button in the **Add Window** pop-up.

All windows are now deleted from the screen.

Display Method

1. Open the Menu and Display pop-up by pressing the Menu key or tap the icon in the **Control Bar**.
2. In the Display pop-up, tap the Delete This Screen button on the screen that needs to be deleted.

All windows are now deleted from the screen.

Delete All Windows from All Screens

1. Open the Menu and Display pop-up by pressing the Menu key or tap the icon in the **Control Bar**.
2. In the Display pop-up, tap the Delete All Screens button on any of the screens.

All windows are now deleted from all screens.

1.10.5. Common Window Settings

All windows have settings.

The settings can be accessed by tapping the MA logo in the title bar or title field.

The settings vary depending on the window. The settings are organized in different sections, visualized by different tabs.

The **Display** section looks like this for the **Fixture Sheet**:

Fixture Sheet Settings - Display tab

There are some common buttons in the settings title bar. The red Delete Window button deletes the window from the screen. The ✖ closes the settings pop-up.

The Save button stores the current settings as a user preference. The Load button is used to load stored preferences. Read more **below** to learn more about how to store and load preferences.

The number of tabs changes depending on the window. Some windows have a lot of settings; some have fewer.

Changing a setting does not close the settings pop-up.

Some settings open small selection pop-ups listing the different properties for the setting.

Display / Sheet

This tab has settings regarding the way the window displays sheet information. The settings here vary a lot depending on the window. The settings that are special for a single window type are described in the topic about the window. Some of these settings are in a Display tab in the Phaser Editor some of the settings are in the Sheet tab.

The following is a list of the settings that are shared between two or more windows:

- **2 Finger Edit:**
  This toggle button enables the possibility to edit objects by using the two-finger gesture or right-clicking.
- **#Columns:**
  This input button sets the number of columns a sheet should display (the settings **Transpose** and **Adjust Columns** must be switched On except in the **DMX Sheet**).

  The DMX Sheet shows all the DMX channels and their output values. Learn more in the **DMX Sheet topic**.

- **Adjust Columns:**
  This On/Off button makes a sheet adjust the column width to match the window size and the number of columns.
- **Appearance:**
  Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.
- **Auto Scroll:**
  This On/Off button activates the auto-scrolling function. This will keep the active object visible in the window by scrolling the sheet or grid.
- **Channel Set**:
  This setting defines the readout of values that are part of channel sets. It has three options:
  - **Value**:
    Displays only the value.
  - **Value + Name**:
    Displays the value and channel set name.
  - **Name**:
    Displays only the channel set name.
- **Color Mode**:
  This switches the color readout between RGB and CMY. The default value is to follow the setting in the **User Profile**. The user profile setting is shown between "<>".
- **Cue Only**:
  It defines if the cue only function is On/Off when editing values.This setting is valid for the **Sequence Sheet in Track Sheet mode** and **Content Sheet**.

  The Sequence Sheet can be in "Track Sheet" mode where the attribute values are shown and can be edited. Learn more in the **Sequence Sheet topic**.

  The Content Sheet shows the cue content. It can show the current, previous, next, or specific cue. The sheet also displays the attribute values, which can be edited in the sheet.
  Learn more in the **Content Sheet topic**.
- **Executors**:
  This is an On/Off button that shows or hides the executors. If executors are hidden and labels

are shown, they look like the executor labels in the Playback Bar on the letterbox screens.This setting is valid for the **Playback** and **Xkeys** windows.
The Playback Window shows on-screen executors and their labels.
Learn more in **Executors topic**.
The Xkeys Window shows an on-screen version of the XKeys buttons and labels.
Learn more in the **Playback Bar topic**.

- **Feature Graphic**:
  Shows or hides a small graphic next to each feature in the sheets showing the features.
- **Feature Sort**:
  This On/Off button activates feature sorting. The selected feature is moved before the other features in the sheets showing features.
- **Fixed Target**:
  This setting defines the sequence a sheet displays if the **Link Type** is **Fixed**. Tapping this setting opens an **Assignment Editor** pop-up where a sequence can be selected.
- **Fixture Appearance**:
  This defines how the appearance of the fixtures is shown in the sheets. There are three options:
  - **None**:
    The fixture appearance is not shown.
  - **Enabled**:
    The appearance of the fixture type is shown.
  - **Graphic**:
    The appearance is shown with a colored background to match the output.

  This is valid for the **Fixture Sheet** and **Content Sheet**.
  The Fixture Sheet is a window that shows all the patched fixtures that have an ID. It has different modes that can use different versions of the attribute values for each fixture.
  Learn more in **Fixture Sheet topic**.
  The Content Sheet shows the cue content. It can show the current, previous, next, or specific cue. The sheet also displays the attribute values, which can be edited in the sheet.
  Learn more in the **Content Sheet topic**.
- **Fixture Graphic**:
  This defines which graphics are displayed in front of the name column in sheets showing the fixture graphic. Resizing the name column to a very small size will hide the graphic.
  This setting has the following options:
  - **None**:
    No graphic is shown.
  - **Flip**:
    Adds the flip indicator for fixtures with position attributes on the left side of the **Name** column.
  - **Simple**:
    Adds a simple square graphic indicating combined color and dimmer values next to the flip indicator in the **Name** column.
  - **Gobo**:
    Adds a gobo image on the simple graphic. It only displays the gobo of one gobo wheel at a time. Gobo wheels in ascending order define which gobo is displayed. For example, when Gobo 1 is set to open, then the gobo of Gobo 2 is displayed.
- **Fixture Sort**:
  This On/Off button activates the sorting of fixtures. The fixtures are sorted in the selection order to the top or left hand side of the sheet showing the fixtures.

- **Font Size**:
  This selects the font size in the window. It is a swipe button that opens a list of sizes from 10 to 32. There is also a **Default** property. The default is the same as size 18.
- **Frame Readout**:
  This defines the frame readout for this window. It can be used to overwrite the default set in the **user profile**.
- **Labels**:
  This is an On/Off button that shows or hides the labels. This setting is valid for the **Playback** and **Xkeys** windows.
  The Playback Window shows on-screen executors and their labels.
  Learn more in **Executors topic**.
  The Xkeys Window shows an on-screen version of the XKeys buttons and labels.
  Learn more in the **Playback Bar topic**.
- **Layer:**
  It selects which layer is displayed in the window. It is a swipe button that opens a list of the layers. A special property is **Auto**. This property makes the window follow the selected layer in the **Encoder Bar**.
- **Layer Toolbar**:
  This On/Off button shows or hides a **layer toolbar** at the bottom with the different Layers.
- **Link Type**:
  This setting defines which sequence is shown in the sheet.
  There are three different link types. The options are:
  - **Fixed**:
    The sheet displays the information from a specific sequence. The selection is made in the Sheet Settings. Read about the **Fixed Target** setting above. It can also be set using the **Assign** and **Sequence** keywords and tapping the sheet's title bar.
  - **Selected**:
    The sheet displays information from the selected sequence.
  - **LastGo**:
    This automatically shows the latest sequence to receive one of the trigger commands (<<<, >>>, Go+, Go-, Goto, Load, On, Select, Top, Temp, Flash, Toggle On, Pause). This includes if the sequence is triggered from a running timecode recording. A sequence can be excluded from LastGo by turning Off the **Include Link Last Go** setting in the **Sequence Settings**. LastGo only shows sequences triggered by the same user profile.

  This is valid for the **Sequence Sheet** and the **Content Sheet**.
  The Sequence Sheet shows the cues in a sequence and all the settings related to cue transition. It also has a mode called Track Sheet that shows the attributes values in the cues.
  Learn more in the **Sequence Sheet topic**.
  The Content Sheet shows the cue content. It can show the current, previous, next, or specific cue. The sheet also displays the attribute values, which can be edited in the sheet.
  Learn more in the **Content Sheet topic**.
- **Merge Cells**:
  Cells can be merged to show a value only once if the adjacent cell has the same value and belongs to the same feature or feature group. For instance, if all red, green, and blue values are "100", then "100" are only shown once.
  - **None**:
    Cells are not merged.

- **Feature**:
  The values of a feature are merged to only be shown once if the two or more adjacent values are the same.
- **Feature Group**:
  The values of a feature group are merged to only be shown once if the two or more adjacent values are the same.

- **Page:**
  It is used to change which executor page the window relates to. This setting is valid for the **Playback window**, **command wing Bar window**, and the **Xkeys window**.
  The Playback Window shows on-screen executors and their labels.
  Learn more in **Executors topic**.
  The command wing Bar is a window that shows labels matching the grandMA3 onPC command wing hardware. It can also be helpful when using the grandMA3 onPC on a single FullHD monitor.
  Learn more in **command wing Bar topic**.
  The Xkeys Window shows an on-screen version of the XKeys buttons and labels.
  Learn more in the **Playback Bar topic**.
- **Preset**:
  This defines how the preset information is displayed in the sheets. There are six properties which are different combinations of these three elements:
  - **ID**:
    Shows the ID number of the preset.
  - **Name**:
    Shows the name of the preset.
  - **Value**:
    Shows the values stored in the preset.

  This is valid for the **Fixture Sheet**, **Sequence Sheet**, and **Phaser Editor**.
  The Fixture Sheet is a window that shows all the patched fixtures that have an ID. It has different modes that can use different versions of the attribute values for each fixture.
  Learn more in **Fixture Sheet topic**.
  The Sequence Sheet shows the cues in a sequence and all the settings related to cue transition. It also has a mode called Track Sheet that shows the attributes values in the cues.
  Learn more in the **Sequence Sheet topic**.
  The Phaser Editor is an editor that can be used to see, create, and edit Phaser information and values.
  Learn more in the **Phaser Editor topic**.
- **Readout:**
  This selects the value readout for fixture attributes. It is a swipe button that opens a list of readout types with the following options:
  - **Auto**:
    This makes the sheet follow the selected readout in the **Encoder Bar**.
  - **Natural**:
    Each attribute has a defined Natural readout. This is defined in the **Attribute Definition**. Selecting this option will show the different readouts defined for the attributes.
  - **Percent**:
    This is a range from 0 to 100.
  - **PercentFine**:
    This is a range from 0.00 to 100.00.

    - **Physical**:
      This uses the physical range defined in the fixture type definition.
    - **Decimal8**:
      This is a decimal range from 0 to 255.
    - **Decimal16**:
      This is a decimal range from 0 to 65 535.
    - **Decimal24**:
      This is a decimal range from 0 to 16 777 215.
    - **Hex8**:
      This is a hexadecimal range from 00 to FF.
    - **Hex16**:
      This is a hexadecimal range from 0000 to FFFF.
    - **Hex24**:
      This is a hexadecimal range from 000000 to FFFFFF.
- **Setup**:
  This changes the window into a setup mode, where the content or setup of the window elements can be manipulated.
- **Sheet Mode**:
  The sheet mode changes how the sheets look. There are four different modes:
    - **Fixture**:
      This shows a matrix with the fixtures in rows and the attributes in columns.
    - **Channel**:
      This shows the fixtures as tiles with the dimmer attribute.
    - **Dimmer+**:
      Looks similar to the **Channel** mode. However, it additionally displays the attributes of the selected feature group. Vertical gray separators are displayed when there is a jump in IDs and when the IDType changes for fixtures that do not have a fixture ID. This mode can display the **Fixture Graphics**, but does not display the **Feature Graphics**.
    - **Sheet/Filter**:
      Similar to **Dimmer+**. However, it displays all attributes unless there is a defined filter in the **Mask** tab of the sheet settings.

  This setting is valid for the **Fixture Sheet** and the **Content Sheet**.
  The Fixture Sheet is a window that shows all the patched fixtures that have an ID. It has different modes that can use different versions of the attribute values for each fixture.
  Learn more in **Fixture Sheet topic**.
  The Content Sheet shows the cue content. It can show the current, previous, next, or specific cue. The sheet also displays the attribute values, which can be edited in the sheet.
  Learn more in the **Content Sheet topic**.
- **Show Grand Master**:
  This shows or hides the Grand Master section in the window. This is valid for the **Custom Master Section window**, **command wing Bar window**, and **Encoder Bar window**.
  The Custom Master Section is a window that shows on-screen versions of the two custom areas, the master area, and the grand master knob. It also shows the labels for these elements.
  Learn more in the **Special Executors topic**.
  The command wing Bar is a window that shows labels matching the grandMA3 onPC command wing hardware. It can also be helpful when using the grandMA3 onPC on a single FullHD monitor.
  Learn more in **command wing Bar topic**.

The Encoder Bar window shows on-screen versions of the encoder bar. It can be useful to create custom interfaces.
Learn more in the **Encoder Bar topic**.

- **Show Master Area** or **Show Master Section:**
  This shows or hides the Master area in the window. This is valid for the **Custom Master Section window** and **command wing Bar window**.
  The Custom Master Section is a window that shows on-screen versions of the two custom areas, the master area, and the grand master knob. It also shows the labels for these elements.
  Learn more in the **Special Executors topic**.
  The command wing Bar is a window that shows labels matching the grandMA3 onPC command wing hardware. It can also be helpful when using the grandMA3 onPC on a single FullHD monitor.
  Learn more in **command wing Bar topic**.
- **Show Title Bar / Title Bar**:
  This shows or hides the window's title bar. It is On by default. If it is Off, then the title bar can be shown temporarily by pressing both MA keys in the control area. In grandMA3 onPC, the title bar can be temporarily shown by pressing Ctrl + Alt on Windows and Ctrl + Option on Mac. This is valid for **Clock Viewer**, **Encoder Bar window**, and **Layout Viewer**.
  The Clock Viewer shows the time. It can be the system time, the time of a timezone, the timer of a timer, or timecode slot.
  Learn more in the **Clock viewer topic**.
  The Encoder Bar window shows on-screen versions of the encoder bar. It can be useful to create custom interfaces.
  Learn more in the **Encoder Bar topic**.
  The Layout Viewer shows a layout. The shown layout can follow the selected layout or always show a specific layout.
  Learn more in the **Layout topics**.
- **Speed**:
  It sets how the speed value is displayed. It has the following options: Auto (following the User Profile setting), Hertz, BPM (Beats Per Minute), and Seconds. This setting is valid for the **Fixture Sheet** and **Phaser Editor**.
  The Fixture Sheet is a window that shows all the patched fixtures that have an ID. It has different modes that can use different versions of the attribute values for each fixture.
  Learn more in **Fixture Sheet topic**.
  The Phaser Editor is an editor that can be used to see, create, and edit Phaser information and values.
  Learn more in the **Phaser Editor topic**.
- **Step:**
  It selects which step to display. Steps are used with **Phasers**. It is a property input button that opens a calculator pop-up. This setting is valid for the **Fixture Sheet** and **Sequence Sheet**.
  The Fixture Sheet is a window that shows all the patched fixtures that have an ID. It has different modes that can use different versions of the attribute values for each fixture.
  Learn more in **Fixture Sheet topic**.
  The Sequence Sheet shows the cues in a sequence and all the settings related to cue transition. It also has a mode called Track Sheet that shows the attributes values in the cues.
  Learn more in the **Sequence Sheet topic**.
- **Time Format**:
  This defines the time format for the windows. This can be used to select a different format

than the default set in the **user profile**. This setting is valid for the **Fixture Sheet** and **Sequence Sheet**.
The Fixture Sheet is a window that shows all the patched fixtures that have an ID. It has different modes that can use different versions of the attribute values for each fixture.
Learn more in **Fixture Sheet topic**.
The Sequence Sheet shows the cues in a sequence and all the settings related to cue transition. It also has a mode called Track Sheet that shows the attributes values in the cues.
Learn more in the **Sequence Sheet topic**.

- **Transpose:**
  This On/Off button flips the columns and rows in windows.
- **View Mode**:
  The view mode defines how the different data and information are displayed in the view. The view mode can be changed in the settings or the title bar of the window.

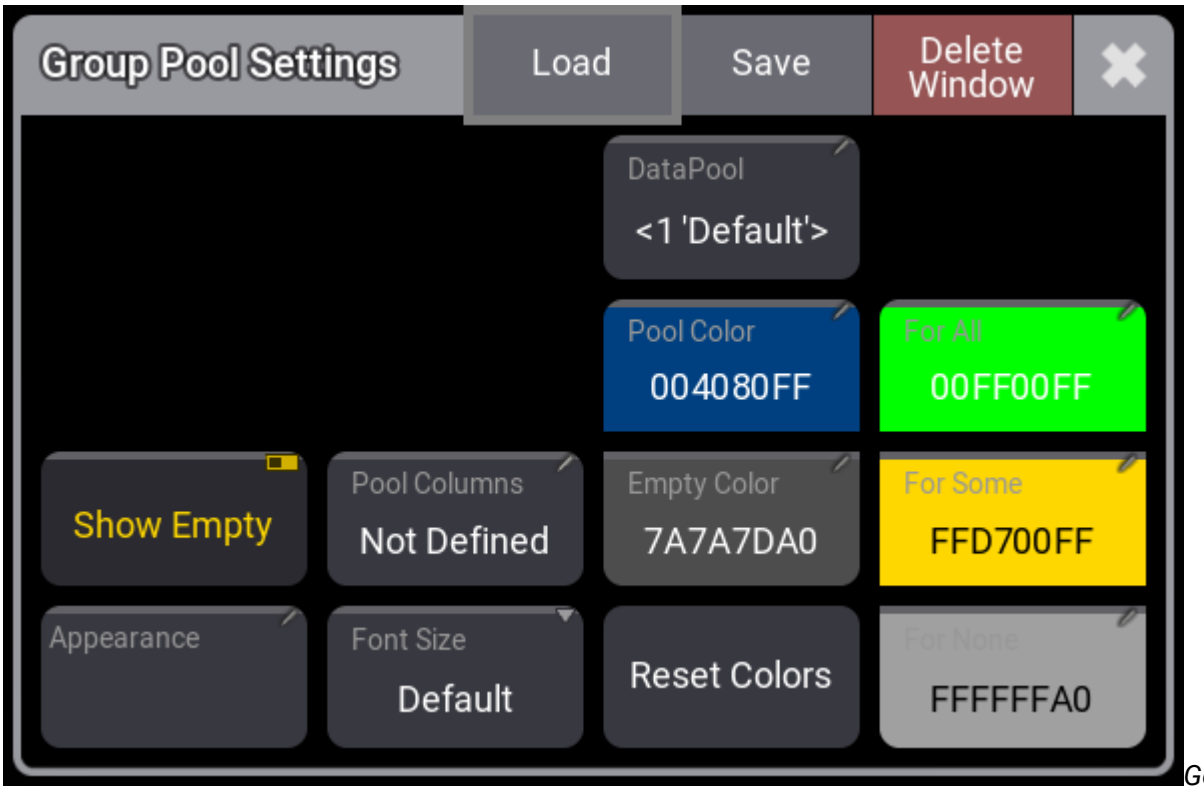
The Agenda Viewer has the following view modes: Sheet, Year, Month, Week, and Day.
The Phaser Editor has the following view modes: Auto, 2D, 1D, and Sheet.
The Sound Viewer has the following view modes: Wave, Sound, and Beat.
The Timecode Viewer has the following view modes: Text, Timeline, and Both.

- **Wing ID:**
  Defines which wing the window displays. Tap this setting to open a small **Select WingID** pop-up where the desired wing can be selected. This setting is valid for the **command wing Bar window** and **Playback window**.
  The command wing Bar is a window that shows labels matching the grandMA3 onPC command wing hardware. It can also be helpful when using the grandMA3 onPC on a single FullHD monitor.
  Learn more in **command wing Bar topic**.
  The Playback Window shows on-screen executors and their labels.
  Learn more in **Executors topic**.

	Hint:
	Showing a window that has the **Show Title Bar** setting in a true full-screen version can be achieved by hiding all other visual elements on the screen using the **Configure Display pop-up** and then turning Off the Show Title Bar setting for the window.

Pool Settings

Most pools have the same few settings. Some pools might have extra settings. The Group and Preset pools have three extra settings regarding the colored bar above each pool object. The **Smart window** behaves much like a pool and shares most of the pool settings.



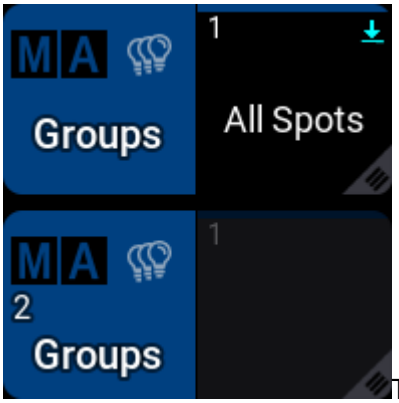
General pool window settings

These are the common settings for pools:

- **Show Empty**:
  This toggle button can hide or show empty pool objects.
- **Appearance**:
  The appearance is applied behind the pool objects.
- **Pool Columns**:
  This defines the width for the pool objects. It does not change the size of the window. It defines how many columns of pool objects are in the window. If the window is wider than the number of columns, then the extra space is displayed as black (default color). If the window is smaller than the number of columns, the pool window can be scrolled horizontally. If the pool has a set width, then there is an icon (⏭) in the upper right corner of the title field.
  The **Not Defined** value dynamically sets the width to match the window size even when the window is resized.
  The **Take Current Width** sets the width to match the current size of the window. It does not dynamically change if the window is resized.
- **Font Size**:
  There are some different font size properties from 10 to 32. There is also a default property. This is the same as size 18. This simply changes the font size on the pool objects.
- **DataPool** (only available for some pools):
  This defines what data pool the pool window shows data from. This makes it possible to have pools showing objects from different data pools. For instance, a group pool window from the default data pool can be shown next to a different group pool window showing groups from a different data pool.
- **Pool Color**:
  This is the color for the title button in the pool.

- **Empty Color**:
  This color is applied to empty pool objects.
- **Reset Colors**:
  This resets the colors to the colors in the default color theme.
- **Use Object Action:**
  When enabled, the selected object action is executed instead of the selected pool action.
  Pool windows with Use Object Action enabled, are marked with a (**+**). Pool objects indicate the selected object action setting with a light grey icon in the background of the object.

The data pool number is shown in the lower right corner of the pool title button. The number is only shown if the pool window supports showing different data pools and if there is more than one data pool in the show.



Two group pools with different data pools

Preset and group pools also have:

- **For All**:
  This color is used when the preset can be used by all of the selected fixtures.
- **For Some**:
  This color is used when some of the selected fixtures can use the preset.
- **For None**:
  This color is used when the preset is not usable by any of the selected fixtures or when none of the selected fixtures are in the group.

Tapping one of the color settings opens an **Edit Color** pop-up.

The buttons in the title bar are the same as the ones for other window settings - read above.

Default Pool Action

This defines the default action executed when a pool object is tapped without a (relevant) keyword in the command line.
Pools can have some of the following actions (the available actions depend on the type of pool):

- **At** (@):
  When there is no selection in the programmer, tapping a preset does nothing. When the programmer has a selection, tapping it calls the preset into the programmer.
- **Call**() - default action for filters:
  This action calls the tapped pool object.
- **SelFix/At** (no icon) - default action for presets:
  When there is no selection in the programmer, tapping a preset will select the fixture that can use the preset. Tapping it again calls the preset into the programmer.

When the programmer has a selection, tapping a preset the first time will call the preset into the programmer.

- **SelFix/Extract** ( ):
  This value acts similarly to SelFix/At, but instead of calling the preset reference into the programmer, the values will be called extracted into the programmer.
- **Select** ( ) - default action for sequences:
  Tapping the pool object selects it.
- **Toggle** ( ):
  Tapping a pool object plays it back or switches it off, depending on its current playback state.
- **Go+** ( ):
  Starts playback of the pool object or goes to the next cue in the sequence.
- **Flash** ( ):
  Flashes a pool object as long as it is tapped. Flash ignores fade times.
- **Temp** ( ):
  Plays back a pool object as long the pool object is pressed. Temp respects the fade times.

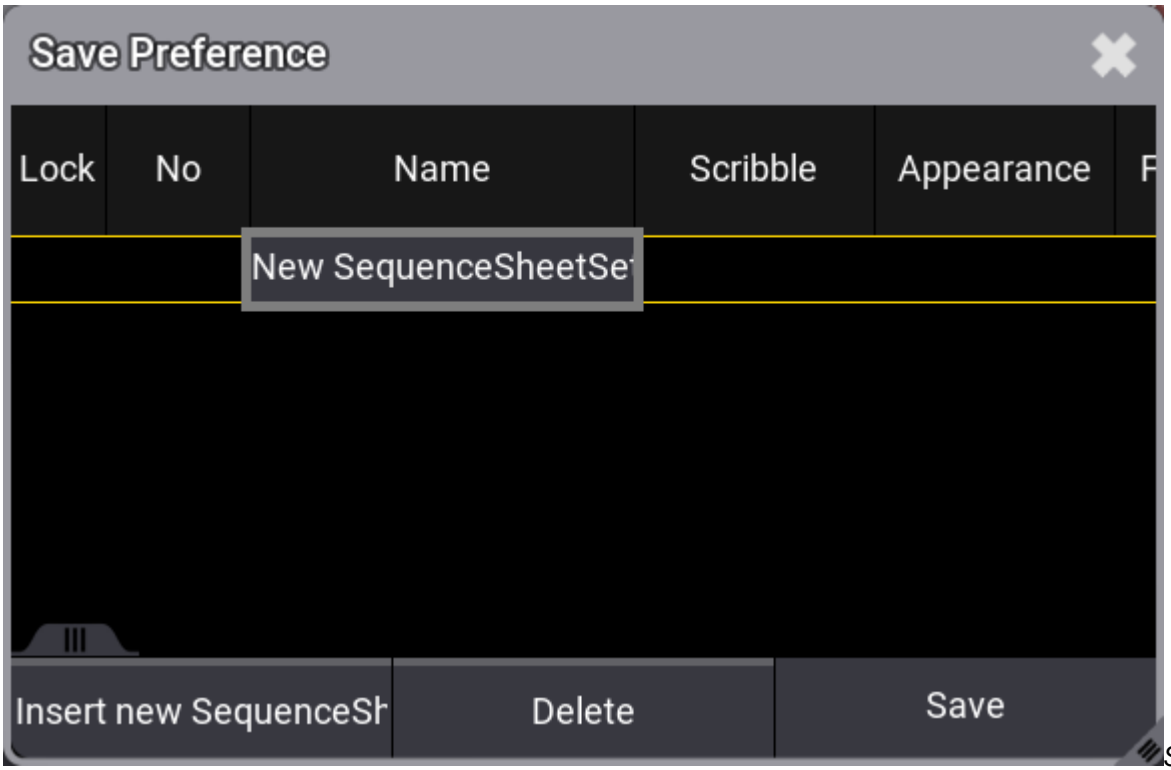
A small icon in the upper right corner of the pool title object indicates the default action.

Pools with default pool actions

Store and Load Preferences

Setting preferences can be stored and loaded for each window and pool. These preferences are stored in the user profile. This means that exporting and importing a user profile includes these preferences.

Tapping Save opens a **Save Preferences** pop-up:



Save Preference pop-up

Save a New Preference

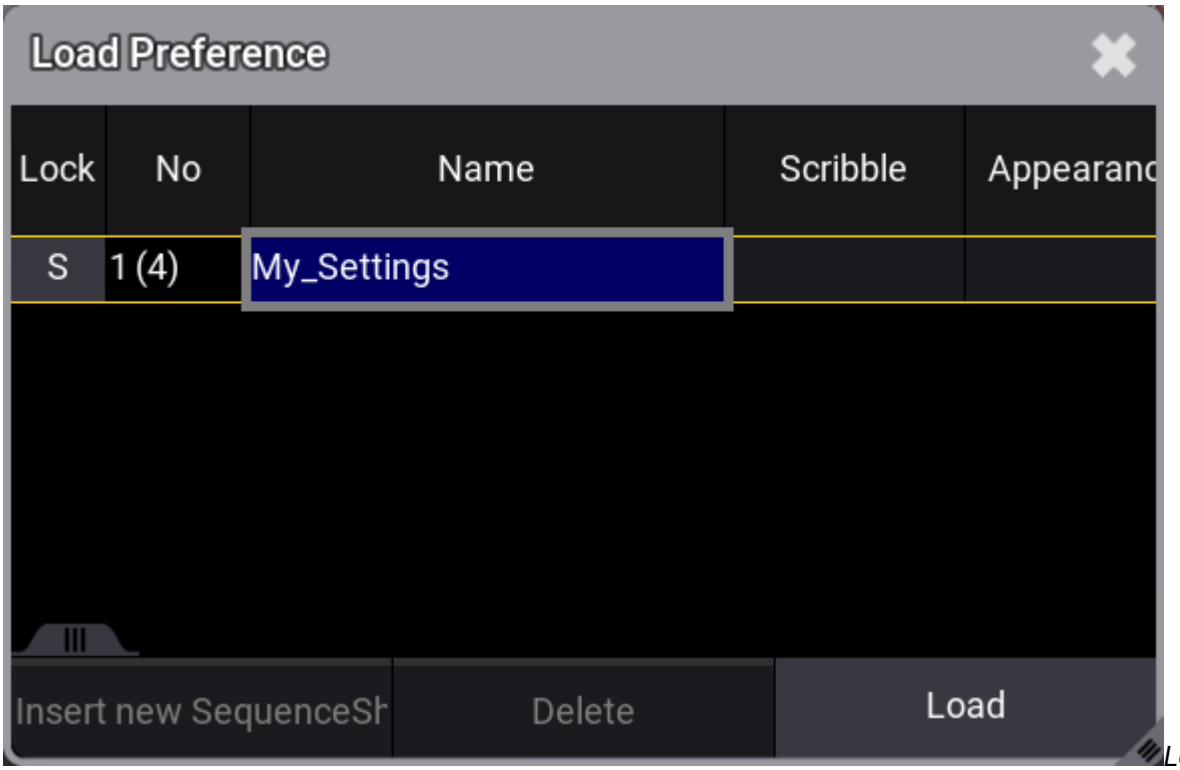
1. Open the **Settings** pop-up.
2. Tap Save in the settings title bar.
3. Tap Insert new [Type] (this button changes a little for each window, and the type shows what type of settings are being inserted).
4. Edit the name field and give it a name.
5. Tap Save in the preference pop-up.

	Hint:
	When saving a new preference, no preference exists yet, and the focus is on "New ...", it is enough to tap Save to save the preference.

Update the Preference

1. Open the **Settings** pop-up.
2. Tap Save in the settings title bar.
3. Tap the desired preference in the list.
4. Tap Save in the preference pop-up.

Load a Preference



Load preference pop-up

1. Open the **Settings** pop-up.
2. Tap Load in the settings title bar.
3. Select the desired preferences.
4. Tap Load in the preference pop-up to confirm.

Delete a Preference

1. Open the **Settings** pop-up.
2. Tap Save in the settings title bar.
3. Select the desired preferences.
4. Tap Delete in the preference pop-up.
5. Close the pop-up by tapping the in the upper right corner.

1.10.6. Title Bar Configuration

Many windows with a title bar can edit the arrangements of the buttons in the title bar.

These settings follow the user profile so that different users can have different default preferences. The setting is also stored with the view. For instance, a view can have several fixture sheets with some buttons in one window and others in a different window.

The title bar buttons can be edited in the window settings. Open the settings by tapping the MA logo in the upper left corner.

The settings title bar has an Edit Title Bar. Toggling this On changes how the setting pop-up works and looks.

Fixture Sheet Setting with Edit Title Bar On

Some settings can have a button in the title bar. These settings can be toggled to **Yes** in the different tabs in the setting pop-up.

The number on the lower right side of the settings shows the order of the button counting from right to left. The latest activated button will have the highest number and be on the left side in the title bar.

The green number in the tab displays how many settings are set to **Yes** out of the possible settings in the tab. For instance, "3/13" is three out of 13 possible settings with a button in the title bar.

Turning the edit mode On also adds an extra tab called Title Buttons.

Title Buttons tab

This tab is split into two sides.

The left side lists all the active title bar buttons. The Selected button can be moved up or down using the arrow buttons on the left side. The selected button can also be deleted by tapping the icon.

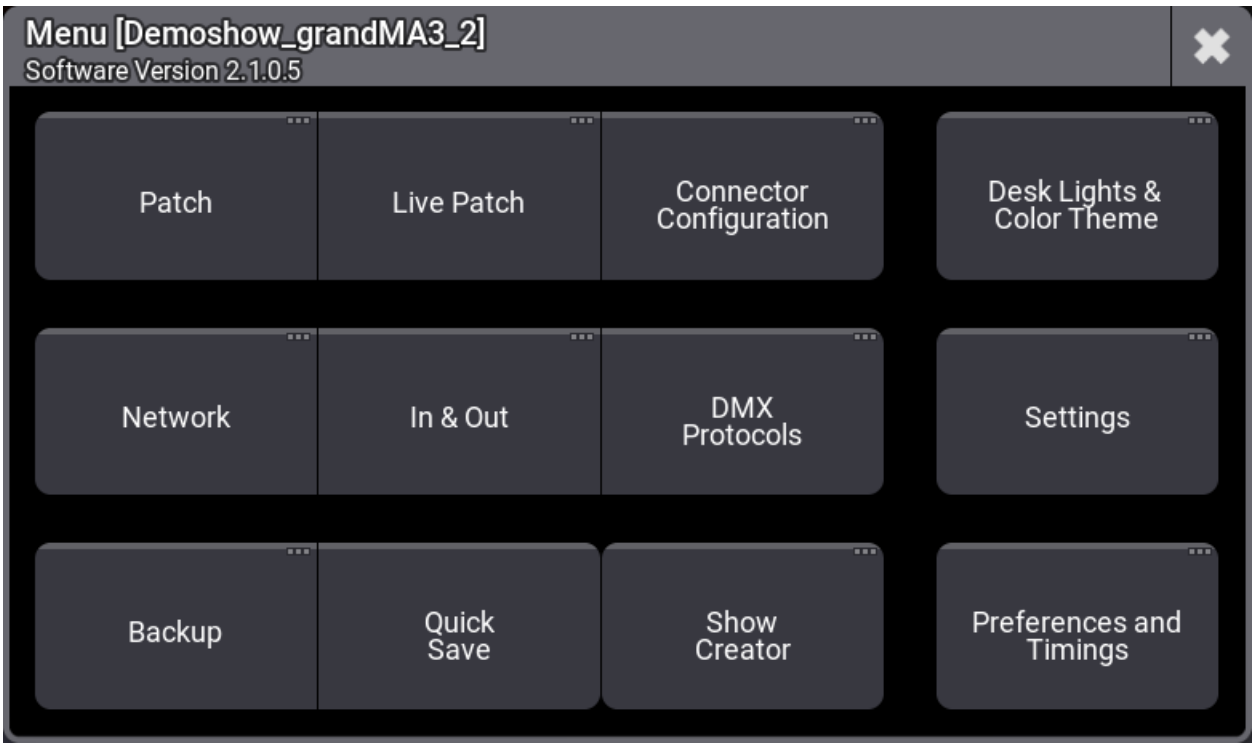
The top button is on the right side of the title bar.

The right side of this tab has two buttons. Load Defaults loads the default button configuration for the window. Delete All delete all the buttons from the title bar.

1.10.7. Menus

There are many different menus in the grandMA3 software. Most are described in relevant sections. For instance, the patch menu is described in the **Patch section**.

There is a **Menu pop-up** that gives access to many of the different menus.



Menu pop-up with buttons that open a lot of menus

This pop-up can be opened in multiple ways:

- On physical hardware, there is a Menu key. Press this to open the pop-up.
- In the **Control Bar** on the left side of most displays, on the onPC, there is a gear icon . Tap this to open the pop-up.
- It can be opened using the command line:

```
User name[Fixture]>Menu "MenuSelector"
```

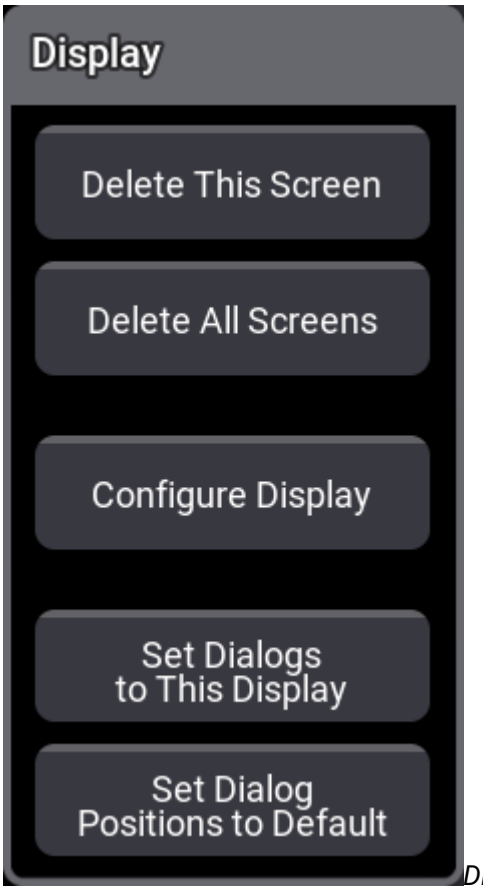
The menu pop-up gives access to the following menus:

- **Patch**:
  This is where fixtures are added and is all about fixture setup. Read more in the **Patch and Fixture Setup section**.
- **Live Patch**:
  This menu gives access to the fixture setup, which can be changed without a new show upload. Read more in the **Patch and Fixture Setup section**.

- **Connector Configuration**:
  The **connector configuration** changes the local physical DMX ports on the MA equipment.
- **Desk Lights & Color Theme**:
  This opens a pop-up that gives access to **customize the look of the software**.
- **Network**:
  This menu is used to set up all about the network and sessions - read more in the **Networking section**.
- **In & Out**:
  This opens the **In & Out menu** with settings for DC remote, MIDI remote, DMX remotes, OSC, and PSN.
- **DMX Protocols**:
  This menu is used to set up DMX **network protocols**.
- **Settings**: This opens a small sub-menu that has these options:
  - **User Configuration**:
    This menu is used to set up users and user profiles - read more in the **Single User and Multi-User Systems section**.
  - **Date and Time**:
    This menu is used to change the time and date in the console. Read more in the **Date and Time topic**.
  - **USB Configuration**:
    This is used to configure USB devices. This should normally not be changed manually.
  - **Software Update**:
    This menu is used to **update the software**.
  - **Touch Configuration** (consoles only):
    This menu is used to assign touch screens to the USB inputs.
  - **Extension Configuration**:
    This menu shows the connected grandMA3 extensions.
  - **onPC Local Settings** (onPC only):
    This is a pop-up with **onPC settings**.
  - **Local Settings** (consoles only):
    This is a pop-up with settings for the **local console**.
- **Backup**:
  The backup menu is used to save and load shows - read more in the **Show File Handling topic**.
- **Quick Save**:
  This is not a menu but a shortcut to saving the show with the same show file name.
- **Show Creator**:
  This opens the **Show Creator** menu that can be used to create presets and groups. It is also possible to import and export different objects of the show file.
- **Preferences and Timings**:
  This is where the defaults are set up for cues and sequences - read more in the **Cue Timing topic**.

Display Overlay

Opening the **Menu pop-up** also opens a **Display overlay**.



Display overlay in the right-hand lower corner of all screens

This pop-up appears on all screens except the letterbox screens.

Most of the actions performed using this pop-up relate to the specific screen where the menu is touched. For instance, deleting a screen (removing the windows from the screen).

- **Delete This Screen**:
  This clears all windows from this screen - read more in the **Remove Windows from a Screen topic**.
- **Delete All Screens**:
  This clears all windows from screens - read more in the **Remove Windows from a Screen topic**.
- **Configure Display**:
  This opens the Configure Display pop-up - read more in the **Configuration of Displays topic**.
- **Set Dialogs to This Display**:
  This makes all menus open on this screen - read more in the **Change Menu Locations topic**.
- **Set Dialog Positions to Default**:
  This resets all menus to open on the default screen - read more in the **Change Menu Locations topic**.

1.10.8. Change Menu Locations

Some menus can be moved between the screens.

Menus that can be moved have a display icon in the title bar:

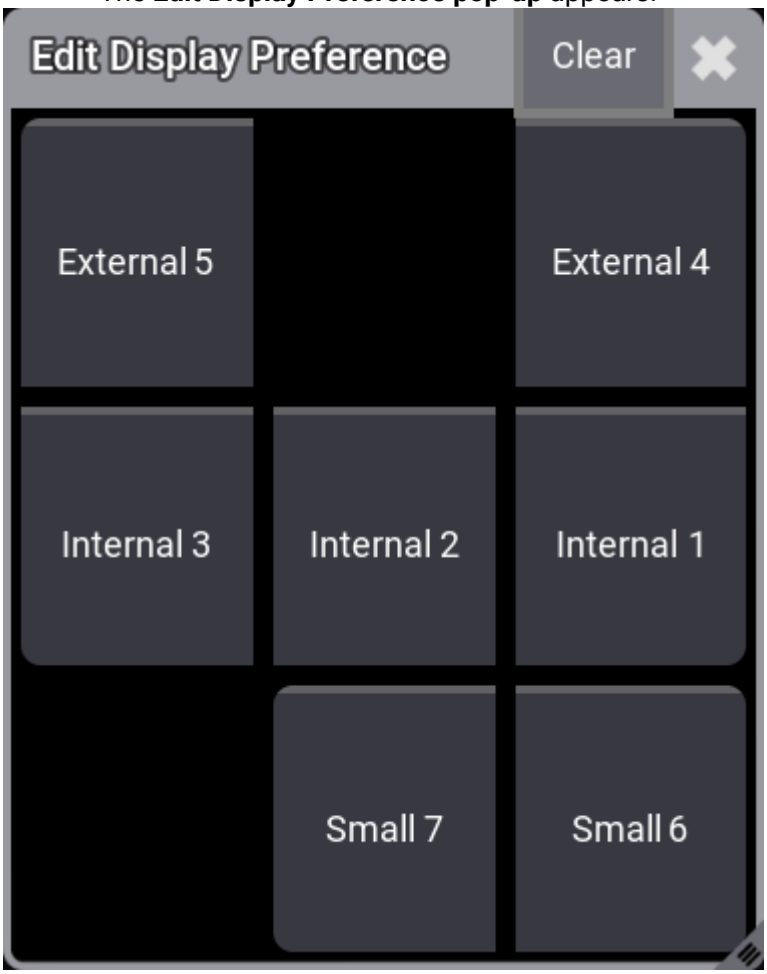

Display icon next to the close X

Move a Menu to a Different Screen

Requirement:

- Open a menu that can be moved. For instance the **Playback Controls** Menu.

1. Tap the display icon in the title bar.
   The **Edit Display Preference pop-up** appears:



Use the Edit Display Preference pop-up to select a new location

2. Tap the button corresponding to the desired screen.

	Hint:

	Tap the Clear button or the  to cancel the move.

Move all Menus to a Screen

It is possible to select a screen where all moveable menus open.

1. Open the **Menu and Display pop-up**.
2. In the Display pop-up tap the Set Dialog to This Display button on the screen where the menus should open.

Reset Movable Menus Screen

The screen selection can be reset. This is useful if a show changes from a smaller hardware platform. For instance, if the show is programmed on a grandMA3 full-size and menus are moved to screen 3 and now the show is opened on a grandMA3 light without screen 3.

1. Open the **Menu and Display pop-up**.
2. In the Display pop-up tap the Set Dialog Positions to Default button on the screen where the menus should open.

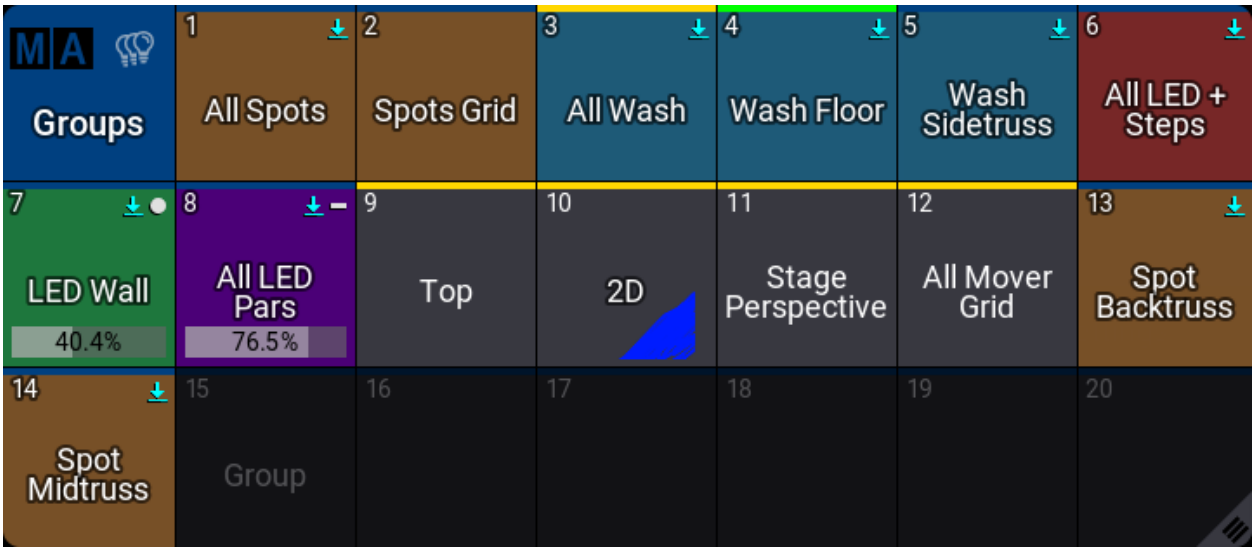
1.10.9. Pool Windows

Most of the show data in grandMA3 are organized in pools, which are part of a **Data Pool**. So, in essence, everything except the patch and fixture setup are stored inside a Data Pool. This makes having several shows with the same fixture setup but different data pools easy.

Pools contain a lot of different data. For instance, the **Groups pool** contains information about the selections of fixtures. The **Views pool** contains information about the arrangement of windows on a screen.

Pools are made like any other window - read more in the **Add Windows topic**. Pool arrangement and resizing are just like any other window - read more in the **Rearrange topic**.

This is an example of the Groups pool:



Groups pool

Each pool has a title field. In the example above it is the first blue square with the MA logo, the pool icon (if available), and the name of the pool.

Tapping the logo opens the settings for the entire pool - read more about the window settings in the **Window Settings** topic. Read the relevant topics for details about the pool or preset specific settings.

Some pools have specific uses. This topic and the following subtopics describe the general concepts. Read the specific topics to learn the details of each type of pool.

Pool Object

All other fields except the title field contain a **Pool Object**. If something is stored in the pool object, it has a lighter gray color (default color) and a label (see the example pool above). It can also contain pool-specific information like a master level or icons that show information about the content or pool object settings.

The number in the upper left corner of each pool object is the unique pool number identifier. It can be used when the object is called. For instance, using **Group 1** in the command line is the first object in the groups pool. The name of the pool object can also be used to call the group. For example, **Group "All Spots"** calls (all) the groups with that name. Asterisk can be used as a joker sign when using object names. Typing **Group "All*"** calls all the group objects where the name starts with **All**.

Working with or using pool objects can be done in many ways. The examples in the subtopics use the command line, but many of the operations could also be performed using the keys, screens, or any combination thereof.

Some pools have **Selected** pool objects. It is visualized with a yellow frame around the selected pool object.

World 1 is selected

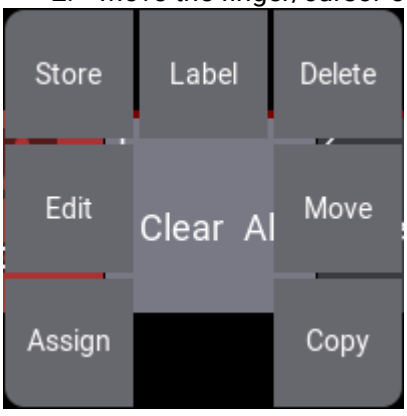
A different pool object can usually be selected by tapping the object in the pool or using the **Select keyword**.

Swipey Commands

Each pool object can open a set of **Swipey Commands**. They give quick access to some common operations with the pool object.

This is how to open the swipey commands:

1. Tap and hold the pool object.
2. Move the finger/cursor outside the pool object while keep pressing the screen.



The open Swipey Commands

3. Keep the screen pressed and move the finger/cursor to the desired command.
4. Release the screen.

These are the available commands with the swipeys:

- **Assign:**
  This adds "Assign" and the pool object type and number in the command line. Waiting for more user input.
- **Edit**:
  This starts the edit mode for the pool object - this is about the content of the pool object.
- **Edit Setting**:
  This opens a small editor that gives access to editing this pool object's settings - this is about the settings for the pool object.
- **Store:**
  This executes "Store" plus the pool object type and number. This is useful for storing new pool objects - for instance, groups.
- **Label**:
  This executes "Label" plus the pool object type and number.
- **Delete:**
  This executes "Delete" plus the pool object type and number - The result is deleting the pool object.
- **Move**:
  This adds "Move" plus the pool object type and number in the command line. Waiting for the user to tap the new location.
- **Copy**:
  This adds "Copy" plus the pool object type and number in the command line. Waiting for the user to tap the new location.

Subtopics

- **Create Pool Objects**
- **Move Pool Objects**
- **Insert Pool Objects**
- **Copy Pool Objects**
- **Lock and Unlock Pool Objects**
- **Delete Pool Objects**

1.10.9.1. Create Pool Objects

This is a general description of how to create pool objects.

The **Store keyword** is used when creating pool objects. This stores the relevant information in the pool object if the relevant information is available.

For instance, to store a preset, the programmer needs to have valid values. If the relevant values are not available, an empty pool object is created.

Some pool objects are a bit more complex. For instance, storing a pool object in the image pool. The pool that can be added on a screen, is not just called "Images" as the title suggests. It is actually image "pool" 3.

So, storing something on pool object 2 needs the following input:

But this only stores an empty pool object.

Some objects are better to **Edit** when creating them. This opens the relevant editor to create the object.

Images are an example of such an object.

This command creates the pool object and immediately opens the editor for the image pool object. Read more about images in the **Images section**.

Some pool objects can be created by pressing and holding an empty pool object. If there is relevant information in the programmer, then this might be stored in the new pool object.

For instance, having a selection of fixtures in the programmer and pressing and holding an empty group pool object for about 2 seconds will create a new group with the fixture selection.

1.10.9.2. Move Pool Objects

Pool objects are moved using the **Move** keyword.

A single object can be moved, or a selection of objects can be moved at once.

Example

Macro 5 needs to be moved to the empty macro pool object 20:

If the destination is not empty, then the existing object will move one place further. If this object is also occupied, it will move further, and so on, until no occupied object has to move anymore. Thus potentially moving all the arranged pool objects.

	Important:
	Please have a look at the **Exchange keyword** for objects exchanging positions.

1.10.9.3. Insert Pool Objects

Pool objects can be inserted between other pool objects of the same type. This can be useful for organizing the pools.

It is done using the **Insert** keyword.

The inserted pool object is inserted before the destination object number.

Example

For example, inserting view 42 between views 9 and 10:

The old view 10 and other views from 10 to the next empty pool object are then moved one number up.

1.10.9.4. Copy Pool Objects

Pool objects can be copied to an empty location using the **Copy** keyword.

A single object or a selection of several objects can be copied at once. Using the selection order, they are positioned at the new location.

Example

Copy sequence 2 at pool object 22 using the following command:

If the destination is not empty, then the old object will be overwritten. There is a pop-up asking if this is desired.

1.10.9.5. Lock and Unlock Pool Objects

Pool objects can be locked using the **Lock** keyword.

If a pool object is locked, then there is a small lock icon in the upper right corner of the pool object.

The lock icon can have three different colors indicating different types of locks:

- **Red**:
  This lock is the System Lock. It is locked by the system and cannot be unlocked. In sheets, this is abbreviated as "SL".
- **Gray**:
  This lock is the Position Lock. It is locked by the software structure and cannot be unlocked. In sheets, this is abbreviated as "PL".
- **Orange**:
  This lock is the User Lock. It is locked by the user, and it can be unlocked. In sheets, this is abbreviated as "UL".

	Hint:
	The big red lock icon that can appear on pool objects is not a locked object using this lock mechanism. It is temporarily blocked by a user editing the object. Learn more in the **Object Ownership topic**.

Example

Locking appearance number 4 is done using the following command:

```
User name[Fixture]>Lock Appearance 4
```

Unlock Pool Objects

User locked objects can be unlocked using the **Unlock** keyword. The syntax is the same as locking except for the keyword.

Example

Unlocking appearance number 4 is done using this command:

```
User name[Fixture]>Unlock Appearance 4
```

1.10.9.6. Delete Pool Objects

Removing pool objects can be done using the **Delete** keyword.

When an object is deleted, grandMA3 will try to make everything look the same. For instance, deleting a preset will move the values stored in the preset into the different cues where the preset was used. This is not always possible, for instance, if an executor is controlling a sequence and the sequence is deleted, then it is gone, and the executor is empty.

Example

Delete view 42:

It can also be a range of numbers, for instance:

Deleting pool objects can be undone with **Oops**. It will bring back the pool objects, but it might not restore links to the object.

1.11. Networking

Networking allows for the expansion of a single standalone console to a complete system with DMX nodes and extra processing powers.

The rear panel of all the consoles has three Ethernet connectors, which can be used to connect to three different networks.

Networking is used for:

- Connecting several MA3 devices in a session
- Output and input Ethernet-based DMX
- Internet connection

MA Session

Sessions are a way to expand a single console. They allow for connecting multiple consoles in a multi-user setup where several programmers can work in the same show. grandMA3 Processing Units can be added to move DMX calculations away from the consoles and into processors located where needed. grandMA3 xPort Nodes listen to a session and function as DMX interfaces, allowing decentralized synchronized DMX distribution.

A session is needed to connect grandMA3 devices. Read about **sessions here**.

Ethernet DMX

grandMA3 can output DMX using Art-Net and sACN. It is possible to set up a specific Ethernet port to output the network DMX. Read more in the **DMX In and Out topics**.

Internet Connection

The grandMA3 system is designed to connect to the Internet, more specifically, to World Servers. These servers can provide different services to the connected stations.

It is recommended to separate the light network from the Internet and use one of the three Ethernet connectors on the back of the MA hardware stations to connect to a network with Internet access. This will maintain a separation between the Internet and the light network and still provide access to the services on the world server.

For more information, see the **World Server topic** and the **Interfaces and IP topic**.

An Internet connection also makes it possible to see the manual videos in the software version of the manual.

Internal Connections

A console uses internal network connections to connect different sections of the internal components. This might be visible at some locations but should never be changed by users.

Enabling or Disabling the Network Connection

The network needs to be enabled to communicate. This includes transmitting DMX using Ethernet.

Turning network On or Off is done from the **Network menu** - read about the network menu in the **Sessions topic**.

In the lower right corner of the network menu, there is a button to toggle the network connection.

 *Network enable button*

If the icon is red, the network is turned Off. If it is green, it is On.

Subtopics

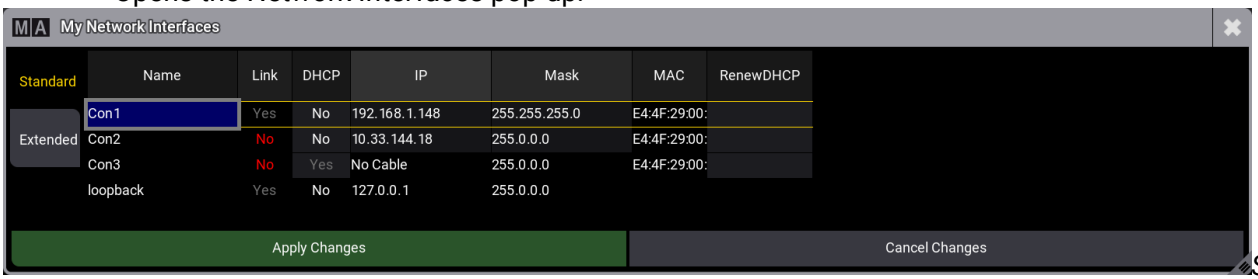
- **Interfaces and IP**
- **Session**
- **Web Remote**
- **Network Design**
- **Regulations and Standards**
- **Station Control**

1.11.1. Interfaces and IP

The Network Interface menu lists the available interfaces, and in the grandMA3 hardware, it can be used to change the settings.

Open the Network Interface Menu

1. Press Menu.
   - Opens the **menu select pop-up**.
2. Tap Network.
   - Opens the Network menu.
3. Tap My Interfaces.
   - Opens the Network Interfaces pop-up:

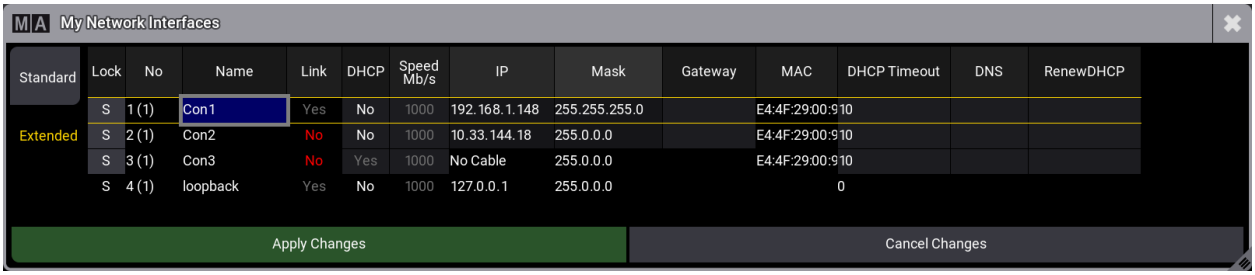


See the Standard tab in Network Interfaces

	Restriction:
	If you see an interface with "imx6" then it is for internal communication inside the console. Do not make any changes to this interface.

The pop-up has two versions: **Standard** and **Extended**. The screenshot above shows the Standard version. The Extended version has more settings, but otherwise, it is the same. It is possible to toggle between the two versions by tapping the tabs on the left.

The Extended version looks like this:



The Extended version of the Network Interfaces pop-up

Every cell with a light gray background can be edited.

The grandMA3 consoles have three network interfaces, with connectors on the rear panel, that can be used to **connect with external equipment**. The connectors are named **Con1**, **Con2**, and **Con3** in the Network Interface menu.

In the grandMA3 onPC, it is not possible to edit the IP addresses in the Network Interface menu. This needs to be done in the computer operating system.

Short Description of All the Available Columns

- **Lock**:
  Changing the value to **Yes** in this cell locks the interface from being edited.
- **No**:
  This is the interface number.
- **Name**:
  This is the name of the interface.
- **Link**:
  This cell says **Yes** if there is an active connection on the connector. It says **No** in red if there is no connection.
- **DHCP**:
  This cell says **Yes** if DHCP is activated for the interface. No means that DHCP is turned Off, and it is possible to manually set a static IP address information. Read about DHCP and Choosing an Address Range below.
- **Slow**:
  This cell says **Yes** if the detected network speed is below 1 Gbit/s.
- **IP**:
  This is the IP address for the interface. If DHCP is set to "No", this cell can be edited on the console to change the IP address for the interface. CIDR notation is accepted as input to also set the subnet mask. It says "No Cable" if it is disconnected. Read about Choosing an Address Range below.
- **Mask**:
  This is the subnet mask for the interface. If DHCP is set to "No", this cell can be edited on the console to change the subnet mask for the interface.
- **Gateway**:
  This is the gateway address for the interface. If DHCP is set to "No", this cell can be edited on the console to change the gateway address for the interface.
- **MAC**:
  This is the MAC address for the interface. This is information only and cannot be edited.
- **DHCP Timeout**:
  This sets the timeout for the software to wait for a response from a DHCP server. If the timeout is reached, then the software automatically sets an IP from the Link-local address block. Read about DHCP and Choosing an Address Range below.
- **DNS**:
  This is the DNS (Domain Name Server) address.
- **RenewDHCP**:
  This can be set to "Yes". When the pop-up is closed with Apply Changes, then the software will make a new request to a DHCP server. The next time the pop-up is opened, then the field will be empty again. Read about DHCP and Choosing an Address Range below.

What is DHCP

DHCP is a system where IP addresses are distributed from a DHCP server. If this is On (the field says "Yes"), it is impossible to set the IP address locally.

Edit the field to turn On or Off DHCP on the console.

Choosing an Address Range

It is recommended that an active choice regarding what IP address range to use be made. The grandMA3 system is built to connect to the Internet (through a properly set up router) to get access to the world server and fixture shares. With this in mind, choosing an IP address range that falls into the **private address space** of IP Addresses is recommended. The private address spaces are ranges of IP addresses that are reserved for use on local LAN networks, and there should not be any servers on the Internet using these ranges. Communication between private IP addresses is usually not routed to the internet.

The IPv4 addresses can be separated into classes (A, B, C, E, and E). The first three of these classes contain the IP addresses that we can use for setting an address. Each of these three classes has a private address space.

RFC 1918 Name	IP Address Range	Number of Addresses	Largest CIDR Block (subnet mask)	Host ID Size	Mask Bits	Classful Description
24-bit block	10.0.0.0 - 10 .255.255.255	16 777 216	10.0.0.0/8(255.0.0.0)	24 bits	8 bits	Single class A network
20-bit block	172.16.0.0 - 172.31.0.0	1 048 576	172.16.0.0/12(255.240.0.0)	20 bits	12 bits	16 contiguous class B networks
16-bit block	192.168.0.0 - 192.168.255.255	65 536	192.168.0.0/16 (255.255.0.0)	16 bits	16 bits	256 contiguous class C networks

Source: ***Wikipedia - Private_network***

The ranges above are often limited further. For instance, a range using 24 mask bits (or a subnet mask of 255.255.255.0).

Another range option is using the link-local range. The valid IPv4 unicast range is 169.254.0.0 - 169.254.255.255 (or 169.254.0.0/16). This range is best known for automatic addressing processes. The link-local range should not be limited by a smaller subnet mask, so the potential number of devices in the same subnet is quite large (65 536 - 256 - 256 = 65 024).

The network interface card assigns the link-local address when a static address is not set and there is no response from a DHCP server.

The link-local range can be used, but it might be better to set a static IP address using one of the ranges in the private address space.

	Important:
	The most important part is that the grandMA3 devices must be in the same IP address range to communicate with each other.

Set the IP Address in the Console

	Restriction:
	The IP address range 192.168.33.x is not allowed to be used for the

	network interfaces Con1, Con2, or Con3.

1. Navigate to the Network Interfaces menu (read above).
2. Make sure DHCP is turned Off (the cell says "No") for the network connector (read above).
3. Edit the IP cell and write the new IP address (IPv4 only). The CIDR notation can be used to set the subnet mask while typing the IP address. For example, **192.168.101.11/24** gives the subnet mask to **255.255.255.0**.
4. Edit the Mask cell and write the subnet mask.
5. Optionally edit the Gateway cell to set a gateway address if needed.
6. Tap Apply Changes to use the new settings.

The changed IP address is used without the need for a reboot.

	**Hint:**
	For more information about the **Auto** option of the MA Net interface, see **Session**.

1.11.2. Session

MA devices are connected in **Sessions.**

Controlling devices (grandMA3 onPC, grandMA3 replay unit, and grandMA3 consoles) are called **Stations**.

Multiple sessions can exist in the same network and are identified by a session name and location.

	**Hint:**
	Learn more about different ways the grandMA3 system can work in a networked environment in the **System Overview**.

Session Rules

The grandMA3 Session is an essential part of a working system.

The rules for accomplishing and maintaining a stable session are listed below to help understand the system's behavior.

- A session always creates a **Master** station.
- A session always creates a unique, matching **Session Index**, which the user cannot set.
- A session always needs a unique, matching **Session Name** across all members.
- A session always needs a matching **Location** across all members.
- All session members (stations) are displayed with a green background in the **Stations** tab in the **Network Menu**.
- All members must have the same **Streaming Version** of the software - the first three numbers of a software version number.
- Devices can be invited or intentionally join the session.
- Devices can be protected from being invited.
- Devices that lost the connection to their session will be auto-invited when coming back online - if stated in the network list of the show file and the Invite button is activated.
- Two devices with the same Session Index in the same network will cause a session data conflict when both are in Master states. The session data conflict only needs user interaction if the system cannot solve it automatically.
- All DMX network protocols will be output from the **Master** station only.
- A maximum of 32 sessions are possible in one network domain with a maximum of 1 fully loaded session (262 144 Parameter) as a total.
- Consoles in session with the same user logged in will work in **Full-Tracking Mode**.
- Consoles in session with different users logged in will work in **Multi-User Mode**.
- The network needs a latency of less than 2ms for data-package transmission (for synchronization) to avoid output jitter!
- The session timeout (before action will be taken when a station loses connection) is 5s!

Many of the elements above can be seen and edited in the network menu.

MA devices recognized in the network can be seen in the network menu.

The menu can be accessed in multiple ways.

	Important:
	When stations are in network overload, up to three progress bars indicate the stations that are most overloaded. This can be the case when transferring huge amount of data in the session, for example when copying a huge data pool or during show data negotiation.

Open the Menu Using the Command Line

Type the following command in the command line input and execute it:

Open the Menu from the Menu Pop-up

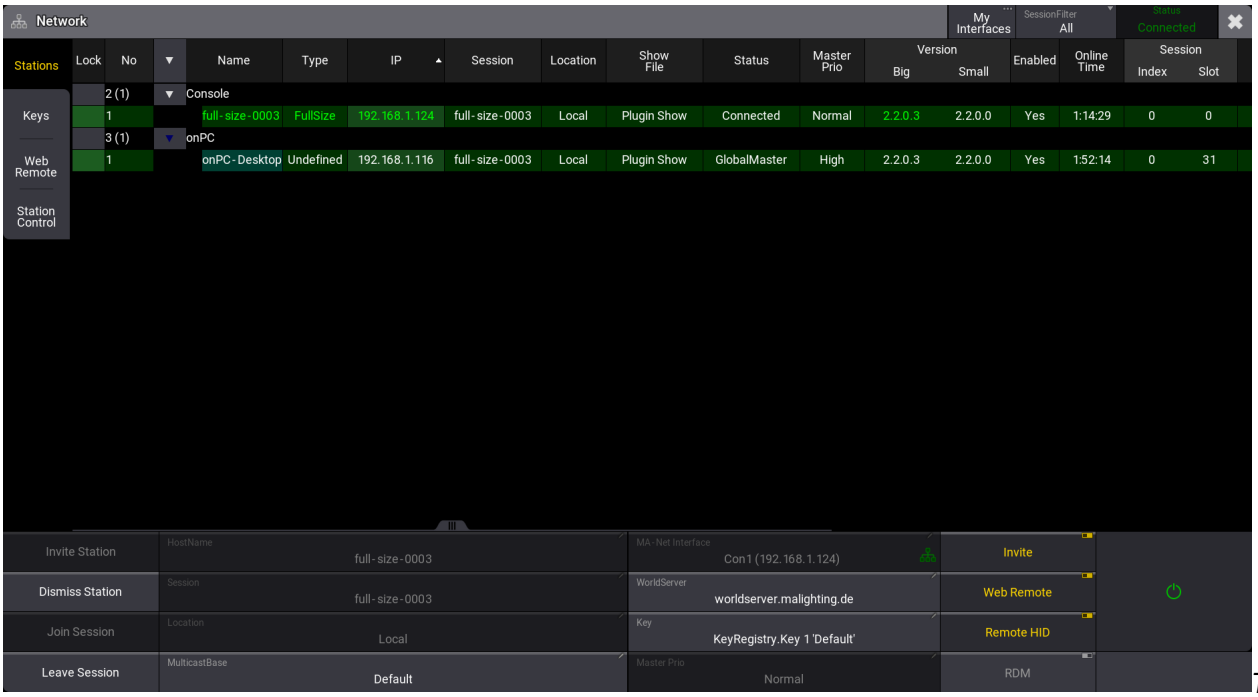
This is a combination of keys and buttons. It executes the same command as above.

1. Press Menu.
2. Tap Network on the menu selection pop-up.

Open the Menu Using the Dedicated Button

The **Command Line** has a dedicated button for the network menu:

Tap this button to open the menu.



The Network menu lists all MA devices in the network

The title bar has the normal close button (✖). Next to this is a button that displays the current status for this device, a filter button, and a button that opens a **My Interfaces pop-up** where the network interfaces are listed and might be modified.

The possible filters are:

- **All**:
  This lists all stations.
- **My Session**:
  This only lists the stations that are in the same session as this station.
- **Not My Session**:
  This lists the stations that are not part of this station's session.
- **Wrong Version**:
  This lists stations that have software version numbers that are different from this station.
- **My Location**:
  This lists the stations that have the same **Location** value. Read about the location setting below.

The tool buttons on the left side have four buttons: Stations, **Keys**, **Web Remote**, and **Station Control**.

There are four buttons on the left side at the bottom: **Invite Station**, **Dismiss stations**, **Join Session**, and **Leave Session**.

Some input buttons change the session settings for the station being operated.

This is a short explanation of each field:

- **HostName**:
  The name of this device. The name can be changed freely. It does not influence the session. It

can be used to identify individual devices. It can only be edited when the station is not in a session.

- **Session**:
  The name of the session. Three elements need to match before devices can connect in the session. The session name, the **Location**, and the **Key**. It can only be edited when the station is not in a session.
- **Location**:
  The session location. It can be anything. It does not need to be the actual location for the session. It is used together with the session name to identify the session. It can only be edited when the station is not in a session.
- **Multicast Base**:
  The multicast base value defines the first three numbers in the multicast IPv4 address for the MA-Net3 session. The options are **Default** and **Alternative**. Learn more about the multicast addresses in the **Protocol Details topic**. Changing this setting while in a session also changes this setting for all grandMA3 devices in the session. Changing it while not in a session only changes the setting for this station.
  Changing the base address opens a pop-up that informs about the possible drawback of the change and asks for confirmation.

	Hint:
	Be careful when changing the Multicast Base address, as it affects the entire network structure. Each grandMA3 device must have the same multicast address to allow communication between the devices.

- **MA-Net Interface**:
  Tapping the Interface field opens a small **Select Interface pop-up**. This lists the possible network interfaces, including an Auto option (Read more about this option **below**). Read more about changing the IP addresses in **Interfaces and IP**.
- **WorldServer**:
  This input field is used to define what world server to contact. Read more about it in the **World Server topic**.
- **Key**:
  The key is a password for the session. Tapping this field opens a small **Select Key pop-up** that lists the possible keys - including the option to create a new one. Read more about key creation in **Create a Custom Key**.
- **Master Priority**:
  This field is used to set the priority for this device. Tapping this field opens the small **Select MasterPriority pop-up**. It can only be edited when the station is not in a session.
  There are several levels of priority: **Never** (not able to be master), **Very Low**, **Low**, **Normal**, and **High**. If stations have the same priority, then the station with the lowest slot number becomes master. Slot numbers are automatically assigned depending on the order of devices joined in the session. The slot number is listed in the station table.

	Hint:
	Each station in a session has a priority. The session needs to have one station, which is the master in the session. If there is only a station with the never priority, then the session is terminated since this station can never become the master.

- **Invite**:
  This On/Off button toggles **Invites**. Disabling this prevents this station from being invited into a session. It is still possible to join a session from this station, but it is an action that needs to be performed locally at the station.

- **Web Remote**:
  This On/Off button toggles the **Web Remote** access. Turning this off prevents web remote connections.
- **Remote HID**:
  This On/Off button toggles **Remote HID**. Learn more about this in the **Remote HID topic**.
- **RDM**:
  This is a global RDM On/Off button. Learn more about RDM in the **RDM section**.
- **Enable button** ():
  This button **enables or disables** the network connections.
  - A red icon means the network connection is disabled. The text beneath the icon describes whether a session is created or joined if the network is enabled.
  - A green icon means the network connection is enabled.
  - A green icon with a red background means the connection is enabled, but the interface is not connected.

Auto Option in MA-Net Interface Select Interface Pop-up

The dropdown for selecting the interface used for MA-Net in the network menu has an **Auto** option.

When the MA-Net interface is set to Auto, the grandMA3 software determines the interface to use. In this case, the IP of the selected interface is set into angle brackets on the MA-Net Interface button, for example, <192.168.0.4>.

The order of the following rules specifies the automatically determined interface in Auto mode:

1. When a Class C IP address (192.168.x.y) is found, the interface with this IP will be taken. No matter the link state of this interface. (With onPC on Mac, the interface link state must be active.)
2. When no Class C IP is found, the software searches for a Class B IP (172.16.x.y). Only if the interface has an active link state.
3. If also no Class B IP address is available, a Class A IP in the range of 10 (10.x.y.z) will be searched only if the interface has an active link state.
4. If this also fails, a Class A IP in the range of 2 (2.x.y.z) will be searched only if the interface has an active link state.
5. The loopback interface will be used if no Class A IP is available.

Changing the selected interface from Auto to the preferred interface is always possible.

After a full install, the first interface will be selected as the default interface on grandMA3 consoles, processing units, and xPort nodes.

Per default, grandMA3 onPC on Windows and macOS will be set to Auto.

The **Stations** view of this menu displays all the MA devices organized in types. The devices with the green background are connected devices.

This is a short description of the columns:

- **Lock**:
  A row can be locked by the user. Editing the field toggles between empty and "UL" (User Locked).
- **No**:
  This is the row number. Notice that a row can be a child of a parent object. If there is a number in parentheses, then it is the number of children in the parent object.
- **Name**:
  This is the name of the device or device type.
- **Type**:
  This shows the type for each device. OnPCs might be listed as "Undefined".
- **IP**:
  This is the IP address for the device's MA-Net interface.
- **Session**:
  This is the name of the session the device is a part of.
- **Location**:
  This is the location for the session. For devices to connect, the session name, location, and key must match.
- **Show File**:
  This is the name of the active show file on the device.
- **Status**:
  This is the device's session status.
- **Master Prio**:
  This is the priority for the station.
- **Version Big**:
  This is the software version for the device. The **Version Big** number indicates the version number that needs to match for stations and processing units to connect in a session.
- **Version Small**:
  This is the software number that needs to match for grandMA xPortNodes to be able to connect to a session.
- **Enabled**:
  This shows if the network connection is enabled.
- **Online Time**:
  This is the time the device has been online in the session. When the station connects to a session, then the online time resets.
- **Session Index**:
  Each session is automatically assigned a number. This is the session number for each device. There is a maximum of 32 sessions in the same network.
- **Session Slot**:
  Each station in a session has a unique slot ID. This is the ID number for each device.
- **Remote IP**:
  Some MA devices are connected to a station. For instance, the grandMA3 extension. This field shows the IP address of the station these devices are connected.
- **Mask**:
  This is the station's sub-net mask.
- **Flow Control Level**:
  This shows information about congestion and data loss in a session. It displays, on a scale of 0 to 255, the intensity of flow control.
- **NACK Count Per 1m/5m/10m**:
  This shows information about negative acknowledgments (Nack). Nack's can happen when network packages fail to properly get though the network.

This shows the amount of nacks in the last one, five, and ten minutes. It can be reset by editing the cell. This also resets the total count (read below).

- **NACK Count Total**:
  This shows the total amount of nacks since the session started. It can be reset by editing the cell. This also resets the nack counters described above.

	Important:
	Please check the network environment of a device if rates of nacks are constantly above 0.

Session Maximums

There are some maximum numbers for a session. These describe some limits in a session that can be good to know.

There should only be one fully loaded session in a network domain. If more is needed, great care should be taken to ensure the network can manage the traffic without creating bottlenecks.

These numbers are for one session.

A session can control up to **262 144 parameters**. The **Parameters topics** explain this and how the parameters can be expanded to the limit.

MA Devices

All MA devices in a session belong to one of the following three categories.

Category A - maximum **32 devices** in a session.

These stations and devices calculate the show file information and parameters for the output. This includes grandMA3 consoles, grandMA3 replay-units, grandMA3 onPC stations, and grandMA3 processing units.

	Hint:
	When adding several devices to the session simultaneously, they will be added following their internal order in the software. This order is represented by the No column in the Network menu.

A grandMA3 onPC counts as one station, no matter how many onPC wings are connected. The grandMA3 onPC wings do not count as devices in the session.

All category A devices should be on the same switch or at least on directly connected switches.

Category B - maximum **64 devices** in a session.

Depending on their task, these stations and devices handle specific show file information. This includes grandMA3 consoles, grandMA3 replay-units, and grandMA3 onPC stations which have session priority “Never”.

Also, grandMA3 processing units can be in category B. The system automatically moves processing units into this category whenever category A is full. In this way, the maximum parameter count in a session can be reached, no matter how many processing units of a specific type (M, L, or XL) are needed.

Category C - maximum **128 devices** in a session.

These are MA devices that use the calculated parameters and are in session. MA Nodes running in grandMA3 mode are category C devices.

	**Important:**
	Network nodes that run in sACN or Art-Net mode do not count against the limitation.

Subtopics

- **Create a Session**
- **Join a Session**
- **Leave a Session**
- **Invite a Station into a Session**
- **Dismiss Stations from a Session**
- **Create a Custom Key**
- **Session Master Selection**

1.11.2.1. Create a Session

Requirement:

Network connections need to be made, and the correct IP address should be set - For more information, see **Interfaces and IP**.

Create a Session with the User Interface

1. Open the Network menu.
2. Tap the buttons at the bottom that need to be edited.
   - Make sure that the correct interface is selected.
   - Make sure the Session name and Location values are correct.
   - Make sure the correct Key is selected.
3. If the network connection is disabled (the network icon () in the **Command Line** is red), then tap the big enable/disable button in the lower right corner of the Network menu.
   - The network icon should become green.

1.11.2.2. Join a Session

Requirement:

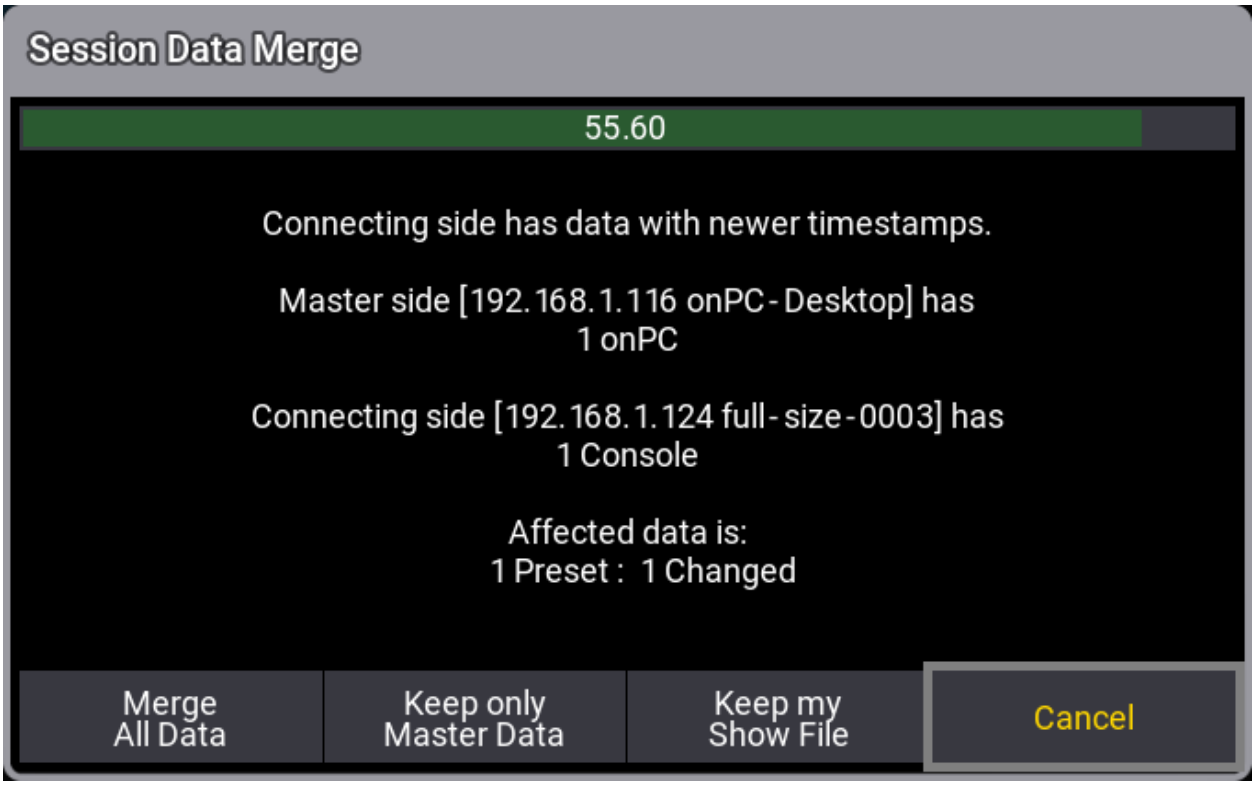
Network connections need to be made, and the correct IP address should be set - For more information, see **Interfaces and IP**. There needs to be a running session that can be joined.

When a console or onPC station wants to join a session or gets invited into a session, the session recognizes that the station joining was already a part of the session at a previous point.

If this is not the case, the station joins the session and gets the show file.

If the same show file is loaded on both sides, the grandMA3 software can merge both show files. This is useful in undetected scenarios, such as a station being disconnected for a time from the rest of the session (maybe because of a defect in the network cable or other causes of failure). If, on both sides (the remaining session and the disconnected station), users worked further on the show file, they had to choose between one of the two show files. This means they had to reprogram the changes of the lost show file to the one that is kept. The Merge All Data option combines the data of both show files in one show file.

This is the small pop-up asking what to do:



Session Data Merge pop-up

The **Session Data Merge** pop-up is visible for 60 seconds. After 60 seconds without a response, the session data merge will be canceled.

To pause the countdown, tap in the text area of the pop-up. When a station, as described above, joins the session again, a pop-up informs the user about a Session Data Merge.

The pop-up lists the affected data. It is the elements that are different between the shows.

There are four options to choose from:

- **Merge All Data**:
  Merges the changes of all connecting stations into one show file.
- **Keep only Master Data**:
  The changes in the show file data of the connecting devices are ignored. The show file of the Master side is maintained. Before overriding the show file of the connecting devices, the show file will be saved automatically on the connecting stations.
- **Keep my Show File**:
  Choosing this option means that the show data on this station will be the new session show data. This deletes any unsaved data on the other stations.
- **Cancel**:
  The connecting stations will not join the session of the Master side and be made Standalone. Then it can be investigated which show file is correct, back up each show file, etc.

Merging the data of two show files is supported for these object types at the moment:

- Sequences with Cue and Cue Part
- Presets
- Recipes
- Programmer and Programmer Parts
- Timecodes
- Macros, Plugins, and PluginComponents
- Pages and Executors
- Images and Videos
- DataPools
- Groups
- All other pools not explicitly mentioned here

	Known Limitation:
	- Changes made to the patch during the disconnection of a station disables the possibility of merging. - Merging the deletion of parts of objects during the disconnection of a station is not fully working yet. (for example, deleting one macro line or deleting a recipe line) - Merging the movement of objects during the disconnection of a station creates duplicates: At the original spot and the new spot. This may result, for example, in having different sequences assigned to the same executor on different stations.

Join a Session Using the User Interface

1. Open the Network menu.
2. Tap the buttons at the bottom that needs to be edited:
   - Make sure that the correct interface is selected.
   - Make sure the correct Key is selected.
3. Tap the station with the desired running session.

4. Tap Join Session.
   - The network icon and the stations should become green.
5. If the **Session Data Collision pop-up** appears, then select the desired option.

Join a Session Using the Command Line

1. Use the **JoinSession keyword** to join existing or create a new session.

	Restriction:
	It is only possible to join a session if the joining station has the same key as the currently active key on the selected station (step 3 above) currently in the session. This means that the station that wants to join a session selects a station already in the session. The station already in the session got a selected key. The joining station needs to have the same key selected to be able to join the session.

1.11.2.3. Leave a Session

Requirement:

The station needs to be a session member to be able to leave the session.

Leave a Session Using the User Interface

1. Open the **Network menu**.
2. Tap Leave Session or turn the network Off using the network connection button.

Leave a Session Using the Command Line

- Use the **LeaveSession keyword** to leave the session.

	**Important:**
	When leaving the session during an active show download, the station will fall back to the previously loaded show file that was loaded before the show download started.

If there are other stations in the session with a priority above never, then the session is still active.

1.11.2.4. Invite a Station into a Session

Stations can be invited into a session if **Invite** is turned On in the **Network menu** on the station being invited.

	**Important:**
	If the priority of the joining station is higher than the current session master, the show file from the joining station will be used instead of the show file used in the session before the invite.

Requirement:

A session needs to be running, and the station being operated needs to be part of the session. See how in the **Create a Session topic**.

Using the User Interface

Stations are located and invited using the **Network menu**.

1. Navigate to the Network menu. See how to use the link above.
2. Locate and tap the station in the list of available stations (stations in the network but not connected to a session).
3. Tap Invite Station at the bottom of the Network menu.

Using the Command Line

Stations are invited using the **Invite keyword**.

	**Restriction:**
	Inviting a station into a session is only possible if the joining station has the same key in the key registry as the one currently selected on the inviting station when the Invite button is tapped.

1.11.2.5. Dismiss Stations from a Session

Stations that are part of a session can be dismissed by another station.

Requirement:

A session needs to be running, and the station being operated needs to be part of the session. The station to be dismissed also needs to be in the same session. See how to create sessions in the **Create a Session topic**.

Using the User Interface

Stations are located and dismissed using the **Network menu**.

1. Navigate to the Network menu - see how using the link above.
2. Locate and tap the station in the list of stations that are connected to the session (stations with a green background color).
3. Tap Dismiss Station at the bottom of the Network menu.

The selected station is now kicked out of the session.

Using the Command Line

Stations are dismissed using the **Dismiss keyword**.

1.11.2.6. Create a Custom Key

Keys are passwords for sessions.

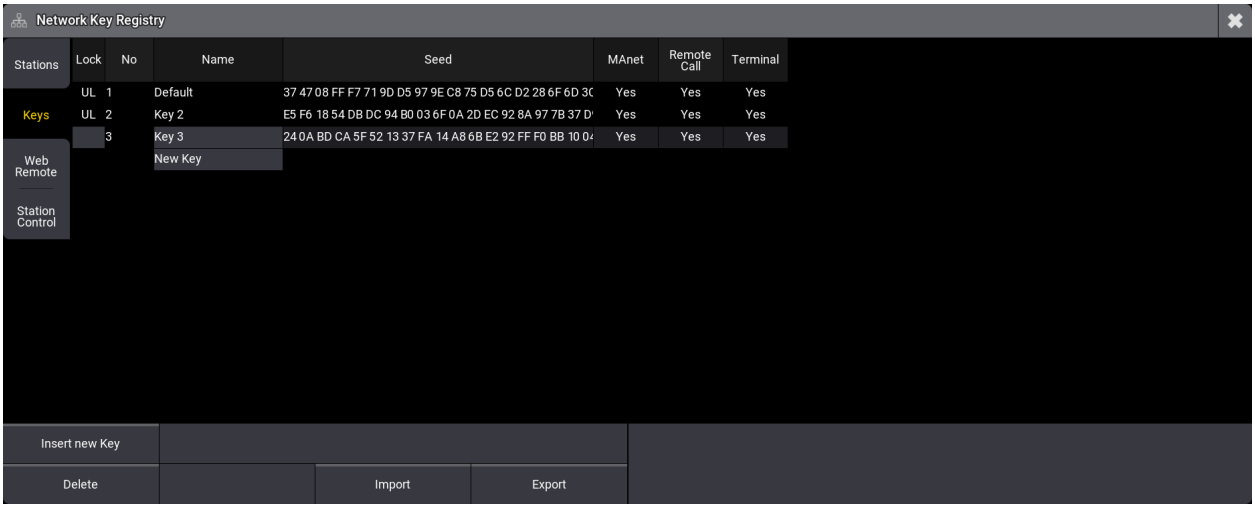
The same default key is a part of every new show file. It is used unless a custom key is created and used instead.

The key is a long hexadecimal hash value generated by a seed word or a set of characters.

Keys can be Inserted/Created, Deleted, Imported, and Exported. Locked keys cannot be edited or deleted.

Keys are used when a **session is created** and when stations are **invited**.

All the described key operations are done from the **Network Key Registry** menu.



Open the Network Key Registry from the Network menu

Open the Network Key Registry Menu

1. Press the Menu key.
2. Tap the Network button.
3. Tap the Keys button on the left menu.

This opens a menu like the one in the image above.

There are a few columns in the menu:

- **Lock**:
  This column indicates if the key is locked. The default key is locked as a default. When a seed is changed, the key is automatically locked. Edit this field to unlock a key.
- **No**:
  This is the key number.
- **Name**:
  This is a user-friendly name for the key. The name can be edited if the key is unlocked.
- **Seed**:
  The seed is a hash value. This can be edited on unlocked keys. Any text input can be a seed to

create a hash value. The hash value cannot be manually input as this would create a new hash using the input as a seed. The original seed input needs to be known to be able to recreate the hash manually. The default value for a new key is the same seed value as the default key.

- **MAnet**:
  When this is set to **Yes**, the key gives access to the MAnet area. This is, for instance, the possibility of joining a session.
- **Remote Call**:
  When this is set to **Yes**, the key gives access to the Remote Call area, which is, for instance, the possibility to use the web remote into a station, network updating the software, **RemoteCommand**, and other commands that affect other stations (for instance, reboot and shut down of other devices).
- **Terminal**:
  When this is set to **Yes**, the key gives access to the Terminal area. This is, for instance, access to a station's terminal interface.

	**Important:**
	Please leave the **MAnet**, **Remote Call**, and **Terminal** columns set to "Yes" for the default key. Changing these might give unexpected results.

Using Keys in Sessions

The key is selected before the session is started. If a custom key is selected, other stations need to have the key in the registry list to join or be invited into a session.

The keys can be exported to a USB memory stick and imported into other stations.

Or, if the seed is known, the custom key can be recreated on other stations using the same seed value.

If a station does not have the key used when the session was created, it cannot be a part of it. The joining station does not need to have the key selected in the **Network Menu** but must be in the **Network Key Registry** list.

Requirement for the next few operations:

The **Network Key Registry** menu is opened.

Create a new Key

Keys are created like most new elements.

1. Select the row where the key should be inserted.
2. Tap Insert new Key.
3. (Optional) Edit the **Name** field to give the key a useful name.
4. Edit the **Seed** field and type a word or a random set of characters. This input is converted to a hash number.

Notice that the row is locked as soon as the seed field changes. It can be unlocked and edited afterward. The lock is to prevent accidental change of the key.

	**Hint:**
	If the same seed text is applied, the key can be reproduced on other

	stations.

Delete a Key

A locked key cannot be deleted. Follow these steps to delete a key.

1. Select the row with the key to be deleted.
2. Edit the **Lock** field to unlock the row - the field should be empty for unlocked.
3. Tap Delete.

The key is now deleted.

Export a Key

Keys can be exported and brought to other stations manually.

1. Select the row with the key to be exported.
2. Tap Export.
3. Select the desired drive in the pop-ups title bar.
4. Give the exported file a name.
5. Tap Export in the pop-up.

The key is now exported to the selected drive.

Import a Key

Keys can be imported into a station.

1. Select the "New Key" row.
2. Tap Import.
3. Select the desired drive in the pop-ups title bar.
4. Select the desired file.
5. Tap Import in the pop-up.

The key is now imported to the station.

1.11.2.7. Session Master Selection

Some rules define what station becomes the master of a session.

A key element to this is the stations' priorities.

Inviting Stations into a Session

A session always has a master. If there is only one station in the session, it is **IdleMaster**. If there are more stations, it is **GlobalMaster**.

Adding another station to the session can have different outcomes:

- The joining station has a **lower priority**:
  The station joins the session and downloads the showfile. The station's status becomes **Connected**.
- The joining station has the **same priority**:
  The station joins the session and downloads the showfile. The station's status becomes **Connected**.
- The joining station has a **higher priority**:
  The station joins the session, takes over the **GlobalMaster** status, and uploads its show to all connected stations. The show is replaced.

Session Master Selection Rules

A set of rules takes effect in a situation where a new master needs to be selected among the remaining stations.

This situation can appear if the current session master disappears. For instance, if the network connection disappears.

It also happens when several stations booting up were previously part of a session.

Three rules define how the master is selected. They are in prioritized order:

1. The station with the highest priority wins. If there is more than one station with the same priority, go to rule two.
2. The stations' online time defines the master. The highest online time (+/- 5 seconds) wins. If more than one station has the same online time (+/- 5 seconds), go to rule three.
3. The station with the lowest IP address becomes the master.

The online time can be seen in the **Network Menu**.

1.11.3. Web Remote

The stations can be remote-controlled by any modern browser connected to the system. This includes Wi-Fi-connected devices.

A maximum of five devices can connect to one station simultaneously.

Each remote device connects to a specific station. The remote browser shows the same controls and encoder view at the bottom of the display as the grandMA3 Compact, grandMA3 Compact XT, and grandMA3 onPC display 1. The user-definable area is the same as screen 1 on any console or onPC. The web remote will try to log in as the user **Remote** when connecting, and the view and windows on screen 1 follow the logged-in user's screen configuration. The web remote can log in as a different user than the one currently logged in on the connected station. Learn more about multiuser setups in the **Single User and Multi User Systems section**.

	Restriction:
	Admin, Guest, and 3D users cannot log in to web remote devices.

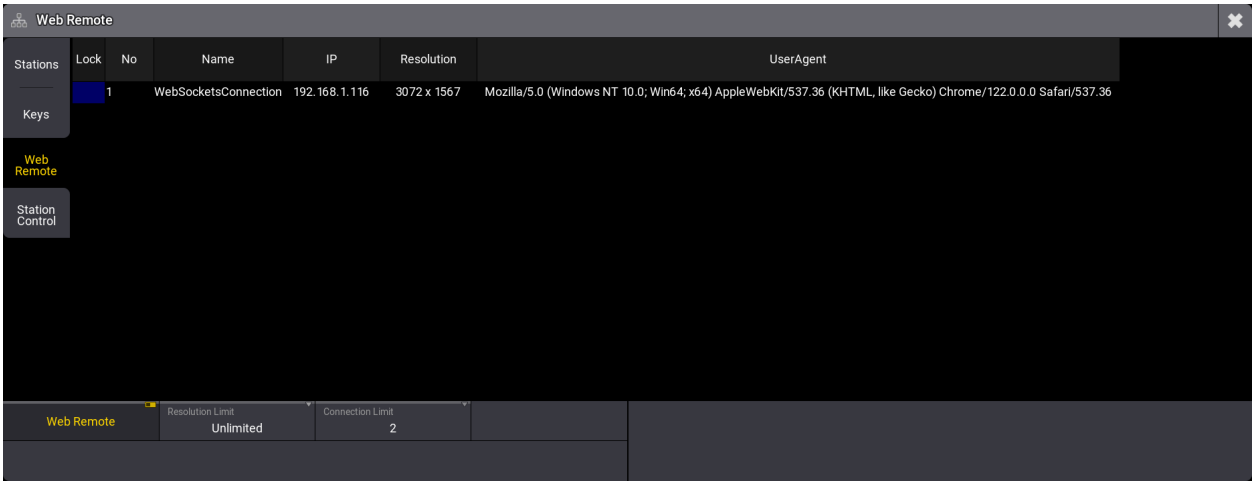
Any window or view that can be created on a console or onPC can be used in the remote. The system will try to provide a high refresh rate of the content, but it is affected by the connection speed and the resolution.

Connect with a Web Remote

Web remotes need to be enabled in the station. This can be done in the **Network Menu** - see the **Session topic** to learn how to access this menu.

The network menu and **Station Control** have a toggle button that turns the web remote On or Off.

The same setting is mirrored in the actual **Web Remote** menu, which can be accessed by tapping Web Remote on the left side of the network menu.



Web Remote menu with one connection

1. Make sure web remote is turned On in the station.

2. Open a browser on a network-connected device.
3. Type the IP address of the station and include **:8080** after the address to specify the port number.
   The result should be a browser window with a grandMA3 display.

Now, there is access to remote control of the station with the user rights provided by the logged-in user.

A new show file has a user called **Remote**. This uses the default user profile but with a different screen configuration. Remote is also a keyword, so if it is desired to log in as this user, type the following in the command line:

MA	User name[Fixture]>LogIn "Remote"
	**Hint:**
	Putting the user name in quotation marks is essential for the software to treat it as a name, not as a keyword.

It is possible to log in as a specific user, including a user password directly in the URL typed in the browser. This is the format:

http://[IP]:8080/?user=[User name]&password=[password]

For example,

http://127.0.0.1:8080/?user=McClane&password=YippeeKiYay

The Web Remote menu shows the connected devices. It is just information about the IP address, operating system, and browser.

Three different settings can be changed in the menu.

The remote connection can be turned On or Off.

A **Resolution Limit** can be set. There are options to limit it to 480p, 720p, 1080p, or unlimited. This limit is the same for every device that connects. If unlimited is selected, the resolution is adopted to the device screen.

Connection Limit can be used to set a number between 1 and 5. This is the maximum amount of simultaneous connections allowed on the grandMA3 onPC. A grandMA3 console or grandMA3 replay unit have a maximum of 2 connections.

Linked Command Lines

A button on the connected remote device is next to the host command line input.

This can be activated to link the remote command line input to the host command line input.

Linked command lines change the background color on both the remote and the host station. It could look like this:

Linked command line example

Notice that the command input in the image above is not a valid command. It is just an example.

Any command input on any linked command line is immediately shown on all the linked command lines where the users are logged in with the same user profile. Any commands are executed at the host station.

1.11.4. Network Design

Creating a good working network for a grandMA3 system can be complex when the system expands.

Small systems might not require expensive switches and infrastructure to work, but pushing the grandMA3 system also pushes the network infrastructure.

General Control System Requirements

- The output of an MA Lighting control system only changes by intention (of the user).
- The output of an MA Lighting control system complies with the standards, proprietary open standards, and protocols (see a complete list in the **Regulations and Standards topic**):
    - DMX512-A.
    - sACN.
    - SMPTE Timecode.
    - Open Sound Control (OSC).
    - DIN SPEC 15800 (GDTF).
    - Art-Net.
    - PSN.
    - MVR.
- The output of an MA Lighting control system is synchronized for all physical output ports over all attached MA devices and all transmitted protocols.
- The output of an MA Lighting control system is considered to be “real-time” (The term “real-time” can be defined by: All latencies between input events and output reactions are determined and known.)
- Show files are upwards compatible, and exceptions shall be announced. (This might not always be possible due to rare technical reasons. If so, MA will state this with the release notes of the software update as ‘known limitations’!)

The list above defines a set of rules that must be true within a correctly configured network.

Network Speed

The amount of data sent in a grandMA3 network can vary depending on the number of grandMA3 devices and parameters.

A grandMA3 system is always defined based on one (1) maximum utilized session in a network. This maximum is defined in the **Session topic**.

With this in mind, a network speed of 1 Gbit/s is required for the MA-Net3 protocol. This means that the ports on the switches should be configured as "Access" ports and have a minimum capacity of 1 Gbit/s. The switches should be "non-blocking".

The maximum latency on the "broadcast domain" should be 2 milliseconds or less.

The MA-Net3 protocol uses multicast, and all grandMA3 devices support IGMP membership requests.

All devices should use the same bandwidth capacity. This means that if there is an older network node translating a DMX streaming protocol (like Art-Net) only capable of 100 Mbit/s speed, this protocol should be separated to a different VLAN (or network) limited to this speed.

In a Gigabit network, the average bandwidth reserved for MA-Net3 is 200 Mbit/s. MA-Net3 does not allow for traffic shaping.

Cable Quality and Lengths

Since the network speed is defined as 1 Gbit/s, the quality of the copper cabling should be Cat.5e or better.

The cable length is defined in a set of standards for fixed installations (EN 50173). These specify a set of rules for installations that should be followed.

For “flying installations” or "one-offs", a different type of cable is used (stranded core / "patch" cables). This type does not support the same lengths as a proper fixed installation (using solid core cables). Using a maximum length of 75 meters / 246 feet between devices is recommended for these systems. This length is tested by MA Lighting using new quality cables. If the cable is damaged, it should not be used.

These lengths are about the distance between devices, such as a grandMA3 console and the switch.

Connecting switches can have higher demands. The speed between the switches might be higher than 1 Gbit/s and thus have higher quality demands on the cabling.

Switch Settings

If more than one switch is used, and the switches are used for more than MA-Net3, then it is recommended to separate the MA-Net3 into a different VLAN.

Please consider that the MA-Net3 should have 1 Gbit/s available, so if the switches do other things as well, there needs to be more than 1 Gbit/s connection between the switches.

When using multiple switches, please ensure the switches support Protocol Independent Multicast (PIM) to route the multicast correctly between the switches.

Switches should have all ports go online simultaneously after a reboot. Delayed port-by-port reboot behavior is not supported at this time.

The IP addresses used in a MA system should follow the recommendation RFC-1918 about address allocation in private internets.

If the MA Worldserver is desired, then a VLAN (or separate network) should be used. This network segment should be routed to the Internet.

EEE or Green Ethernet

Energy-Efficient Ethernet (IEEE 802.2az) or Green Ethernet is a system supported by some network switches. The idea is to save some energy on the network ports that do not have a lot of traffic. This includes some negotiations between the switch and the connected device. Unfortunately, this does not

always result in a successful setting, and the network port could be disabled and enabled when the negotiation fails. This is not good in a MA-Net3 setup.

Disabling any energy-saving functions in the network switch is highly recommended.

If the switch is unmanaged with EEE enabled, it is recommended not to use it.

Bandwidth Calculation for Best Practise Networks

The DMX output of MA devices needs a refresh rate of 30Hz. Therefore all “real-time” data needs to be present within 33ms.

To avoid any visible jitter, the synchronization data packet is allowed to have a maximum latency over the whole system of 2ms.

Within each refresh of 33ms, all necessary data must be transferred to the end devices.

The maximum possible number of parameters will be 262 144 with 24-bit DMX data. Only a mixed calculation is possible with a 1 024 DMX universe limit. The maximum for 262 144 parameters each of 16bit will end up in 1024 DMX universes and therefore:

1 024 Universes * 512 packets * 10 bit * 30 Hz = 158 MBit/s

This will be the average bandwidth needed to allow the “real-time” traffic (as described in the requirements above) to pass through. At the same time, the latency cannot be more than 2ms. This is a worst-case scenario with the maximum possible DMX data.

There is more data to be transmitted than just the DMX data; therefore, the average bandwidth for Gigabit systems is set to 200 MBit/s.

Higher bandwidth for faster show upload might be required, especially in bigger installations.

More bandwidth is needed if additional DMX-over-Ethernet protocols are used in parallel and/or Web-Remote(s).

The calculation will be the same, and the bandwidth of additional DMX-over-Ethernet data must be added.

DCSP and QoS

The different types of grandMA3 network traffic is divided into five categories. Each category can have its own setting defining how the switches should prioritize the network packages for this category. These settings can alter the traffic flow and should only be done if a specific need exists. Learn more in the **Station Control topic**.

Testing the Network Environment

A grandMA3 network system will be recognized as a smooth operating system if there are no data retransmissions and no DMX dropouts on any device in the session.

Any retransmission of data packages in the overall system will be shown in the System Monitor of every device within the session. Session stability and reliability are measures of network quality and vice versa.

The **Network Menu** has columns showing the number of negative acknowledgments (NACKS) for each connected station in the system. NACKS are indicators of packages being dropped or not getting through to the recipient.

The **Network Menu** is used to see all the stations and sessions in a network. Learn more in the **Session topic**.

Retransmission means a data package has not reached its destination in order or at all. The question arises of which retransmission count is acceptable. In other words, when to worry about network stability.

The answer cannot be a quantity – the answer is a situation-dependent measure.

Here are some guidelines for quality insurance of the network session handling:

- A few Retransmissions (0 – 50) on a regular or irregular basis – The system is considered stable.
- Multiple Retransmissions (> 50) on an irregular basis – The system has issues.
- Multiple Retransmissions (>100) on a regular basis – The system needs investigations!
- High number of Retransmissions (> 200) on an irregular basis – The system needs investigations!

	Important:
	During a show upload, the number of retransmissions can be very high due to the heavy use of communication on all layers, and retransmissions are expected. The system is considered to be stable.

A network speed test can be performed to test the network speed. This will test the connections between the station activating the test and the selected stations. The result will show if the connection has issues. Do not perform this test while in a show. Learn more in the **NetworkSpeedTest keyword topic**.

DMX dropouts are not allowed within the session at any time.

Subtopics

- **Protocol Details**
- **Wireless Networks WLAN - WiFi**

1.11.4.1. Protocol Details

The grandMA3 system supports different network protocols.

	**Hint:**
	See a list of protocols in the **Regulations and Standards topic**.

The following are some details about each protocol that might be useful for setting up rules in network switches and routers.

MA-Net3

General Traffic

Transport layer: UDP - Multicast

Port number: 30020

IP addresses (default Base address):

- 236.4.1.0 - Administration group for all devices and sessions
- 236.4.1.1 - General communication (session data)
- 236.4.1.2 - Extended communication (session data)
- 236.4.1.3 - Extended DMX (session data)
- 236.4.1.4 - Reserved (session data)

Additional Information:

The IP addresses mentioned above belonging to the session are for session index number 1. Subsequent session indexes use different addresses using the formula "BaseIP+4*X" where "X" is the session index number, and "IP" is the address mentioned above. This only changes the last octet of the address.

So the default IP addresses specific for session index 3 are:

- 236.4.1.13
- 236.4.1.14
- 236.4.1.15
- 236.4.1.16

This allows for 32 sessions in a network.

Alternate Multicast Base

The default multicast base address (236.4.1.X) is within an IANA reserved block (according to RFC5771 - IANA Guidelines for IPv4 Multicast Address Assignments).

If this causes any problem in the network environment, then an alternative multicast base address can be selected in the **Network Menu**.

The option called **Alternative** uses the 239.4.1.X address as a base. The fourth number is the same as described above.

Alternate Traffic

Transport layer: TCP - Unicast

Port number: 30022+

Additional Information:

This is used when a station needs to send a lot of information to a different station, for instance, with a network update.

MA Worldserver

Transport layer: TCP - Unicast

Port number: 30021

Additional Information:

This must be routed to the Internet to access the official worldserver from MA Lighting.

MA Web-Remote - HTTP

Transport layer: TCP - Unicast

Port number: 80

MA Web-Remote - Websocket

Transport layer: TCP - Unicast

Port number: 8080

Art-Net (revision 1.4db)

Transport layer: UDP - Broadcast

Port number: 6454

and

Transport layer: TCP - Unicast

Port number: 6454

Additional Information:

In Auto-Mode, grandMA3 Art-Net will be sent as Unicast unless there are more than five receivers (default value in Broadcast Threshold) of the same DMX universe in the network. Then the protocol switches automatically to Broadcast.

The default Broadcast Threshold is set to five but can be changed in the **Art-Net menu**.

sACN

Transport layer: UDP - Multicast

Port number: 5568

IP addresses: 239.255.x.y

and

Transport layer: TCP - Unicast

Port number: 5568

Additional Information:

The default uses a Multicast address for each sACN DMX universe corresponding to a specific universe. The first two numbers in the IPv4 address are fixed. The two following numbers are calculated to match the universe number.

For instance, sACN universe 1 is on IP 239.255.0.1, and sACN universe 256 is on IP 239.255.1.0

PSN - PosiStageNet

Transport layer: UDP - Multicast

Port number: 65565

IP addresses: 236.10.10.10

Additional Information:

The Multicast address and port number can be changed in the **PSN menu**.

OSC - OpenSoundControl

Transport layer: TCP / UDP

Port number: 8000

Additional Information:

The transport layer protocol, address, and port number can be changed in the **OSC menu**.

NDI - Network Device Interface

mDNS communication

Transport layer: UDP - Multicast

Port number: 5353

IP addresses: 224.0.0.251

Alternate Discovery

Transport layer: TCP - Unicast

Port number: 5959

Base TCP Connection for Stream

Transport layer: TCP - Unicast

Port number: 5961+

UDP Shared with TCP Connections

Transport layer: UDP - Multicast

Port number: 5960+

UDP / TCP Receiving

Transport layer: UDP / TCP

Port number: 6960+

UDP / TCP Sending

Transport layer: UDP / TCP

Port number: 7960+

MVR-xchange

mDNS communication for discovery

Transport layer: UDP - Multicast

Port number: 5353

IP addresses: 224.0.0.251

MVR-xchange

Transport layer: TCP

Port number: 42424

Additional Information:

Only TCP exchange is supported at the moment. Learn more about MVR-xchange in the **MVR-xchange topic**.

SFTP

Transport layer: TCP - Unicast

Port number: 22

SNMP

Transport layer: UDP - Multicast

Port number: 161 and 162

Additional Information:

SNMP v1 and v2c. Public community.

AVAHI

Transport layer: UDP - Multicast

Port number: 5353

1.11.4.2. Wireless Networks WLAN - WiFi

A wireless connection should only be used with the web remote.

Do not try to run a session on a WiFi connection. Please read the **Network Design topic** and understand why it is not possible.

Wireless networks (WiFi) manipulate the data packages to optimize the wireless network. This works for standard HTTP and HTTPS traffic but not for MA-Net3.

Please use proper access points for the wireless network.

The access points shall be connected to a port on a network switch, and all access points should be in a separate VLAN. This ensures that all multicast traffic can be filtered out of the wireless network.

The access points must be filtered to only allow MA Web Remote traffic “on air” to avoid delays caused by too much traffic between the wireless device used for MA Web Remote and the rest of the MA System. This filtering could be done in the switch or the access points.

In the software, the maximum resolution transmitted for the Web Remote can be set (and limited) to allow for better performance in a slow wireless network - see the **Web Remote topic**.

1.11.5. Regulations and Standards

The grandMA3 system conforms to and suggests a series of network-related regulations and standards.

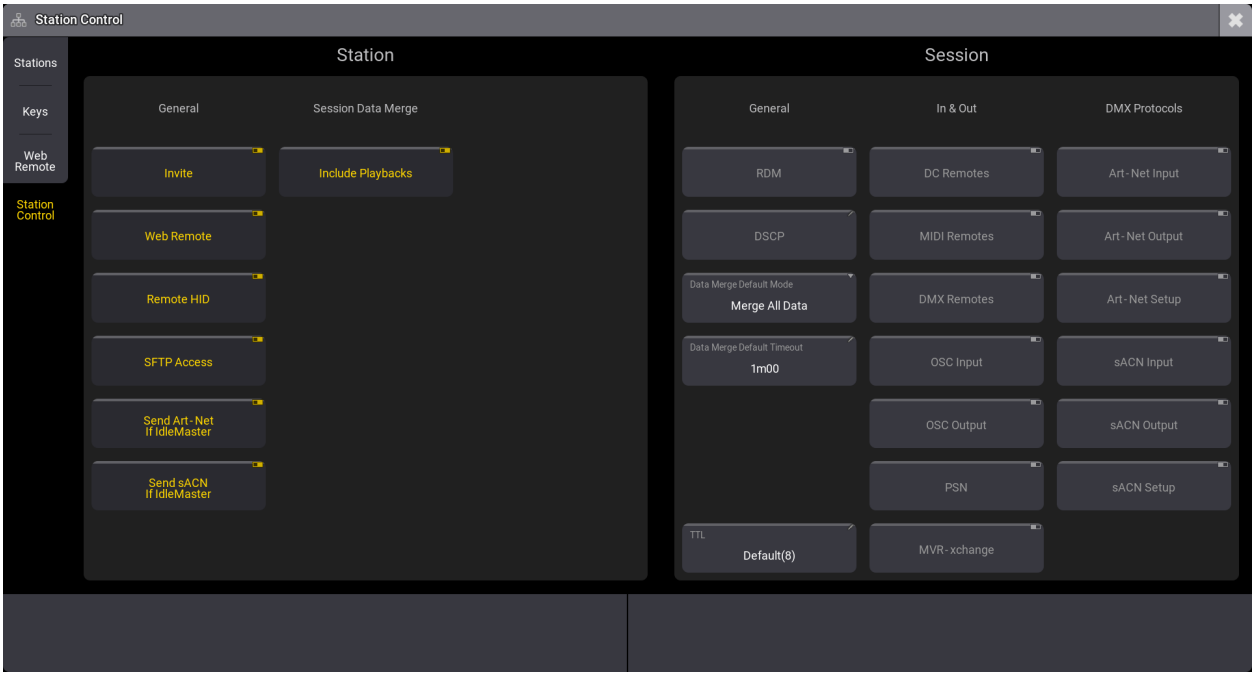
- ISO/IEC 11801:2002 - International standard for structured data cabling
- EN 50173 - European standard for structure data cabling
- ANSI/TIA-568 - Standard for commercial building cabling products and services
- ANSI/IEEE 802.3 IEEE - Standard for Information Technology (Cabling)
- ANSI/IEEE 802.11 IEEE - Standard for Information Technology (WiFi)
- RFC 1918 - Address Allocation for Private Internets
- RFC 2236 - Internet Group Management Protocol
- RFC 4594 - Configuration Guidelines for DiffDerv Service Classes
- ANSI E1.11 - 2008 (R2018) Entertainment Technology USITT DMX512-A Asynchronous Serial Digital Data Transmission Standard for Controlling Lighting Equipment and Accessories (DMX)
- ANSI E1.31 - 2018 Entertainment Technology Lightweight streaming protocol for transport of DMX512 using ACN (sACN)
- ANSI E1.20 - 2010 Entertainment Technology RDM Remote Device Management Over DMX512 Networks (RDM)
- Art-Net 4 - revision 1.4db
- PSN 2.03
- NDI 4
- DIN SPEC 15800:2022 – Entertainment Technology - General Device Type Format (GDTF)

Some of the protocols have some more details in the **Protocol Details topic**.

1.11.6. Station Control

The Station Control menu gives quick access to settings regarding the station and session communication.

Many of the settings in this menu can be found at other locations in the system.



Station Control menu The menu is split into two sides. The left side is settings related to this specific station. The right side is settings that relate to the entire session.

Station

The station side has two columns: **General** and **Session Data Merge**.

General has the following toggle buttons:

- **Invite**:
  This is the same as the **Invite** setting in the **Network Menu**.
- **Web Remote**:
  This is the same as the **Web Remote** setting in the **Network Menu**.
- **Remote HID**:
  This is the same as the **Remote HID** setting in the **Network Menu**.
- **SFTP Access**:
  This setting toggles whether this station can be accessed using the SFTP connection. Learn more about SFTP in the **SFTP Connection topic**.
- **Send Art-Net If IdleMaster**:
  This is the same as the **Send Art-Net If IdleMaster** setting in the **Art-Net Menu**.
- **Send sACN If IdleMaster**:
  This is the same as the **Send sACN If IdleMaster** setting in the **sACN Menu**.

Session Data Merge only has one toggle button:

- **Include Playbacks**:
  This defines if the changes of playbacks are to be considered in the **Session Data Merge**. If it is enabled, the changes made on playbacks will be taken into account.

  Session Data Merge happens when stations connect in a session, and the data from two stations are merged.

Session

The session side has three columns: **General**, **In & Out**, and **DMX Protocols**.

General only has the following buttons:

- **RDM**:
  This is the same as the **RDM** setting in the **Network Menu**.
- **DSCP**:
  DSCP settings define the Quality of Service (QoS) setting for the data packages. This is information added to the package, and the network switches have settings defining the prioritization of the data packages. Read more **below**.
- **Data Merge Default Mode**:
  This defines the default data merge mode. This is the preselected option in the **Session Data Merge pop-up**. This option will be selected when the timeout (read below) expires unless the user manually selects an option. The options are:
  - **Cancel**:
    The connecting stations will not join the session of the Master side and be made Standalone. Then, it can be investigated which show file is correct, back up each show file, etc.
  - **Merge All Data**:
    Merges the changes of all connecting stations into one show file.
  - **Keep Only Master Data**:
    The changes in the show file data of the connecting devices are ignored. The show file of the Master side is maintained. Before overriding the show file of the connecting devices, the show file will be saved automatically on the connecting stations.

  This pop-up allows the user to select how data is merged between the session and connecting stations. It is described in the **Join a Session topic**.

- **Data Merge Default Timeout**:
  This defines the timeout time for the **Session Data Merge pop-up**. **Unlimited** is an option. This disables the timeout, and the user must make a manual selection. If the time is set to 0, then the pop-up will not appear and the default mode (see above) is used.

  This pop-up allows the user to select how data is merged between the session and connecting stations. It is described in the **Join a Session topic**.

- **TTL** (Time To Live):

The TTL value specifies the lifespan of an IP packet. Each time the packet passes the next hop (for example, a router or gateway), its TTL is reduced by one, and the packet is discarded when the value reaches zero. Tap TTL and set a value using the calculator. The Default value is "8".

	**Hint:**
	Admin rights are needed to change DSCP, Data Merge Default Mode, Data Merge Default Timeout, and TTL.

In & Out has the following toggle buttons:

- **DC Remotes**:
  This is the same as the **Enable Input** setting in the **DC Remotes menu**.
- **MIDI Remotes**:
  This is the same as the **Enable Input** setting in the **MIDI Remotes menu**.
- **DMX Remotes**:
  This is the same as the **Enable Input** setting in the **DMX Remotes menu**.
- **OSC Input**:
  This is the same as the **Enable Input** setting in the **OSC menu**.
- **OSC Output**:
  This is the same as the **Enable Output** setting in the **OSC menu**.
- **PSN**:
  This is the same as the **Enable Input** setting in the **PSN menu**.
- **MVR-xchange**:
  This is the same as the **Enable** setting in the **MVR menu**.

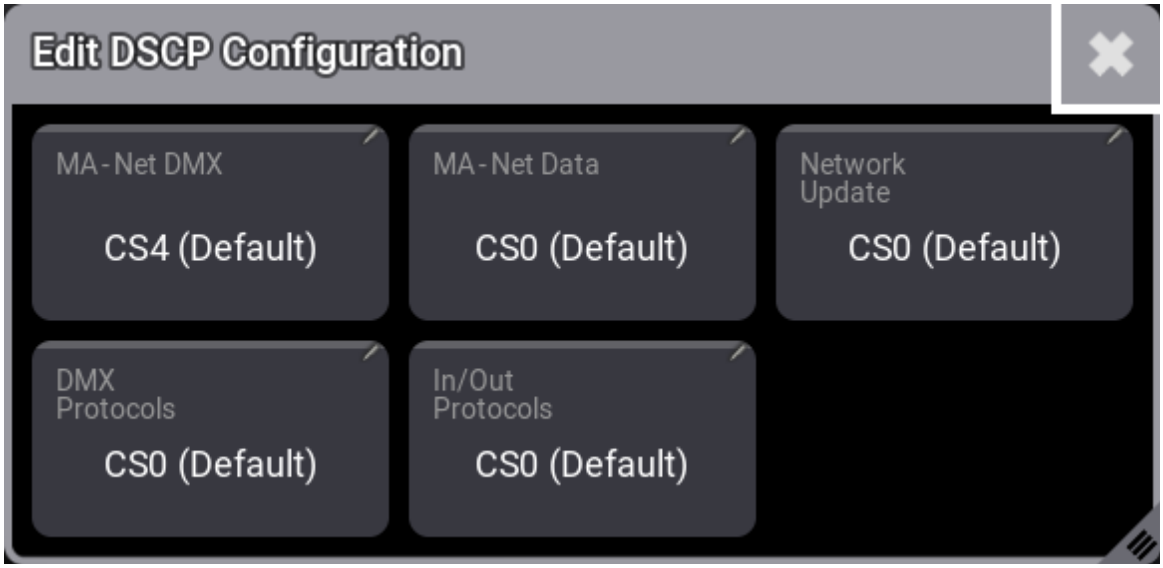
DMX Protocols has the following toggle buttons:

- **Art-Net Input**:
  This is the same as the **Art-Net Input** setting in the **Art-Net Menu**.
- **Art-Net Output**:
  This is the same as the **Art-Net Output** setting in the **Art-Net Menu**.
- **Art-Net Setup**:
  This is the same as the **Art-Net Setup** setting in the **Art-Net Menu**.
- **sACN Input**:
  This is the same as the **sACN Input** setting in the **sACN Menu**.
- **sACN Output**:
  This is the same as the **sACN Output** setting in the **sACN Menu**.
- **sACN Setup**:
  This is the same as the **sACN Setup** setting in the **sACN Menu**.

DSCP

The network data packages from a session are divided into five different categories. Each category has a setting defining the DSCP value. The higher the value, the higher the priority.

Read about the setting above. Editing the DSCP setting opens the **Edit DSCP Configuration pop-up**.



Edit DSCP Configuration pop-up.

The five categories are:

- MA-Net DMX
- MA-Net Data
- Network Update
- DMX Protocols
- In/Out Protocols

The different options for each category are:

- CS0 - DSCP Value 0
- CS1 - DSCP Value 8
- AF11 - DSCP Value 10
- AF12 - DSCP Value 12
- AF13 - DSCP Value 14
- CS2 - DSCP Value 16
- AF21 - DSCP Value 18
- AF22 - DSCP Value 20
- AF23 - DSCP Value 22
- CS3 - DSCP Value 24
- AF31 - DSCP Value 26
- AF32 - DSCP Value 28
- AF33 - DSCP Value 30
- CS4 - DSCP Value 32
- AF41 - DSCP Value 34
- AF42 - DSCP Value 36
- AF43 - DSCP Value 38
- CS5 - DSCP Value 40
- Voice-Admit - DSCP Value 44
- EF - DSCP Value 46
- CS6 - DSCP Value 48
- CS7 - DSCP Value 56

It is recommended that the default value be changed only if there is a specific need to do so.

Learn more about DSCP on **Wikipedia (external link)** or **iana.org (external link)**.

	Restriction:
	On grandMA3 onPC for Windows® the DSCP values are overwritten with the default value CS0.

1.12. DMX In and Out

There are several ways to get DMX in and out of the grandMA3 system.

DMX ports on stations can generate DMX as a standalone device.

DMX ports on grandMA3 devices that are part of a session can also be DMX outputs and inputs.

This is set up from the **Output Configuration** menu. Please read about it in the **DMX Port Configuration topic**.

DMX can also be transferred using standard network protocols. This is **Ethernet DMX**.

There are different rules for the different kinds of DMX.

Generally, each DMX port on a grandMA3 device can be an input or an output.

DMX Refresh Rate

The DMX standard used by the system (ANSI E1.11 - 2008 (R2018)) supports a wide range of refresh rates, and it is allowed only to send some of the DMX channels.

The grandMA3 system implements DMX using the following rules:

- The entire DMX universe is sent each time.
- The maximum DMX rate is 30 Hz.
- The minimum DMX rate is 1 Hz.
- The default refresh rate is 30 Hz.
- The refresh rate can drop down (not below the minimum) if there are no changes in the DMX values in a universe and the XLR port is to **"RDM"** or when DMX is output using Art-Net or sACN.

  DMX output can be XLR ports on compatible grandMA3 hardware or via Ethernet DMX. The XLR ports set to "Out" do not slow down the DMX refresh rate. The other methods can slow it down.

- Universes can have different refresh rates. For instance, a universe without any changes can be at 1Hz, and a universe with a running phaser can be at 30Hz.

See more about the used refresh rates in the sub-topics.

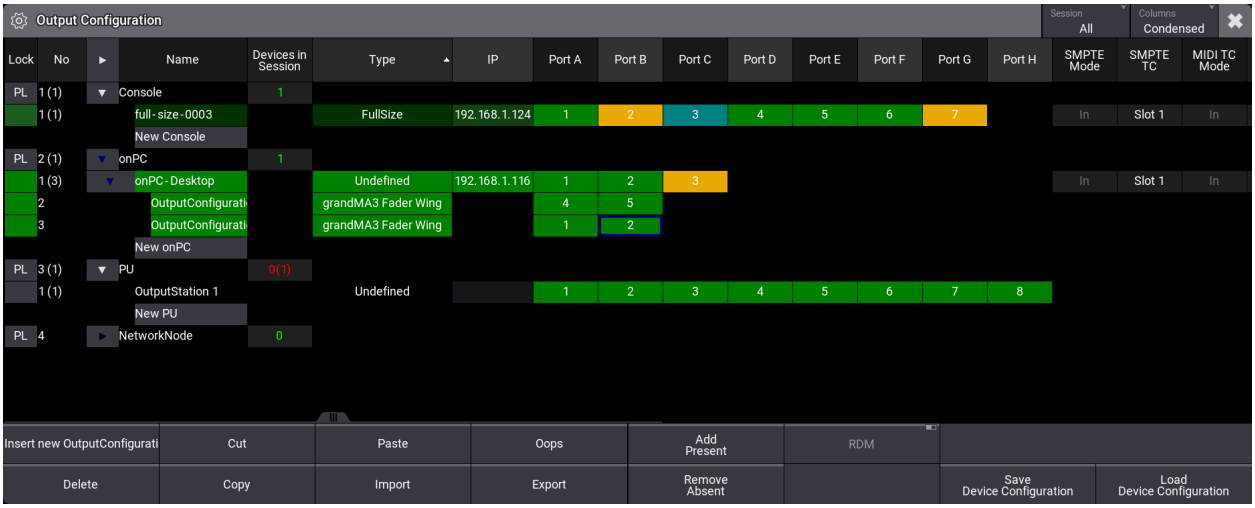
Subtopics

- **DMX Port Configuration**
- **Ethernet DMX**

1.12.1. DMX Port Configuration

The DMX outputs and inputs are configured in the **Output Configuration** menu. They are labeled from **A** and alphabetically upwards.

There are many columns for SMPTE, MIDI, DC, and Ethernet interfaces, but these elements are not described in this topic. Please read about them in the **Timecode section**, **Remote In and Out topic**, and **Ethernet DMX**.



Output configuration menu with console onPC and PU

Access the menu by pressing the Menu key and then tap Connector Configuration.

-OR-

Use the command line to open the menu:

MA	User name[Fixture]>Menu "ConnectorConfig"

	Hint:
	See **Ethernet DMX** for how to send and receive DMX using Ethernet.

The menu lists grandMA3 devices. There is a button in the title bar called Session. This filters the list to show **All** devices in the network or only the ones **In Session**.

The devices are organized in Console, onPC, PU, and NetworkNode sections that can be unfolded to see the devices in each category. Real hardware devices are matched to devices in this list by IP addresses.

There is a **Devices in Session** column. It shows the number of each device type in session with the current device in green. If a device is missing, the number is red, and the expected number is in parentheses next to the actual number of devices.

The menu has different modes for displaying columns. There is a button in the title bar called Columns. This has three modes:

- **Full**:
  This shows all columns.
- **Condensed**:
  This mode condenses the XLR columns to one for each port and displays all other columns.
- **XLR Only**:
  This only shows the XLR ports in a condensed version.

	**Hint:**
	Read the **What are Timecode Slots topic** to learn about the SMPTE and MIDI settings. Read the **DC Remotes topic** to learn about the DC Start setting. Read the **Ethernet DMX topics** to learn about the Art-Net and sACN settings. Read the **OSC topic** and the **PSN topic** to learn about these settings.

DMX Ports

Each DMX port has four different settings (only visible when in **Full** mode):

Mode:

Each port can have four different modes:

- **Off** (gray background color in the cell):
  The port is turned Off.
- **Out** (green background color in the cell):
  DMX is sent out without RDM traffic. The universe is defined in the XLR field. This is the default mode.
  DMX refresh rate is a constant at 30 Hz.

  DMX refresh rates define how many times a DMX universe is sent per second. Learn more about refresh rates in the **DMX In and Out Topic**.

- **RDM** (dark sea green background color in the cell):
  DMX is sent out, and RDM is active. The universe is defined in the XLR field.
  DMX refresh rate is variable and can change between 1 Hz to 30 Hz.

  DMX refresh rates define how many times a DMX universe is sent per second. Learn more about refresh rates in the **DMX In and Out Topic**.

- **In** (yellow background color in the cell):
  DMX is received and merged into the universe specified by the XLR number. The merge uses the priority defined in Prio.

XLR:

This is a number that defines what universe the port relates to. Fixtures can be patched in universes 1 to 1 024.

Merge:

This option is only visible when the port mode is **In**.

The options are:

- **Off**:
  The incoming DMX is not merged.
- **Prio**:
  The priority set in the **Prio** column is used. The highest priority wins.
- **HTP**:
  The highest DMX value is used.
- **LowTP**:
  The lowest DMX value is used.

Prio (Priority):

The priority is used for merging DMX inputs. It is only used and can only be edited when the port mode is **In**, and the merge is set to **Prio**.

Editing this field opens a small select pop-up with the options.

The options are (from highest priority to lowest):

- **Super**:
  This priority is the LTP priority above any other playbacks and even above the programmer.
- **Prog**:
  The Programmer priority is between Super and the other priorities.
- **Highest**:
  Highest LTP priority - like LTP but higher than both LTP and High.
- **High**:
  High LTP priority - like LTP but a higher priority than normal LTP.
- **LTP (Latest Takes Precedence)**:
  This is the normal LTP priority. The newest attribute value is prioritized over the old value.
- **Low**:
  Low LTP - This is a lower LTP priority.
- **Lowest**:
  Lowest LTP - This is a lower priority than both LTP and Low.

	**Important:**
	The port configuration is stored in each device where it is set.

Import and Export of Entire Device Configuration

The device configuration for all the devices in the list can be stored in a single configuration file. This configuration can then be loaded again to restore the settings based on the file. This makes storing and loading configurations for different shows or setups easy.

	**Hint:**
	The device configuration file includes the Output Configuration and the DMX Protocol (**Ethernet DMX** protocols) settings.

Save Device Configuration

1. Make sure the configuration is as desired.
2. Tap Save Device Configuration. This opens the **Save Device Configuration as:** pop-up.

3. Type a name for the configuration and ensure the desired drive is selected in the title bar of the pop-up.
4. Tap Save.

Load Device Configuration

1. Tap Load Device Configuration. This opens the **Load DeviceConfiguration from:** pop-up.
2. Select the desired configuration file.
3. Tap Load.

Import and Export of Output Configuration for Single Devices

Each device keeps its own output configuration. The configuration of single devices can be exported to a file and then imported again.

Export an Output Configuration

1. Select the devices that need to be exported.
2. Tap the Export button -> a file browser opens.
3. Select the desired drive.
4. Give the file a name and tap enter/please.

A file can contain the configuration for one or multiple devices. It is just a matter of selecting one or several devices before the export.

Import an Output Configuration

1. Select the desired devices.
2. Tap the Import button -> a file browser opens.
3. Select the desired drive.
4. Select the files that have a configuration for the selected devices.

Add and Remove Devices

The list of devices is automatically updated with new devices in the network.

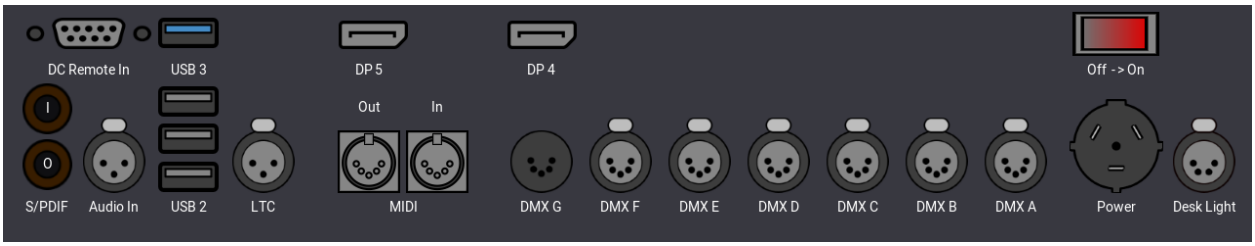
Tapping Add Present updates the list with new devices.

Tapping Remove Absent removes devices from the list that are no longer in the network.

Devices can be added without being present in the network. Unfold the desired device type in the list and tap the New [device type] followed by Insert new [device type]. The grandMA3 onPC's can be expanded further by adding up to two grandMA3 Fader Wings to each grandMA3 onPC.

Show Connectors

On the light and full-size consoles, there is a button called Show Connectors. This can be toggled On and shows a connector overlay on the letterbox screens. This indicates where the different connectors are located on the back of the console. The XLR ports are labeled DMX A to DMX G in this overlay.



Connector overlay on the right letterbox screen (grandMA3 full-size model)

The overlay can also be toggled using the following command:

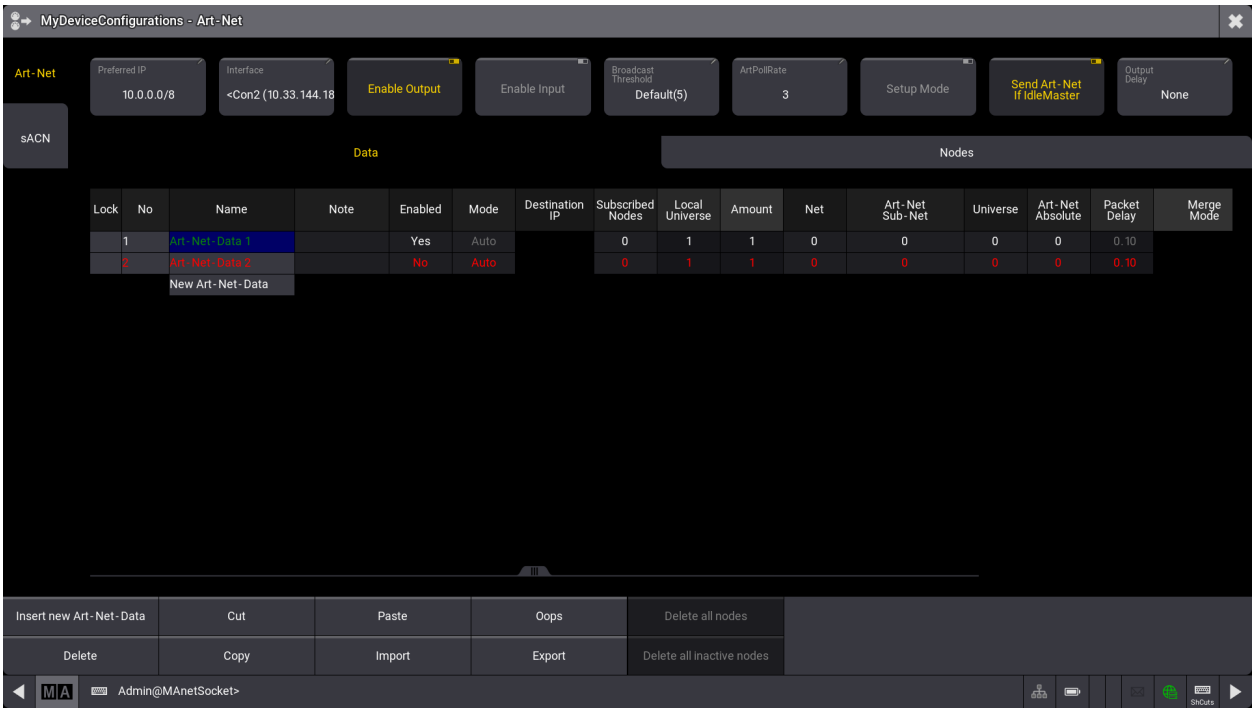
```
User name[Fixture]>Menu "ConnectorView"
```

1.12.2. Ethernet DMX

There are currently two supported protocols for sending and receiving DMX using the network: sACN and Art-Net.

It is the session master station that outputs the network DMX. The visual feedback regarding sending or receiving network DMX can look different on the stations in the session, so it should be viewed and changed on the master station to see the actual output/input status.

Both are set up and changed in the **DMX Protocols** menu:



DMX protocol configuration for Art-Net

Access the menu by pressing Menu and then tap DMX Protocols.

Tapping the DMX Protocols button opens the menu for the first protocol. In the example above, it is the Art-Net menu.

The menus can also be opened using the command line:

For Art-Net:

```
User name[Fixture]>Menu "ArtNet"
```

For sACN:

```
User name[Fixture]>Menu "sACN"
```

On the left side menu, there are two buttons, one for each menu.

See the subtopics for information about the two protocols.

Subtopics

- **Art-Net Menu**
- **sACN (streaming ACN) Menu**
- **Transmit DMX Using Art-Net**

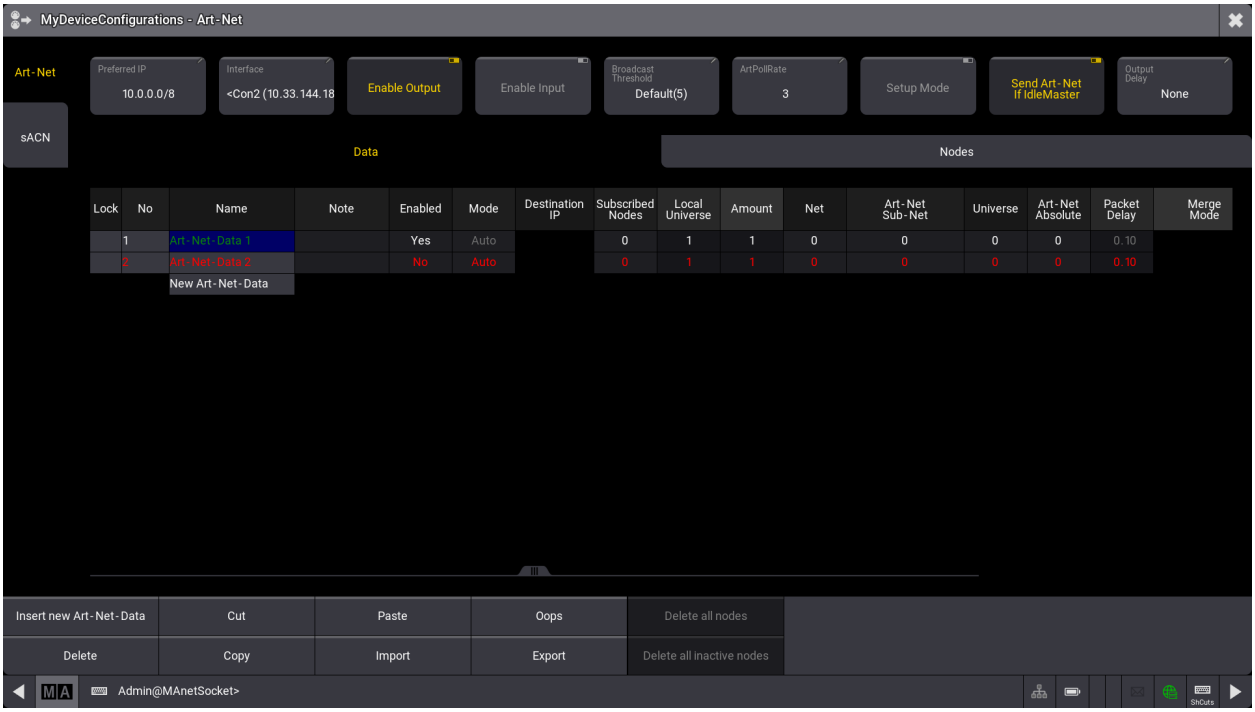
1.12.2.1. Art-Net Menu

Art-Net is a royalty-free protocol developed by Artistic Licence (**https://artisticlicence.com/**).

grandMA3 supports Art-Net 4.

	Hint:
	The DMX output via Art-Net can vary in refresh rates. It follows the rules specified in the **DMX In and Out topic**.

It is configured in the Art-Net menu:



Art-Net configuration menu

See the **Ethernet DMX topic** for information on how to open this menu.

See the **Transmit DMX using Art-Net topic** for an explanation of the steps needed to transmit Art-Net.

Config Buttons

There are some buttons at the top of the menu:

- **Preferred IP**:
  This is the preferred IP address or address range used by the Art-Net protocol. The Interface can be set to **Auto**, allowing this setting to select the interface that matches the preferred setting. Tap this input button to open a small number input pop-up. The preferred number has to be input as an IP address with the subnet mask written in CIDR notation. For instance, 10.0.0.0/8 tells the system that it prefers an IP address that starts with 10. The rest of the

numbers do not matter (the "/8" is the same as the subnet mask 255.0.0.0). This setting is individual for each station.

- **Interface**:
  Tap this button to open the Select Interface pop-up to select the desired network interface. This interface will be used for all Art-Net in and out. The **Auto** option can be used to allow the **Preferred IP** setting to select the interface. The interface is written inside "<" and ">" when it is selected by the preferred setting. This setting is individual for each station.
- **Enable Output**:
  This On/Off button must be On for Art-Net to be transmitted. Data also needs to be configured for output - read more below.
- **Enable Input**:
  This On/Off button must be On for Art-Net to be received. Data also needs to be configured for input - read more below.
- **Broadcast Threshold**:
  This input button sets the amount of Art-Net receivers for a universe before the master starts to send the universe as broadcast. This is only valid if the mode is set to **Auto**. If the number of receivers is below the threshold, the universe is sent as Unicast. If it is above, then the universe is sent as Broadcast.
- **ArtPollRate**:
  This input sets the time between each ArtPollRequest packet sent by the master station.
- **Setup Mode**:
  This On/Off button is used to toggle the setup mode. This mode can be used to transmit and receive configuration data only. If the output and input are turned Off, only the configuration data is transmitted and received.
- **Send Art-Net If IdleMaster**:
  This On/Off button defines if the station transmits Art-Net data when it is Idle Master. In a session, the Global master transmits the network DMX. If the station is not in a session with other devices, it is Idle Master. Turning this setting On will make the station output network DMX when it is Idle Master. This must be On if a single station is to output networked DMX. Learn more about standalone devices and networked devices in the **System Overview section**. This setting is individual for each station.
- **Output Delay**:
  This can set an output delay between 0ms and 30ms. This delays the entire Art-Net output compared to the outputs coming from MA-Net devices.

Important:

- Enabling Setup Mode allows Art-Net configuration data to be sent and received even when **Enable Output** and **Enable Input** is Off.
- When Setup Mode is Off, and output is enabled, DMX and configuration data are transmitted.
- When Setup Mode is Off, and input is enabled, DMX and configuration data are received.

There are two tabs below the buttons. They are **Data** and **Nodes**. Data is used to set up output and input. Nodes can be used to see the discovered nodes in the network. If the nodes support it, this can also be used to configure the ports on the node.

Data Tab

The data tab is a grid of rows and columns. Each row is an Art-Net configuration. The text is red if the row is invalid or not enabled. The name text flashes green when Art-Net is transmitted. It flashes dark yellow when an Art-Net input signal is received.

This is a short description of the columns:

- **Lock**:
  The row can be locked to prevent changes.
- **No**:
  This is the row number.
- **Name**:
  Each row can have a name. This can be used as short info for the row.
- **Note**:
  A note can be added to each row for information purposes.
- **Enabled**:
  This **Yes** or **No** field is used to enable the row transmitting or receiving Art-Net. No is the default.
- **Mode**:
  The mode defines what the row is doing. There are four options:
  - **Broadcast**:
    This transmits Art-Net using broadcast.
  - **Unicast**:
    This transmits Art-Net using unicast.
  - **Auto**:
    This transmits Art-Net. It uses the **Broadcast Threshold** number to determine if universes should be transmitted using broadcast or unicast. If the number of universe subscribers (determined by ArtPollRequests) are below the threshold, then it is sent using unicast. Is it above the threshold, then it is broadcasted.
  - **Input**:
    The row receives Art-Net and merges it into the defined universe.

Restriction:
The total input count (Art-Net and sACN) must not exceed 128 Universes.

- **Destination IP**:
  This field is only active if the unicast mode is selected. This is the IPv4 address of the receiving device.
- **Subscribed Nodes**:
  This field provides information about the number of subscribing devices. This is determined using ArtPollRequests.
- **Local Universe**:
  This is the grandMA3 universe to be transmitted or the universe that should receive incoming Art-Net DMX. This is the first universe in the range if the amount is more than one.
- **Amount**:
  This is the amount of grandMA3 universes to be transmitted or received.
- **Net**:
  The net number is a value between 0 and 127. There are 128 different nets since Art-Net III. Each net is a complete group of **Sub-Nets** and **Universes**. This allows for addressing a total of 32 768 Art-Net universes. To be compatible with Art-Net I and Art-Net II devices, please use net 0.
- **Art-Net Sub-Net**:
  There are 16 sub-nets in Art-Net. They can be input in decimal numbers from 0 to 15 or 0 to F in hex numbers.
- **Universe**:
  There are 16 universes in each sub-net. They can be input in decimal numbers from 0 to 15 or 0 to F in hex numbers.

- **Art-Net Absolute**:
  This is the absolute universe number. It is calculated based on the net, sub-net, and universe numbers. It can also work the other way. A universe number can be input here, and then the net, sub-net, and universe numbers are calculated based on the input.
- **Packet Delay**:
  A delay can be set up between each transmitted universe. This can be helpful for older nodes with slower network cards. Sending many universes at once can flood the node. Adding a small delay helps. This setting is only available for outputting modes.
- **Merge Mode**:
  The Merge Mode defines how incoming sACN merges into universes. When changing the Merge Mode for a configuration line, all universes defined by Local Universe and Amount are changed together. In the DMX Universes-tab of the Patch and Live Patch menu or by editing a universe in the universe pool, it is still possible to independently change the Merge Mode for single universes. As soon as two or more universes of a configuration line have different Merge Modes, the Merge Mode cell will display **...** to indicate this. The Merge Mode and Input Priority are described in the **DMX Port Configuration topic**.
- **Input Priority**:
  This is the priority of the received Art-Net. This can only be changed when the mode is input. Follow the link above to learn more.
- **Timecode Slot**:
  Any received ArtTimeCode is sent to the timecode slot number defined here. The mode does not need to be set to Input to receive the timecode, but Input needs to be enabled. This setting can only be changed when the mode is input. There are a total of 8 timecode slots. Read more about them in the **What are timecode slots topic**.
- **Enable RDM**:
  RDM via Art-Net can be enabled for the Art-Net universes specified in the row.

Nodes Tab

Each node detected is a row. Each node has one or more sub-rows with "binds". Each bind can only have 4 ports. If a device has more than 4 ports, then the device has multiple binds.

- **Lock**:
  The row can be locked to prevent changes.
- **No**:
  This is the row number.
- **Name**:
  Each row gets a name from the device. The name can be edited if the device supports it.
- **IP**:
  The IP address of the device. The address can be edited if the device supports it.
- **Net**:
  Each bind has a net number. The number can be edited if the device supports it.
- **Sub-Net**:
  Each bind has a sub-net number. The number can be edited if the device supports it.
- **Output Port**:
  This is the universe number of output ports on the device. The number can be edited if the device supports it.
- **Input Port**:
  This is the universe number of input ports on the device. The number can be edited if the device supports it.

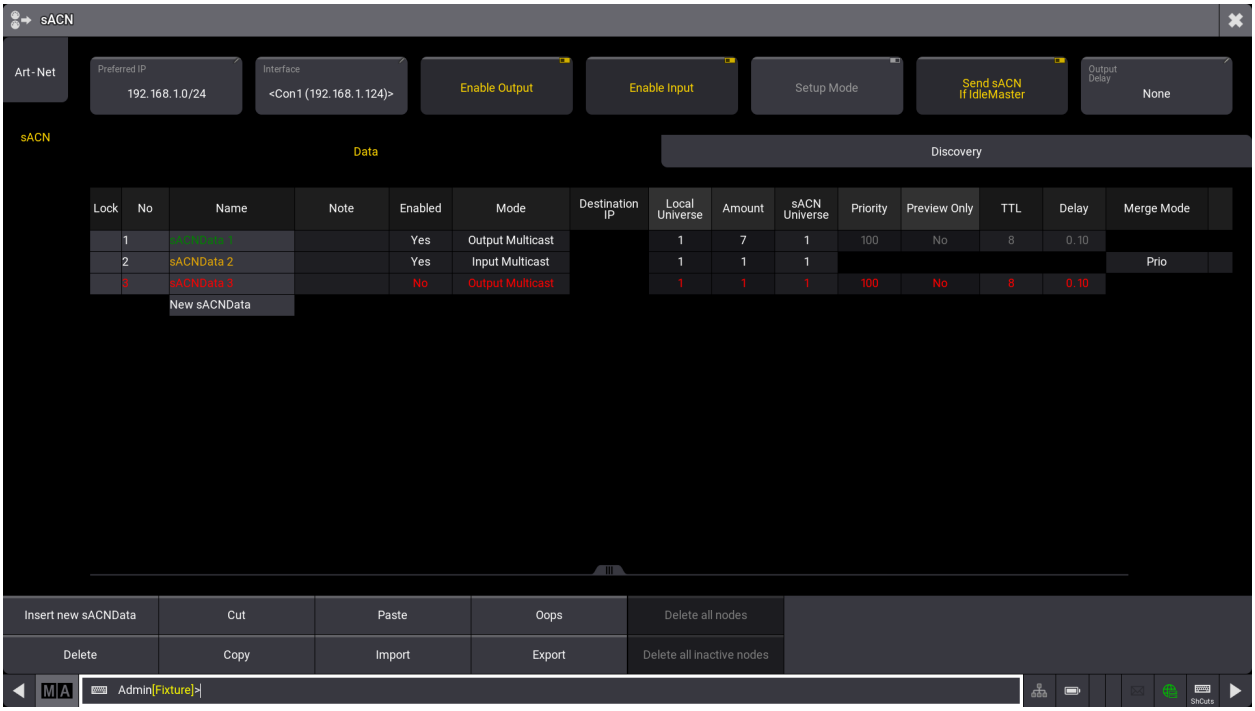
1.12.2.2. sACN (streaming ACN) Menu

ACN (and streaming ACN) is an ANSI/ESTA international standard. Further readings: **https://en.wikipedia.org/wiki/Architecture_for_Control_Networks**.

ACN (Architecture for Control Networks) is a suite protocol. It uses a lot of elements that are currently not supported by grandMA3. However the ACN protocols also have a version for transporting DMX data. It is called 'Lightweight streaming protocol for transport of DMX512 using ACN' or more popular "streaming ACN" or "sACN". It is the international standard number E1.31.

	Hint:
	The DMX output via sACN can vary in refresh rates. It follows the rules specified in the **DMX In and Out topic**.

It is configured in the sACN menu:



sACN menu

See the **Ethernet DMX topic** for information on how to open this menu.

Config Buttons

There are some buttons at the top of the menu:

- **Preferred IP**:
  This is the preferred IP address or address range used by the sACN protocol. The Interface can be set to Auto, allowing this setting to select the interface that matches the preferred setting. Tap this input button to open a small number input pop-up. The preferred number has to be

input as an IP address with the subnet mask written in CIDR notation. For instance, 192.168.1.0/24 tells the system that it prefers an IP address that starts with 192.168.1. The last number does not matter (the "/24" is the same as the subnet mask 255.255.255.0). This setting is individual for each station.

- **Interface**:
  Tap this button to open the **Select Interface pop-up** to select the desired network interface. This interface will be used for all sACN in and out. The **Auto** option can be used to allow the **Preferred IP** setting to select the interface. The interface is written inside "<" and ">" when the preferred setting selects it. This setting is individual for each station.
- **Enable Output**:
  This On/Off button must be On for the master to transmit sACN. Data also needs to be configured for output - read more below.
- **Enable Input**:
  This On/Off button must be On for the master to receive sACN. Data also needs to be configured for input - read more below.
- **Setup Mode**:
  This On/Off button is used to toggle the setup mode for nodes. This allows configuration data to set up the nodes without sending sACN DMX data into the network.
- **Send sACN If Idle Master**:
  This On/Off button defines if the station transmits sACN data when it is Idle Master. In a session, the Global Master transmits the network DMX. If the station is not in a session with other devices, it is Idle Master. Turning this setting On will make the station output network DMX when it is Idle Master. This must be On if a single station is to output networked DMX. Learn more about standalone devices and networked devices in the **System Overview section**. This setting is individual for each station.
- **Output Delay:**
  This can set an output delay between 0ms and 30ms. This delays the entire sACN output compared to the outputs coming from MA-Net devices.

Important:

- Enabling Setup Mode allows sACN configuration data to be sent and received even when **Enable Output** and **Enable Input** is Off.
- When Setup Mode is Off and output is enabled, then DMX and configuration data are transmitted.
- When Setup Mode is Off and input is enabled, then DMX and configuration data are received.

Below the buttons, there are two tabs: **Data** and **Discovery**. Data is used to set up output and input, while Discovery can be used to see the transmitting nodes in the network.

Data Tab

The data tab is a grid of rows and columns. Each row is an sACN configuration. If the row is not valid or not enabled, then the name is in red text. The name text flashes green when sACN data is transmitted. An orange name indicates a possible set-up conflict.

This is a short description of the columns:

- **Lock**:
  The row can be locked to prevent changes.
- **No**:
  This is the row number.

- **Name**:
  Each row can have a name. This can be used as short info for the row.
- **Note:**
  A note can be added to each row for multiline information.
- **Enabled**:
  This Yes or No field enables the row to transmit or receive sACN. Yes is the default.
- **Mode**:
  The mode defines what the row is doing. There are four options:
  - **Output Multicast**:
    When choosing Output Multicast, sACN will be sent as multicast to the relevant multicast addresses.
  - **Output Unicast**:
    When choosing Output Unicast, a valid IP address has to be entered in the **Destination IP** column. Universes configured in this row will be sent as unicast to this IP address.
  - **Input Multicast**:
    Input Multicast will join the Multicast group of the relevant DMX Input Universe. Input Multicast is limited to max. 20 Universes. A warning pop-up appears if more than 20 rows are configured as Input Multicast, and all rows beyond multicast input row 20 will be invalid.
  - **Input Unicast**:
    Input Unicast is not limited and receives sACN data for the relevant universe without joining any multicast group.

	Restriction:
	The total input count (sACN and Art-Net) must not exceed 128 Universes.
- **Destination IP**:
  This field is only active if the output unicast mode is selected. This is the IPv4 address of the receiving device.
- **Local Universe**:
  This is the grandMA3 universe to be transmitted or the universe that should receive incoming sACN DMX. If the amount is more than one, then this is the first universe in the range.
- **Amount**:
  This is the amount of grandMA3 universes to be transmitted or received.
- **sACN Universe**:
  This is the sACN universe number the grandMA3 universes is transmitted to or the universe number that is listened to if Input is selected. If the amount is more than one, then this is the first universe in the range.
- **Priority**:
  The allowed value is 0 to 200. The highest number has the highest priority. The default value is 100. This priority is used for transmitted sACN.
- **Preview Only**:
  sACN data can be sent as preview data. This can, for instance, be used to send DMX to visualizers.
- **TTL** (Time To Live):
  Time To Live is a number used to tell routers and some switches how far through the network the sACN data should be transmitted. This is only relevant for output modes. The default value is 8 and this should usually not be changed.
- **Delay**:
  A delay can be set up between each transmitted universe. This can be helpful for older nodes

with slower network cards. Sending many universes at once can flood the node. Adding a small delay helps.

- **Merge Mode**:
  The Merge Mode defines how incoming sACN merges into universes. When changing the Merge Mode for a configuration line, all universes defined by Local Universe and Amount are changed together. In the DMX Universes-tab of the Patch and Live Patch menu or by editing a universe in the universe pool, it is still possible to change the Merge Mode for single universes independently. As soon as two or more universes of a configuration line have different Merge Modes, the Merge Mode cell will display **...** to indicate this. The Merge Mode and Input Priority are described in the **DMX Port Configuration topic**.
- **Input Priority**:
  This is the priority of the received sACN. sACN input of grandMA3 ignores sACN priorities and uses this priority instead.

Discovery Tab

This tap displays the transmitting nodes in the network. Each node detected is a row, and each node has one or more sub-rows with "pages".

- **Lock**:
  The row can be locked to prevent changes.
- **No**:
  This is the row number.
- **Name**:
  Each row gets a name from the device. The name cannot be edited.
- **Universe List**:
  This is a list of the universes the node transmits.

1.12.2.3. Transmit DMX Using Art-Net

This is a short example of setting up DMX transmission using Art-Net. Learn more about the Art-Net menu in the **Art-Net Menu topic**.

The idea in this example is some lights are patched in universes one to five. The fixtures are connected to Art-Net nodes and the nodes need to get the Art-Net data from the grandMA3. There is also a media server that uses a single universe. It is patched in universe 21 and needs to receive Art-Net universe 21 (hex number 1:5).

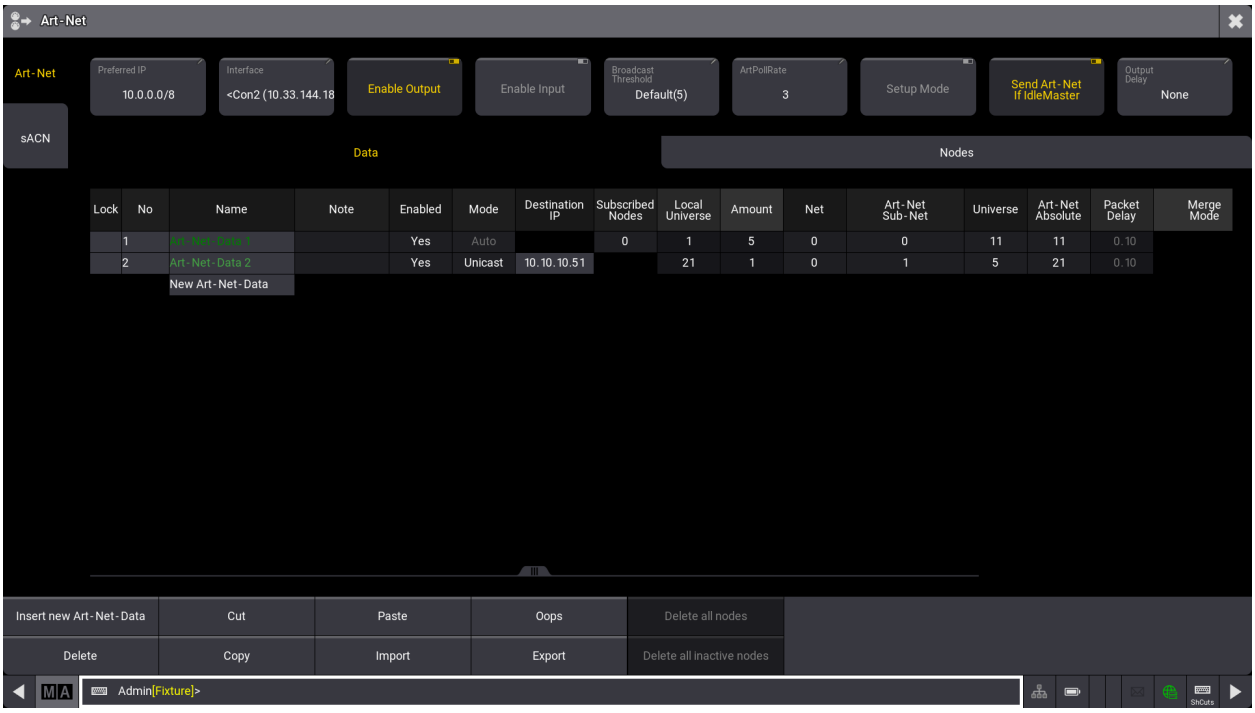
The Art-Net nodes are set up to receive from Art-Net universe 11 (hex number 0:B), so the grandMA3 universe one needs to be sent to this Art-Net universe. The media server is a specific device with a defined Art-Net IP address, and this data can be sent using unicast to it. In this example, the media server IP address is set to 10.10.10.51.

The example assumes a patched setup that matches the universes. This is not actually needed for demonstration purposes. If the grandMA3 onPC is used to try this example, then there might not be parameters to allow actual output.

These are the steps needed:

1. Open the Art-Net menu by pressing Menu and then tap DMX Protocols. Make sure the Art-Net menu is visible by tapping Art-Net.
2. The first line is going to be universe one to five. Edit the following cells to match these options:
    a. Enabled = Yes
    b. Mode = Auto
    c. Local Universe = 1
    d. Amount = 5
    e. Art-Net Absolute = 11
3. A second line is needed for the universe that needs to go to a specific destination. Tap New Art-Net-Data below line 1.
4. Tap Insert new Art-Net-Data in the button menu at the bottom of the screen.
5. Now the line exists. Edit the new line to match these options:
    a. Enabled = Yes
    b. Mode = Unicast
    c. Destination IP = 10.10.10.51
    d. Local Universe = 21
    e. Amount = 1
    f. Art-Net Absolute = 21 (notice that this automatically calculates the correct Art-Net Sub-Net and Universe number)
6. Finally, tap the Enable Output until it is On to activate the transmission of DMX data via Art-Net

That was it. It should look something like this:



Final Art-Net example setup

If the station is not connected to other stations or nodes in a session, it is **Idle Master**. If the station needs to output Art-Net in a setup like this, the **Send Art-Net if Idle Master** setting needs to be turned On. If the station is in a session with other stations, then it is the **Global Master** in the session that transmits the Art-Net data.

The menu can be closed by tapping the cross ( ) in the upper right corner.

1.13. Single User and Multi User Systems

The grandMA3 system can handle small and big systems with one or many users simultaneously.

grandMA3 systems are often referred to as a **Session**.

An Ethernet network can have several sessions running.

A session can have one, some, or many MA devices as members or listeners.

This page describes some differences between the two primary ways to have a system.

Single User Session

In a single-user session, there is one operator/programmer. This person might have one or several stations (consoles or onPC).

When there is only one user, all stations are logged in as the same **User**.

This is also called a **Full Tracking Backup**. The stations are all in sync and share views, and the programmer content is the same on all stations.

This is useful for the single operator. If one station fails, the operator can move to another station and continue working without losing anything.

A version of this involves a single operator adding multiple **Users** with the same **User Profile** but different **Screen Configurations**. This allows the user to have different views on the stations' screens while still having the same programmer content.

The single operator will usually have full admin rights to the entire system.

Multi User Session

In a multi-user session, there are several operators. They might use the same stations and take turns (working in shifts or at different process phases). It can also be a system with multiple users working simultaneously on different stations.

For this setup, the operators must create more **Users** and, most likely, more **User Profiles**. Each operator will then log into the station using their User and may control the complete system or just a part of the system.

The stations share the show file, and the sequences are all in sync, but the users can have different views on the screens, and their programmer is not shared - the output is.

Sometimes systems are a combination of multi-users and full tracking backups; There are several operators (Multi User), but each operator has two (or more) stations logged in with their user (Full Tracking Backup).

In a multi-user system, it is possible to have users with different operator rights. There are several levels of rights. They span from complete access to only being allowed to change the view.

Subtopics

- **Create User**
- **User Settings**
- **Object Ownership**
- **Screen Configuration**

1.13.1. Create User

A **User** is also a login name.

Some things are connected to the user. For instance, each user can have different operating rights and different languages. Read more in the **User Settings topic**.

The users are stored in a **User pool**. Read more about creating windows in the **Add Windows topic**.

The User pool with logged-in Admin user

The currently logged-in user has a yellow frame. It is the selected user.

Tapping one of the other users logs in as this user. If the user has a password, then a **Please Login pop-up** appears.

Please Login pop-up asking for password

	Important:
	The password is case-sensitive.

Type the correct password to log in.

	Important:
	The password for the Admin user get reset when loading the show. The default password for the admin is empty or nothing. The password of the Guest cannot be changed. The default password for the guest is empty or nothing.

A show file always has some default users**.** They use the same **User Profile**.

User Profiles are the ones that hold information about programmer content, selected sequence, current page, views, cameras, etc.

If two users share the same user profile, they share all the information stored in it.

If the user profiles differ, two operators can have different selections, programmer values, views, etc. - **Multi User System**.

Each **User** must have a **User Profile** assigned - Read more about this and how to create user profiles in the **User Settings topic**.

Create a New User

The easiest way to create a new user is to edit an empty object in the user pool, for instance, by using swipeys.

This opens the **Edit User pop-up**

Edit User pop-up with a new user

The name should be changed at a minimum. This is also the name used when logging in using the **LogIn keyword**.

The big area on the right side of the pop-up shows the user's selected appearance (the image above shows an appearance).

The rest of the settings in the editor are explained in the User Settings section.

1.13.2. User Settings

Many settings are connected to the **User** and the **User Profile**.

User Settings

There are some different user settings.

- **Name**:
  This is the name of the user. This is the login name used with the **LogIn keyword**.
- **Scribble**:
  A user can have a scribble assigned. The scribble is only used in the **User pool**.
- **Appearance**:
  A user can have an appearance. The appearance is used in the user pool and when the station is locked.
- **Note**:
  A note can be added to the user.
- **Password**:
  This is an optional password that can be set to the user. It needs to be provided when the user is logging in.
- **Profile**:
  Each user has a User Profile assigned. Read more **below**.
- **ScreenConfig**:
  Different screen configurations can be created. This is the configuration called when the user logs in. Learn more in the **Screen Configuration topic**.
- **Rights**:
  There are six different levels of rights. Read more **below**.
- **Keyboard**:
  The on-screen keyboard can have different layouts. This setting can be used to select one of the available layouts.
- **Privacy Policy** (Only in the User Configuration menu):
  This setting can be toggled between Yes and No. Yes means that this user has agreed to the **Privacy Policy**. The policy can be found in Menu - Settings - Software Update - Privacy Policy.
- **WebRemoteLogin** (Only in the User Configuration menu):
  This is information only. It indicates if the user can log in using the web remote.

There are two places in the GUI where the settings can be viewed and edited.

The first place is the Users pool. Editing a pool object opens an editor where the settings can be changed. Learn more about the Users pool in the **Create User topic**.

The second place is the **User Configuration**. Navigate to it using these steps:

1. Press the Menu key.
2. Tap the Settings button.
3. Tap the User Configuration button.

This opens the list of users in the show. Edit any field to change the setting.

User Profile Settings

The user profiles contain most of the settings relevant to the users. The programmer information, views, selected elements, values, preferences, and much more.

Many of these elements are stored and selected by using the software. Some settings can be set in the User Profiles menu.

Navigate to the menu:

1. Press the Menu key.
2. Tap the Settings button.
3. Tap the User Configuration button.
4. Tap the Profiles button on the left menu.

Some settings here can also be changed in relevant places in the software. For instance, the value readout setting can be changed in the **Encoder Bar**.

- **Name**:
  This is the name of the user profile.
- **Note**:
  This is a note for the user profile.
- **DMX Readout**:
  This is used to change the readout of DMX values. This is useful when editing **fixture types**.
- **Normal Value**:
  This is the intensity value used when the **Normal keyword** is used - typically by pressing the **At key** twice. It is a DMX value, so the DMX readout setting affects how to input a value in this field.
- **Wheel Resolution**:
  This can be used to change the resolution of the wheels on the consoles. The options are: Coarse, Normal, and Fine.
- **Wheel Mode**:
  This setting changes how the wheels work. Read more **below**.
- **Knob UI Style**:
  This defines how on-screen rotating knobs can be adjusted. There are three modes:
  - **None**: This disables using gestures to adjust the values of knobs.
  - **Rotate**: Change the value by rotating around the encoder or knob.
  - **Drag**: This can open a special pop-up that can be used to edit the value. Read more in the **Gestures topic**.
- **Encoder UI Style**:
  This defines how on-screen encoders can be adjusted. There are two modes:
  - **None**: This disables using gestures to adjust the values of encoders.
  - **Rotate**: Change the value by rotating around the encoder or knob.
  - **Drag**: This can open a special pop-up that can be used to edit the value. Read more in the **Gestures topic**.
- **Precise Edit**:
  This function is used when editing values in sheets. It can be difficult to hit a field precisely in a

sheet with fingers. Turning this Off makes it easier to select a field without accidentally creating a new selection. Learn more about this in the **Gesture topic**.

- **Screen Encoder**:
  When this option is enabled, the rightmost dual encoder becomes a screen encoder. Turning the inner encoder moves the focus in a vertical direction while the outer encoder scrolls in a horizontal direction. To create a lasso selection in a grid, like in the patch menu, press, hold and turning the inner encoder. Press and release the inner encoder or the dual encoder key to edit the currently focussed object.
  The screen encoder label in the encoder also displays which display the encoder is currently active. This option is enabled by default.
- **Time Key Target**:
  The **Time key** can have two different targets: Cue or Fixture. This defines if the key defaults to keywords relating to cue timing or fixture layers.
- **TCSlot**:
  This displays the currently selected timecode slot. Read more in the **What are Timecode Slots** topic.
- **Value Readout**:
  This is the default value readout. Many sheets can be set to show a readout, which can be a specific readout type or follow this default type. This setting can be changed in the **Encoder Bar**.
- **Speed Readout**:
  This is the default speed readout. The options are Hertz, BPM, and Seconds. Speed is used in **Phasers**.
- **Preset Readout**:
  This changes the way presets are displayed in Fixture and Sequence Sheets. The preset can be displayed with a combination of three elements:
  - **ID**: The ID number of the preset.
  - **Name**: The name of the preset.
  - **Value**: The values in the preset.
- **Help Popup Zoom Factor**:
  This is the default zoom factor in the help pop-up.
- **Overlay Fade**:
  This time sets a fade time used by pop-ups and menus in the user interface. The default time is 250 ms.
- **Time Readout**:
  This is the default time readout for the user profile. Read more about the readout **below**.
- **Frame Readout**:
  This is the default frame readout for the user profile. Read more about the readout **below**.
- **Color Readout**:
  This is the default color readout. This default is used in the **encoder toolbar** and most sheets - unless a different readout is selected for the sheet.
- **Oops Views**:
  This setting toggles if view operations are oopsable. **Create Oops** needs to be enabled for this function to work.
- **Oops Programmer**:
  This setting toggles if programmer operations are oopsable. **Create Oops** needs to be enabled for this function to work.
- **Oops Selection**:
  This setting toggles if selection operations are oopsable. **Create Oops** needs to be enabled for this function to work.

- **Create Oops**:
  This setting toggles if the Oops function in general is turned On or Off. "Yes" means that the Oops function is On.
- **Oops Confirmation**:
  - **Always**: All oops actions need confirmation.
  - **Main**: This will require a confirmation for the following actions:
    - Create
    - Delete
    - Exchange
    - File actions while loading from a file.
    - Insert
    - Layout
    - Move
    - Remove
  - **Never**: Actions are oopsed without confirmation.
- **Mirror SpecialExecutor Pages**:
  This setting is for the custom section of the **grandMA3 extension**. If the setting is "Yes" then the extension has the same assignment as the custom section of the station the extension is connected to. If the setting is "No", then each extension can have its own assignments in the custom section. grandMA3 extensions with the same WingID are always mirrored - they are essentially defined as the same. Learn more about connecting grandMA3 extensions in the **Connect grandMA3 extension topic**.
- **Show Appearance In Cue Input**:
  This setting is used to define if the cue appearances are shown in pop-ups where cues can be selected. For instance, the pop-ups that appear using the **Goto** and **Load** commands without a specified target.
- **Show Settings In Editors**:
  Toggling this shows or hides the settings in some editors. This setting is also in the title bar of the editors, where this can be toggled.
- **Single Digit Input**:
  If this is set to "Yes", then all input integer dimmer values below 10 are multiplied by 10. The following rules apply:
  - Whole numbers from 1 to 9 are applied as values 10, 20, 30, 40, 50, 60, 70, 80, or 90.
  - Digits with decimal places, for example, 1.5, are deemed as 1.5.
  - When specifying any attribute in a command (Attribute "Dimmer" ...), single digit input is not applied.
  - When using the At command without any additional attributes being specified, the value is taken by the natural readout of the dimmer of the user profile.
- **Resolve Executor Assignments**:
  This setting is used in the **Command Editor**. When it is "Yes" and a command is edited by pressing executor buttons, then the handle of the object is used instead of the reference to the executor.
- **Preview Variables**:
  This setting changes how variables are previewed in the **Command Editor**. When this is "Yes", the command preview displays the specified variable's content. This only works if the variable already exists at that time.
- **Show Connectors**:
  This setting shows or hides the Connector Overlay. Learn more about the Connector Overlay in the **Output Configuration topic**.

- **DMX Tester Address Mode**:
  The following settings relate to the **DMX Tester**.
  - **Uni**: This splits the DMX address into universes and universe addresses.
  - **Abs**: This shows the DMX address as an absolute DMX address.
- **DMX Tester Retain Mode**:
  Toggles the retain mode in the DMX tester.
- **DMX Tester Mode**:
  - All: All DMX addresses are available for testing.
  - Patched: Only patched addresses can be tested.
  - Unpatched: Only unpatched addresses can be tested.
- **DMX Tester Test Value**:
  This is the test values used by the DMX tester.
- **Move Grid Cursor**:
  This is the same setting as the one available in the **Selection Grid** window. Learn more in the **Selection Grid Window topic**.

User Attribute Preferences

From the profile setting, the **User Attribute Preferences** can be accessed by tapping Edit Encoder Bar.

This is used to set the user profiles preferred readout type for the natural readout. It also has different resolutions and encoder press resolution multipliers.

Resolution multipliers define the factor by which an encoder changes a value when the encoder is pressed or, in the case of the dual encoders, the difference between the inner and outer ring.

At the top, there are some settings:

- **Time Layer Resolution**:
  Sets the encoder resolution for the time-related layers.
- **Phase Layer Resolution**:
  Sets the encoder resolution for the phaser-related layers.
- **Dual Encoder Factor**:
  This defines the multiplier the outer encoder ring uses in relation to the inner encoder ring.
- **Dual Encoder Press Factor**:
  This defines the multiplier for the outer encoder ring when the dual encoder key is pressed while turning the outer encoder ring.
- **Link Resolution**:
  This setting defines how the encoder resolution is linked between features. The options are **Single** and **Feature Group**. Learn more about this setting in the **Encoder Toolbar topic**.

The rest of the pop-up lists all the attributes in rows.

Three different columns with values can be changed for each attribute:

- **Natural Readout**:
  The Natural readout allows defining the desired readout type per attribute. For instance, dimmer values are best displayed in percentage readout. In contrast, pan or tilt values are better readable using the physical readout, which displays the pan and tilt values as degrees. This setting selects the desired readout when Natural is selected in sheets and encoders. The user-defined readout has a higher priority than the readout defined for the attribute. The

user-defined readout can be linked to the default (from the attribute definition). This option is called Default, and the actual readout is shown in angle brackets.

- **Encoder Resolution**:
  The encoder resolution defines how big an attribute value change will be when turning a dual encoder by 1 click. This setting can also be changed by pressing MA and tapping the channel function area in the encoder toolbar. Learn how in the **Encoder Resolution topic**.
- **Encoder Press Factor**:
  This multiplier is used when the inner encoder is pressed and turned for this attribute.
- **Ignore Channel Functions**:
  When this is set to "Yes", the Channel Sets area displays the channel sets of all channel functions at the same time

Learn more about the different resolutions available in the **Encoder Resolution topic**.

Wheel Mode

There are four different level wheel modes:

Additive:
Additive keeps the difference between dimmer values until they reach 0% or 100%, using the level wheel. After 0% or 100 % is reached, the values will be leveled out.

Example:
Fixture 1 has a value of 50, and fixture 2 is at 60. When the level wheel is used to take them both up to 100 and then back down, they both go down from 100 at the same time and at the same level.

Incremental:
Incremental would always keep the difference between the dimmer values, even if you reached 0% or 100% by using the dimmer wheel.

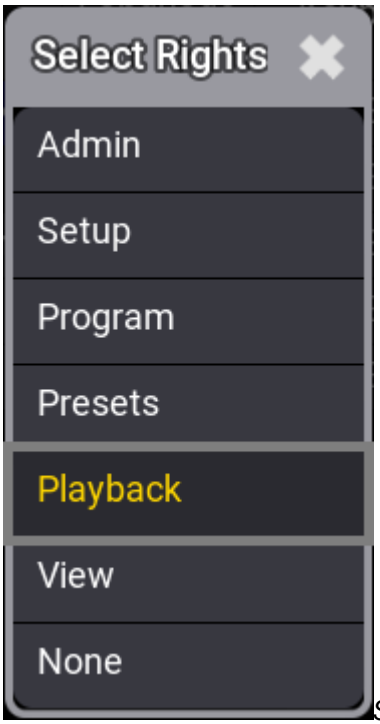
Example:
Fixture 1 has a value of 50, and fixture 2 is at 60. When the level wheel is used to take them both up to 100 and then back down, fixture 1 will start coming down first, and then fixture 2 will follow when fixture 1 is at 90.

Prop.+ (Proportional positive):
When using the level wheel to turn up the dimmer values, the difference in the dimmer values will **decrease**. Turning up to 100% will make all channels reach 100% at the same time.

Prop.- (Proportional negative):
When using the level wheel to turn up the dimmer values, the difference in the dimmer values will **increase**. Turning down the values will make all channels reach 0% at the same time.

User Rights

There are seven different levels of user rights in the system. They are a user setting - read more above.



Select Rights pop-up is used to select one of the rights levels

The settings are:

- **Admin**:
  This is the right to change everything in the console, system, and show.
- **Setup**:
  This will limit access to some of the elements in the console. There are other console settings that can be accessed.
- **Program**:
  At this level, the user cannot make major changes to the patch. It only gives access to the "Live Patch". Most programming operations can be done.
- **Presets**:
  This level allows for updating existing presets. But the user cannot edit the cue content.
- **Playback**:
  This level allows playback and running a programmed show. But the user cannot store anything.
- **View**:
  With this user right, it is not allowed to use a programmer. The user is allowed to call views and log in as a different user.
- **None**:
  The user is only allowed to log in as a different user.

Time and Frame Readout

The applied values are used in most places the time is displayed.

	Hint:
	The Readout can be changed for individual sheets in the **window settings**.

There is a hierarchy of the readout/format settings. The default is set in the user profile. If this default is changed in a window, then the window setting is used.

Timecode is an example of an area with many layers in the hierarchy. Read more about the timecode in the **Timecode section**.

	Hint:
	The user profile defined time readout is used in the encoder bar even if the readout for an, for instance, fixture sheet is changed separately.

The timing calculator value indicator is aligned with the preselected frame readout.

To provide better optical representation, the trailing zeros from frames or seconds are always suppressed. When the frame readout is set to a frame unit, the times will always display trailing zeros.

To easily distinguish between fractions of a second and frames, fractions of a second are separated from second using a dot (.), while frames are separated from seconds using a colon (:).

The time options are:

- **10d11h23m45**:
  The time is separated into days, hours, minutes, and seconds using letters as separators.
- **251h23m45**:
  This is separated into hours, minutes, and seconds using letters as separators. The hour number can become more than 24 if time is more than a day.
- **10.11:23:45**:
  The time is separated into days, hours, minutes, and seconds using a dot and colons as separators.
- **251:23:45**:
  This is separated into hours, minutes, and seconds using colons as separators. The hour number can become more than 24 if time is more than a day.

The frame options are:

- **Seconds**
- **24 fps**
- **25 fps**
- **30 fps**
- **60 fps**

	Hint:
	If frame readouts (24, 25, 30, 60 fps) are used, fractions are separated by a colon, and fractions of seconds are divided by dots.

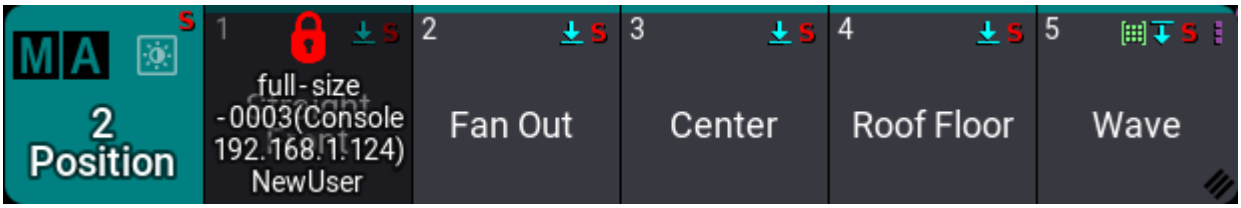
Fps means frames per second.

1.13.3. Object Ownership

A multi-user setup presents a situation where multiple users can try to edit the same object. This could cause unpredicted problems.

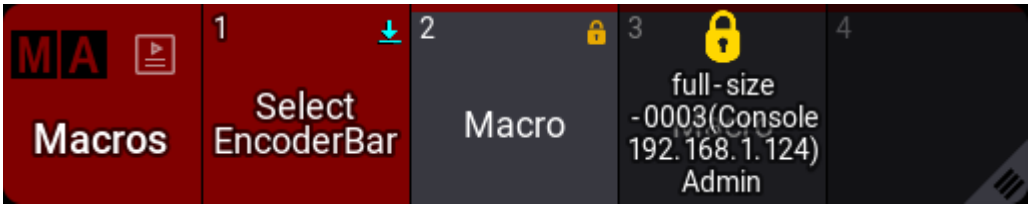
The grandMA3 system has a locking mechanism that temporarily prevents multiple users from editing the same object.

When a user starts to edit an object, the other users are presented with a pulsing lock icon and some text offering information about the station and the user editing the object.



Position preset 1 is being edited by a different user

An object can be fully locked by another user editing the entire object. A big red lock indicates this (see the example above). A big yellow lock indicates that an object is partially locked. This means that only a part of the object is being edited and is currently locked. Other elements of the object can still be edited.



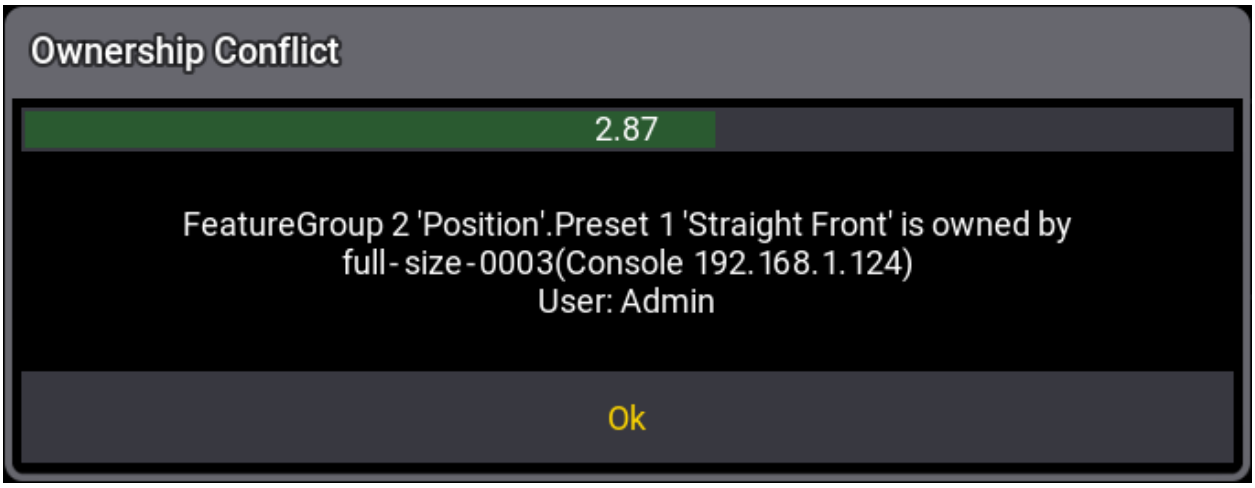
A macro line is being edited by a different user

All users can use the objects while they are being edited, but using the object before the edit is done might not reflect the edited values.

The user editing the object has ownership of the object while it is being edited. The station being used to edit the object will have a green frame in the relevant window while it is being edited.

Suppose the editing user is logged into several stations. In that case, other stations with this user display a yellow frame around the windows to indicate the user has started editing an object on a different station.

If a different user tries to edit a temporarily locked object, a pop-up appears with additional information:

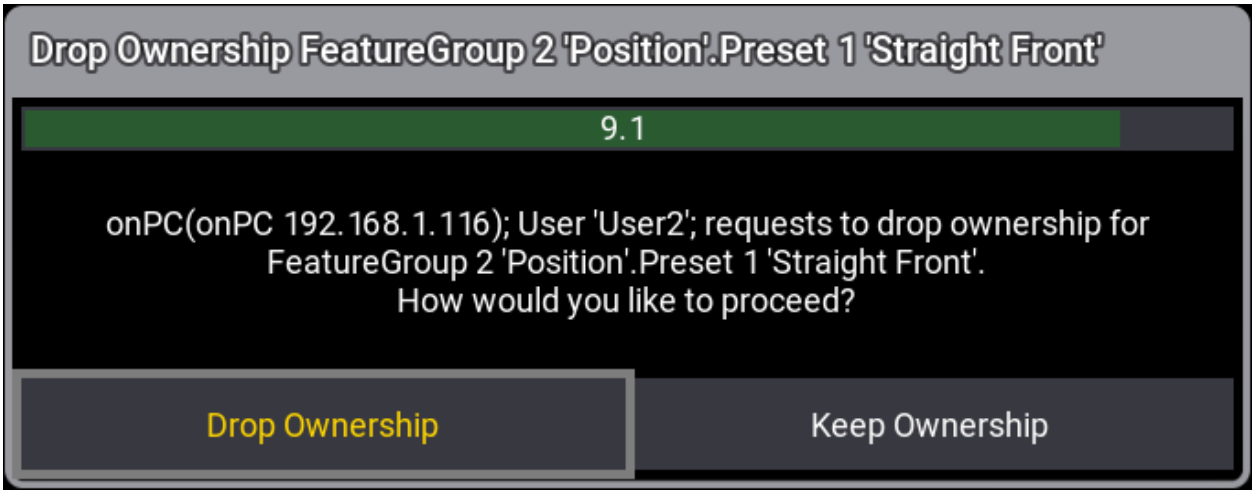


Ownership Conflict pop-up

The pop-up has a 10 seconds countdown that automatically closes the pop-up. It can be closed before the countdown ends by tapping Ok.

A user can try to get ownership of a temporarily locked object using the **DropOwnership keyword**.

This presents a pop-up at the stations belonging to the user who owns the object:



Drop Ownership pop-up

In this pop-up, the current owner can choose to drop the ownership by tapping Drop Ownership or keep it by tapping Keep Ownership.

This pop-up also has a countdown of 10 seconds. If the time runs out, then the ownership is dropped. This is the default action that allows other users to get objects currently owned by a user who has left the stations and simultaneously allows a user to keep ownership as an active choice.

The **ListOwnership keyword** can list all or specific ownerships in a session. The result is shown in the **Command Line History window**.

	Hint:
	Objects can also be locked by automatic processes that start as a follow-up of the user's actions. For example, when a user updates a preset, the Total Reference Update

	mechanism locks the needed objects with ownership while running. The automatic process owns the objects while the update process is running.

1.13.4. Screen Configuration

Screen Configurations contain information about which windows are visible in the different **User-Defined Areas** and the size of the user-defined areas. The screen configuration also contains information about what is assigned to the **View Buttons**.

The User-Defined Areas are the areas on screens 1 through 7 where windows can be added and arranged to be stored and recalled as views. Learn more about this in the **User-Defined Area topic**.

The view buttons can be used to store and recall views and clear the screen. They can also have other objects than views assigned. Learn more about View Buttons in the **View Bar and View Button topic**.

The screen configurations do not have a pool with objects to interact with.

Each user has a screen configuration assigned. This can be changed in the **User Settings**.

Screen configurations can be assigned to Executors and View Buttons. This makes it possible to change the current user's assigned screen configuration quickly. Screen configurations can also be called using the **ScreenConfiguration keyword**. Follow the link to the keyword to see examples of creating, calling, and assigning screen configurations.

The screen configurations can be used to have different sets of views on the screens and different View Button setups.

Examples of Use

Example 1 - Operator with Main and Backup Console:

A single operator has two consoles.

If the operator has one User with a User Profile, then both consoles would be logged in with this User, and the consoles would have the same programmer content and the same views on the screens.

Instead of having a mirrored console, the operator could expand the workspace and actively use the backup console to show other views. In this case, the operator creates a second User, "MyBackupConsole". This user has the same User Profile but has a different Screen Configuration. The backup console then uses this new User.

Now, the backup console still has the same programmer content but can show different views than the main console. The operator can now actively use both consoles and still have the backup function in case of failure.

Example 2 - Operator with a Console and a 3D Computer:

A single operator has one console and a computer with powerful graphic capabilities.

The desire is to use the computer as a 3D visualizer that the operator can use to show the playback state and to see **blind** programmer content.

The blind function hides programmer values from the output. This can be used to program elements without it being shown in the output. Learn more in the **What is the Programmer topic**.

To be able to follow the operator into blind, the onPC in the computer needs to log in as a User with the same User Profile as the operator, but to avoid it showing the same views as the console, it can be a User with a different Screen Configuration. Now, the onPC can show a single big **3D Viewer window**, and the operator can freely use the console.

The 3D Viewer shows a virtual 3D space with the stage setup. Learn more in the **3D Viewer topic**.

Example 3 - Operator on a Console and Designer with a Web Remote:

An operator is using a console to program a show. The Lighting Designer has a desire to see some information about the sequence and fixture values.

One way to accomplish this is by allowing the designer to connect their laptop to the lighting network using WiFi.

The designer can open a browser window and log in on the console using their own User with their own Screen Configuration. The designer User has limited rights but uses the same User Profile as the operator to see programmer content and to be able to manipulate some attribute values.

If the operator needs to help the designer set up the windows or change a setting, then the operator can change to the same Screen Configuration, make the changes, and then change back to their own Screen Configuration.

Create a New Screen Configuration

The examples in the **ScreenConfiguration keyword** include an example of creating a new Screen Configuration using the command line.

It is also possible to create a new screen configuration when editing the **ScreenConfig** setting in the **User Settings**. Edit the setting and tap New. This does not give access to label the screen configuration. This still needs to be done using the command line.

Copy a Screen Configuration

The examples in the keyword topic (link above) show how to create a new Screen Configuration, but it could also be relevant to copy an existing screen configuration.

This can be done using the following syntax:

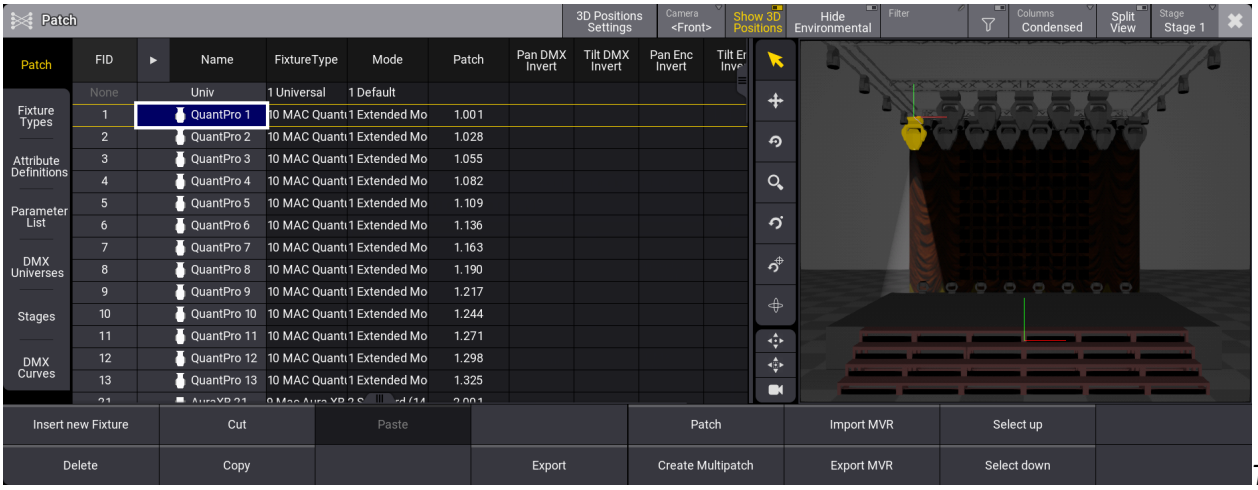
Copy ScreenConfiguration ["ScreenConfiguration_Name" or ScreenConfiguration_Number] At ScreenConfiguration ["ScreenConfiguration_Name" or ScreenConfiguration_Number]

The copy operation can be used to create a new screen configuration or copy into an existing one.

1.14. Patch and Fixture Setup

Fixtures need to be added to the show file before they can be controlled or **operated**.

Fixtures are added to the show using the **Patch** menu.



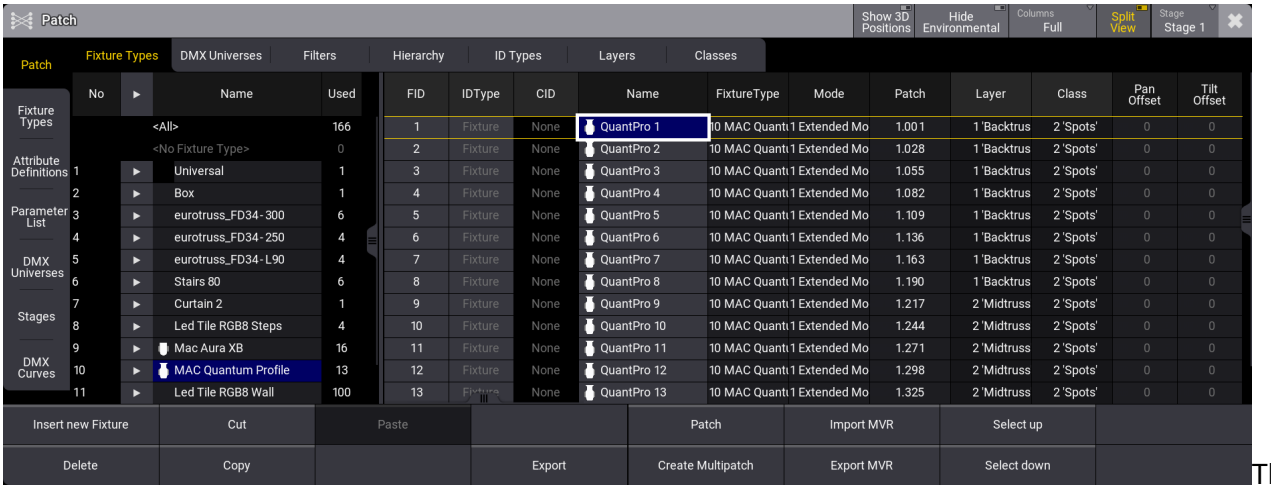
The patch menu with some fixtures patched

The patch menu gives access to **Fixture Types**, **Attribute Definitions**, **Parameter List**, **DMX Universes**, **Stages**, and **DMX Curves**.

The patch menu has two different modes. The image above shows the **Condensed** mode. This mode hides a lot of columns that might not be needed. The other mode is called **Full**. This shows all columns in the patch menu. The mode can be toggled by tapping Columns in the title bar.

The fixture list can also be filtered using the filter settings in the title bar or by activating Split View.

Split view filters the fixtures by different column properties.



The full Patch menu in Split View

The different split options appear as tabs that can be selected to filter the fixture list.

The list of fixtures is separated into two sides. The left side lists the different elements of the selected split filter. The right side lists the fixtures that qualify based on the object selected on the left side.

The tabs and split options (in Full mode) are:

- **Fixture Types**:
  The left side lists the imported fixture types, including <All> (all fixtures) and <No Fixture Type> (fixtures without an assigned fixture type). Learn more about fixture types in the **Fixture Types section**.
- **DMX Universes**:
  The left side lists the DMX universes, including <All> (all universes) and <Unpatched> (fixtures without an assigned DMX address). Learn more about DMX universes in the **DMX Universes topic**.
- **Filters**:
  The left side lists the filters in the Filter pool, including <All> (all fixtures). Learn more about filters in the **World and Filters section**.
- **Hierarchy**:
  The left side lists the hierarchical structure of the patch. It lists the parent elements and can be unfolded. The right side displays the elements directly dependent on the selected object on the left side.
- **ID Types**:
  The left side lists the different defined ID types, including <All> (all fixtures) and <No ID> (fixtures without an assigned FID or CID). Learn more about ID types in the **What are Fixtures topic**.
- **Layers**:
  The left side lists the different defined layers, including <All> (all fixtures) and <No Layer> (fixtures without an assigned layer). Learn more about layers in the **Classes and Layers topic**.
- **Classes**:
  The left side lists the different defined classes, including <All> (all fixtures) and <No Class> (fixtures without an assigned class). Learn more about classes in the **Classes and Layers topic**.

It is possible to have a small version of the 3D stage open in the patch. It can be toggled On or Off with Show 3D Positions in the title bar. It is visible in the first example image above.

When it is On, it is visible on the right side of the menu. It can be adjusted in size by moving the separator. Also, when it is On, a 3D Positions Settings appears in the title bar. Tapping this opens a limited version of the settings for the 3D viewer.

The settings have two tabs: **Misc** and **Label**.

The tab Misc offers these settings:

- **Wireframe**:
  Enables the rendering of the 3D scene as wireframe. By default, this setting is disabled.
- **Beam**:
  Changes the visualization of the beam. Available values are "Simple" and "Line". Simple is the default value.

- **BodyQuality**:
  Changes the visualization of the fixture body:
  - **Box**:
    The whole bounding box of the fixture is visualized as one box.
  - **Standard** (Default):
    The fixture is visualized with its original meshes until the vertex limit of 1 200 vertices is reached. If the overall vertex count of the fixture exceeds 1 200 vertices, the fixture is visualized with default meshes.
  - **Ultra**:
    The fixture is visualized with its original meshes, no matter its vertex count.

The Label settings are the same as for the normal 3D (link below).

When the 3D area has focus (for instance, when tapped), the encoder toolbar changes to the 3D encoder toolbar.

The 3D area in the patch does not visualize the current DMX output. The selected fixture in the patch shows an intensity output of 100%. The zoom of a fixture is visualized with the default value of the zoom attribute or, respectively, with the value of the Beam Angle of its Beam Geometry if a fixture does not have a zoom attribute. Other functions of the fixture (for example, color, iris, or gobo) are not visualized in the 3D area in the patch.

Learn more about 3D and the setting in the **3D Viewer topic**.

Subtopics

- **What are Fixtures**
- **Add Fixtures to the Show**
- **Add Multipatch Fixtures**
- **My Virtual Rig (MVR)**
- **Live Patch**
- **DMX Sheet**
- **DMX Universes**
- **Remove Fixtures from the Show**
- **Position Fixtures in the 3D Space**
- **3D Viewer**
- **Render Quality**
- **Camera Pool**
- **Stages in grandMA3**
- **Classes and Layers**
- **Attribute Definitions**
- **Parameter List**
- **DMX Curves**

1.14.1. What are Fixtures

Fixtures are the different devices that can be controlled by the grandMA3.

Fixtures are added using the patch menu. Each fixture is a row in the patch menu. Each row has some settings organized in columns. Two of the columns are **Fixture Type** and **Mode**. Fixture Type is the definition or description of the physical fixture and the DMX definition of the fixture might be separated into different modes. Each fixture type can have several different modes. Only one mode can be selected for each fixture in the patch. Read more about editing and creating fixture types in the **Fixture Types section**.

Fixtures contain different **Attributes** or **Parameters**. The parameters can have different values. Individual parameters are the ones manipulated when controlling the fixtures' attributes. For instance, changing the dimmer attribute for ten fixtures changes the dimmer parameter on each of the ten fixtures. Read more about attributes in the **Attribute Definitions topic**.

Fixtures in the grandMA3 are selected using an ID. There are 12 different **ID Types**. These ID Types can be used to organize the fixture list. Six of the ID Types are **Universal**, **Fixture**, **Channel**, **MArker**, **Multipatch**, and **PSR** - they are locked and cannot be renamed. **Universal** is a fixture type that is automatically created and should not be edited manually. **PSR** is a fixture type the system uses when utilizing the **Partial Show Read** function. It should not be used when manually adding an ID.
Multipatch (MP) ID type is a special type used to create multi-patch versions of a fixture - learn more in the **Add Multipatch Fixture** topic.
The remaining six ID types can be renamed to match the needs of the show. These are called **Custom ID** because they can be customized. The default names are **Houselights**, **NonDim**, **Media**, **Fog**, **Effect**, and **Pyro**. Each of these is a keyword. Renaming them changes the corresponding keywords. Read about the ID type keywords following these links: **Fixture**, **Channel**, **Universal**, **Houselights**, **NonDim**, **Media**, **Fog**, **Effect**, **Pyro**, **MArker**, **Multipatch**, and **PSR**.

Partial Show Read is used to access elements from a different show file. The function needs to compare the patch, and that is why this ID type exists. When the patch is compared and possibly combined, then elements can be transferred from the other show into the recurrent one. Learn more in the **Partial Show Read topic**.

A fixture can have a **Fixture ID** (FID) and one other **ID Type** (CID) with the same or a different ID number. A fixture needs at least one ID to be **operated**.

Fixtures are added to a stage. In a new empty show, there is a standard stage (Stage 1). More stages can be added to organize the setup. Read more about Stages in the **Stages topic** and about placing fixtures in the **Position Fixtures in the 3D Space topic**.

Fixtures can have **Class** and **Layer** information. These are two different ways to organize or group the fixtures. Read more in the **Classes and Layers topic**. The patch menu can be filtered to hide or show fixtures, for instance, using specific classes or layers.

Another way to organize the patch is to add fixtures as a "child" (structurally below a "parent") to another fixture. A special **Grouping** fixture is very useful for this. It is an "empty" fixture in the generic

manufacturer group. This can be useful if there are many fixtures of one type, for instance, 100 LED panels. A grouping fixture would be added to the patch, and the LED panels would be added inside or below the grouping fixture. The grouping fixture can have an ID. Selecting this ID then controls the LED panels inside or below the grouping fixture. An example of this structure can be seen in the demo showfile.

Fixtures also need to be patched to a DMX address in the grandMA3. If it does not have a DMX address, then the system does not know where to send the parameter values.

Example 1

A row is added to the patch menu. A Robe MegaPointe is assigned to the row. Now, the fixture exists in the patch menu. The fixture needs an ID to be selected and controlled. The fixture gets FID (Fixture ID) 1. Now, the fixture can be selected using the **Fixture 1** command. For the fixture to react, it needs to have the same DMX address as set in the fixture itself, and it needs to be connected to a DMX port set to the correct universe. The fixture gets address 3.54. The universe number is 3, and the DMX address in that universe is 54 - this needs to match the fixture setting.

Example 2

A row is added to the patch menu. A generic dimmer (8-bit) is assigned as the mode. The fixture is a part of the building and is used for house lights. The ID Type is changed to **Houselights** and CID (Channel ID / Custom ID depending on the ID type) is changed to 101. It does not get an FID. It can now be selected by pressing Channel three times to get the houselights keyword and the numbers 1 0 1.

The fixture gets the Patch address 1.101 (universe 1, address 101).

Hint:

The channel key is used to get all other ID types than **Fixture**. Pressing it multiple times toggles through the ID types.

1.14.2. Add Fixtures to the Show

This is the process for adding new fixtures (or devices) to the show and the patch, using the GUI.

Quick Steps in a new show:

1. Open the Patch Menu.
2. Follow the steps in the wizard:
    a. Select the fixture type.
    b. (optional) Add a custom name.
    c. Type the quantity.
    d. Type the first ID number.
    e. (If in **Full** mode) Select a Layer and class (**None** is an option).
    f. Type the patch address for the first fixture.
    g. Tap Apply.
3. Close the Patch menu and tap Save and Exit to save the changes.

These are steps needed for adding the first fixtures to a new show - read below for details.

Tap the video below to see the example.

Quick Steps in a show with existing fixtures:

1. Open the Patch Menu.
2. Select the row where the new fixtures should be inserted and tap the Insert New Fixture button.
3. Make sure the desired source is selected.
4. Select the desired fixture type (use the filter to limit the list) and tap Select.
5. Fill in the rest of the fields in the wizard and tap Create !.
6. Edit any desired field in the patch grid before closing the Patch menu.
7. Close the Patch menu and save the changes.

These are steps needed to add more fixtures in an existing show - read below for details.

Tap the video below to see the example.

Everything about adding fixtures is done from the Patch menu.

Navigate to the Patch Menu

The Patch menu needs to be open to add fixtures.

1. Press the Menu key.
2. Tap the Patch button in the menu pop-up.

The patch menu is now open.

The open Patch menu - in condensed mode - with some fixtures

When the menu is opened the first time, a wizard helps to add the first fixtures to the show, and instead of the patch menu, there is a guide through the fixture selection and the **Insert New Fixtures** pop-up (see below).

Insert a Device in the Patch

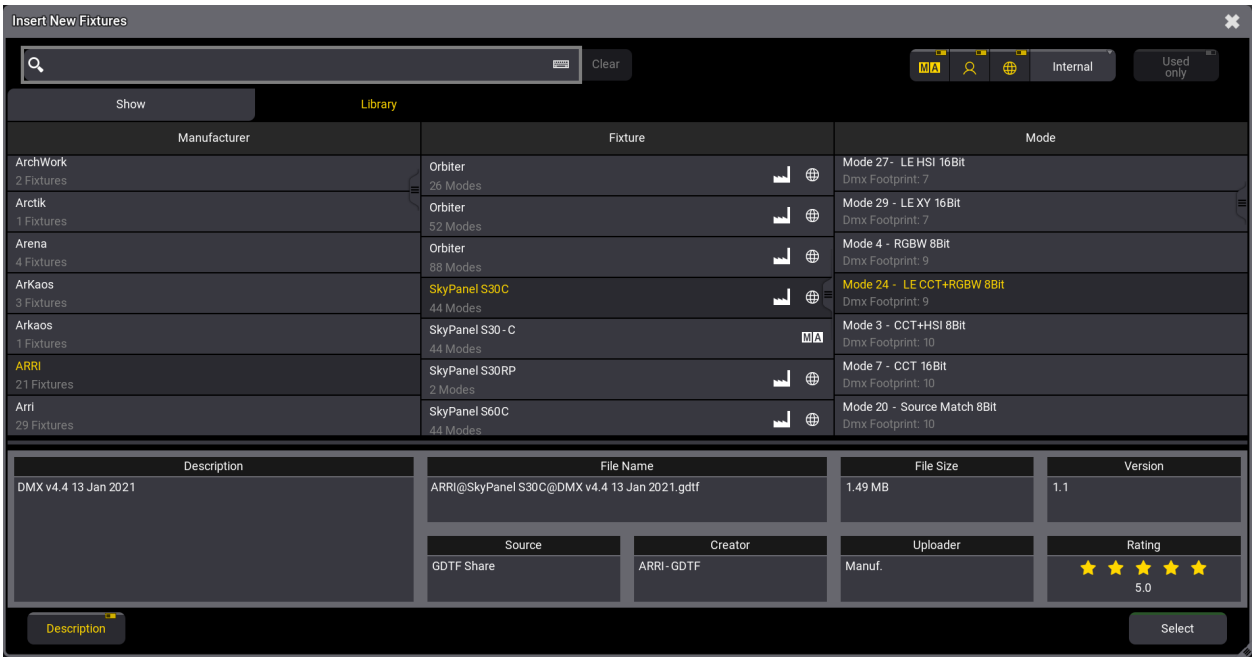
Each fixture needs a row in the patch menu.

The fixtures belong to a "parent" stage object. The default parent is **Stage 1**. Several stages can be created - read more in the **Stage topic**.

The selected stage can be changed using the stage button in the title bar.

1. Select the stage where the fixtures should be added using the button in the title bar.
2. Tap the New Fixture area on the list if it needs to be added to the bottom, or tap an existing fixture to insert new fixtures above the selected fixture.
3. Tap Insert new Fixture.

This opens the **Insert New Fixture** wizard pop-up on the **Select DMX Mode to use** part.



Select a fixture in the libraries

Below the search bar are two tabs: **Show** and **Library**. Show lists the fixture types already imported into the show file. This list can be limited to only showing the patched fixture types. This can be done by activating the Used only button. The **Library** tab lists the different fixture types that can be used. With library selected, there are extra buttons available on the right side.

One button selects the drive. This makes it possible to select the internal drive, previously installed versions library, or an external USB drive. In the image above, it is the Internal button. The button is labeled to show the selected drive. Next to this button, there are small toggle buttons that select different sources.

The sources are MA (grandMA2 converted fixture files and grandMA3 fixture files), User, and Shares (GDTF share and grandMA3 fixture share). The share button is only available when there is an active connection to a World Server (learn more in the **World Server topic**).

	Restriction:
	grandMA2 fixture files cannot be loaded directly into the grandMA3. The grandMA2 converted source mentioned above is the grandMA2 library converted to match the grandMA3 structure. Fixture files are converted when a grandMA2 file is stored as a grandMA3 file. Learn more in the grandMA2 manual / Using the Backup Menu / Save as grandMA3.

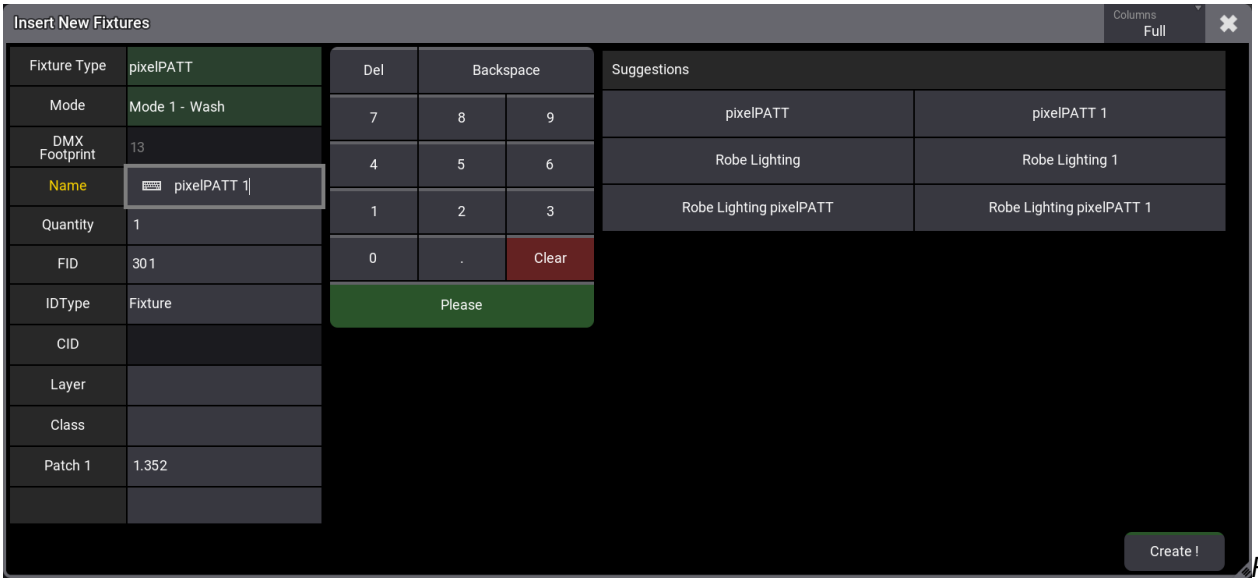
4. Select the desired drive and activate the desired sources.

The fixture libraries are sorted by manufacturers in the left column. Selecting a manufacturer lists the fixtures from that manufacturer in the center column. Selecting a fixture lists the available modes in the right column.

At the bottom, a Description button can show an area with extra information about the selected fixture type file.

5. The list can be filtered by typing a word in the **Search** input. The search is across manufacturers, fixtures, and modes (If a fixture has multiple modes, it will show them all, but only fixtures that include the searched mode).
6. Select the desired fixture and mode from the list and tap Select.

This reveals the **Insert New Fixtures Wizard**. This has a list of elements that need information to add the fixtures to the patch. The wizard comes in two different modes, depending on the **Columns** mode of the patch. This mode can be toggled in the Columns button in the title bar. When the mode is **Full,** then there are more elements (ID Type, CID, Layer, and Class). This is the full version of the pop-up:



Fill out all the desired fields in the right column to insert fixtures

7. All the fields in the right column can be filled with information. The required fields have a value suggestion. These can be edited to suit the needs. All the information can also be edited afterward in the patch menu. The area on the right side adapts to help fill out the selected field. It might have suggestions for input.
8. Accept the suggested values, or edit them to suit the needs.
9. Tap Create ! to add the fixtures.

A row now represents each new fixture in the patch menu. Read below for an explanation of each column in the patch menu and detailed information about changing some values.

Short Explanation of Most Columns in the Patch Menu

The patch menu has many columns. This is a short explanation of each. Remember that the patch menu has a condensed and full mode. Select Full to see all the columns.

- **FID**:
  This is the Fixture ID of the fixture. Read more in the **Assign an ID to fixtures** below.
- **IDType**:
  This is the ID Type of the fixture. Read more in the **Assign an ID to fixtures** below.
- **CID**:
  This is the fixture's CID. Read more in the **Assign an ID to fixtures** below.

- **Name**:
  This is the name of the fixture. If there are sub-fixtures, then a right-pointed arrow can be tapped to unfold the sub-fixtures.
- **FixtureType**:
  This is the name of the selected fixture type.
- **Mode**:
  This is the mode of the selected fixture.
- **Patch**:
  This is the first DMX address of the fixture. Read more in the **Assign DMX address to fixtures** below.
- **Layer**:
  Fixtures can be organized in Layers. This is the layer information for the fixture. Read more in the **Classes and Layers topic**.
- **Class**:
  Fixtures can be organized in Classes. This is the class information for the fixture. Read more in the **Classes and Layers topic**.
- **Offset Pan**:
  This offset value can offset the DMX output for the Pan attribute. It is applied just before the DMX output, which means the offset is not shown in the programmer, cues, or presets. The offset can be useful if, for instance, a fixture is hung differently than what has been programmed, especially in a touring situation where it is not a permanent change.
- **Offset Tilt**:
  This is the same as the Offset Pan but for the tilt attribute. Read above.
- **DMX Invert Pan**:
  The Pan DMX output can be inverted by editing this field. This can be useful depending on how the fixtures are mounted.
- **DMX Invert Tilt**:
  The Pan DMX output can be inverted by editing this field. This can be useful depending on how the fixtures are mounted.
- **Enc Invert Pan**:
  Editing this field allows the pan to be inverted for the encoder rotation, which can be useful depending on how the fixtures are mounted.
- **Enc Invert Tilt**:
  Editing this field allows tilt to be inverted for the encoder rotation. Depending on how the fixtures are mounted, this can be useful.
- **Pos X**:
  This is the fixture's position on the X-axis in the 3D window.
- **Pos Y**:
  This is the fixture's position on the Y-axis in the 3D window.
- **Pos Z**:
  This is the fixture's position on the Z-axis in the 3D window.
- **Rot X**:
  It is the rotation of the fixture on the X-axis in the 3D window.
- **Rot Y**:
  It is the rotation of the fixture on the Y-axis in the 3D window.
- **Rot Z**:
  It is the rotation of the fixture on the Z-axis in the 3D window.
- **Scale X**:
  This can scale the fixture/object on the X-axis in the 3D window. The scale value is a factor,

with 1 being the default. A higher number makes the object bigger, and a lower number makes it smaller.

- **Scale Y**:
  This can scale the fixture/object on the Y-axis in the 3D window. The scale value is a factor, with 1 being the default. A higher number makes the object bigger, and a lower number makes it smaller.
- **Scale Z**:
  This can scale the fixture/object on the Z-axis in the 3D window. The scale value is a factor, with 1 being the default. A higher number makes the object bigger, and a lower number makes it smaller.
- **Gel Color**:
  Here, a color that will be added to the fixture's output can be defined. This is useful for adding gels to conventional fixtures. This is visualized in the console and 3D Viewer.
- **Note**:
  This adds a multiline note to the fixture.
- **Beam Angle**:
  Here, the fixture's beam angle can be defined. This is useful for conventional fixtures where different angles might be needed. This is visualized in the 3D window.
- **Shadow Quality**:
  This setting defines the render quality of the shadows created when this fixture's light beam hits a different object that casts shadows. A higher level of quality improves the real-world look, but it also increases the calculations needed for the 3D visualization. The options are **None**, **Low**, **Medium**, **High**, and **Very High**.
  Cast Shadows need to be set to Yes for the shadows to be rendered (see below). This setting is not available for environmental fixture types.
- **Cast Shadow**:
  Each fixture can cast a shadow or not. This setting defines if this fixture should cast a shadow.
- **Follow Target**:
  Turning this to Yes makes the fixtures or 3D objects something that can be selected or picked using the **Follow** function in the 3D Viewer.
- **Selectable 3D**:
  This is a Yes or No (text is hidden) field. Yes means that the fixture can be selected using the selection tool in the **3D Viewer**. Turning this off (No option) for stage elements that do not need to be controlled can be useful.
- **Visible 3D**:
  This is a Yes or No (text is hidden) field. Yes means that it is shown in the **3D Viewer**.
- **Target Space**:
  The MAker fixtures use this. This is the selected target space for the MArker fixture. It defines the available space where fixtures can move their light beams when selecting this MAker fixture. Learn more in the **MAker Fixture topic**.
- **Movement Space**:
  The MAker fixtures use this. This is the selected movement space for the MArker fixture. It defines the space wherein the MAker fixture can move. Learn more in the **MAker Fixture topic**.
- **Master React**:
  This setting has three options: **None**, **Group**, and **Grand**. This defines if the intensity is affected by a master. If Grand is selected, then group masters will also affect the fixture. If Group is selected, then they are not affected by the grandmaster. None makes the fixture ignore both groups and grandmasters.
- **Grid Rot and Inv**:
  These six settings allow for modifying the arrangement of subfixtures in the selection grid. For

instance, if a linear LED bar has multiple LED cells defined in the fixture type, it will use the information in the fixture type to show the individual cells when the subfixtures are shown in the selection grid. The Grid settings can be used to rotate and invert the representation of subfixtures in the Selection Grid.

- **Appearance**:
  An appearance can be assigned to the fixture.

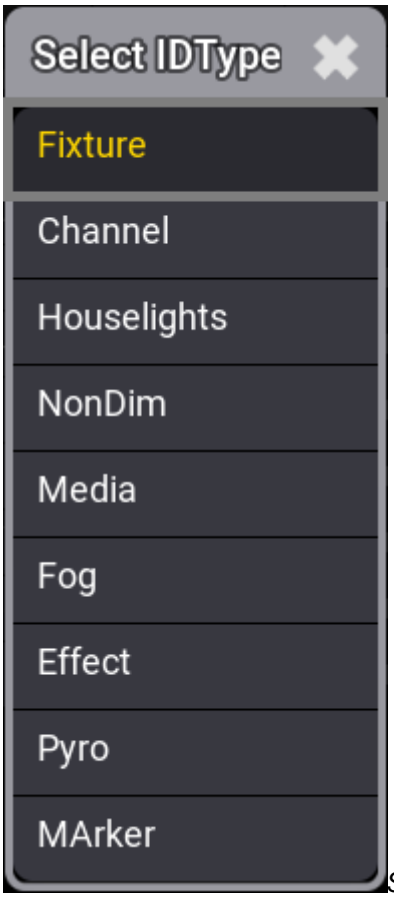
Assign an ID to Fixtures

The fixtures need at least one ID to be selected and controlled.

There are two types of ID numbers for each fixture. Fixtures can have both or just one of the two - but it needs at least one.

The **FID** is the default fixture ID. The number here is used with the **Fixture** keyword.

The **CID** is the second ID. This can be used if the **IDType** differs from "Fixture". Editing the IDType field opens the **Select IDType** pop-up.



Select the desired IDType

This lists the different valid IDTypes. Select one that is not "Fixture" to use the CID.

	Important:
	There are other hidden types called **Universal**, **Multipatch**, and **PSR**. Universal is a special type the software automatically creates for a special universal fixture added to the patch. Multipatch is used for special

	multipatch versions of fixtures. PSR is used with Partial Show Read. These IDTypes cannot be used for normal fixtures.

1. Select the ID fields in one of the two ID columns (FID or CID) for the fixture rows where an ID is to be assigned. The selection order is important.
2. Type a number on the keyboard and assign the number by pressing Please.

Now, the fixtures have an ID. The numbers are assigned sequentially based on the selection order.

Assign DMX Address to Fixtures

The fixtures need to be assigned a DMX universe and address before being able to create any output.

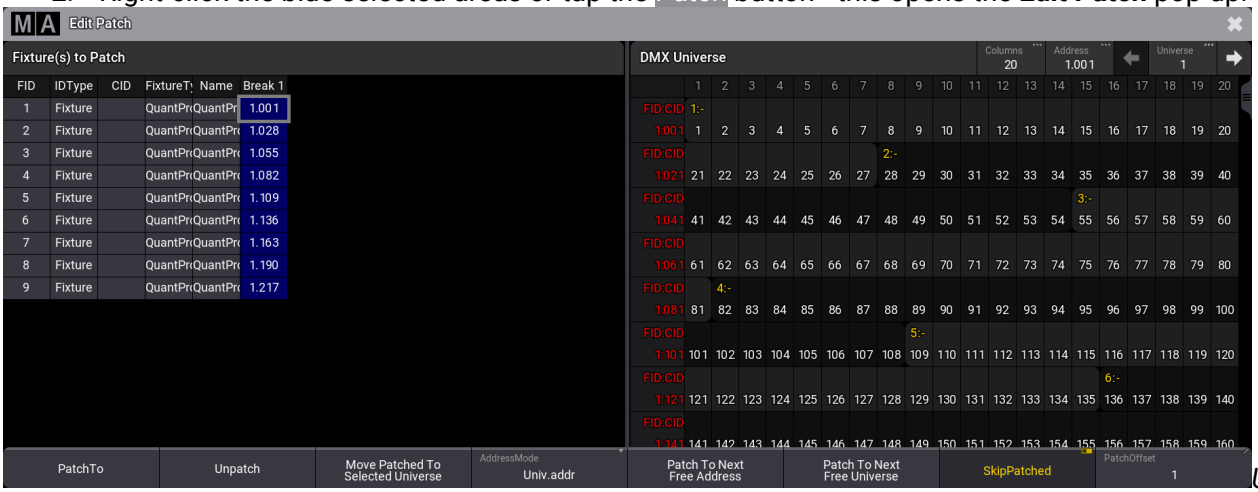
There are two primary options for giving the fixtures an address. One is to type the desired address directly in the patch field, and the other is to use the dedicated **Edit Patch** pop-up.

Type the number:

1. Select the fields in the Patch column for the fixture rows where the address will be assigned. The selection order is important.
2. Type the DMX universe and address separated by a dot (for example, **2.1**) on the keyboard and assign the address by pressing Please.

Use the Edit Patch menu:

1. Select the fields in the Patch column for the fixture rows where the address will be assigned. The selection order is important.
2. Right-click the blue selected areas or tap the Patch button - this opens the **Edit Patch** pop-up.



Use the Edit Patch pop-up to get a visual overview of the DMX universe

This pop-up is divided into two sides. On the left side, there are all the selected fixtures. On the right side is a view with all the DMX universes and addresses. The purpose of the menu is that fixtures can be selected on the left side, and the universe list on the right side shows where there is available space in the universes. There are several ways to navigate universes and assign the fixtures to the selected DMX address.

The numbers on the right side are red if the universe is not granted.

The pop-up also opens a dedicated encoder toolbar:

Edit patch encoder toolbar is used for easy navigation

This toolbar can be used to select fixtures, universe, and address.

This is one way to do it:

1. Select the fixtures in the correct order.
2. Use the encoders to navigate to the desired address.
3. Tap the **Address** encoder to assign the fixtures to the address.

The Edit Patch pop-up has several buttons at the bottom:

- **PatchTo**:
  Opens a small pop-up where the desired patch address can be typed.
- **Unpatch**:
  This removes the patch address from the fixtures.
- **Move Patched To Selected Universe**:
  This moves the fixtures to the selected universe and keeps the DMX address.
- **AddressMode**:
  This toggle button changes how the DMX address is displayed. They can be split up into universes and a range DMX address from 1 through 512 - this option is called **Univ.addr**. The other option is **Absolute**, which shows the addresses continuously starting from number 1 and upwards.
- **Patch To Next Free Address**:
  This gives the fixtures the next available DMX addresses.
- **Patch To Next Free Universe**:
  This patches the fixtures to the next empty universe.
- **SkipPatched**:
  This will skip addresses that have patched fixtures when scrolling through the universes.
- **PatchOffset**:
  This can set the desired number of DMX channels between the fixtures. If the number is lower than the number of channels the fixtures use, they are patched as close as possible. If the number is more than the used channels, then the PatchOffset number is used.

The **Edit Patch** menu auto closes when all the selected fixtures are patched.

Filtering the Patch Menu

Filtering in the patch menu can be useful when there are a lot of fixtures and stage elements.

There are several ways to filter the fixtures in the patch menu.

An existing filter can be assigned using the Filter input in the title bar. Tapping this opens a small select pop-up that lists all existing filters (read about creating a filter in the **Create a Filter topic**). Two other options are **None** (no filter) and **New** (create new filter). Select the desired filter.

Filtering needs to be turned On for the filter to be active. Tap the filter icon ( ) (between Filter and Columns) to turn it On. This button toggles On or Off.

Turning filters On also exposes column filters. An extra row is displayed above the fixture rows just below the column titles. Any column value can filter the patch simply by editing the filter field for a column. Only one filter value can be applied to each column.

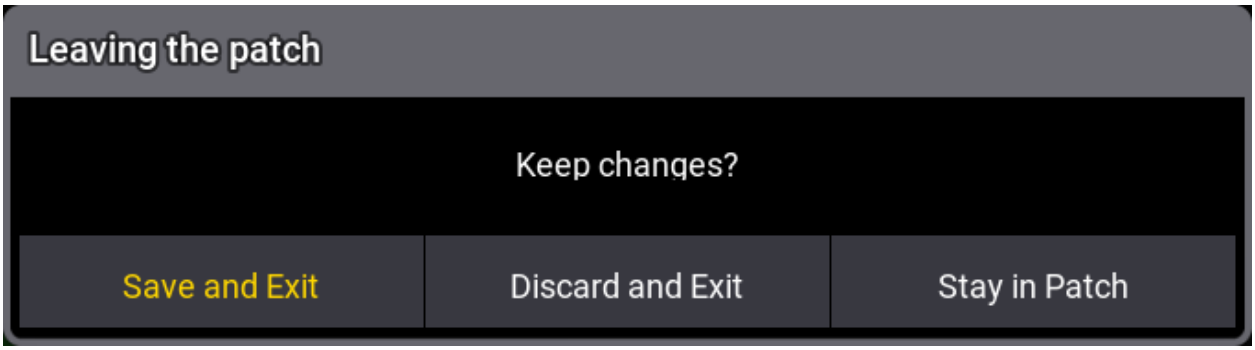
Edit a column filter field to assign a value. Some fields open a small selection pop-up with a list of available values, while others are input fields where text needs to be written.

The column filters can be combined with a filter from the pool.

The **Split View** is also a way to filter the patch. Learn more about the split view in the **Patch and Fixture Setup topic**.

Closing the Patch Menu

There is a pop-up asking what to do when exiting the patch menu.



Leave patch pop-up with options

This asks if the changes should be kept.

Tap Save and Exit to save the changes and leave the patch menu. Tap Discard and Exit to cancel any changes and leave the patch menu. Tap Stay in Patch to stay in the patch menu.

1.14.3. Add Multipatch Fixtures

Multipatch fixtures are virtual fixtures where every parameter follows the primary fixture (the fixture they mirror).

These multipatch fixtures can be used when multiple physical fixtures are controlled by the same ID. The virtual fixtures do not cost any extra parameters as long as the primary fixture has a patched and granted DMX address. They can have their own patch address and there will be DMX output to these addresses. For more information about Granted in DMX Universes, see **DMX Universes**.

The multipatch fixtures can be assigned to **layouts** and positioned in the **3D**. They are not part of the selection grid and can, therefore, not be used in MAtricks.

Changing the parameters on a selected multipatch fixture actually changes the parameters of the primary fixture. All multipatched fixtures to this primary fixture will reflect the changed values.

Changing the location of the primary fixtures in the patch or adding multipatch fixtures to other fixtures might renumber all the multipatch fixtures.

Multipatch fixtures can be selected individually. The selection color is light red instead of yellow (default colors). In the fixture sheet, the primary ID will flash between yellow and light red when a multipatch fixture is selected.

They can be selected using the command line using their unique multipatch CID or by an index number of the primary fixture.

Example

Have a setup like this:

FID	▶	Name	FixtureType	Mode	Patch
None		Univ	1 Universal	1 Default	
1		QuantPro 1	10 MAC Quanti	1 Extended Mode	1.001
MP 1		<QuantPro 1>	<10 MAC Quan	<1 Extended Mode	6.001
MP 2		<QuantPro 1>	<10 MAC Quan	<1 Extended Mode	6.028
2		QuantPro 2	10 MAC Quanti	1 Extended Mode	1.028
MP 1		<QuantPro 2>	<10 MAC Quan	<1 Extended Mode	5.001
MP 2		<QuantPro 2>	<10 MAC Quan	<1 Extended Mode	5.028

Patch with two fixtures with two multipatched fixtures each

The first multipatch fixture on FID 2 has the hidden multipatch CID number of 3 - it is the third multipatch fixture in the entire patch.

It can be selected by this syntax: **Multipatch 3** or **Fixture 2 Multipatch 1**.

	Hint:
	The hidden multipatch CID should not be used to select the fixtures. The CID number will possibly change if changes are made to patch. The number is not fixed. It is recommended to use the index number based on the primary fixture (the second example above).

Learn more about the keyword in the **Multipatch keyword** topic.

Add Multipatch Fixtures

Requirement:

Have some fixtures patched to the show.

1. Enter the Patch menu by pressing Menu and then tap Patch.
2. Select the primary fixture that needs multipatched fixtures.
3. Tap Create Multipatch.
4. Enter the number of needed multipatch fixtures in the **Amount of MP fixtures** pop-up and confirm it by tapping Please.

The new multipatched fixtures can now get a DMX patch address. The multipatch fixtures are limited in what can be changed, but name and location information is part of what can be changed for the fixtures.

5. When all needed information is assigned to the new multipatch fixtures, the patch menu can be closed by tapping and saving the changes.

Tap the video below to see the example.

1.14.4. My Virtual Rig (MVR)

MVR (My Virtual Rig) is a file format that is used to share data for a stage set up between a lighting console, a visualizer, a CAD program or similar tools. This allows for transferring parametric and geometric data between different programs.

It is a complementary system to the GDTF files.

MVR is described in the GDTF help pages (external link): **https://gdtf-share.com/help/en/help/mvr/index.html**

Import MVR

MVR files can be imported into a show file. This is done from the **Patch menu**. The MVR file needs to be in the correct folder for the software to find it. The folder is ../gma3_library/mvr. It is the same location if the files are on a USB stick.

There is a button at the bottom of the patch menu called Import MVR. Tapping this opens an **Import MVR** pop-up. This is used to browse and select the MVR file. The source drive can be selected in the title bar by tapping the drive selection button (next to the).

Select the desired file in the list and tap Import MVR to import the MVR file.

The content of the MVR file is added to the patch and the information in the MVR files means that there might be added **Layers, Classes**, stage elements, or fixtures.

It is a good idea to store the show file **before** importing the MVR. In case it adds unwanted elements or in worst case corrupts the show.

Export MVR

Exporting the patch to an MVR file is done from the Patch menu. The file is created in the [folder with grandMA3]\shared\resource\lib_mvr folder on the selected drive.

At the bottom of the menu, there is a button called Export MVR. Tapping this opens an **Export MVR** pop-up. This can be used to select the desired drive and give the file a name. It also lists already existing MVR files on the selected drive.

The MVR file contains the entire patch including fixture files, stage elements, and organizational elements like the **Classes and Layers**.

It is a good idea to store the show file **before** exporting it to the MVR file.

	Important:
	In order to export an MVR file or GDTF file that contains user meshes and gobos, save the show file to the local hard drive before exporting.

MVR-xchange

The MVR menu in the In & Out menu offers a platform to exchange and manage MVR files via a network connection. See **MVR-xchange**, to get more information.

1.14.5. Live Patch

The **Live Patch** menu is a version of the **Patch menu** with limited functionality. It looks like the patch menu, but some buttons are missing and some columns have a dark background - these columns cannot be edited.

The idea behind Live Patch is that it is only possible to change the elements that do not require a new show upload to the system.

This means that fixtures/objects cannot be added or removed.

In the live patch it is possible to change the following columns:

- Name
- Patch
- Offset Pan
- Offset Tilt
- DMX Invert Pan
- DMX Invert Tilt
- Enc Invert Pan
- Enc invert Tilt
- Position and Rotation
- Gel Color
- Note
- Beam Angle
- Shadow Quality
- Cast Shadow
- Follow Target
- Selectable 3D
- Visible 3D
- Master React
- Values for automatic position calculation
- Grid rotation and invert values
- Appearance

The automatic position calculation columns are for the four points used for automatic fixture position calculation. Read more about this in the **Position Fixtures in the 3D Space** topic.

Follow the link to the Patch menu (above) to read more about the other columns.

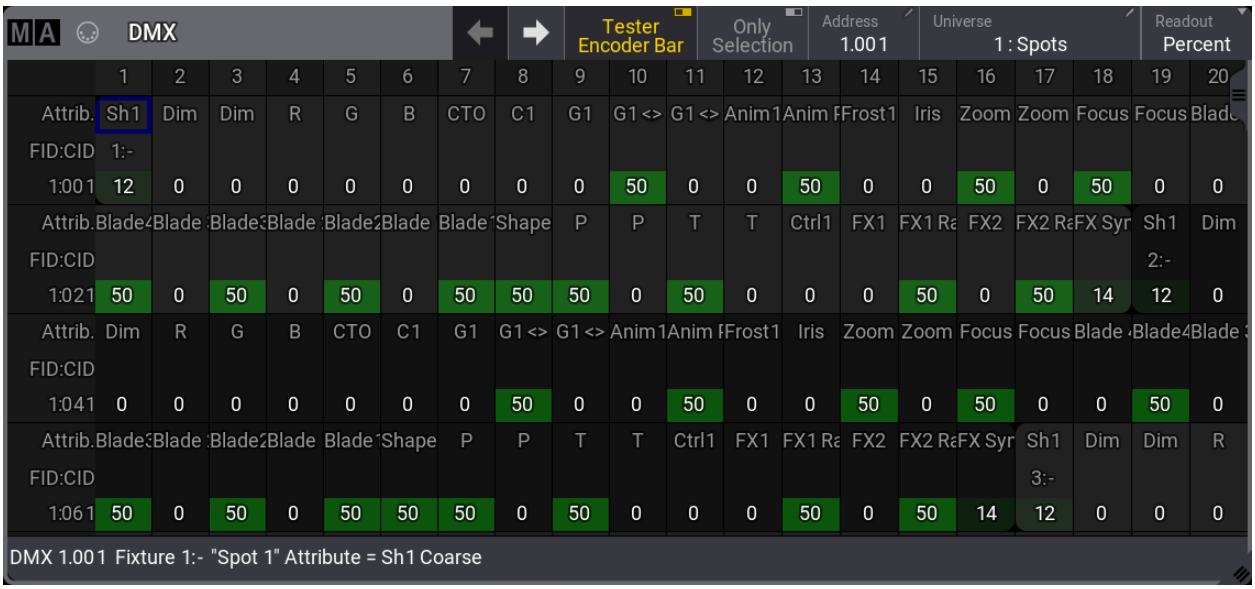
Changes made in the live patch are changed immediately.

The lowest button on the left side gives access to an additional RDM Fixtures menu. This lists fixtures discovered using RDM. Learn more about RDM in the **RDM section**.

1.14.6. DMX Sheet

The DMX sheet displays the actual DMX output from the system. It shows the result from the sequences, programmer, incoming merged DMX, and any masters that might limit the output.

The sheet can be created as a window on any empty space on the screens. The minimum size is 1.5 squares wide and 2 high. It is created like any other window using the **add windows**.



DMX sheet with patched fixtures

The main part of this window displays a big grid with every DMX address represented by its own square. This makes it a very long list (all 1024 universes are there).

The first column on the left side is a label for the second column.

There are two ways to see the DMX address: Absolute DMX address and the normal Universe divided. This can be changed in the **Sheet Settings**. The absolute address mode displays the DMX addresses as a continuous number. This means that the first address in the second universe will not be written as "2:001" but as "513" (512 addresses from the first universe + 1 from the second).

The top row displays the column number. Since the grid is a matrix, the column number should be added to the address shown in the vertical bar - the actual calculation for the square number is the number in the left column plus the number in the top row minus one.

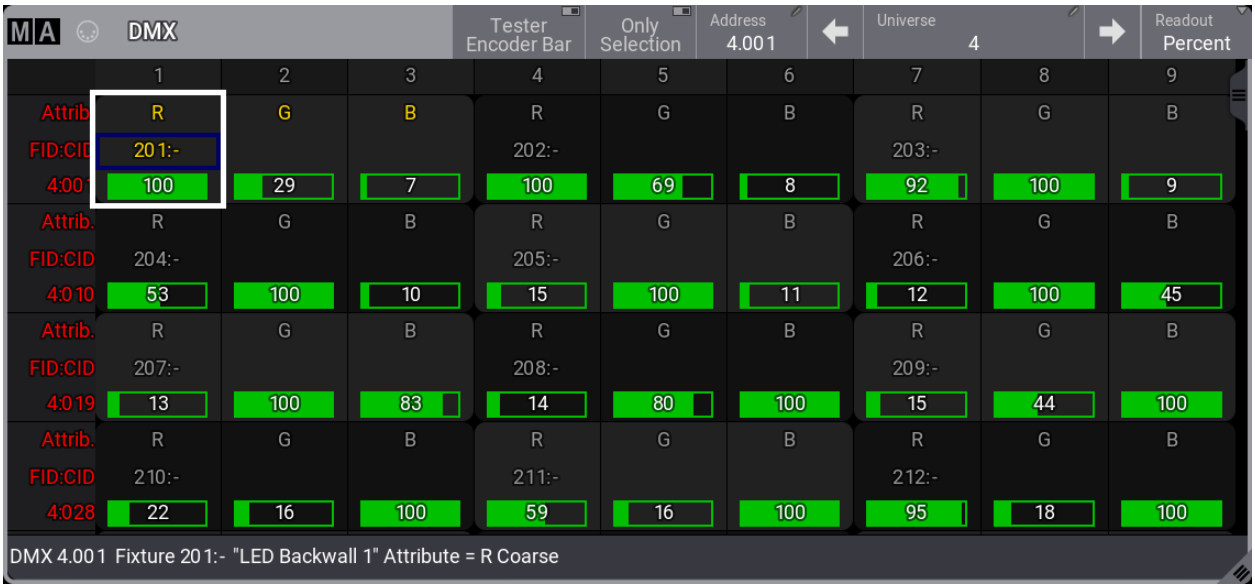
Each square can display up to three different types of information:

- **Value**:
  Toggling this setting shows or hides the DMX value of the DMX channel in the sheet.
- **Attribute**:
  Toggling this setting shows or hides the attribute related to the DMX channel in the sheet.

- **ID**:
  Toggling this setting shows or hides the FID and CID number assigned to the fixture patched to the DMX channel in the sheet.

The three elements can be turned On or Off individually in the mask settings (read the **Sheets Settings** below for more about the settings).

If a fixture uses more than one DMX channel, the rounded corners are on the first and last attribute/DMX channel. The background color alternates between two shades of gray to indicate different fixtures.



Detail of the DMX sheet

In the example above, some RGB LED fixtures are patched. The selected fixture (FID 201) gets a yellow text color in attributes and FID:CID.

If labels in the leftmost column have red text, then the channels/universes are not granted. Read more about getting more parameters in the **Expand the Amount of Parameters topic**.

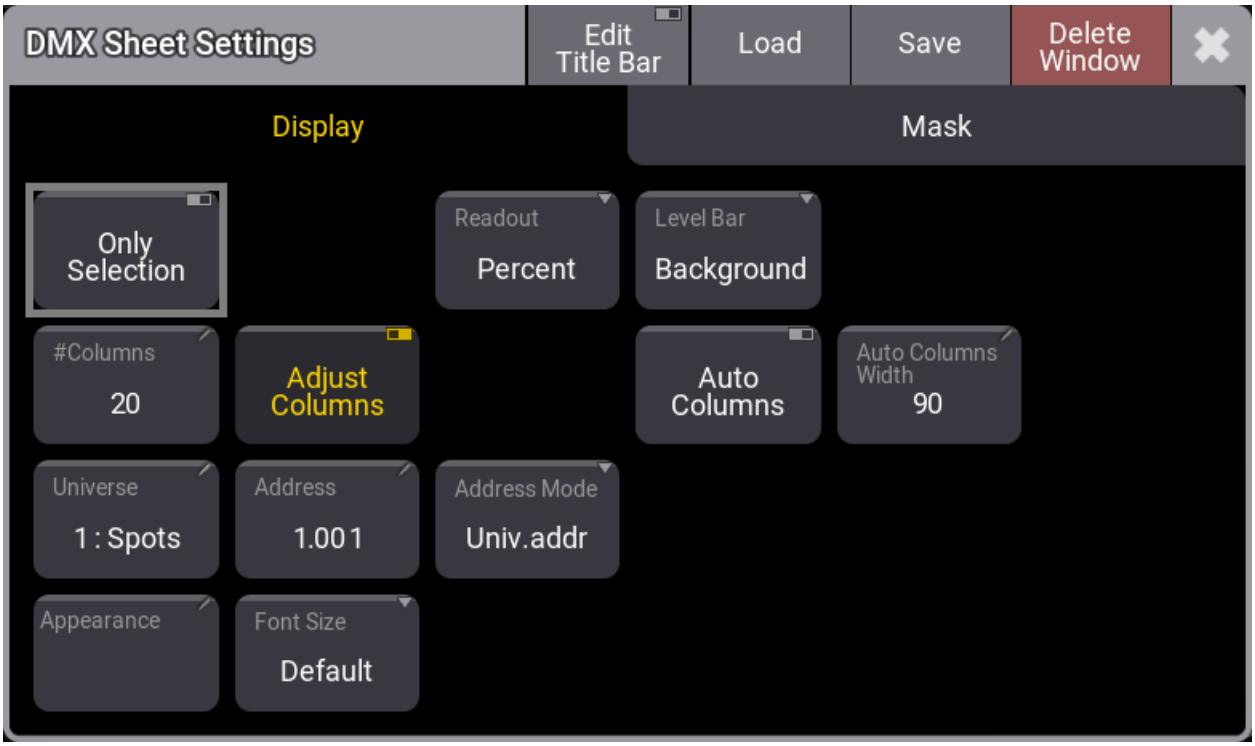
If the cursor is moved to a square, information about that square is displayed in a text line at the bottom of the window (see the image above for an example).

The two images show the two different ways the value can be visualized in the DMX sheet. The first is called **Background**, and the second is **Bar**. They can be changed or turned Off in the **Sheet Settings**. The setting is called **Level Bar**.

Sheet Settings

The DMX sheet has some settings that define how the sheet shows the data.

Tap the MA logo in the upper left corner of the window to open the settings. It opens the Display tab as a default.



DMX Sheet Settings - Display tab

This is a short description of all the Display settings for the DMX Sheet:

- **Only Selection**:
  When this setting is On, the DMX sheet will automatically scroll to the universe of the first fixture selected in the programmer. It automatically turns Off if a specific universe is selected.
- **#Columns**:
  This input button sets the number of columns a sheet should display (the settings **Transpose** and **Adjust Columns** must be switched On except in the **DMX Sheet**).

  The DMX Sheet shows all the DMX channels and their output values. Learn more in the **DMX Sheet topic**.

- **Universe**:
  This input button can change the universe a sheet should display. Tapping it opens a number input where a universe number can be typed. There is a special option called **Selected**. This makes the sheet scroll to the selected universe.
- **Appearance**:
  Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.
- **DmxTestBar** (only when Edit Title Bar is On):
  This setting is only visible when the Edit Title Bar is On. Thus, changing the setting is only possible when the setting is visible in the title bar.
  The setting is an On/Off button that shows or hides the DMX tester encoder bar when the DMX Sheet has focus. Learn more in the **DMX Tester section**.
- **Adjust Columns**:
  This On/Off button makes a sheet adjust the column width to match the window size and the number of columns.

- **Address**:
  This input button opens a calculator where a DMX address can be typed. This address defines the address the sheet should scroll to.
- **Font Size**:
  This selects the font size in the window. It is a swipe button that opens a list of sizes from 10 to 32. There is also a **Default** property. The default is the same as size 18.
- **Readout**:
  This selects the value readout for DMX channels. It is a swipe button that opens a list of readout types with the following options:
  - **Percent**:
    This is a range from 0 to 100.
  - **Decimal**:
    This is a decimal range from 0 to 255.
  - **Hex**:
    This is a hexadecimal range from 00 to FF.
- **Address Mode**:
  This toggle button changes how the DMX address is displayed. It can be split up into universes and a range of DMX addresses from 1 through 512 - this option is called **Univ. add**. The other option is **Absolute**, which shows the addresses continuously starting from number 1 and upwards.
- **Level Bar**:
  This input field changes the level bar setting in the sheet. There are three options:
  - **Off**:
    The value only visualizes the level.
  - **Background**:
    The background changes from dark green to bright green as the value rises.
  - **Bar**:
    The value is visualized as a bar that moves from left to right as the value rises.
- **Auto Columns**:
  This On/Off button activates a function that automatically adjusts the number of columns in the DMX Sheet.
  Adjust Columns need to be active for this to work. **#Columns** is automatically adjusted by this function.
- **Universe Up** (only when Edit Title Bar is On):
  This is not a traditional setting. It is a button that can be visible in the title bar. Tapping the button will select the next universe.
  This setting is only visible when **Edit Title Bar** is On. Thus, using the button is only possible when it is visible in the title bar.
- **Auto Column Width**:
  This On/Off button activates a function that automatically adjusts the number of columns in the DMX Sheet.
  Adjust Columns need to be active for this to work. **#Columns** is automatically adjusted by this function.
- **Universe Down** (only when Edit Title Bar is On):
  This is not a traditional setting. It is a button that can be visible in the title bar. Tapping the button will select the previous universe.
  This setting is only visible when **Edit Title Bar** is On. Thus, using the button is only possible when it is visible in the title bar.

The mask tab contains the three On/Off toggle buttons called **Value, Attribute**, and **ID**. They are described in the text above.

DMX Tester

The DMX channels can have output from the DMX tester.

This can be done using the **DMXAddress** and **DMXUniverse** keywords.

It can also be done using the **DMX Tester** encoder bar. It can be shown by turning on the Tester Encoder Bar in the title bar of the DMX Sheet.

Encoder bar - DMX Tester

This encoder bar allows fast access to DMX addresses, a test value, and the option to patch the selected address to an existing fixture. A row of buttons gives fast access to some useful functions.

On the left side of the bottom row, there is a button that changes the address mode for the encoders. It can be Uni for **Universe** and **Address** separation on the first and second encoder or Abs for absolute DMX addressing on the second encoder.

The third encoder (Tester Output) defines the DMX value used for the test. Changing the readout for the DMX sheet also changes the readout for this encoder.

The fourth encoder can be tapped to open a **Fixture(s) to Patch** pop-up. This can be used to select an existing fixture that should be patched to the selected DMX address.

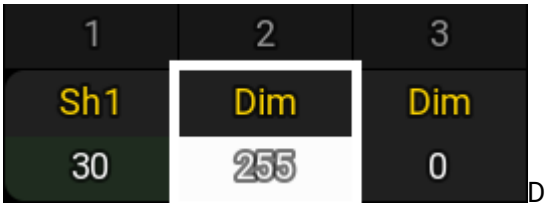
DMX addresses can also be selected by tapping on the DMX sheet.

The buttons above the encoder control have the following functions:

- **Retain**:
  Turning this On will keep the value from the tester active in output for any DMX address selected using the tester. Retain Off releases the previous addresses from the tester.
- **DMX Channels**:
  This button toggles between three different values:
  - **All**:
    Any DMX address can be selected using the encoders.
  - **Patched**:
    This allows only patched DMX addresses to be selected using the encoders.
  - **Unpatched**:
    This allows only unpatched DMX addresses to be selected using the encoders.
- **Select All**:
  Tapping this selects all the DMX addresses affected by the DMX tester.
- **Release Selected**:
  This will release the currently selected DMX address from being affected by the DMX tester.
- **Release Unselected**:
  This will release all other DMX addresses except the currently selected one from being affected by the DMX tester.
- **Release All**:
  This will release all DMX addresses from being affected by the DMX tester.

- **Park Selected**:
  This can park the currently selected DMX address at the current DMX value.
- **Unpark Selected**:
  This can be used to unpark the currently selected DMX address.

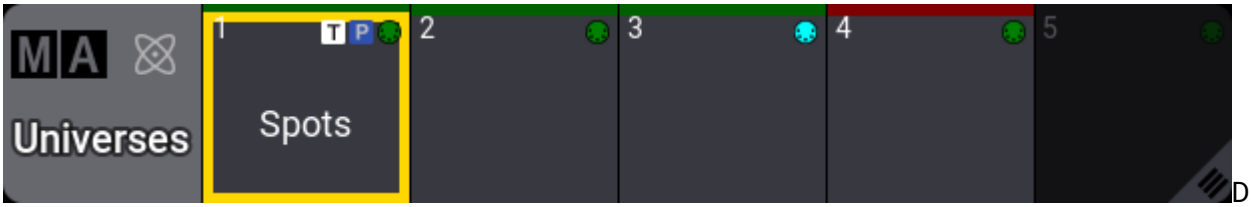
When a DMX tester affects a DMX channel, the background color for the value is white.



DMX address 2 is being tested

The **DMX Universe pool** shows when a universe has parked DMX addresses with a small P icon. Universes where the tester affects DMX channels also have a small T icon.

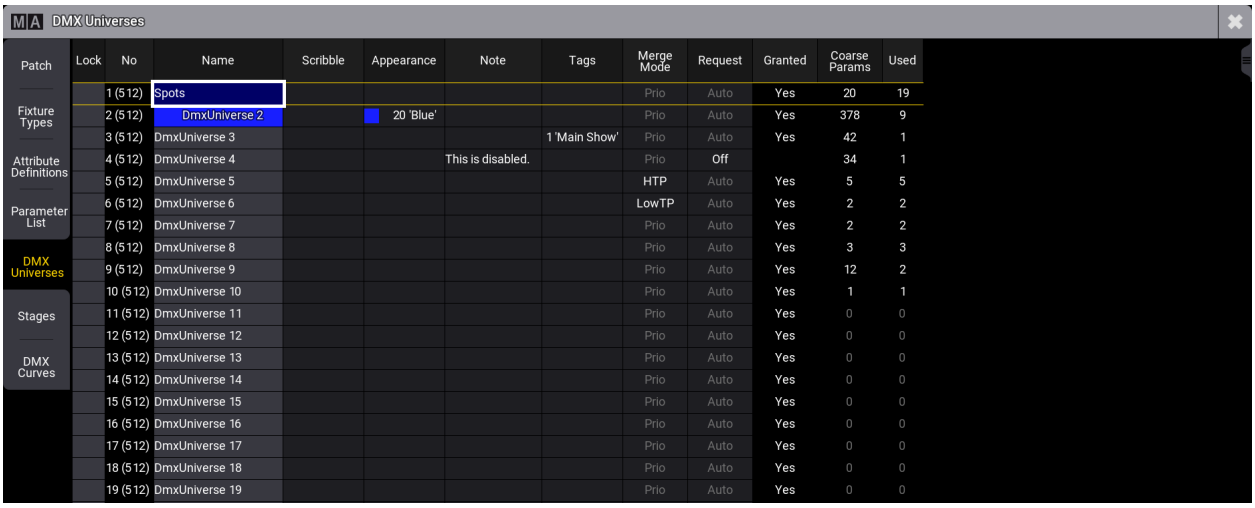
It could look like this in the pool:



DMX universe pool with parked addresses and actively testing addresses

1.14.7. DMX Universes

All the 1024 DMX universes are listed in the DMX Universes menu. This can be accessed by Menu - Patch - DMX Universes.



DMX Universes menu

The menu has several columns showing information about each universe.

- **Lock**:
  This shows **UL** if a user locks the row.
- **No**:
  This is the universe number.
- **Name**:
  This is the name of the universe. It can be a good idea to give the universes a name for a better overview.
- **Scribble**:
  A **scribble** can be assigned to the universe.
- **Appearance**:
  An **appearance** can be assigned to the universe.
- **Note**:
  A note can be added to the universe. Universes with a note have the note icon ( ) in the pool object - see below. Learn more in the **Notes topic**.
- **Tags**:
  Tags can be added to the universe. Universes with a tag have the tag icon ( ) in the pool object - see below. Learn more in the **Tags topic**.
- **Merge Mode**:
  This is the merge mode used with incoming DMX data. These are described in the **DMX Port Configuration topic**.
- **Request**:
  A universe can be requested or not. Requesting a universe means a desire to send DMX data to the universe. This request might not be granted - generally because there are not enough

unlocked parameters (read more in **Expand the Amount of Parameters topic**). The setting has three options:
    - **Auto** (default): This automatically requests the universe if something is patched.
    - **On**: This requests the universe.
    - **Off**: This does not request the universe. It can be helpful to turn off the request if there are not enough unlocked parameters, but a test is needed for a specific universe, for instance, during preprogramming.
- **Granted**:
  This shows if the output of the universe is granted. It says "Yes" if the universe is requested and granted. This field is information only.
- **Coarse Params**:
  This is the number of parameters patched to the universe.
- **Used**:
  This is the number of fixtures patched to the universe.

Universe Pool

The universe pool has a pool object for all the 1024 universes that the system can handle.

The pool is a window created like any other window. Learn how in the **Add Windows topic**.

DMX universe pool

Each pool object represents a DMX universe.

The pool object shows some information about the universe.

There is a colored bar at the top of the pool object. This is green if the universe is requested and granted. It flashes bright green when the DMX data is outputting. It is red if it is not granted.

Pool objects have a **P** (Prio), **H** (HTP), or **L** (LowTP) to indicate the **Merge Mode**. Learn more in the **DMX Port Configuration topic**.

A small blue park icon (P) indicates if the universe contains parked DMX channels.

There is a small white tester icon (T) to indicate if there are DMX channels affected by the DMX Tester.

There are also icons for Tags ( ) and Notes ( ) if the universe has any of these attached.

If a universe is assigned to a DMX port, there is a small port icon. This can have different colors depending on whether the port is set to output (green), input (orange), output and input (bright cyan), if it is output with RDM enabled (dark cyan), or if it does not output DMX (black).

There are indicators to show if the universe is output using sACN (a small "S") or Art-Net (a small "A"). These are black and flash green when the DMX output values are updated.

Indicators also show if DMX data is merged into the universe from sACN or Art-Net. This is indicated with an orange text: **sACN In** and **Art-Net In**. This is a dark orange text to show the assignment. It flashes orange when there is an incoming signal. See universes 8 and 10 in the example image above.

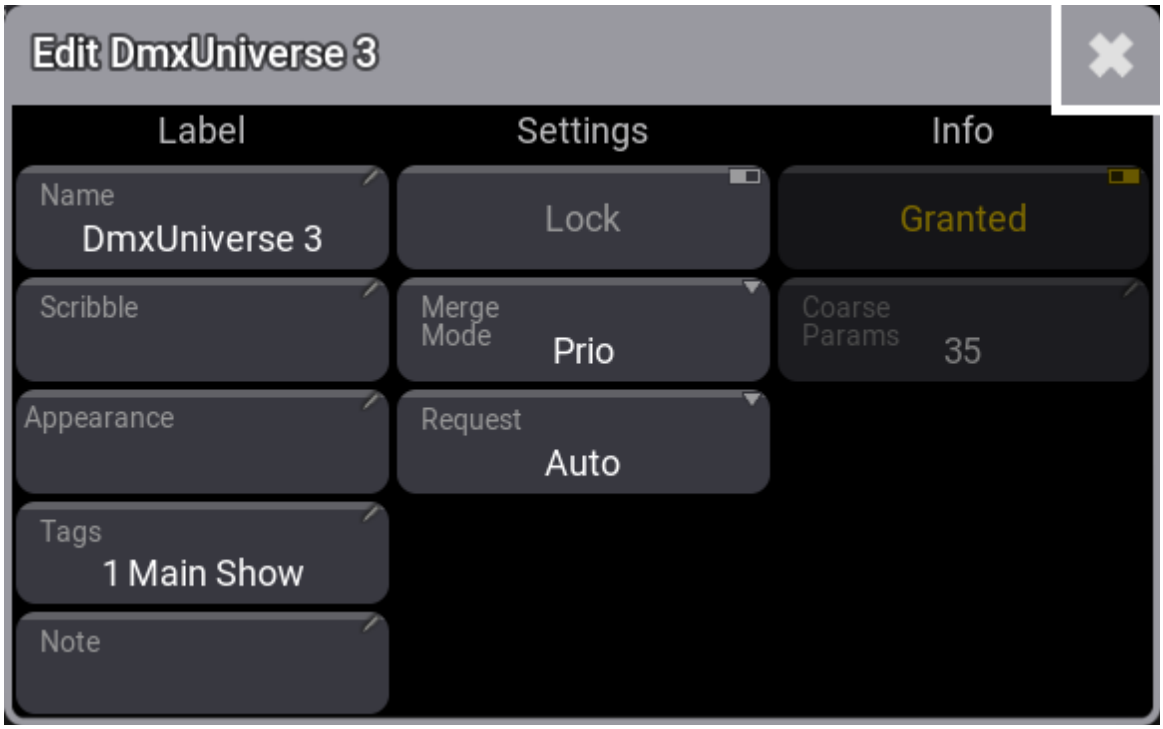
The final indicator is the PSN In. This appears on universes with MArker fixtures mapped to a PSN source. It also flashes orange when a PSN signal is received. Learn more about mapping PSN in the **PSN topic**.

Read the topics in the **DMX In and Out section** to learn how to set up the DMX ports and how to set up Art-Net and sACN.

Moving a universe in the pool also changes the universe for the fixtures patch in the universe but keeps the respective DMX start addresses.

A universe can be selected by tapping it in the pool or using the **Select keyword** together with the **DMXUniverse keyword**. The **DMX Sheet** can be set up to show only the selected universe in the sheet settings - using the universe setting.

Editing a pool object opens the Edit DMX Universe pop-up:



Edit DMX Universe pop-up

It is possible to edit the same settings as in the DMX Universe menu.

1.14.8. Remove Fixtures from the Show

Removing or deleting a fixture row in the **Patch menu** removes all programmed information from the show.

It can not be done from the Live Patch menu.

Removing the DMX address or the ID number does not remove the programmed values, but the fixture can no longer be edited, and it does not produce any output. It is also not visible in the **Fixture Sheet**.

Remove a Fixture from the Patch

Requirement: Patched fixtures.

1. Open the Patch Menu.
    a. Press Menu
    b. Tap Patch
2. Select the fixtures in the patch.
3. Tap Delete
4. Close the Patch menu and tap Ok to accept the changes.

The fixtures are deleted and all programming with those fixtures is lost. It cannot be oopsed.

1.14.9. Position Fixtures in the 3D Space

The **3D Window** shows a 3D visualization of the fixtures and objects in a 3D space.

The fixtures need to be positioned in the 3D space for this to truly be a powerful tool. The virtual fixtures should be positioned and rotated to match the real-world fixtures.

There are three primary ways to change the fixture position: **Using the patch**, **using the 3D Viewer**, or **position calibration**.

Position Fixtures Using the Patch

The best way to position the fixtures from the patch is from the **Live Patch**:

1. Press Menu.
2. Tap Live Patch.
3. Make sure the menu is in **Full** column mode.

In the patch, there are rows for each fixture, and there are columns with position and rotation values:

Pos			Rot		
X	Y	Z	X	Y	Z
0.000	0.000	0.000	0.00	0.00	0.00
0.000	0.000	0.000	0.00	0.00	0.00
0.000	0.000	0.000	0.00	0.00	0.00
0.000	0.000	0.000	0.00	0.00	0.00
0.000	0.000	0.000	0.00	0.00	0.00
0.000	0.000	0.000	0.00	0.00	0.00

Position and Rotation values in the Patch.

New fixtures are always added at the zero-point location and with zero rotation.

The zero points are 0.000 meters for all three position axes (X, Y, and Z) and 0.00° for all three rotation axes.

- X position is usually regarded as the Stage Left and Stage Right indication. A positive value is in the stage left direction.

- Y position is usually the Downstage and Upstage direction. A positive value would move the fixture more upstage.
- Z position is the height. A positive value would move the fixture above the stage.
- X rotation is rotating the fixture around the fixture's own X-axis. A positive value is rotating the top of the fixture towards downstage.
- Y rotation is rotating around the fixtures' own Y-axis. A positive value rotates the top towards stage left.
- Z rotation is rotating around the fixtures' own Z-axis. A positive value rotates the fixture counter-clockwise, as seen from the top.

The grandMA3 software currently only works with meters and degrees.

Do the following steps to edit the fixtures' position and rotation:

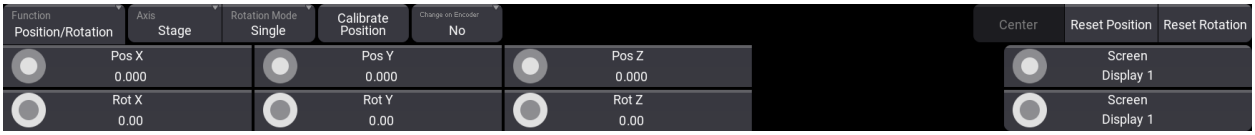
1. Locate the rows for the fixtures that need to be positioned.
2. Edit the needed fields.
3. Type the new position.

The Show 3D Positions can be activated to show the 3D space with the fixtures. The fixtures represented by the selected rows are highlighted in yellow in the 3D space.

Position Fixtures Using the 3D Viewer

The 3D Viewer can be used to position and rotate the fixtures. It needs to be in **Setup** mode, which can be activated by tapping Setup in the title bar.

This mode changes the encoder toolbar into this:



Use the encoders to position and rotate fixtures

Different buttons in this toolbar change how the fixtures are affected by the encoders. Some standard functions do not change; others change depending on selected functions.

The general workflow is this:

1. Select the fixtures that need to be moved.
2. Change the settings in the toolbar to match the wanted action.
3. Use the encoders to adjust the values.

The values on the encoders show the relevant values depending on the selected fixtures. Turning the encoders changes the values and thus the position or rotation of the selected fixtures.

These are the different standard buttons for position and rotation:

- **Function**:
  This button toggles between two different modes: **Position/Rotation** and **Arrangement**. This changes the function of the encoders to either change the position and rotation of the selected fixtures or to different arrangement types. Read about the Arrangement tool **below**.
- **Axis**:
  The axis is used when a fixture is moved or rotated. The two options here are **Stage** and **Object**. This means that the selected fixtures can be positioned and rotated using their own axis (Object) or the stage or world axis (Stage).
- **Rotation Mode:**
  When multiple fixtures are selected, the rotation mode can be changed between **Single** or **Group**. This determines if rotation is done for each fixture or if the selection of fixtures is treated as a group and thus rotated as one object.
- **Calibrate Position**:
  This opens the Position Calibration menu. Read more about it **below**.
- **Change on EncoderEvent**:
  This determines when changes are distributed through the network to other stations. When it is set to **Yes**, then the new position is sent immediately. When it is **No**, then the new information is sent two seconds after the encoders stop turning.
- **Reset Position:**
  Tap this button to reset the fixture position to 0 in all positions.
- **Reset Rotation**:
  Tap this button to reset the rotation to 0 on all axes.

Position Arrangement Tool

The position arrangement tool consists of three different arrangement or layout types. It can be used to arrange the selected objects in a **Line**, **Grid**, or **Circle**.

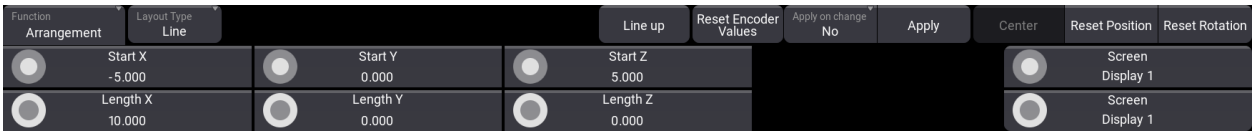
The tool is accessed by changing the Function in the encoder toolbar to **Arrangement**. Each of the three layout types has its own encoder functions and their own buttons. There are some common arrangement buttons:

- **Layout Type**:
  There are three types of arrangement: **Line**, **Grid**, and **Circle**. These different types change the available buttons in the encoder bar and the functions of the encoders.
- **Reset Encoder Values**:
  Resets the values on the encoder to the default values.
- **Apply on Change**:
  This controls whether arrangement changes are applied and distributed immediately (Yes) or only marked by a purple indicator of the would-be location (No).
- **Apply**:
  If **Apply on Change** is set to No, then this button needs to be tapped to confirm the new arrangement location.

When an arrangement is being adjusted, then there are purple fixtures in the 3D window with ID numbers to indicate where the fixtures would end up if the arrangement settings are applied.

Line

This is used to position the fixture on a single row.

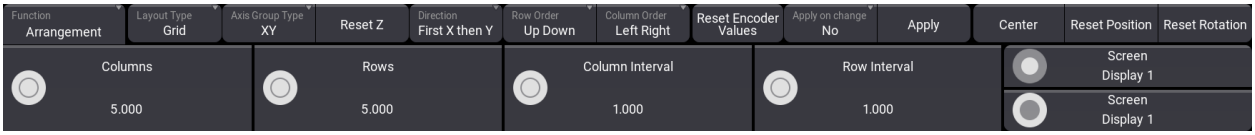


Line Arrangement Encoder Toolbar

The encoders change to set a start and a length for all three axes. There is a special button called **Line up**. Tap this to align the base of the fixture to match the line.

Grid

The grid arrangement moves the fixtures into rows and columns. It is a 2D grid.



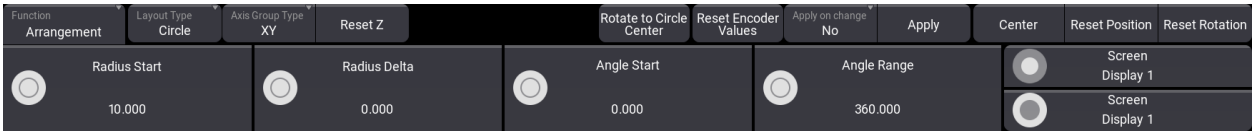
Grid Arrangement Encoder Toolbar

The encoders change to set the number of columns and rows and the interval spacing between the columns and rows. There are some special buttons:

- **Axis Group Type**:
  This sets the orientation of the grid. It shows the two axes that are used as columns and rows.
- **Reset Z** / **Reset X** / **Reset Y**:
  This button resets the position on the third axis. The one not affected by the grid.
- **Direction**:
  This changes the direction of the grid. It changes the order of the two axes.
- **Row Order**:
  Tap this button to reverse the direction of the row.
- **Column Order**:
  Tap this button to reverse the direction of the row.

Circle

The circle arrangement tool is used to position fixtures in circles and spirals.



Circle Arrangement Encoder Toolbar

The encoders change to set the Radius Start size, Radius Delta (sets if the radius changes size), Angle Start, and Angle Range. Angle Start sets where the first fixture is on the circle. Angle Range is used when creating semi-circles (value below 360) or even multiple circles (value above 360).

There are some special buttons:

- **Axis Group Type**:
  This sets the orientation of the circle. It shows the two axes that are used.

- **Reset Z** / **Reset X** / **Reset Y**:
  This button resets the position on the third axis. The one not affected by the circle.
- **Rotate to Circle Center**:
  This rotates the fixture bases to match the circle. Tapping this button several times rotates the fixtures, each time to a different rotation.

Position Fixtures Using Position Calibration

The fixture position calibration system calculates the fixture position based on the pan and tilt values needed to hit three or four known points in the real world. While using only points one, two, and three can be enough, the best result is achieved using all four points.

The four points do not need to be the same for all fixtures. Each fixture can have its own four points. The points are visible in the 3D window as a red, green, blue, and yellow octahedron when the window is in setup mode and the **Position Calibration** pop-up is open.

The pop-up is opened by tapping the Calibrate Position on the encoder bar - read **above**.

Position calibration pop-up with some selected fixtures and their values

The pop-up shows the selected fixtures as rows and four colored sections of columns.

Each color section has X, Y, and Z values for the calibration point, and the Pan and Tilt values needed to hit the point.

At the bottom, there are buttons to store and recall the pan and tilt values for each point. There is also the Solve needed to start the position calculation.

To record the pan/tilt position that matches the calibration point, tap Store Px. Call Px can be used to recall a stored position to refine it.

	Hint:
	Four calibration points are recommended. Using three calibration points deteriorates the quality of the calibration.

To calibrate fixtures, use this workflow:

1. Open the **Position Calibration** pop-up.
2. Select the fixtures to calibrate.
3. Define the real-world coordinates of the calibration points (P1 - P4) in the position calibration pop-up for each fixture.
4. Position each fixture using pan and tilt to each of the calibration points, and store these positions.
5. Tap Solve.
6. Close the pop-up.

Now the fixtures should move and rotate in the 3D window to match the real-world values.

Using the Command Line

Storing calibration points, calling calibration points, and solving the calibration can be done using the **Action keyword**.

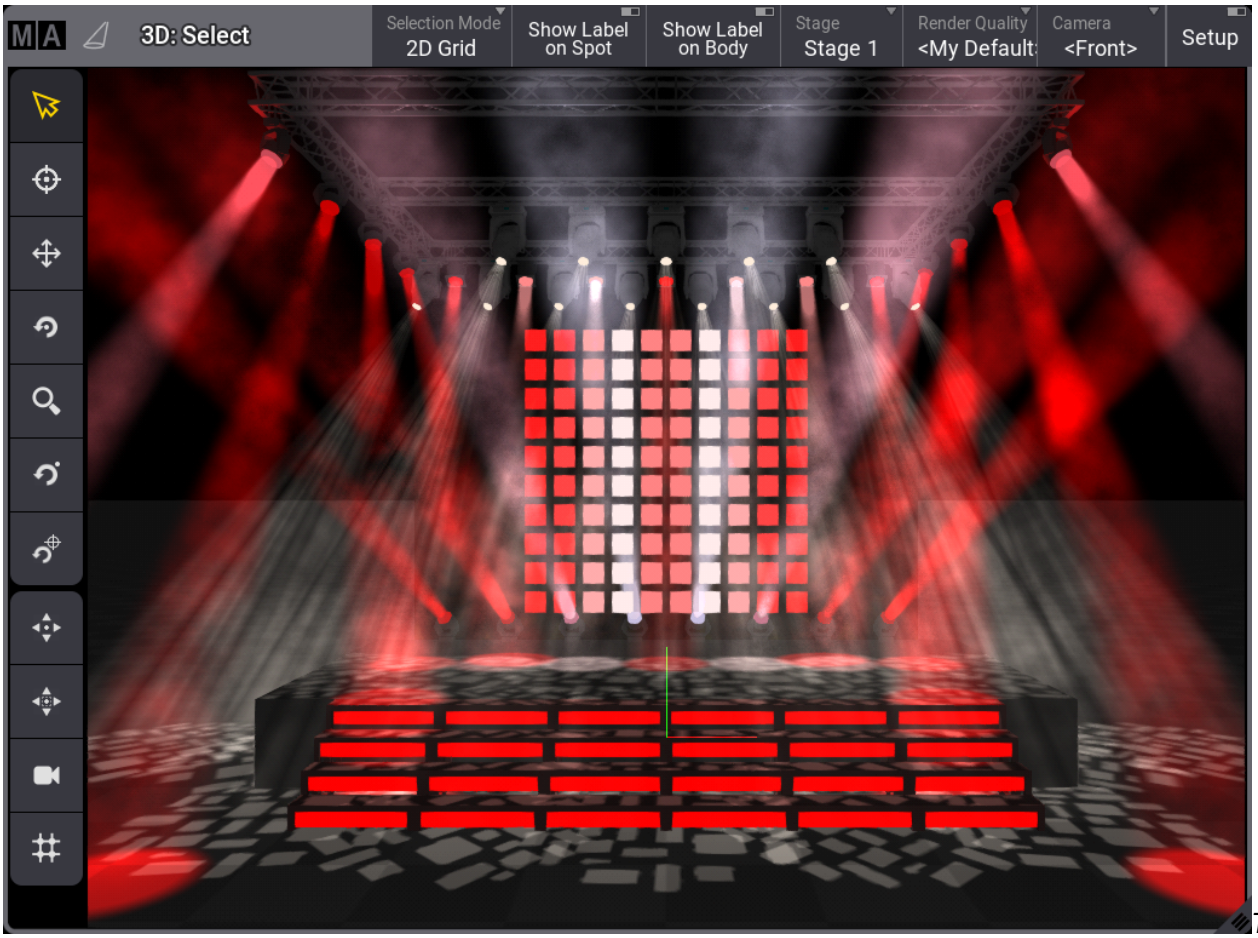
1.14.10. 3D Viewer

The 3D Viewer shows a 3D visualization of the virtual space where fixtures and 3D elements can be positioned and rotated.

The fixtures can project a light beam that moves and changes color when the values for the fixtures are changed.

Haze can be rendered to make the beams look more realistic in the air. Learn more in the **Render Quality topic**.

All the fixtures and other stage objects can be **positioned and rotated** in the **Patch**, **Live Patch**, or using this window.



The 3D Viewer with some fixtures

The **Stage** object can be visualized as a box or just a floor. The box can be looked into, but not out of. Read more **below**.

Fixtures can be interacted with in the 3D Viewer like the **Fixture Sheet**. For instance, it can be used in combinations with commands.

Virtual cameras are used to see 3D space. These define the position, direction, and other settings from where and how the 3D space is viewed. The cameras are stored in the **Camera Pool**.

	Restriction:
	When on an onPC station, the driver of the GPU has issues with OpenGL, the 3D Viewer informs the user about the driver issue instead. In this case, the 3D Viewer does not try to render 3D scenes anymore.

Title Buttons

There are several default buttons in the title bar. The available buttons in the title bar can be edited using the window settings. Learn more in the **Title Bar Configuration topic**.

Some of the default buttons are swipe buttons that open a list of the available options.

Three toggle buttons turn On or Off labels and turn On or Off the Setup mode.

The others give fast access to select Selection Mode, Stage object, Render Quality, and Camera selection.

There can be several stages in the patch. Each 3D Viewer can show one of these stages or all stages. Read more about stages in the **Stages topic**.

Read about all the settings **below**.

Left Side Tool Buttons

There can be a toolbar on the left-hand side of the window. These tools control what happens when the window is touched and change camera positions (read more about the cameras below).

The ones that change the touch mode are tapped to select the mode, and when the window is touched, the mode dictates what happens.

The selected mode is displayed in the window title bar. Most tools are grayed out and disabled if the selected camera is locked.

This is the explanation of the different tool buttons:

- **(Select):**
  Sets the touch mode to **Select**. This is used to select fixtures in the window. The fixtures can be tapped or selected using a selection lasso.
- **(Follow):**
  Sets the touch mode to **Follow**. This makes all the selected moving lights point to the touched stage area. The follow function obeys the **Align** settings.
- **(Move):**
  The mode is set to **Camera Move**. Move means changing the camera's position without changing the camera's pan and tilt values. The scroll wheel on a mouse moves the camera forward and backward in the view based on the location of the pointer (not necessarily the center).
- **(Rotate Center):**
  Sets the mode to **Camera Orbit**. This mode orbits the camera around the center of the window,

keeping it pointed toward the center. The scroll wheel on a mouse moves the camera closer to or away from the location of the pointer (not necessarily the center).

- **(Zoom):**
  Changes the mode to **Camera Zoom**. This mode moves the camera in and out of the 3D Viewer. This can also be done in the other modes using a scroll wheel on a mouse.
- **(Rotate Pivot):**
  Changes the mode to **Camera Pivot**. This pivots the camera around a pivot point that does not have to be the center. The pivot point can be set (read about the next button) and is remembered until a new point is set. The scroll wheel functions just like the rotate center mode.
- **(Set Pivot):**
  This button is used to set a new pivot point in the window. The rotate pivot mode is selected as soon as the point has been set (by clicking or touching the window).
- **(Focus)** - only available when **Setup** is On:
  This tool can focus static fixtures (fixtures without position attributes). Select the desired fixture, tap this tool, and tap the location in the 3D Viewer where the fixture should be pointed. This tool affects the rotation values of the fixture.
- **(Fit):**
  Tapping this button moves the camera to fit the entire stage area into the view.
- **(Fit Selected):**
  Tapping this button moves the camera to fit the selected fixtures into the view without orbiting the camera.
- **(Camera):**
  This button reloads the selected camera to the settings set in the **camera pool object**.
- **(3D view to Selection Grid)**:
  Tapping this tool creates grid values for the **Selection Grid**, using the current view of the fixtures.

Moving the Camera

The cameras are pool objects in the **Camera Pool**. A camera can be moved to different locations and pointed in different directions.

This information can be edited in the pool, but the position and direction are easier to change in the 3D Viewer - read about the left-hand side tool buttons above.

A scroll wheel on a mouse moves the camera closer to or away from the location of the pointer.

Editing the camera in the pool allows changing the camera mode and type. Read more in the camera pool topic (link above).

Moving the Fixtures

The fixtures' position and rotation can be set in the **patch** or **live patch**. But they can also be positioned live in the 3D Viewer.

Read more about this in the **Position Fixtures in the 3D Space topic**.

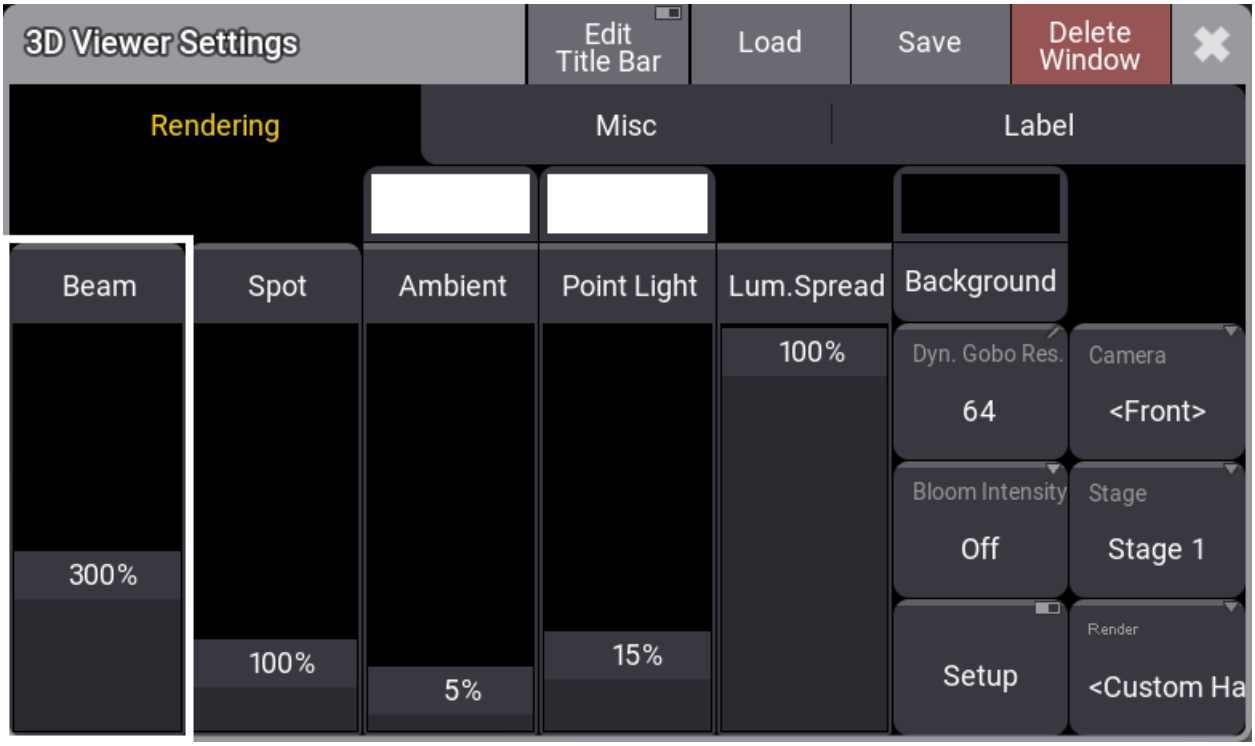
3D Viewer Settings

The settings can be opened by tapping the MA logo in the upper left corner of the 3D Viewer.

This opens a settings pop-up. This pop-up has tabs called **Rendering**, **Misc**, and **Label**.

Rendering Settings

The rendering settings are about light levels, colors, and rendering quality.



Rendering tab in 3D View Settings

There are four on-screen faders:

- **Beam:**
  This is the visibility of the light beam from all fixtures.
- **Spot**:
  This is the general intensity of the visualization of the light beam reflection where it hits a surface.
- **Ambient**:
  This is the ambient light level inside and outside the stage box. It is a very diffuse light that removes some of the contrast in the 3D Viewer.
- **Point Light**:
  This is a light source from the direction of the camera. It is used to light up the elements in the 3D space.
- **Lum. Spread** (Luminous Spread):
  This controls how visible the difference in brightness between different fixture types is. The fixture type with the highest luminous intensity defines this intensity as 100% brightness in the 3D Viewer window. When fixture types are used that have a much lower Luminous Intensity, they will be rendered much darker at their 100% dimmer output.

To allow adjustment to this intensity difference (and therefore deviate from reality) to better see the darker fixture, the Luminous Spread setting can be used with different values:
    - When this setting is set to 100%, the Luminous Intensities are rendered as before.
    - Setting the Luminous Spread to 0% renders all fixtures beams with the same maximum brightness.
    - The default of Luminous Spread in the 3D Viewer window settings is set to 80%.

The ambient and point light can be colored, similar to putting a gel in front of the light. The area above the faders can be tapped to open a color selector pop-up.

The **Background** can also be colored. The ambient light must be turned up for this color to be visible.

There are some buttons:

- **Dyn. Gobo Res.** (Dynamic Gobo Resolution):
  This is the resolution of dynamically created gobos.
- **Bloom Intensity**:
  The blooming effect can be turned On or Off. Tapping this button toggles between the two.
- **Setup**:
  Turning the setup On makes it possible to move the fixtures and 3D objects using the 3D Viewer and encoder bar. Read more about this in the **Position Fixtures in the 3D Space topic**.
- **Camera**:
  This is used to select one of the cameras from the camera pool. If it is set to follow the selected camera, the name is inside angled brackets. Read more about cameras in the **Camera Pool topic**.
- **Stage**:
  This selects what stage the 3D Viewer shows. A single stage can be selected, or all stages can be shown. Read more about stages in the **Stages topic**.
- **Render**:
  This is used to select one of the render qualities. Read more in the **Render Quality topic**.

Some of these are swipe buttons, so remember that the options can be reached easily by swiping out of the button.

Misc Settings

The second tab is called **Misc**.

Misc tab in the 3D View Settings

The Misc. tab has the following settings:

- **Show fps:**
  Turning this On displays frames per second information in the upper right corner of the window.
- **Wireframe:**
  Turning this On shows the 3D Viewer as a wireframe instead of a shaded view.
- **Priority**:
  Turning this On gives the 3D Viewer a high priority. It is only recommended to turn this On when used on a computer with a high-quality graphics card. It takes away resources from a console interface, making it react slower to user input.
- **Mark Faulty Meshes**:
  This marks meshes that are unloaded or faulty.
- **Show Selection**:
  Selected fixtures are marked with a yellow body color for the primary fixtures and a light red color for multipatch fixtures when this is On. Learn more about multipatch in the **Add Multipatch Fixtures topic**.
- **Lens Shading**:
  This defines whether the selection is shown on the lens of a geometry type “Beam”. The shaded selection is drawn when it is On.
- **Touch Mode**:
  Touching and swiping in the 3D Viewer can interact with the window differently. This setting defines how. Tapping it toggles through the following options: **Select, Follow, Focus**(Only valid when Setup is On)**, Camera Orbit**, **Camera Zoom**, **Camera Pivot, Camera Move**, and **Camera Set Pivot**.
  The Camera options move the camera - Read more **above**. The **Select** option is used to select fixtures in the view.

- **Selection Mode:**
  This defines how the fixtures are selected and positioned in the **Selection Grid**. The selection mode has two different options:
    - **2D Grid**:
      The 2D grid selection offers two different modes, **Planar** and **Perspective**. Using the lasso selection to select fixtures in the 3D Viewer adds them as a two-dimensional selection to the selection grid. In planar mode, the camera position does not affect the selection order. Only the real position of the fixtures is significant. In perspective mode, the camera's orientation also influences the selection's order in the selection grid.

      To select fixtures in the 3D Viewer using the planar selection, draw a lasso starting with a horizontal or vertical mouse movement. The color of the lasso changes to green when the selection is locked to planar mode. To select fixtures in perspective mode, draw a lasso starting with a diagonal mouse movement. The color of the lasso changes to cyan when the selection is locked to perspective mode.

    - **Linearize**:
      The selection is linearized to only the X-axis of the selection grid depending on which direction the selection lasso was created (top/bottom - left/right). This selection mode is indicated with a yellow lasso.

	**Hint:**
	The projection distortion of the camera of the 3D Viewer may affect the position of the fixtures in the selection grid if the perspective selection is used. To prevent this, use a 2D camera in the 3D Viewer.

- **Mark Unpatched**:
  This marks the body of unpatched fixtures with a dark red color when On.
- **Prism Lines**:
  This shows a line for each prism facet when the **Beam Quality** is set to **Line**.
- **Point of Origin**:
  This shows a point of origin axis marker. It is typically at the zero point of the stage.
- **Show Pigtails**:
  This shows the "Pigtail" element if the GDTF model defines it.
- **Arrangement**:
  This is a collection of settings for visualizing the purple arrangement "ghosts" markers. They are seen when using the arrangement tool to position fixtures.
    - **Mark Type**:
      There are two options: **Dynamic** or **Small**. Dynamic shows a box matching the size of the fixture. Small is just a small square marker.
    - **Depth**:
      Set if the depth of the 3D space is considered when showing the marks. This could lead to marks being hidden behind objects. If the depth is switched off, the marks are always drawn in front of other objects.
    - **Alpha**:
      Define the transparency of the marker objects.
- **Render Environmental**:
  This toggles if the environmental elements are rendered or hidden.
- **Draw Target Spaces**:
  This setting show or hide a square wireframe for MArker fixtures' target spaces.
- **Show Title Bar**:
  This setting shows or hides the title bar for the window.

- **Show Tool Bar**:
  This shows or hides the toolbar on the left-hand side. The toolbar is always visible when **Setup** is On.

Label Settings

The tab is called Label. It is settings about the labels that can be turned On or Off for the spots and the fixture bodies.

Label tab in the 3D View Settings

- **Show Label on Body**:
  This On / Off button enables labels to be drawn on the fixture's body.
- **Show Label on Spot**:
  This On / Off button enables labels to be drawn in the center of the fixture's spot. A fixture can also be selected using the rubber band selection on a label.
- **Select by Label**:
  This setting is possible when **Show Label on Spot** is active. It makes it possible to select fixtures by selecting the spot labels in the 3D Viewer when this setting is On.
- **Add Fixture ID** / **Add CID** / **Add Patch** / **Add Name**:
  These On/ Off buttons define the information displayed on the label.
- **Background Alpha**:
  This on-screen fader sets the transparency of the label's background.
- **Text Alpha**:
  This on-screen fader sets the transparency of the text displayed on the label.
- **Font Size**:
  This button sets the font size of the text displayed on the label. Tapping this toggles through the available Font Sizes. It can be swiped to open a small Select Label Font Size pop-up with all the size options.
- **Max Label Count**:
  This input field sets the maximum count of labels that are displayed at the same time. If the

number of labels exceeds the maximum label count, the labels closest to the camera will be displayed.

- **Selection Only**:
  This On / Off button defines whether labels are displayed for all fixtures or only for selected or partly selected fixtures.
- **Selection Priority**:
  This On / Off button defines whether labels of selected fixtures are displayed on top of not selected fixtures.
- **Spot Subfixture ID**:
  This On / Off button defines if the label of the spots shows the Fixture ID of the corresponding sub fixture (On). Otherwise, all spots of a fixture show the Fixture ID of the main fixture (Off).
- **Reset Properties**:
  Tap this button to reset the label properties to their factory defaults.

Setting Up a 3D Computer

The grandMA3 consoles can show a 3D Viewer with the fixtures and the stage setup. However, the consoles are optimized to be the human interface used to program the light.

If a high-quality and high-framerate 3D render machine is needed, the best solution is to have a high-performing computer with good graphics cards. This computer must have the grandMA3 onPC installed and optimized for 3D graphics.

The grandMA3 onPC needs to be in a session with the console. The grandMA3 onPC could be logged in as the default 3D user who uses a different **Screen Configuration** but has the same user profile as the default admin user. The grandMA3 onPC should be logged in with a user with the same profile as the user looking at the 3D computer. This ensures that the 3D computer follows the user into preview mode.

Screen Configurations contain information about which windows are visible in the different User-Defined Areas and the size of the user-defined areas. The screen configuration also contains information about what is assigned to the View Buttons. Learn more in the **Screen Configuration** topic.

The 3D computer could have one big 3D Viewer with all the render settings set to the best quality, and the 3D Viewer setting called **Priority** should be On (see settings above).

The window can also be set to hide the title bar to maximize 3D viewing space using the **Window Settings**. Other elements, like the view buttons, encoder bar, etc., can also be hidden using the **Configure Display pop-up**.

1.14.11. Render Quality

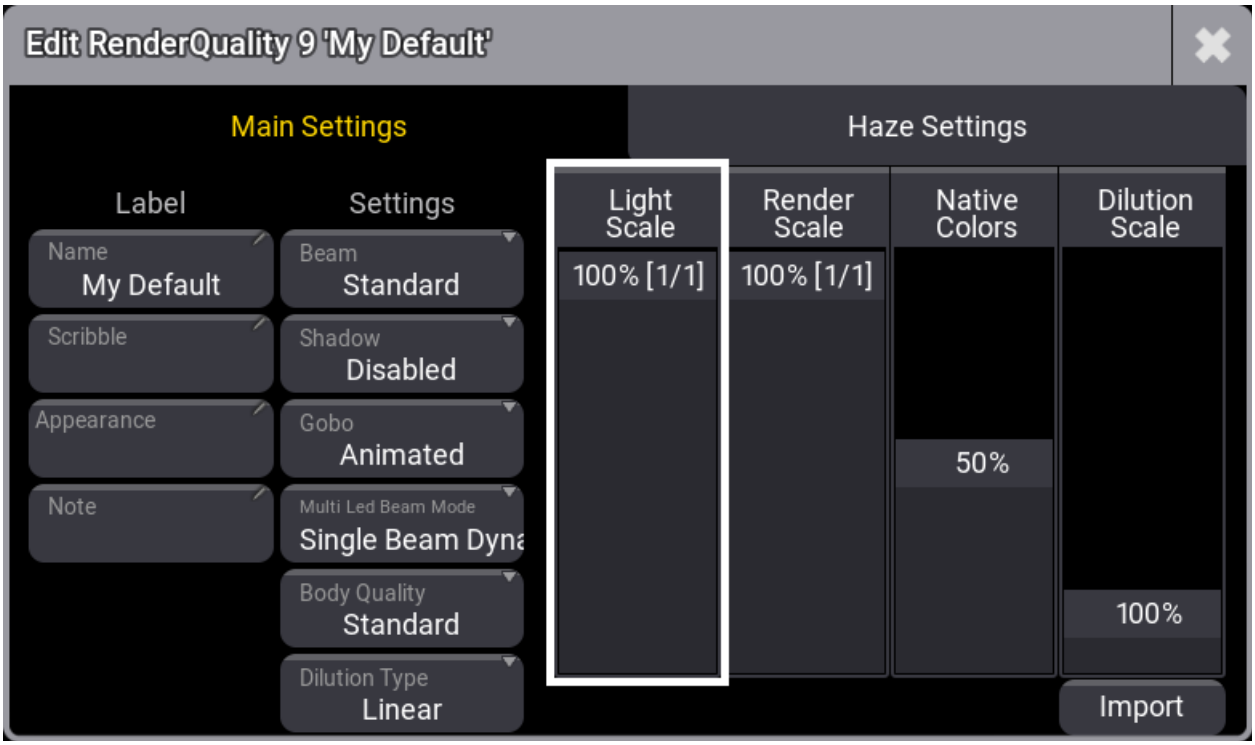
The render quality defines how the light and fixture bodies are rendered in the **3D Viewer**.

All the render settings are stored in a Render Quality pool object and can be easily selected in the pool.

Render Qualities pool

There are some predefined render qualities in the pool. There is always a selected object. The 3D Viewer can use any render qualities or be linked to the selected pool object.

New pool objects can be made and edited to change the render settings.



Render Quality Editor - Main Settings

The editor has two different tabs. The one called "Main Settings" deals with the general settings defining how detailed the rendering should be.

The main settings contain the following:

- **Name**:
  This is the name of the pool object. Tapping it allows changing the name.
- **Scribble**:
  This is the assigned scribble, and tapping it allows selecting a scribble or creating a new one.
- **Appearance**:
  This is the assigned appearance. Tapping allows for selecting an appearance or creating a new one.
- **Note**:
  A note can be added to a render quality.
- **Beam**:
  This toggles through a list of different simulation qualities that can be selected for the light beam. It goes from low to high quality. The higher quality uses more computer resources.
  - **No Beam**:
    This does not render any beams. This mode does not show any gobos or spot rendering.
  - **Line**:
    This renders a simple line as the beam. This mode does not show any gobos or spot rendering.
  - **Standard**:
    This is a simple light beam from the fixture.
  - **High**:
    This mode adds more reality to the light beam.
  - **High Fancy**:
    This is a more precise calculation of light dissipation.
- **Shadow**:
  Turns On or Off if the beam shall throw shadows when hitting another object. Therefore, the 3D object needs the setting Cast Shadow enabled within the patch, and the **fixture's Shadow Quality** must be set to a value above **None**.

  Each fixture has a Shadow Quality setting in the Patch. Learn more in the **Add Fixture to the Show topic**.

- **Gobo**:
  This setting defines whether gobos shall be rendered within the beam and spot. It has three different options:
  - **Disabled**:
    Gobos are not rendered.
  - **Enabled**:
    Gobos are rendered.
  - **Animated**:
    This means that in addition to the gobo, gobo animations like gobo shake or gobo wheel spin are rendered. When gobo is only set to enabled, these animations will not be rendered.
- **Multi LED Beam Mode**:
  This defines how multi-emitter fixtures render the light beam. The options are:

    - **Separated Beams**:
      This renders a beam for each emitter.
    - **Single Beam Mean Color**:
      This renders a single beam using the mean color.
    - **Single Beam Dynamic Gobo**:
      This takes the output of the emitters and creates a "virtual dynamic gobo". This is then rendered as a single beam.
- **Body Quality**:
  This defines the quality of the fixture body rendering. The options are
    - **None**:
      No meshes of fixtures are visualized. Nevertheless, environmental objects are displayed.
    - **Box**:
      A whole fixture's body is visualized as one box. This box has the dimensions of the fixture.
    - **Low**:
      Every geometry of a fixture is visualized as a box. This is not changed if one of the fixture's meshes is a default cylinder.
    - **Simple**:
      Fixtures with a common geometrical structure of a moving head (Base - Yoke - Head - Lens with or without Pigtail) are visualized with grandMA3 default meshes. All other fixtures are visualized in "Low" mode.
    - **Standard**:
      Fixtures with a vertex count above 1 200 are visualized in "Simple" quality mode. The meshes of fixtures with a vertex count lower than 1 200 are not changed.
    - **High**:
      Fixtures with a vertex count above 10 000 are visualized in "Standard" quality mode. The meshes of fixtures with a vertex count lower than 10 000 are not changed.
    - **Ultra**:
      The fixtures are visualized with their original meshes, no matter the vertex count.
- **Dilution Type**:
  This defines the dilution intensity of the beam and the spot relative to the distance. These are the options:
    - **None**:
      The beam and spot keep their brightness over the distance.
    - **Linear**:
      Corresponds to a linear attenuation of the intensity over the distance.
    - **Correct**:
      Corresponds to a more realistic attenuation.
- **Light Scale**:
  This changes the resolution of the light rendering in the 3D Viewer. 100% means that the 3D Viewer is rendered with its native resolution. 10% means that the resolution of the light in the 3D Viewer is divided by 5 in width and height.
- **Render Scale**:
  This changes the resolution of the whole rendering in the 3D Viewer. 100% means that the 3D Viewer is rendered with its native resolution. 10% means that the resolution of the 3D Viewer is divided by 5 in width and height.
- **Native Colors**:
  This fader interpolates between the native representation of colors and the intensities of the beams and spots. An automatic method that changes the colors and intensities so that the

rendered beams are easier to separate. A higher value for native colors can make distinguishing between individual overlapping beams impossible but provides stronger colors and higher contrasts.

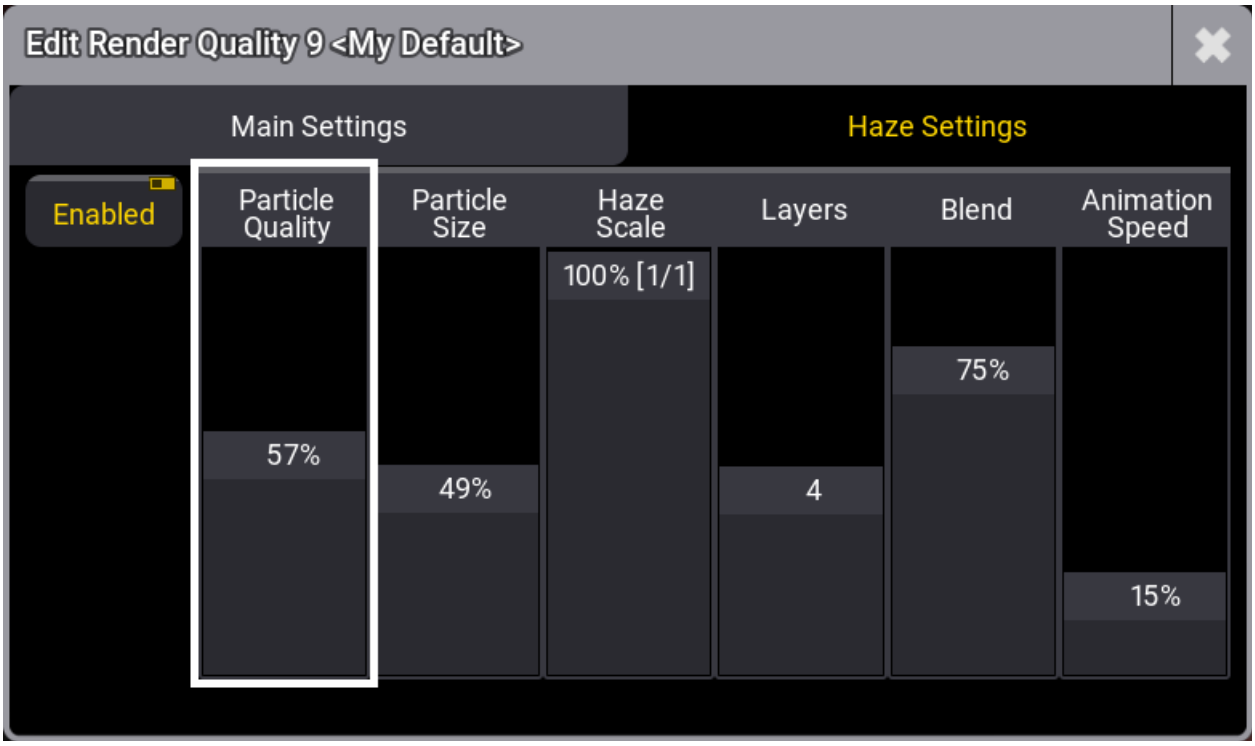
- **Dilution Scale**:
  This defines the beam intensity dilution of the beam dilution. The higher the value, the less the beam is attenuated, resulting in a beam that remains visible over further distance.

Render Scale and Light Scale can be used in combination. Setting both to 1/2 means that the whole 3D Viewer is rendered with half of its resolution, while the light is rendered with a quarter of its resolution.

The Import button allows for importing the settings of the **exported** render quality objects.

Pool objects can be exported using the **Show Creator menu** or the **Export Keyword**.

The other tab looks like this:



Render Quality Editor - Haze Settings

These settings are about an animated haze that can be visualized in the 3D Viewer. This requires some graphic powers that exceed the intended use of the graphics cards in the consoles. It is recommended that this feature be used on a grandMA3 onPC with a powerful graphics card.

Using this feature makes the 3D Viewer look more realistic.

The 3D Viewer must activate the **Priority setting** to visualize the haze animation.

The priority mode is used to prioritize the 3D rendering. It takes away resources from displaying other UI elements. Learn more about it in the **3D Viewer topic**.

The haze exists inside a volume space. The space is defined by the location of the patch's non-environmental devices positioned in the stage space.

If this volume space is too big, the haze rendering is disabled. If this happens, a warning will appear in the upper left corner of the 3D Viewer: "Haze has been automatically disabled as the maximum volume defined by the non-environmental fixtures is in [size of volume]% overload.". The haze rendering is enabled again when the volume space is made smaller. It is an internal setting that defines whether or not the haze is rendered. This does not change the "Enable" setting mentioned below.

Another restriction to the haze rendering is the show data limit on the show file. As soon as 90% of the 10GB data limit is reached, the haze rendering will be disabled, and a corresponding warning text will be displayed in the 3D Viewer.

These are the haze settings:

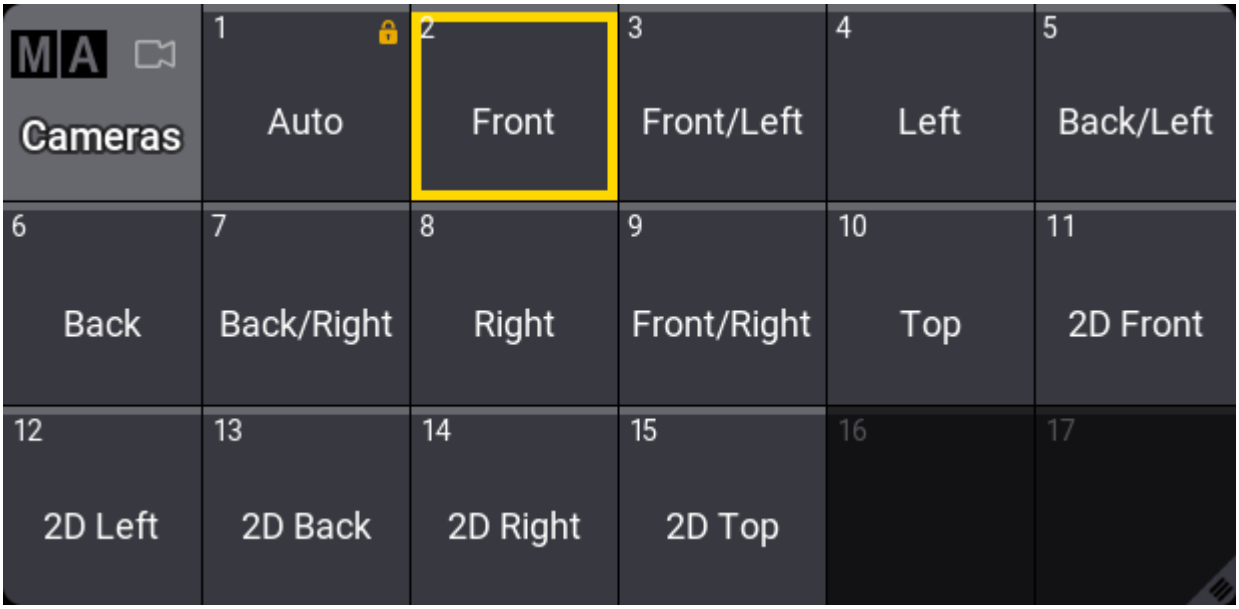
- **Enable**:
  This toggles if the haze rendering is activated. Limitations might internally disable the haze rendering, so this setting defines if the rendering is desired.
- **Particle Quality**:
  This defines how accurately the clouds of haze are rendered. The higher the value, the better the quality.
- **Particle Size**:
  This defines the size of the haze particles. A low value makes the haze more like clouds. A higher value makes the haze more dense.
- **Haze Scale**:
  This changes the resolution of the haze texture. The higher the number, the more detailed the texture, and then more resources are needed to calculate the rendering.
- **Layers**:
  This defines how many layers of haze are used in depth. The higher the value, the more spatial the haze looks.
- **Blend**:
  This defines how well the haze is blended in the air. The lower the values, the more it blends into the air and seems more even. A higher value makes it appear more realistic.
- **Animation Speed**:
  The speed at which haze moves around. The haze will not be animated if the 3D viewer does not have Priority enabled.

1.14.12. Camera Pool

The cameras are used in the **3D viewer** to see the fixtures in the 3D virtual stage.

Multiple cameras can be arranged to see fixtures and 3D objects from different angles and with different camera settings.

The cameras are all in the **Camera Pool**.



Camera pool with some cameras and a selected camera

There are some default cameras in the pool in a new show. Most of these can be edited, but the "Auto" camera is locked.

The "Auto" camera is a special camera that zooms and pans to put the selected fixtures in the center of the 3D window. When no fixtures are selected, it shows the entire stage.

A camera can be selected by tapping it in the pool or using the command line. For example, selecting camera number 3 can be done with this command:

The selected camera has a yellow frame.

Editing an empty pool object creates a new camera that copies the currently selected camera.

Editing an existing camera object opens the **Edit Camera** pop-up:

Edit Camera pop-up for the front camera.

This pop-up contains all the values needed for the camera.

Changing the values here changes the camera.

- **Name:**
  This input box can be used to change the name of the camera.
- **Scribble**:
  A scribble can be assigned to the camera pool object.
- **Appearance**:
  An appearance can be assigned to the camera pool object.
- **Note**:
  A note can be added to the camera.
- **Camera Mode:**
  The cameras can have a 3D perspective or a flat 2D projection. Editing this field allows selecting one of the predefined 2D camera angles. Selecting a 2D camera disables some of the other values in this pop-up.
  The 3D camera mode gives access to all values in this pop-up.
- **FOV (Field Of View):**
  The Field of view or Field of vision describes how wide or narrow the camera looks at the 3D stage.
- **Roll**:
  The roll value can be used to roll the camera. The value specifies the degree to which the camera will be rolled. A positive value rolls the camera clockwise. A negative value rolls the camera counterclockwise. The default value is "0", which makes the camera horizontal.
- **Zoom**:
  The zoom value is very useful with 2D cameras. The zoom value goes from 0.01 to 1. It goes from narrow to wide.
- **Pos X:**
  This is the camera position on the X-axis.

- **Pos Y:**
  The camera's position on the Y-axis.
- **Pos Z:**
  The camera's position on the Z-axis.
- **Pan:**
  This value pans the camera. Positive values turn the camera clockwise.
- **Tilt:**
  This tilts the camera up and down. A positive value points the camera down.

Moving the camera and changing the pan and tilt values directly in the 3D window might be easier. There are nice tools for manipulating the camera. Read more about this in the **3D window** topic.

The values are applied to the camera when Reload Camera is tapped.

Close the pop-up by tapping the  in the upper right corner of the pop-up.

1.14.13. Stages in grandMA3

In grandMA3 there are virtual stages. The fixtures **added to the patch** are placed in a stage. The fixtures can be **positioned** in this 3D virtual stage.

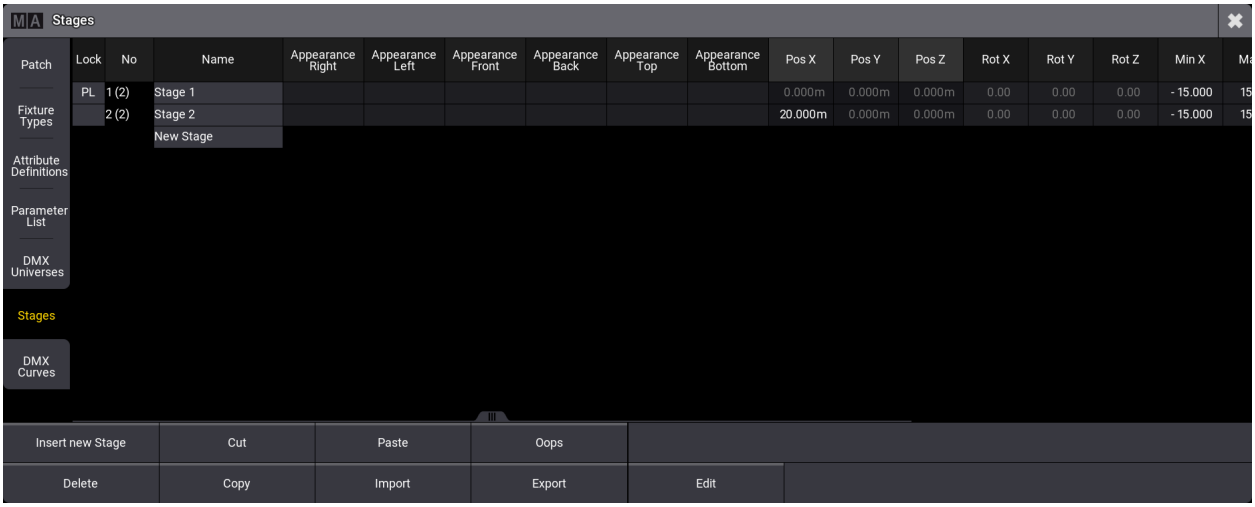
It is possible to create more stages. This could be useful in a house with several physical stages, a television station with several studios, a theme park with several different areas, or for adding a festival rig to a touring show.

The stages have location and dimension information.

Add a Stage in the Patch

1. Press Menu.
2. Tap Patch in the menu pop-up.
3. Tap Stages on the left side.
4. Tap Insert new Stage at the bottom.

A new stage has now been added above the line that had focus.



Stages menu with two stages

The stages can be named for organizational purposes.

The standard size for a new stage is 30 meters wide (X) and deep (Y), with a zero in the middle. The height (Z) is from zero to 15 meters high.

The stage displayed in the 3D window will automatically expand the visible box but will not change the dimension defined for the stage object.

The stage setup allows for changing the positions and rotation of the stages. A 3D Viewer can show one or all of them.

The buttons at the bottom make it possible to insert new stages.

The stages can also be deleted. Be careful with this!

	Important:
	Fixtures are patched inside stages. Deleting a stage also deletes all the fixtures in that stage, including all the information programmed about those fixtures.

If the stages need to be reorganized, they can be copied, cut, and pasted to move them in the list.

A stage can also be exported and imported. Remember that the stage contains the fixtures. The exported stage object contains the patched fixtures and any other stage objects inside the stage.

For more information about adding fixtures to the stage, please follow the first link at the top.

Edit a Stage

A selected stage in the stage menu can be edited by tapping Edit at the bottom. This opens the **Edit Stage** menu.

Edit stage menu for stage 1

The top part of this edit has input fields for the same settings as the **Stage** menu.

The lower part is used to add and edit a stage **Space** object and stage **Elements**. The elements are flat surfaces that can be combined to create boxes or areas that the light can hit.

A stage space defines a volume in the stage. This space is relevant when using XYZ programming. It defines the size of the space the fixtures can point to. Read more about this in the **XYZ section**.

A stage space can have multiple stage elements inside. These elements are defined by four corners, each with an X, Y, and Z coordinate.

Stage spaces and elements can be created, deleted, copied, cut, pasted, oopsed, imported, and exported like many other objects.

Export and Import are especially nice if a lot of time has been spent on creating a 3D set.

	Hint:

	More complex 3D objects can be added to the Patch. They are found in the "Set" manufacturer.

A new stage always has a Stage Space and a default Stage Element (the "floor").

Adding MArker fixtures automatically adds stage spaces for Target and, if needed, for the Movement. Read more about this in the **MArker Fixture topic**.

1.14.14. Classes and Layers

Fixtures can be organized and grouped in both **Class** and **Layer**.

It is inherited from the MVR (My Virtual Rig) import and from the CAD drawing programs where fixtures can be organized using classes and layers.

In grandMA3 there is no real difference between a class or a layer.

Each fixture can have one of each assigned.

The class and/or layer can be used when applying **filters in the patch**.

Classes and layers can be created manually.

Create a Class or Layer

The following description is for a layer, but the process is the same for classes. Just exchange all the places it says "Layer" or "FixtureLayer" with "Class".

1. Press Menu.
2. Tap Patch in the menu pop-up -> this opens the patch menu.
3. Activate Full columns in the title bar.
4. Activate Split View in the title bar.
5. Tap Layers.
6. Tap the location in the list where the new layer should be or tap New FixtureLayer at the bottom of the list.
7. Tap Insert new FixtureLayer.

A new class or layer is now added.

The name can be changed.

Classes and layers can also be created by editing the class or layer field for the fixture. See the description below, but instead of selecting an existing class or layer, tap New. This creates a new class or layer, and the name pop-up appears to give the new class or layer a name.

Assign a Class or Layer to a Fixture

Classes and layers can be assigned to fixtures in the **patch menu** when it is in **Full** column mode or when the **Split View** is active in layers or classes.

When the column mode is full, they also appear as options in the Fixture Wizard (read about it following the link above).

Editing the class or layer cells opens small select pop-ups where the existing classes or layers can be selected. There is also the option to select **None**. This can be used to remove a class or layer from a fixture.

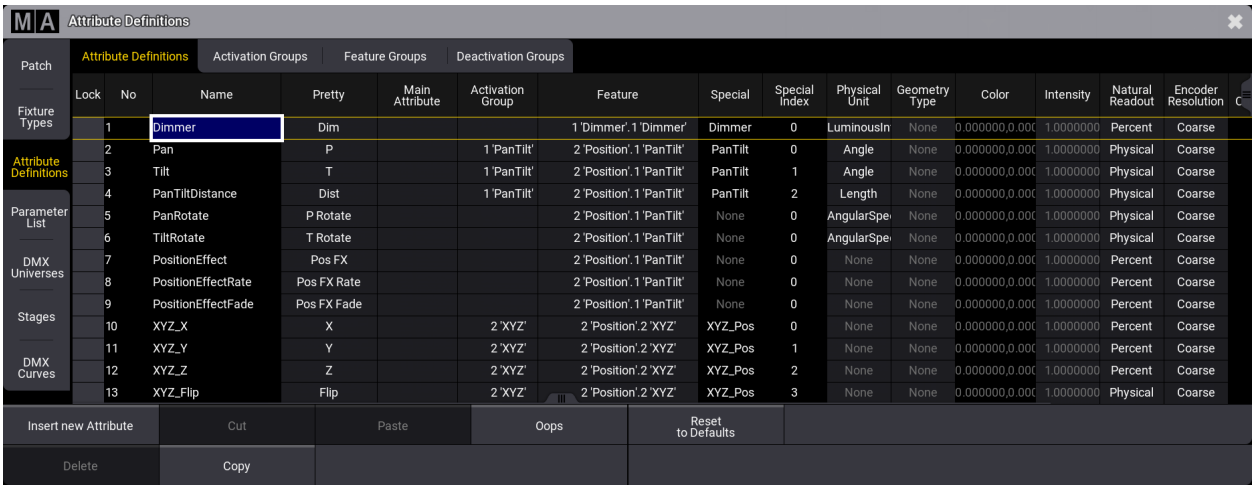
1.14.15. Attribute Definitions

Attributes are the building blocks of **fixture types**. The same building blocks are used throughout the console and they are what is controlled using the **Encoder bar** when **operating fixtures**.

Attributes definitions describe the relation between Main Attributes and sub-attributes. Attributes are organized in **Activation Groups**, **Feature Groups**, and **Deactivation Groups**.

There is a set of 366 predefined attributes with a complete setup and relations. They can be edited but it is also possible to create custom attributes.

It is all done from the sub menus in the patch menu.



Attribute Definitions

The attribute definitions have the following columns:

- **Lock**:
  This column can be used to lock the rows and protect them from editing.
- **No**:
  This is an auto-generated row number.
- **Name**:
  This name needs to be a unique identifier. When it is typed, it is locked and cannot be edited.
- **Pretty**:
  This is the name displayed above the encoders. This input accepts local country characters if typed using grandMA3 onPC.
- **Main Attribute**:
  This is used to define a hierarchy between the attributes. Editing a cell in this column opens a small **Select Main Attribute** pop-up. Selecting the main attribute makes this attribute a subattribute of the main attribute.
- **Activation Group**:
  Activation groups are used to activate a group of attributes as soon as one of the members in the group is activated. Read more in the **Activation Group topic**.

- **Feature**:
  Features are part of the hierarchy system. Attributes need to be a part of a feature. Read more in the **Feature Group topic**.
- **Special**:
  This is information only. It shows how the attribute is visualized in the **3D window**.
- **Special Index**:
  This is information about the index number for the special value.
- **Physical Unit**:
  The physical units describe the physical properties of the attribute. Read more **below**.
- **Geometry Type**:
  This can be used as a standard geometry type when using the attribute in new Fixture Types. Read more about geometry types in the **Fixture Types topic**.
- **Color**:
  Any attribute can have a color but it is only relevant for the attributes that use "ColorComponent" as the physical unit. Here the color information defines the color of the LED emitter.
- **Intensity**:
  The intensity value can be used to define the relationship between the Color Components. It is used by the color engine to compensate if, for instance, the blue emitters are much brighter than the green. This intensity information can also be defined in the fixture types. The information in the attribute definition is used if nothing is defined in the fixture type.
- **Natural Readout**:
  This is the readout type used when **Natural** readout is selected.
- **Encoder Resolution**:
  This defines the resolution of the encoder controlling the attribute.
- **Log Channels**:
  This is information only. It shows how often the attribute is used in logical channels of fixture types added to the show.
- **Channel Functions**:
  This is information only. It shows how often the attribute is used in channel functions of fixture types added to the show.
- **Hide**:
  This can be used to hide the attribute on the interface. It will not be visible on encoder bars or on any sheets. If the attribute is matched with a DMX channel then it will still be visible in the DMX sheet.

Add an Attribute Definition

It is recommended to use one of the factory-defined attributes, if possible.

Custom attribute definitions can be created following these steps:

1. Press Menu.
2. Tap Patch.
3. Tap Attribute Definitions on the left side.
4. Select the line in the list, where the new definition should be above.
5. Tap Insert New Attribute at the bottom.
6. Fill in all the relevant cells.
7. Close all the menus and tap Ok to accept the changes to the setup.

Now the definition exists and can be used to create new custom devices.

Physical Units

The physical units are used to describe the physical part of the attribute - if relevant.

For some attributes, this is not relevant. For instance, the selection of a Gobo is linked to an image of the gobo projected, but it is not defined as a physical unit (it is a wheel slot information). Physical unit information is relevant for the attributes that define the rotation of the selected gobo. So the attribute defining the gobo position (index) uses **Angle** as a physical unit. The attribute defining the continuous rotation of the gobo uses **AngularSpeed** as the physical unit.

Editing the cells for physical units (see the column descriptions above) opens the **Select Physical Unit** pop-up.

There are many physical units available on the list. Select the one matching the needed unit.

Subtopics

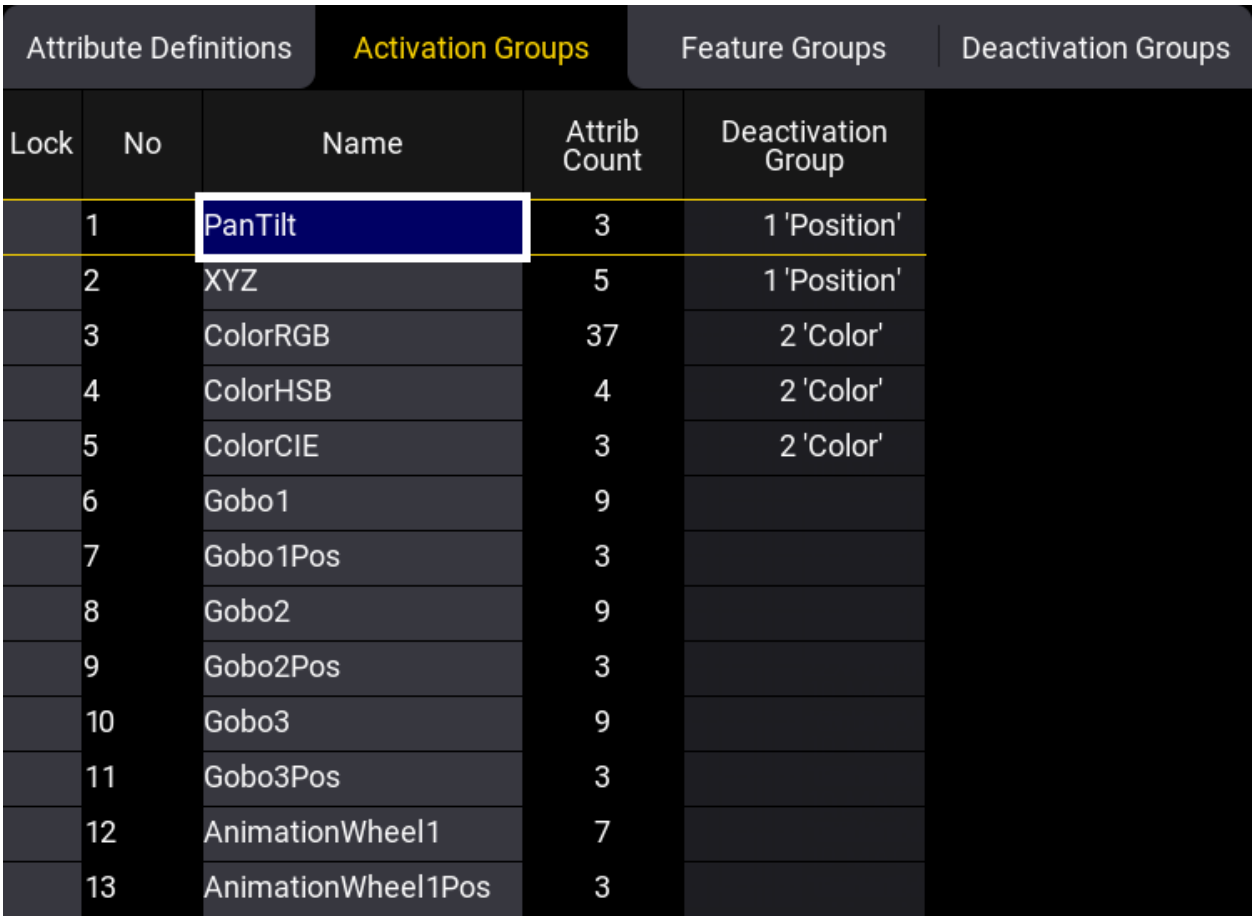
- **Activation Group**
- **Feature Group**
- **Deactivation Group**

1.14.15.1. Activation Group

An activation group is used when there are several attributes where it makes sense that they are activated together.

For instance, storing pan and tilt values together often makes sense. Having them in the same activation group makes this possible. Activating one of them also activates the other.

There are factory-defined groups that are used with the factory-defined attributes. Custom groups can be made and then used in the **Attribute Definitions**. The group needs to be created before it can be used.



Lock	No	Name	Attrib Count	Deactivation Group
	1	PanTilt	3	1 'Position'
	2	XYZ	5	1 'Position'
	3	ColorRGB	37	2 'Color'
	4	ColorHSB	4	2 'Color'
	5	ColorCIE	3	2 'Color'
	6	Gobo1	9	
	7	Gobo1Pos	3	
	8	Gobo2	9	
	9	Gobo2Pos	3	
	10	Gobo3	9	
	11	Gobo3Pos	3	
	12	AnimationWheel1	7	
	13	AnimationWheel1Pos	3	

The standard Activation Groups

The menu has the following columns:

- **Lock**:
  This can be used to lock the row from being edited.
- **No**:
  This is an auto-generated row number.
- **Name**:
  This is the name of the activation group.

- **Attrib Count**:
  This is a counter showing how many attributes are part of the group.
- **Deactivation Group**:
  This column selects a **Deactivation Group** for two or more activation groups. Read more about this in the **Deactivation Group topic**.

	**Important:**
	Editing the existing Activation Groups is not recommended. Editing them can result in not automatically storing all expected values.

Create and Use a New Activation Group

To create a new activation group:

1. Press Menu.
2. Tap Patch.
3. Tap Attribute Definitions on the left side.
4. Tap Activation Groups on the top tabs.
5. Select the line in the list, where the new group should be above.
6. Tap Insert New ActivationGroup at the bottom.
7. Edit the name field to add a descriptive name.
8. Optionally edit the **Deactivation Group** cell to add a deactivation group.
9. Tap Attribute Definitions on the top tabs.
10. Assign the new group to the relevant attributes in the **Attribute Definitions** menu.
11. When finished, close the menus and tap Save and Exit in the pop-up asking if the changes should be kept.

For more information on how to use the **Activation Group** read the **Attribute Definition topic**.

1.14.15.2. Feature Group

Feature groups are part of the structure and hierarchy of the entire show.

This structure is visible and used in the **Feature Group Control Bar** in the default **Encoder Bar**.

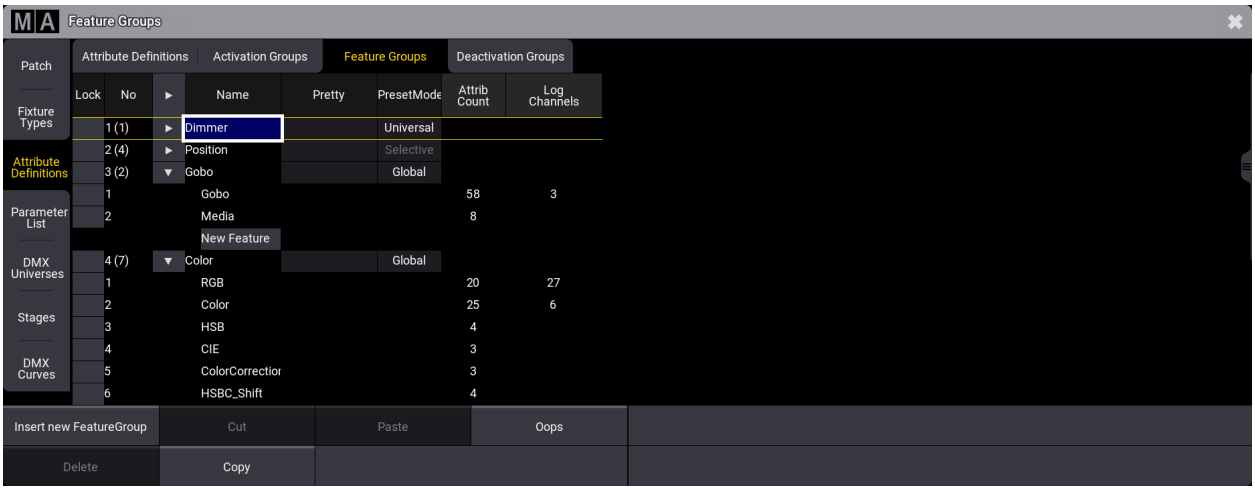
	Important:
	Be aware that editing the Feature Groups settings can have serious consequences for the structure of the show. Editing the existing groups is only recommended if the consequences is known. Maybe use the custom **Encoder Bars** to manage encoder set up.

A **Feature Group** contains at least one **Feature**.

The feature is assigned to attributes in the **Attribute Definitions**. All attributes need to have an assigned feature.

A feature group automatically has a **Preset Pool** where the attributes in the feature group can store values. Read more about presets in the **Presets topics**.

The feature group and feature need to be created before they can be used.



The standard Feature Groups with Features

The menu has 6 columns:

- **Lock**:
  This can be used to lock the row from being edited.
- **No**:
  This is an auto-generated row number for the feature group. The number in parentheses is the number of features inside the group.
- **Name**:
  This is the name of the feature. This needs to be a unique name.

- **Pretty**:
  This is a custom label for the feature group. It can have local characters or a different spelling. For instance "Color" could be labeled "Colour".
- **Preset Mode**:
  This is the default type used when storing values in the preset pool.
- **Attrib Count**:
  This is information only. This is a counter showing how many attributes use the feature.
- **Log Channels**:
  This is information only. It shows how often the feature is used in logical channels of fixture types added to the show.

Create and Use a New Feature Group and Feature

Create a new feature group and feature following these steps.

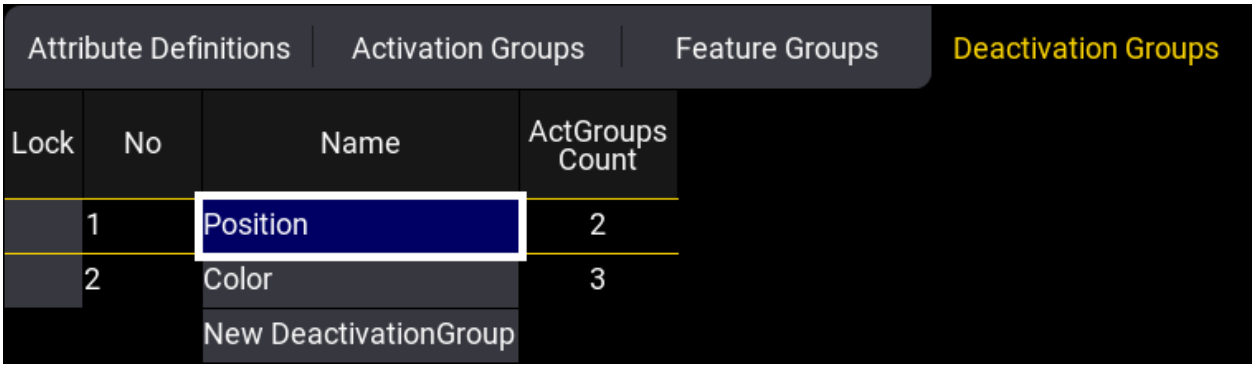
1. Press Menu on the console.
2. Tap Patch to select the patch menu.
3. Tap Attribute Definitions on the left side.
4. Tap Feature Groups on the top tab.
5. Select the row in the list, where the new group should be above.
6. Tap Insert New FeatureGroup at the bottom.
7. Fill in the name of the group.
8. Tap the triangle icon ▶next to the new group to expand the group.
9. Tap the New Feature area in the new group.
10. Tap Insert and write the name of the feature. Repeat until all the needed features have been created.
11. Tap Attribute Definitions on the top tab.
12. Assign the new features to the relevant attributes.
13. When finished close all the menus and tap Save and Exit in the pop-up asking if the changes should be kept.

For more information on how to use the **Feature Group** read the **Attribute Definition topic**.

1.14.15.3. Deactivation Group

Deactivation Groups are groups of **Activation Groups**. A deactivation group makes sure that only one set of valid attribute values are active for the actual DMX channels being output. This is to avoid DMX channels getting conflicting information. The deactivation group knocks out other groups of attributes when a value is assigned to a different activation group than the one already active.

For example, when having active pan and/or tilt values in the programmer, the deactivation group "Position" takes care, that these attributes will be knocked out when activating XYZ attributes instead.



Default deactivation groups

By default, there are 2 automatically generated deactivation groups: Position and Color.

The menu has the following columns:

- **Lock**:
  This can be used to lock the row from being edited.
- **No**:
  This is an auto-generated row number.
- **Name**:
  This is the name of the activation group.
- **ActGroups Count**:
  This is the number of activation groups using the deactivation group.

The deactivation groups are assigned to activation groups in the **Activation Groups** tab.

	Important:
	Editing the existing Deactivation Groups is not recommended. Editing them can result in conflicting values being stored.

Create and Use a New Deactivation Group

Create a new deactivation group following these steps.

1. Press Menu.
2. Tap Patch.
3. Tap Attribute Definitions on the left side.

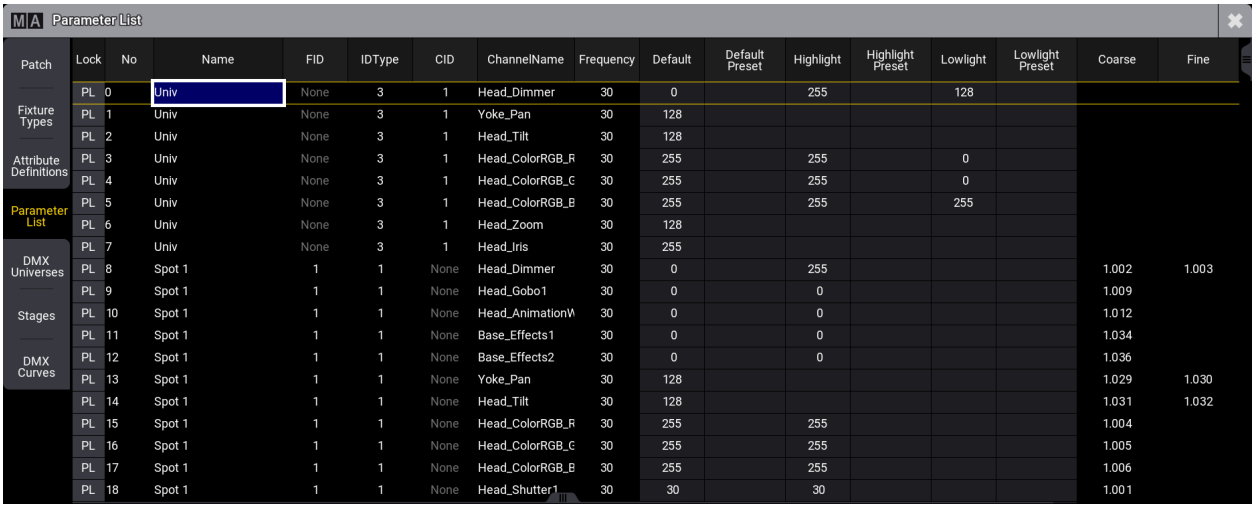
4. Tap Deactivation Groups on the top tabs.
5. Select the line in the list, where the new group should be above.
6. Tap Insert New DeactivationGroup at the bottom.
7. Edit the name field to add a descriptive name.
8. Tap Activation Groups on the top tabs.
9. Assign the new deactivation group to the relevant activation groups in the **Activation Groups** menu.
10. When finished, close the menus and tap Save and Exit in the pop-up asking if the changes should be kept.

1.14.16. Parameter List

The Patch menu gives access to the list of all the parameters by tapping the Parameter List button in the menu on the left of the **Patch Menu**.

These parameters are also called RTChannels (Realtime Channels).

This menu lists all the parameters in the show.



Parameter List menu

The menu is a big table with the parameters in rows. This is a short explanation of the different columns:

- **Lock**:
  This indicates a "PL" (Position Locked) if the row is locked.
- **No**:
  This is the parameter number.
- **Name**:
  This is the name of the fixture.
- **FID**:
  This is the FID of the fixture using the parameter.
- **IDType**:
  This is the ID number of the fixture ID type.
- **CID**:
  This is the CID for the fixture using the parameter.
- **ChannelName**:
  This is the name of the channel in the fixture definition.
- **Frequency**:
  This is the DMX output frequency.
- **Default**:
  This is the default value for the parameter.
- **Default Preset**:
  If the default value is referencing a preset, then the preset number and name are shown here.

- **Highlight**:
  This is the highlight value for the parameter.
- **Highlight Preset**:
  If the highlight value is referencing a preset, then the preset number and name are shown here.
- **Lowlight**:
  This is the lowlight value for the parameter.
- **Lowlight Preset**:
  If the lowlight value is referencing a preset, then the preset number and name are shown here.
- **Coarse**:
  This is the DMX address for coarse control of the parameter.
- **Fine**:
  This is the DMX address for fine control of the parameter.
- **Ultra**:
  This is the DMX address for ultra-fine control of the parameter.
- **DMX Curve**:
  This field can be used to select one of the existing DMX curves. Learn about DMX curves in the **DMX Curve topic**.

Only the default, highlight, lowlight values and presets, and the DMX curves can be changed in this list.

A filtered version of this list is also shown when a fixture is edited. It is filtered to only show the attributes (or RTChannels) for the fixture being edited.

Change the Default, Highlight, and Lowlight Values

The values in the Default, Default Preset, Highlight, Highlight Preset, Lowlight, and Lowlight Preset columns can be edited in this menu or when editing the fixture.

It is also possible to store the programmer's content to these elements. It automatically assigns presets that might be in the programmer into the preset column.

To reset any of the special values, set the corresponding attributes to remove values in the programmer and Store /Merge them to the desired type of special values. Or set any value for the corresponding attributes and do a Store /Remove.

It is done using the **Default**, **Highlight**, and **Lowlight** keywords and the **/Remove** and **/Merge** option keywords.

Examples:

Store a new default position for a group of fixtures.

1. Clear the programmer for any existing values and selection.
2. Select the desired fixtures.
3. Give them the desired position for instance from a preset.
4. Be careful to not have undesired values in the programmer (for instance, intensity values).
5. Execute this command: **Store Default**.
6. Clear the programmer and verify the new default position.

Remove the currently stored highlight values for a selection of pan and tilt attributes.

1. Clear the programmer for any existing values and selection.
2. Select the desired fixtures.
3. Give the pan and tilt attributes some value - the actual value does not matter it only matters that there are active programmer values for the desired attributes.
4. Be careful to not have undesired values in the programmer (for instance, intensity values).
5. Execute this command: **Store Highlight /Remove**.
6. Clear the programmer and verify the new highlight position.

For resetting the special values of default and lowlight, follow the example above but replace Highlight with Default, or Lowlight.

In addition, storing with the **/Release** option keyword to the special values will do the same as Remove.

	**Important:**
	For which fixtures the special values will be changed when using the command line approach, the Use Selection setting and the If not empty setting of the Store Options are important. To learn more about the Use Selection setting and the If not empty setting, read in the **Store Options topic**.
	**Restriction:**
	It is only possible to store special values using absolute values. Relative values will be ignored.
	**Restriction:**
	Presets with more than 1 step ("Phaser presets") will only use step 1 when using them as special values.

1.14.17. DMX Curves

Attributes use DMX curves. As a default, they use a linear transition from 0% to 100%. Other DMX curves can be created and assigned to parameters.

There are three types of DMX curves: MinMax, Switch, and Custom.

	Restriction:
	A show file can contain 9 999 DMX curves.

Importing fixture types that use special DMX curves adds the curves to the list.

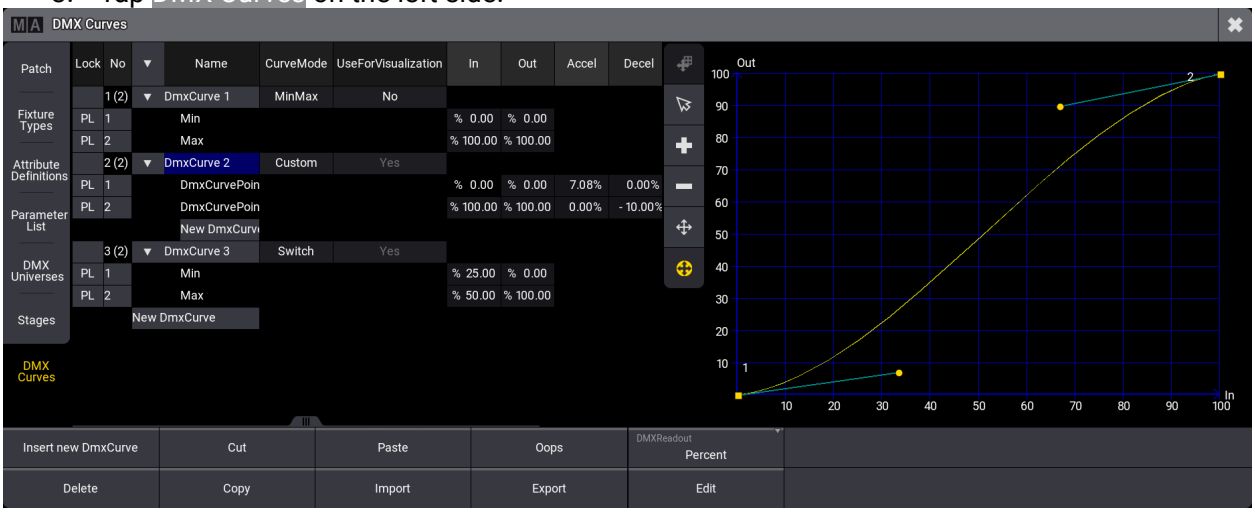
Curve values can be seen and edited in different readouts. These can be changed in the menu at the bottom by tapping the DMXReadout button.

Other buttons at the bottom give access to Cut, Copy, and Paste the curves in the list. There is also a button that deletes a curve. If the curve is used, it might be impossible to delete it.

The curves are displayed, created, and edited in the DMX Curves menu found in the patch.

Open the DMX Curves Menu

1. Press Menu.
2. Tap Patch.
3. Tap DMX Curves on the left side.



DMX Curve menu

The DMX Curve menu is split into left and right sides. The left side lists the curves, and the right displays the selected curve.

The curve settings are listed on the left side list. The curves can be unfolded (tap the white right-pointing triangle), and the different points on the curves can be seen and edited in the table mode. In and Out show the relationship between the incoming value and what it is translated to in the curve.

The right side can also be used to edit curves. It has a standard toolset for selection (), addition of point (), removal of point (), move point (), and move point handle (). Not all tools not available for all curve modes.

The column called **UseForVisualization** defines whether the DMX output, as defined by the DMXCurve, shall be visualized in the 3D Viewer.

	**Important:**
	When enabling this property for DMXCurves created by color-calibrated fixtures, the colors in the 3D Viewer will not match the actual output on stage!

All DMX curve changes are made inside the patch menu. This means that the menu needs to be closed and changes saved for them to take effect. To finish all DMX curve changes, leave and save the patch changes. Or exit without saving to cancel any changes.

Create a New MinMax Curve

MinMax curves are linear DMX transitions from a minimum to a maximum value. These can be useful to limit output values in, for instance, dimmers or the pan and tilt movements of fixtures.

1. Open the DMX Curves menu.
2. Tap a location in the curves list on the left side where the new curve should be.
3. Tap Insert new DmxCurve at the bottom - the default curve mode for a new curve is **MinMax**.
4. Edit the name to give it a custom name.
5. Edit the Min and Max values to match the needs.

Create a New Switch Curve

Switch curves use two points to create a switch point where the output value instantly changes from one output value to the other.

1. Open the DMX Curves menu.
2. Tap a location in the curves list on the left side where the new curve should be.
3. Tap Insert new DmxCurve at the bottom.
4. Edit the CurveMode value to open the small **Select CurveMode** pop-up.
5. Tap Switch in the pop-up.
6. Edit the name to give it a custom name.
7. Edit the Min and Max values of the two points to match the needs.

Having two points in the switch allows for setting a value needed for switching on and another value for switching down. This can help with hysteresis.

Example:

1. Create a new switch curve.
2. Set the **Min** point to In = 70% Out = 0%
3. Set the **Max** point to In = 80% Out = 100%

This creates a switch that needs a value of 80% and above to turn On (output 100%) and a value of 0% to 70% to turn Off (0% output)

Create a New Custom Curve

Custom curves can be used for many different things, for instance, to match the emitter output of LED fixtures.

1. Open the DMX Curves menu.
2. Tap a location in the curves list on the left side where the new curve should be.
3. Tap Insert new DmxCurve at the bottom.
4. Edit the CurveMode value to open the small **Select CurveMode** pop-up.
5. Tap Custom in the pop-up.
6. Edit the name to give it a custom name.
7. Points can be added using the Add tool on the right side or by creating a New DMXCurvePoint in the table view on the left side.
   To create a point using the tools:
   a. Tap the Add tool (+)
   b. Tap in the curve where the point should be added
8. Edit the Min and Max values and the Accel (acceleration) and Decel (deceleration) of the points to match the needs. This can be done in the table or by using the graph editor on the right side.

	**Important:**
	A curve might give a warning if the output at some point is lower than a previous output value. Curves with a warning have the name displayed with a red text color.

Export and Import Curves

Curves can be exported to a drive.

1. Select the desired drive.
2. Tap Export.
3. Select the desired drive.
4. Write a name in the name input field.
5. Tap Export in the pop-up.

Curves can be imported to the show.

1. Select a location in the curve list.
2. Tap Import.
3. Select the desired source drive.
4. Select the desired curve.
5. Tap Import in the pop-up.

Assign Curve to a Parameter

The curves can be assigned to parameters in the **Parameter List**.

1. Open the Parameter List in the menu on the left side.
2. Locate the parameter that needs to use the custom curve.
3. Edit the DMXCurve field and select the custom curve in the small select pop-up.

Assign Curve to a Fixture Type

DMX Curves can be assigned to an attribute in a **Fixture Type**.

1. Open the Fixture Types in the menu on the left side.
2. Select the desired fixture in the list.
3. Tap Edit in the menu at the bottom.
4. Tap the DMXModes tab at the top.
5. Edit the DMX Curve field for the desired attribute.
6. Select the wanted curve in the pop-up.
7. Close the FixtureType editor pop-up by tapping the X in the upper right corner.

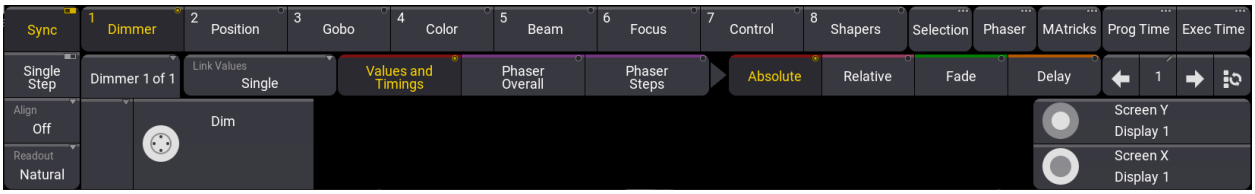
1.15. Operate Fixtures

When the fixtures are added and patched in the show, see **Patch and Fixture Setup**. The next step is to select and operate them.

Different fixture types can do various things. Most fixtures have some intensity control, but if it is a moving light, it will also have pan and tilt control. If the fixture can change color, it will have a color wheel, an RGB, a CMY, or a scroller. If it is a media server, there are many other settings to control, such as which clip to play. All these different control elements are called **Attributes**.

	Hint:
	Knowing the patched fixtures and their capabilities (attributes) is important. Do not waste time trying to mix the perfect color if the fixture has a fixed color wheel.

One of the primary ways to operate fixtures is via the **Encoder Bar**.



Encoder Bar

The **Encoder Bar** gives fast access to available attributes for the selected fixtures.

The **Dual Encoders** change function depending on the selected **Feature Group**, for instance, Position.

There is a window for color control, the **Special Dialog**. This is an excellent tool for selecting a color independently of the fixture's color system.

The **Smart View** provides quick access to fixture-defined values. For instance, most fixtures with a gobo wheel have information about what value is needed to select the gobo slots.

Dimmer

The dimmer is a little special because it is a universal attribute. Almost all fixtures have some intensity control.

There are many ways to set the dimmer value on selected fixtures. Here are some of them:

- Use the level wheel to set the value.
- Use the numeric keys to set a specific value, for example: At 3 0 Please.
- Press Full to give the fixture 100% intensity.
- Select **Dimmer** in the encoder bar and use the dual encoders to set a value.
- Select **Dimmer** in the encoder bar and press on the inner ring of the dual encoder to open the **calculator**. Use this to set a specific value.

- Tap and hold a cell in the fixture sheet under the **Dimmer** column to open the calculator and set a value.
- Use a dimmer preset.
- Enter the **Normal Value** by pressing At At. See **User Settings**.
- **Select Fixtures**
- **Fixture Sheet**
- **What Is the Programmer**
- **Clone**
- **Graphics**
- **Encoder Resolution**
- **Special Dialog**
- **Gels Pool**
- **Selection Bar**
- **Align**
- **SMArt View**
- **Selection Grid**

1.15.1. Select Fixtures

First, fixtures must be selected to operate and change the values of attributes (for example, dimmer, pan, tilt, or zoom).

To select fixtures, use one of the following ways:

- Type the fixture ID in the command line using the numeric keys.
- Tap at the fixtures in a **Fixture Sheet**.
- Draw a lasso around the fixtures in the **3D Viewer**.
- Tap a Group Pool object. See **Create Groups**.
- Tap a Preset Pool object. See **Create New Presets**.

	**Hint:**
	See the **System Colors** topic to learn more about the font colors used when selecting or deselecting fixtures in the fixture sheet.

Examples

Requirements:

- 10 patched fixtures. See **Add Fixtures to the Show**.
- A fixture sheet window is open.

To select fixtures 1 thru 5, press the following hardkeys on the console or use the command section menu  :

Fixture 1 Thru 5 Please

This command is visible in the command line input:

Fixtures 1 thru 5 are selected. The five selected fixtures are in a yellow font in the Fixture Sheet window.

To exclude fixtures from a selection, for example, types:

Fixtures 1 to 5 are selected, 6 to 8 are deselected, and 9 and 10 are selected.

To make the same selection of fixtures as shown in the example above using a different syntax, type:

MA	User name[Fixture]>Fixture 1 Thru 5 + 9 Thru 10

To select fixtures in two commands, type:

MA	User name[Fixture]>Fixture 1 Thru 5
MA	User name[Fixture]>Fixture 9 + 10

To select the fixtures 9 and 10 in the example above, + or Thru can be used.

	**Important:**
	Some keys are used to navigate the sub-selection of fixtures, which is part of the **MAtricks** function. The keys have an on-screen version called the **Selection Bar,** which can also be used to select fixtures.

Recursive Selection of Fixtures

Some fixtures have multiple sub-fixtures. The following examples show how to select fixtures and their sub-fixtures in different orders and hierarchies using an additional dot (.).

	**Hint:**
	See the First Patch topic in the Quick Start Guide to learn more about sub-fixtures.

Requirement for this example:

- 10 Ayrton Alienpix - RS Ex 16-bit patched. See **Add Fixtures to the Show**.
- Fixtures spot numbered from 301 to 310.
- A fixture sheet window is open.

To select fixture 301, type:

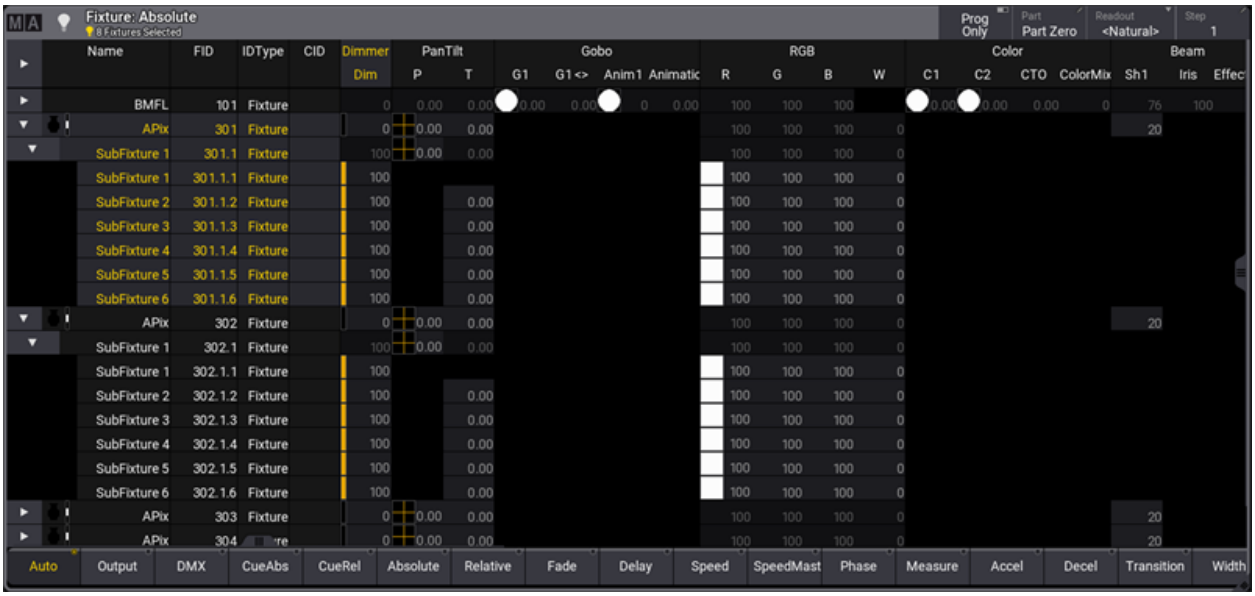
MA	User name[Fixture]>Fixture 301

Only the main fixture is selected.

- A dot (.) can be added to the selection syntax of the main fixture to quickly select all sub-fixtures.

To select fixture 301 and all its sub-fixtures, type:

MA	User name[Fixture]>Fixture 301.



Fixture 301 and the sub-fixtures are selected.

To select the main fixtures and all sub-fixtures of fixtures 301 thru 303, type:

All AlienPix (fixtures and sub-fixtures) are selected.

- The dot selects the sub-fixtures downwards from the specified level. Therefore, it is also possible to select only the pixels of an AlienPix.

To select all pixels of an AlienPix, type:

All pixels of an AlienPix are selected.

To select pixels of an AlienPix in particular, type:

```
User name[Fixture]>Fixture 301.1. Thru 4 + 6
```

Particular selection of pixels.

General Fixture Selection

To select all fixtures and sub-fixtures, type:

```
User name[Fixture]>Fixture Thru .
```

To select all parents of all fixtures, type:

```
User name[Fixture]>Fixture Thru
```

1.15.2. Fixture Sheet

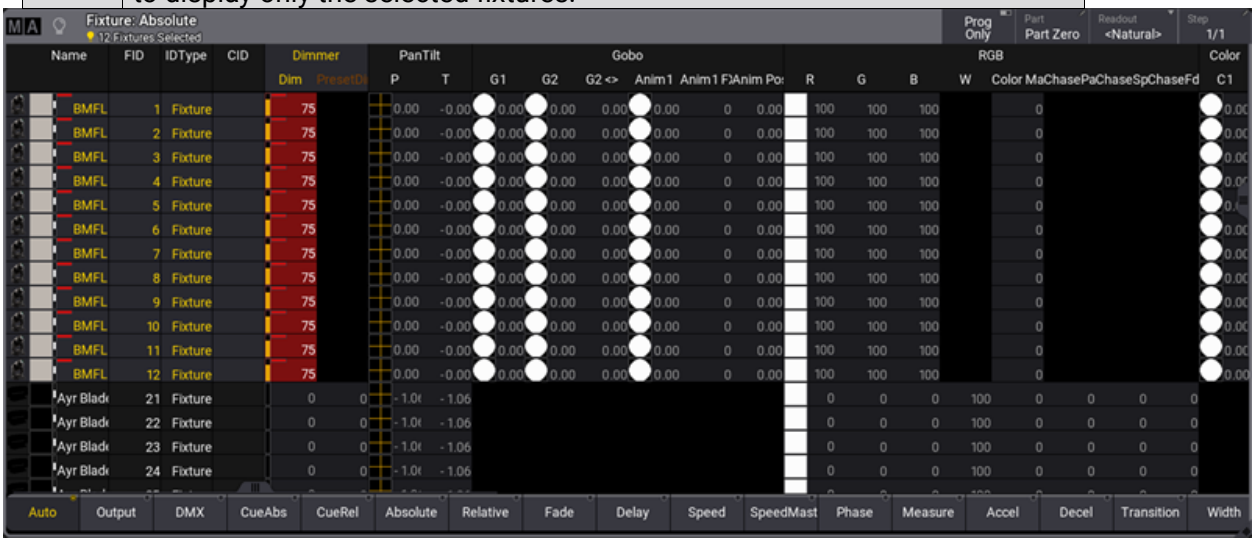
The fixture sheet shows all the patched objects in the show file with an FID or a CID. This is usually every object that needs to be controlled in the show. Read the **Patch and Fixture Setup** topic to learn how to add objects to the patch.

The fixture sheet is a window that can be created like any other window. For more information, see **Add Window**.

Filters, worlds, and the selection can be assigned to the fixture sheet. This will, for example, reduce the amount of displayed fixtures in the sheet. Information about the filter is displayed in the title bar. For more information, see **Worlds and Filters**.

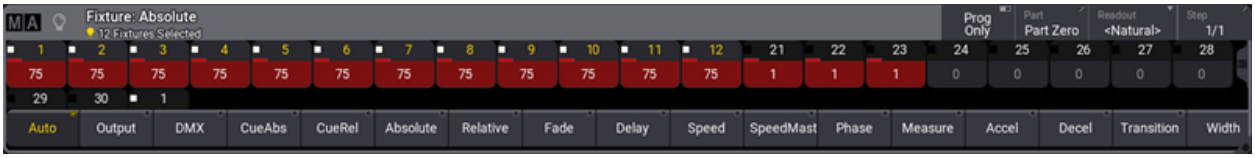
Hint:

Press Assign Fixture Fixture and then tap the title bar of the Fixture Sheet to display only the selected fixtures.



Fixture Sheet in fixture mode

The sheet has four modes: **Fixture**, **Channel**, **Dimmer+,** and **Sheet/Filter.** These modes can be set in the **Window Settings**.



Fixture Sheet in channel mode

Hint:

The sheet uses system colors. For more information, read the **System Colors** topic.

The **Channel** mode shows all objects with an ID and an intensity attribute, but only the ID and its intensity values. The **Fixture** mode shows the same objects but the values of every available attribute. The

Dimmer+ mode is similar to the Channel mode but additionally displays the attributes of the selected feature group. The **Sheet/Filter** mode is similar to Dimmer+ but additionally displays all attributes unless there is a defined filter in the Mask tab of the sheet settings.

The active world filters the sheet. If this world is different than world one, then the world name and number are displayed in the title bar.

The fixture sheet is divided into rows and columns. In the **Fixture** mode, each row represents a fixture or a sub-fixture.

The first column is the fixture name. To see the sub-fixtures for the current fixture, tap on the arrow on the left side of the fixture name column.

The second column is the FID (Fixture ID). Every fixture patched in the showfile must have a unique FID, CID, or both. If a fixture contains sub-fixtures, these IDs will be the fixture number, a dot, and a sub-number.

The next column shows the IDtype of the fixture. Learn more about the ID types in the **What are Fixtures** topic.

The next column is the CID (Channel ID / Custom ID depending on the ID type). This is the other ID a fixture can have.

The next columns display the values for the fixture's different attributes, depending on the mask and display settings. Some attributes might have symbols next to them, showing the selected gobo or the result of the color attributes.

Channel mode only shows the ID (it prioritizes the CID if available), the intensity value, and a small square showing the current color and intensity combination.

In the title, there are four buttons.

- **Prog Only:**
  Displays only the programmer data.
- **Part**:
  This is the Programmer Part - read more about it in the **What is the Programmer** topic.

When the programmer has values from different parts, all cells of attributes that are not part of the selected programmer part are displayed in the fixture sheet with a darker background color.

Fixture sheet displaying values for part 1 (Dimmer) with a stronger and brighter red background color.

- **Readout**:
  The readout defines how the attribute values are displayed. The options are:
  - **Auto** - This makes the sheet follow the readout set in the encoder toolbar. Notice that the button in the title bar does not say Auto; it displays the selected readout in angle brackets.
  - **Natural** - For more information, see Patch and **Fixture Setup - Attribute definitions**.
  - **Percent** - This displays the value as a percentage from 0% to 100%. It is a whole number. **Percent** Keyword.
  - **PercentFine** - This also displays the value in percent. The range is the same, but the resolution is higher. The value has two decimal numbers. **PercentFine** Keyword.
  - **Physical** - These are the physical values defined in the fixture type definition. **Physical** Keyword.
  - **Decimal8** / **Decimal16** / **Decimal 24** - This displays the value in a decimal number in the three available resolutions (256, 65 536, or 16 777 216). **Decimal8**, **Decimal16**, and **Decimal24** keywords.
  - **Hex8** / **Hex16** / **Hex24** - Displays the value in a hexadecimal number in the three available resolutions (FF, FFFF, or FFFFFF). **Hex8**, **Hex16**, and **Hex24** keywords.
- **Step**:
  This is the step number - learn more about steps in the **Phaser section**.

	Hint:
	There is a **user profile setting** called **Value Readout**. The Auto option uses this setting, and the encoder toolbar changes it.

To read more about all the other different settings and masks, read the **Window settings** topic.

	Hint:
	It is possible to select fixtures by tapping on them in the Fixture Sheet.

1.15.3. What Is the Programmer

The programmer is a temporary memory where the edited values are placed. The values can then be stored or released.

Every user has their own programmer.

The programmer has three levels:

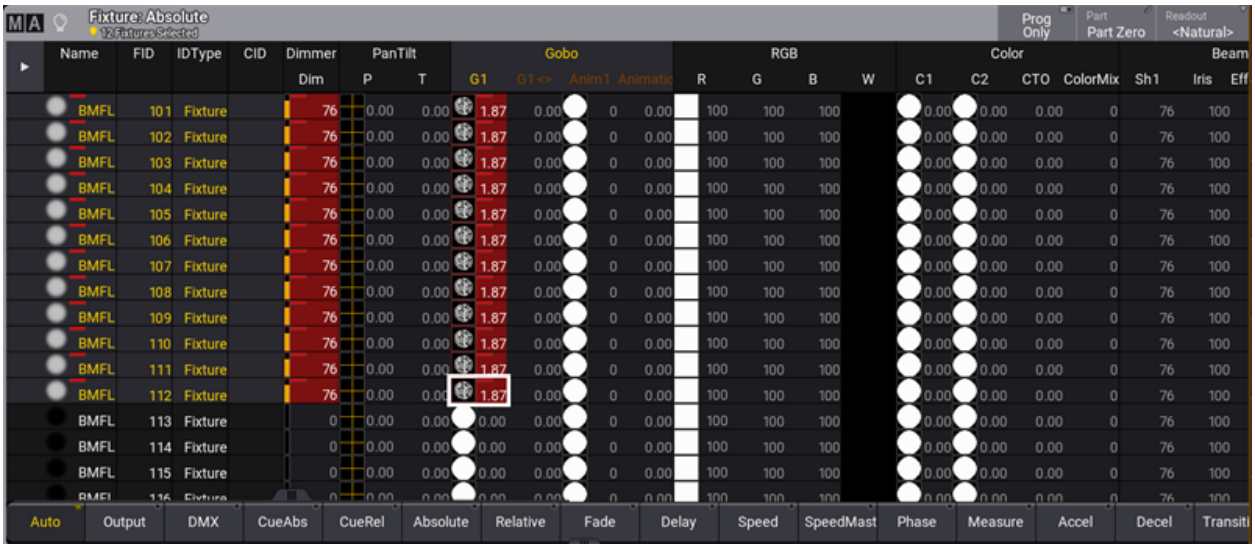
- Selected fixture
- Active programmer values
- Inactive programmer values

The programmer's values usually affect the system's output. However, there is a Blind function that allows hiding the programmer's values from the output.

Selected fixtures will be affected by encoder input and any intensity changes. They can be identified with a yellow name and ID text color.

Both active and inactive programmer values can affect the output. The difference is that the active values will be stored using the default **Store Options** (Use Selection = Active).

The Fixture Sheet colors for active attribute layers are explained in detail in the **System Colors** topic.



Values in programmer

When a fixture is selected, it is possible to press Please twice to activate all attributes for the selected fixture. This puts all the current values for all the fixture attributes in the programmer. Press Please once more, and the attributes will be inactive in the programmer.

To release values from the programmer, press Off and then tap the value to deactivate it. Feature groups can also be released from the programmer using Off and then tapping the encoder bank button.

When there are values in the programmer, and a fixture is selected, it is possible to use the clear button to clear each of the three levels:

- Press Clear to deselect the fixture but keep the active and inactive values in the programmer.
- Press Clear again to keep the values in the programmer but as inactive values.
- Press Clear a final time to clear the programmer and release all the values.

Completely clearing the programmer can also be done by holding Clear for more than one second.

The associated keywords are **Clear**, **ClearSelection**, **ClearActive**, and **ClearAll**.

The Clear button in the AT overlay executes the Clear command. For more information, see **At Overlay**.

Layers

The programmer has several layers. Normal values - for instance, a selected static position - are in the **Absolute** value layer. An absolute value can be affected by a value in the **Relative** layer. For more information, see **Encoder bar**. A dark purple marker indicates relative values. The **Fade** layer can be used to adjust and see individual fade times. Individual fade times are indicated with a green marker. The **Delay** layer can be used to adjust and see individual delay times. An orange marker indicates them. These values are best looked at in the **Sequence Sheet** in tracking mode. Learn more about **Individual Attribute Timing** in Cues and sequences topic.

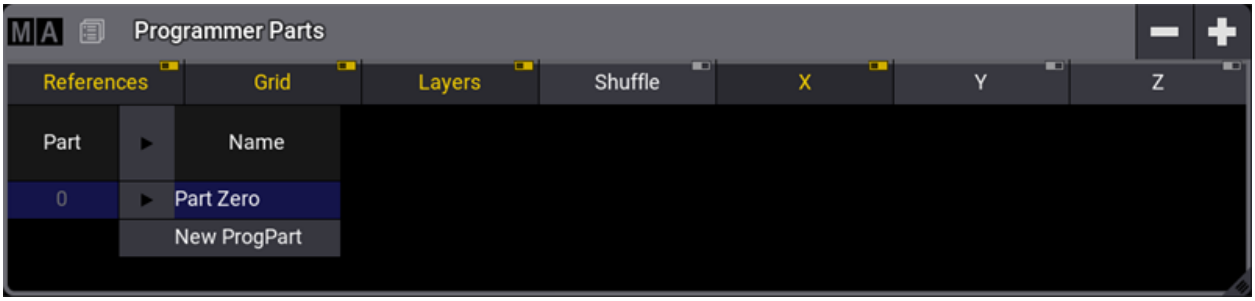
	Important:
	After calling the values of a Preset, the next time the preset is tapped, the programmer values for the Layer not included in the preset will be discarded.

Programmer Parts

The programmer can have several parts (Maximum: Part Zero + 239 individual parts = 240 parts). The ProgrammerParts window shows all parts of the current programmer. Parts can be labeled here directly.

Using cue parts is a good way to organize your show better. The primary information for every cue is in part 0. Part numbering starts at Part 0. Presets can be called into the programmer parts automatically when calling the preset. This then creates the respective cue parts when storing the programmer's content into a cue. Learn more about pool settings at **Preset Pool — Preset Pool Settings**.

Open a Programmer Parts window to see all active programmer parts. For more information, see **Add Windows**.



Programmer Parts

Programmer parts can be deleted or created using the buttons in the title bar. A part can be selected by tapping the part in the window. A new programmer part can also be created using the command line and the **Programmer Keyword**.

To create programmer part 3, which is labeled "Color", type:

```
User name[Fixture]>Store Programmer 3 "Color"
```

Select the programmer part simply by writing the keyword and the number or name.

```
User name[Fixture]> Programmer "Color"
```

When a programmer part is empty, it will be deleted when entering a different cue part.

For example, to store cue 12 with part cue 3, type:

Store Cue 1 2 Cue 3

```
User name[Fixture]>Store Cue 12 Part 3
```

Freeze

By default, the executors have a higher priority than the programmer. That means the programmer's values can change if a running executor has stored values for the exact attributes. To give the programmer a higher priority than the executors, enable Freeze. This keeps the adjusted values in the programmer even when the executor is executed.

Blind

Blind is a function that hides programmer values from the output. The blind mode can be toggled On and Off by pressing Blind or using the **Blind Keyword**.

Turning on the blind mode without any values in the programmer does not change the output. Changing values while in blind mode does not affect output. If the programmer is cleared before leaving the blind mode, then the output is not affected. Leaving blind with programmer values will add these values to the output. Entering blind mode with values in the programmer removes them from the output, meaning the output is changed.

Program Time

The programmer's value changes are usually done immediately. However, using the **Program Time** function, a timed or manual fade can be used. Read more about it in the **Time Control** topic.

Preview

In preview mode, the user can have values entered in a separate programmer and will not generate output on a stage.

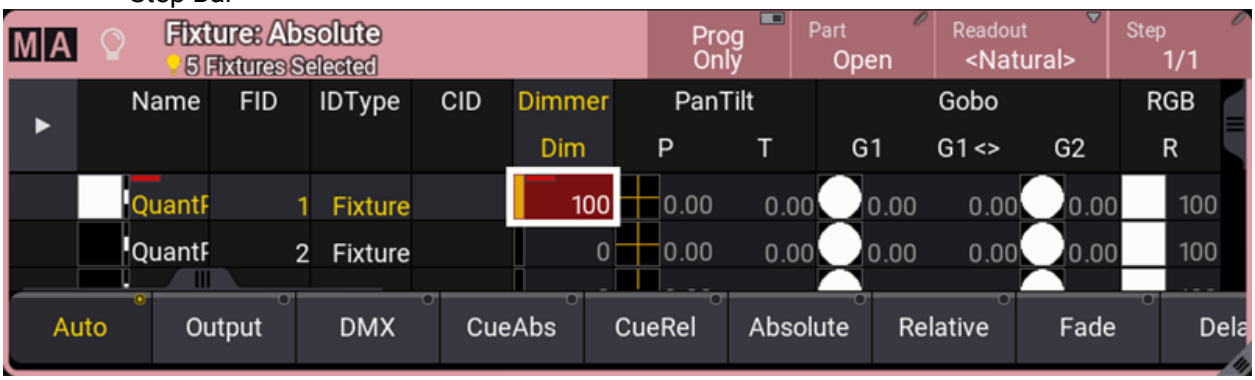
The preview programmer behaves the same as the live programmer. There is one preview programmer in a session.

⚠	**Warning:** When multi-users work in a session, be aware that everyone works in one shared preview programmer. As a result, one user can modify other users' values in the preview programmer.

To enter the preview mode, press Prvw.

When Preview is active, the Prvw key is lit up, and the borders of the following windows will turn red:

- 3D Viewer
- Align Bar
- At Filter, window, and overlay
- DMX Sheet
- Fixture Sheet
- Layout Viewer
- MAtricks Editor, window, and overlay
- Phaser Editor, window, and overlay
- Programmer Parts
- Selection Grid
- sMArt
- Step Bar



Fixture Sheet in Preview Mode

Editing cues and presets in preview mode will turn the window frame yellow.

Fixture Sheet in Edit Preview Mode

To exit the preview mode, press Prvw again. The border of the windows listed above will turn gray.

The output of the preview programmer will be displayed in the corresponding windows. If the values of the preview programmer were not manually cleared, leaving and entering the preview mode will bring back the previous values.

1.15.4. Clone

Cloning can be used for the following occasions:

- To clone data in the patch from existing fixtures to new fixtures, for example, if an additional truss is to be used in a rig on a stage.
- To transfer attribute values from fixtures to fixtures in the programmer.
- To transfer values from attribute to another attribute, for example, from Gobo1 to Gobo2.
- To transfer values from fixtures to fixtures throughout the show file, including Sequences, Groups, Presets, Worlds, and Layouts.

Cloning can be triggered via the command line or a clone dialog in the user interface. For more information about the command line syntax, see **Clone Keyword**.

How to use the clone feature is described in the following topics.

Subtopics

- **Clone Fixtures in the Patch**
- **Use Clone to Transfer Values**

1.15.4.1. Clone Fixtures in the Patch

The cloning feature is a useful tool for duplicating existing fixtures and their data to new fixtures in the Patch menu for immediate use. The Copy and Paste functions automatically create a complete clone of the show data for the corresponding fixtures.

The following topics offer additional information:

- For general information about cloning, see **Clone Function**.
- To understand the Patch menu, see **Patch and Fixture Setup** and **Add Fixtures to the Show**.
- To position fixtures in the 3D viewer, see **Position Fixtures in the 3D Space**.

Take a look at the following example to get a better understanding of cloning fixtures in the patch.

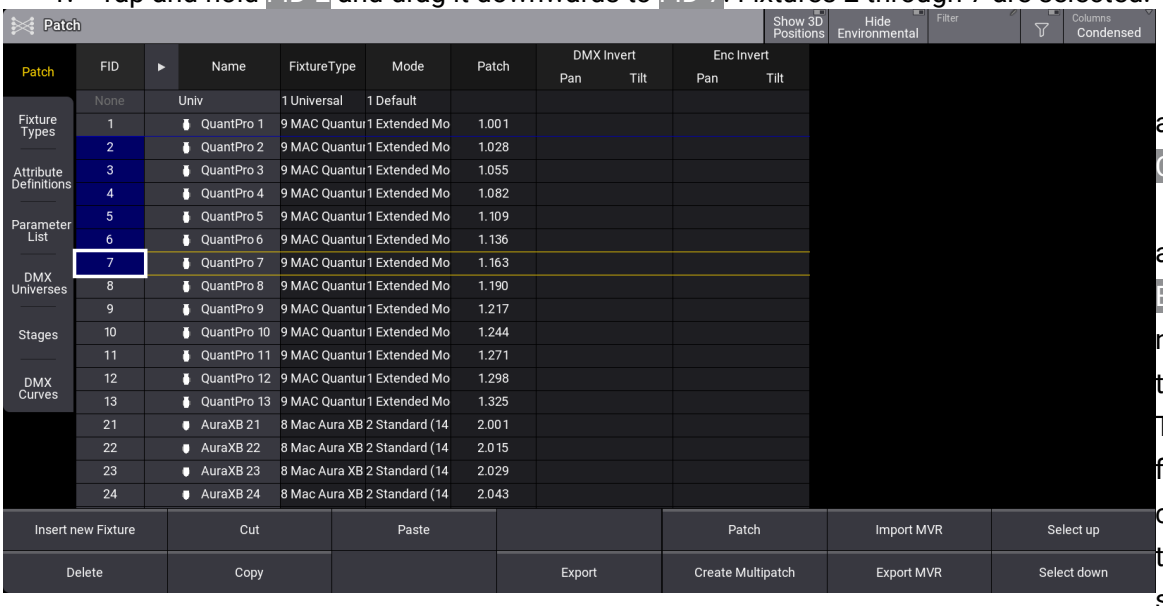
Example

Tap the video below to see the example.

Requirements:

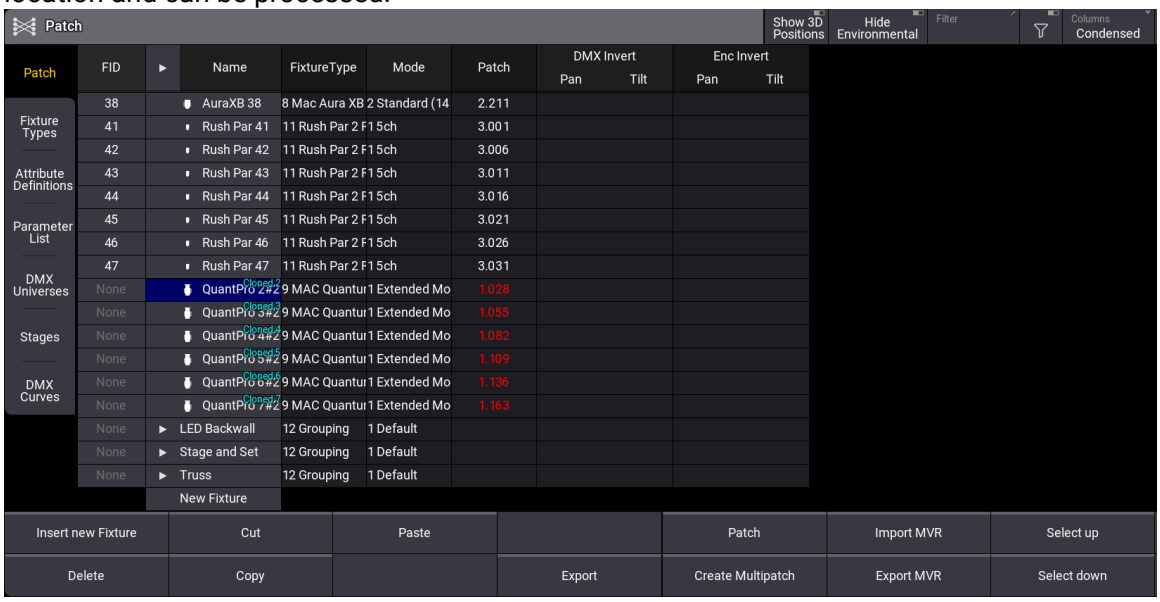
- Load the grandMA3 demo show file and enter the Patch menu.

1. Tap and hold FID 2 and drag it downwards to FID 7. Fixtures 2 through 7 are selected.



2. Tap Copy.
3. Tap LED Backwall and then tap Paste. The copied fixtures are cloned to the specified

location and can be processed.



- After copying and pasting fixtures into the Patch, the pasted fixtures get a cyan text (*Cloned <FID>*) on the right side of the Name cell.
- The number indicates the source fixtures. In the example above, the fixtures 2 through 7 were the source fixtures.

	**Hint:**
	The show data will be copied when leaving the Patch. After closing and reopening the Patch menu, the cyan text is no longer displayed.

1.15.4.2. Use Clone to Transfer Values

For more general information about Clone, read the **Clone** topic first.

Clone Window

To open the Clone window:

- Press MA + X1 | Clone and then press Please.
- Type **clone** into the command line and press Please.
- Type **clo** into the command line and press Please.

The Clone window opens.

The Clone window is separated into the following areas:

- **Left:** Clone Source.
- **Right:** Clone Destination.
- **Middle**: Object button bar. For more information, see **Cloning Show Data**.

	Important:
	Each area can be focused separately by tapping the screen in the corresponding area. A white frame indicates the currently focused area.

Adding Fixtures

To set fixtures into the Clone Source:

1. Select fixtures in the programmer. For more information on selecting fixtures, see **Select Fixtures**.
2. Tap Add Selection in Clone Source. Fixtures are set. The total number of fixtures on each side is indicated by the number in parantheses.

or

1. Tap the source area. The area is focused.
2. Tap the ➕ in the tool area on the left. A pop-up opens.
3. Select fixtures or groups in the pop-up.
4. Tap Add. Fixtures are set.

	Hint:
	On console, use the 2 Finger Edit gesture to add fixtures.

To clear the complete clone list:

- Tap Clear List.

To replace fixtures in the Clone Source:

1. Tap, hold, and drag a finger over the fixtures you want to replace. The selected fixtures will be highlighted in blue.
2. 2 finger edit the highlighted area. A pop-up opens.
3. Select the replacing fixtures in the pop-up. The selected fixtures will be highlighted in blue.
4. Tap Replace. The selected fixtures are replaced.

	**Hint:**
	The steps above work similarly for setting fixtures into the Clone Destination.

To organize the fixtures in the selection, use the following tools located on the left side of the Clone window:

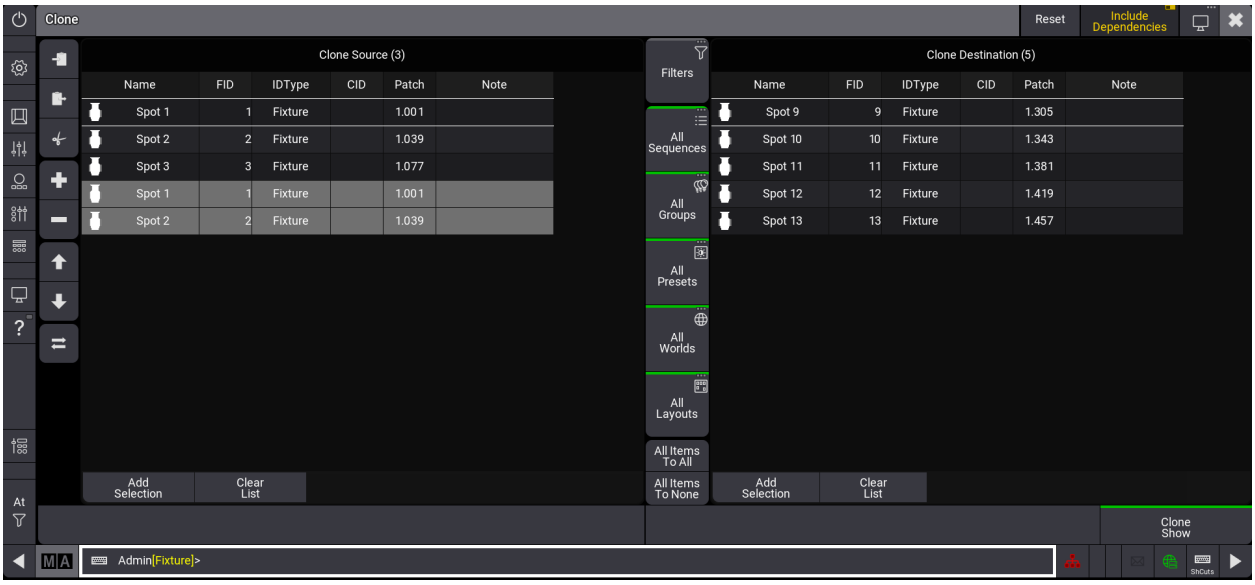
-  : Copies the selected fixtures from the focused side.
-  : Pastes the selected fixtures to the focused side.
-  : Cuts the selected fixtures from the focused side.
-  : Opens the Add Fixtures pop-up.
-  : Deletes the selected fixtures from the focused side.
-  : Moves the selected fixtures up from the focused side.
-  : Moves the selected fixtures down from the focused side.

The fixture selection is synchronized in line on both sides of the clone window. The first line of the clone source uses the first line of the clone destination as the target.

Example

In the image below, Fixture 1 is set as the clone source for Fixture 9 and Fixture 2 is set as the clone source for Fixture 10.

If one area has fewer fixtures than the other, the side with fewer fixtures is automatically adjusted. The adjusted fixtures are displayed with a grey background.



Clone Window - Selection of fixtures

Cloning Programmer Values in the Clone Window

There are two methods for copying fixture values from fixture to fixture for programmer values only:

- You can use the At keyword in the command line input, for more information see **At Keyword**.
- Or use the Clone keyword in the command line input. Single attributes of a fixture source can be cloned as well. For more information, see **Clone Keyword**.

When executing the clone command using the clone keyword, the Clone window opens. For example, when entering **Clone Fixture 1 At Fixture 2** in the command line.

- To execute the clone command directly, use the If keyword. For example, by entering **Clone Fixture 1 At Fixture 2 If Programmer** into the command line input. the Clone window will not open and the priority pop-up opens. For more information, see **If Keyword**.
- To execute the clone command without opening the clone window or the priority pop-up, use the /NoDependencies keyword. The command will be executed in low priority. For more information, see **/NoDependencies**.

Example

Requirements:

- The grandMA3 demo show file is loaded and the clone window is open.

To clone fixtures in the programmer using the Clone window:

1. Add Fixtures to the Clone Source and Clone Destination areas as described above.
2. Tap All Items To None, located in the object bar between both Clone areas. The button in the lower right corner switches to Clone Programmer and the object indicator bar turns grey.
3. Tap Clone Programmer. The fixture values are cloned.

	Hint:
	Cloning only programmer values is indicated by gray-colored indicator bars above the object buttons and the Clone button. For more information on the colors, see **Cloning Show Data**.
	Hint:
	When cloning only programmer values, Include Dependencies has no effect and therefore the button is grayed out. For more information, see **Include Dependencies**.

Cloning Show Data

Tap the video below to see how to clone show data using the user interace.

Object buttons are displayed in form of a bar in the middle of the window. The object buttons use filters to restrict the current cloning process for the following objects:

- Sequences
- Groups
- Presets
- Worlds
- Layouts
- Additionally, the At Filter or Filter Pool can be used. For more information, see **At Filter** or **Filters**.

Depending on the selected items of an object, the color of the indicator bar of the object buttons changes.

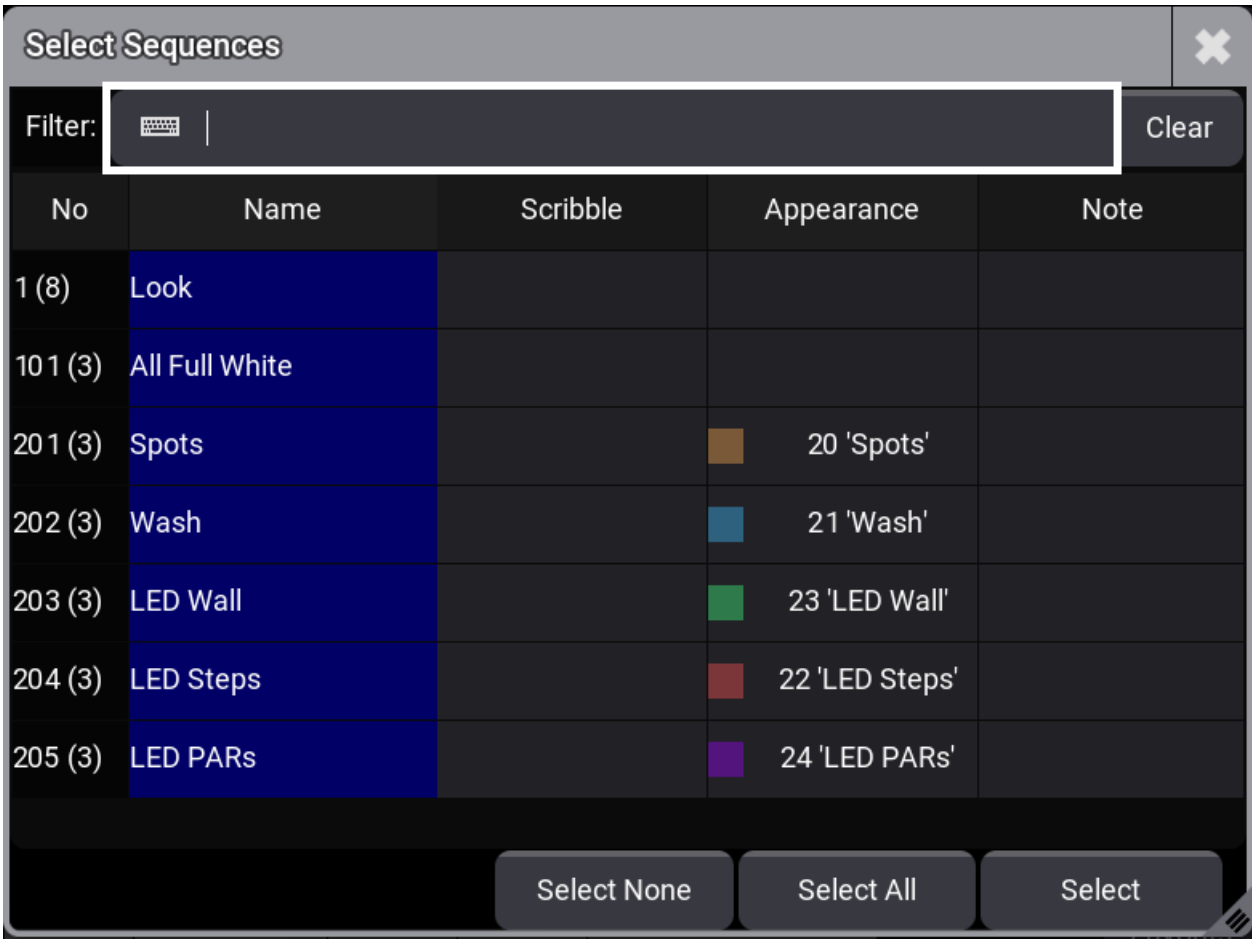
- Green: All object items will be cloned.
- Yellow: Only selected object items will be cloned. The Clone Show button changes to Clone Selected Objects.
- Gray: No object items will be cloned.

	Hint:
	If an object does not contain any items, the object's indicator bar will be colored in gray.

To restrict the cloning process:

1. Tap an object button., for example, All Sequences. A pop-up opens.
2. Select a Sequence and then tap Select. The pop-up closes.

The cloning process has been restricted to the selected object.



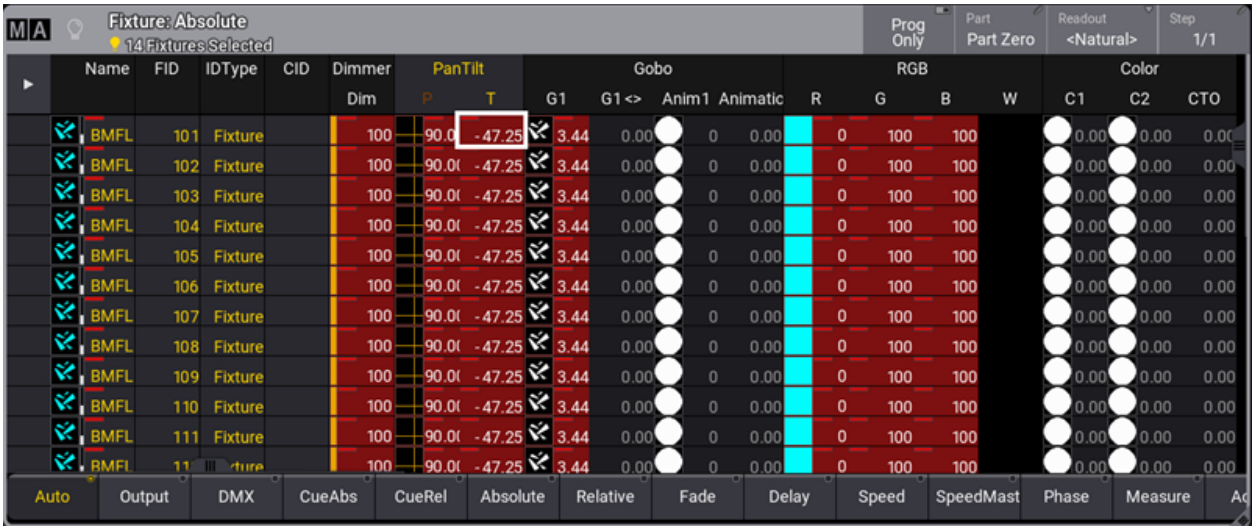
Clone - All Sequences pop-up

- To disable all items for a specific object, tap Select None in the pop-up.
- To enable all items for a specific object, tap Select All.
- To select all items for all objects, tap All Items To All in the Object bar. Tap Clone Show to execute the clone command.

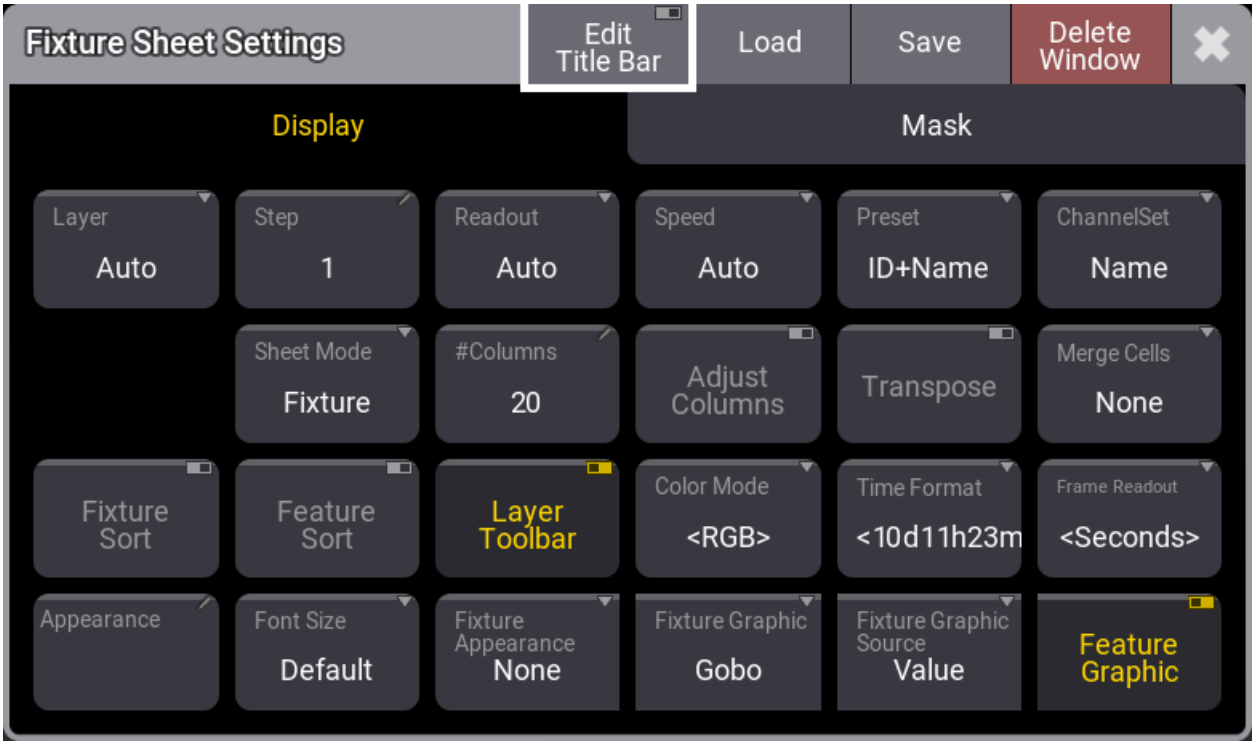
1.15.5. Graphics

Feature Graphic

The attribute values of the **Dimmer**, **Position**, **Gobo**, and **Color** features can be displayed with a small symbol graphic representing the attribute values.



- Tapping Feature Graphic in the Fixture Sheet, the Content Sheet, and the Sequence Sheet settings window will show or hide these symbols in the corresponding sheet.

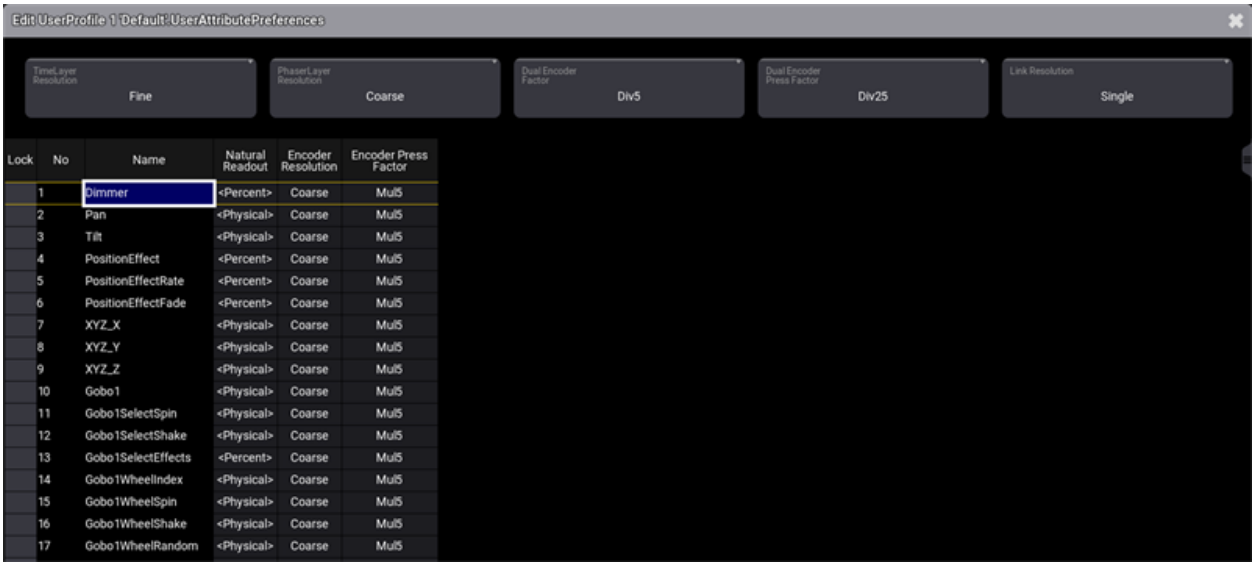


1.15.6. Encoder Resolution

Sometimes precise handling is required, and you may want to change the resolution of an encoder.

The default encoder resolution of attributes can be defined from the user configuration menu:

1. Press Menu, tap Settings, and tap User Configuration. The user configuration menu opens.
2. On the left side of the window, tap Profiles.
3. Tap Edit Encoder Bar. The Default User Attribute Preferences menu opens.



It is also possible to change the encoder resolution directly on an encoder.

To do so:

1. Press and hold MA, this will display the possible resolutions in the channel function area of each encoder.
2. While holding MA, tap Link Resolution and select **Single** or **FeatureGroup.** Then tap the resolution area of an encoder; the encoder's selected resolution will change to the next resolution.
3. When the desired resolution is selected, release MA.

	Hint:
	- For one revolution, an encoder has 24 clicks.

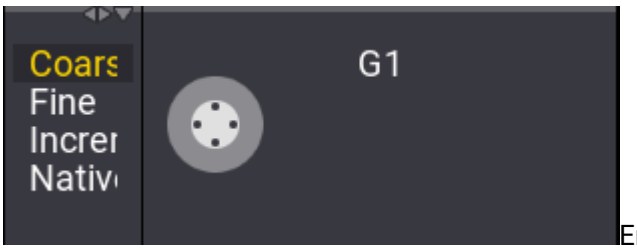
	- 5 turns of an encoder are needed to cross the whole range of the attribute from its minimum to maximum.

The possible resolutions are:

1. **Coarse**: One encoder turn click will change the value depending on the readout:
    - Percent: 1
    - PercentFine: 1
    - Physical: (MaxValue-MinValue)/ (24*5)
      When multiple fixture types are selected, the smallest physical range of a fixture is taken to determine the size of one click. This allows having the same value change when for example, turning the tilt encoder.
    - Dec8: 255/(24*5), therefore 1 click equals 2.125
    - Dec16: 65 535/(24*5), therefore 1 click equals 546.125
    - Dec24: 1 677 216/(24*5), therefore 1 click equals 13 976.8
    - Hex8: 255 (=FF)/(24*5), therefore 1 click equals 2.125
    - Hex16: 65 535 (FFFF)/(24*5), therefore 1 click equals 546.125
    - Hex24: 1 677 216 (=FFFFFF)/(24*5), therefore 1 click equals 13 976.
2. **Fine**: Fine has a 10x finer resolution than coarse.
3. **Increment**: When the resolution is set to Increment, one encoder turn click will change the lowest digit of the displayed readout.
4. **Native**: The value Layers absolute and relative offer this mode to have direct access to the smallest possible value change of the parameter resolution.

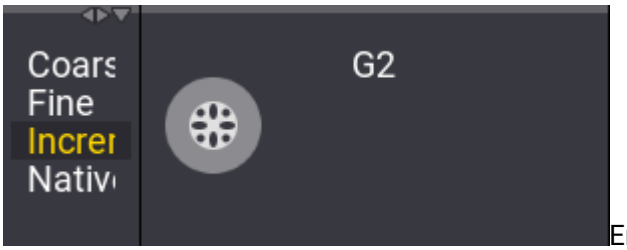
	Hint:
	When Native mode is selected, if a dimmer channel is based on 8/16/24 bit, one encoder turn always results in a one digit change in DMX output.

An encoder displays the current resolution in the center of the encoder symbol within the encoder bar:

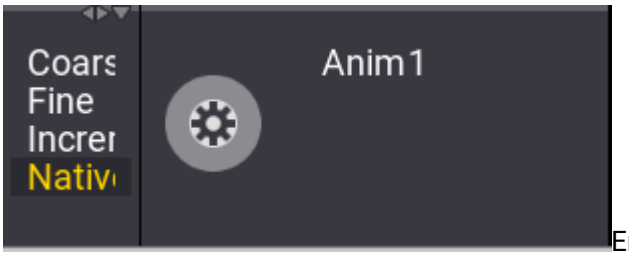


Encoder resolution coarse

Encoder resolution fine



Encoder resolution increment



Encoder resolution native

Furthermore, the factors of value change between turning the inner encoder and all other encoder actions can be defined. See **User settings** for more information.

All factors can be selected from a list of predefined factors:

Predefined Factors	Background Used Factors
Div50	0.02
Div25	0.04
Div10	0.1
Div5	0.2
Div2.55	0.39
Div2	0.5
One	1
Mul2	2
Mul2.55	2.55
Mul5	5
Mul10	10
Mul25	25
Mul50	50
Disable	0

1.15.7. Special Dialog

The Special Dialog window provides a user-friendly way to control attributes within a feature group in a single window. These attribute functions can be accessed through different tabs in the Special Dialog window. To learn more about attributes and feature groups, see **Attribute Definitions** and **Feature Group**.

The Special Dialog window can be found in the Add Window dialog - Common / Tools - Special Dialog.

For more information about adding windows, see **Add Window**.

Tab on the following video to get an overview of the Special Dialog and the Settings.

The Color tab and the Shapers tab on the left side in the Special Dialog window can be used to switch between the different functions. In the Special Window settings, these tabs can be shown or hidden by toggling Tabs. For more information about window settings in general, see **Window settings**.

The Special Window settings are separated into three tabs: Display, Color, and Shapers.

Display Settings

- Tabs**:** Enables/disables the tabs on the left side of the Special Dialog window.
- Tab**:** Tap to switch between the different Special dialog windows.
- Link Encoder Bank**:** Links an encoder bank to a specific Special Dialog window.

	Hint:
	Pressing MA in the title bar opens the settings of the currently open tab.

To learn more about the specific functions of the Special Dialog window, read the following topics.

Subtopics

- **Using the Color Picker**
- **Using the Shapers Dialog**

1.15.7.1. Using the Color Picker

The **Color Picker** is part of the Special Dialog window and can be selected via a tap on the left side of the window or via the window settings. For more information, see **Special Dialog**.

	Important:
	Color Picker windows in show files created in software version 1.9.7.0 or earlier are migrated to the Special Dialog window when loading the show file in software version 2.0 or later.

The Color picker can be used to select a color in fixtures with a mixed color system or color wheel(s).

The color picker provides convenient access to mix the desired color using several color mixing and selection options. The method is independent of the fixture's color mix system (LED emitters or color subtraction) and color wheel.

Open the Color Picker

Open a Special Dialog window. See **Add window**.

Tap Color on the left side of the window.

RGB and HSB color space:

Tap Color Space in the title bar of the color picker to switch the color picker's color space. There are four options:

- **Fixture Type:** The color space is defined by the emitters of the respective fixture type.
- **Standard:** Plasa Standard E1.54 for Color Communication in Entertainment Lighting.
- **Rec.2020:** ITU-R BT.2020 or Rec. 2020 is an audiovisual industry standard for ultra-high definition (UHDTV).
- **Rec.709:** ITU-R BT 709 or Rec. 709 is an audiovisual industry standard for high definition (HDTV).

Tap MA in the top left corner of the Special Dialog window to open the settings.

Special Dialog Window Settings - Color

The Mode can be selected using the buttons in the title bar. This is a short description of the different modes.

- **CIE:** A CIE color space area picker with Brightness, Quality, x, and y on-screen faders.
- **HSB:** An HSB area with Brightness and Quality on-screen faders.
- **Fader:** On-screen faders to adjust RGB, CMY, HSB, Brightness, and Quality.
- **Book:** This is a swatch book with colors from different gel manufacturers.

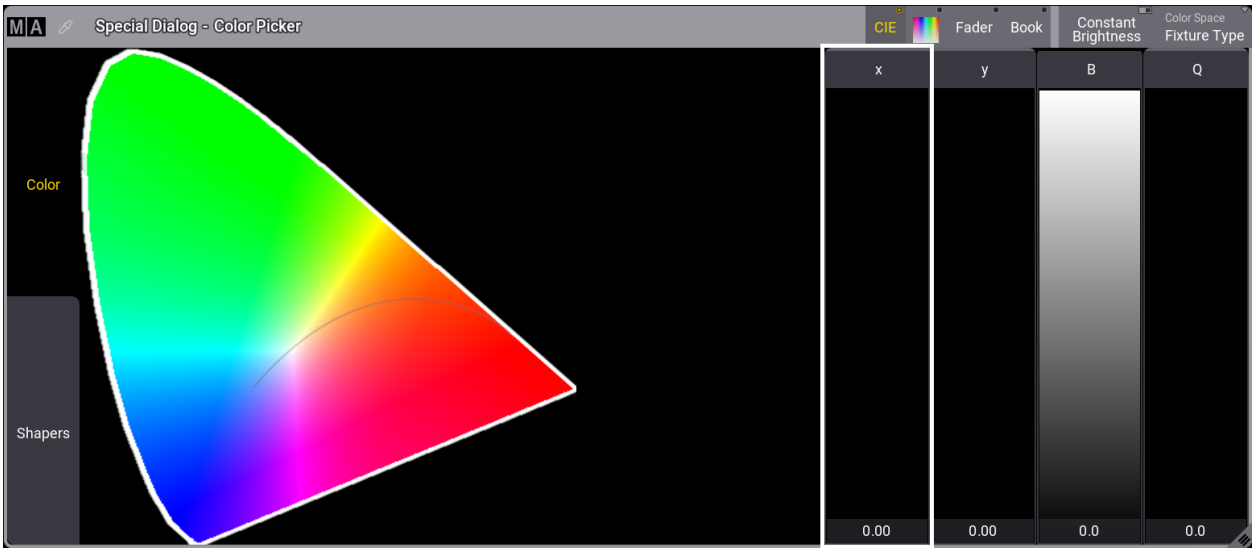
Tap Edit Title Bar to define which buttons will be displayed in the title bar of the color picker window.

Special Dialog Window Settings - Edit Title Bar

Quality

The Q fader or Quality fader is available when the fixtures have a color mix system of more than three colors. It controls how the colors are mixed.
Q at 100 results in a kind of small-band mixing (the specialized emitters are used). 0% results in a broadband mix. That uses as many emitters as possible to mix the color.

CIE



Special Dialog - Color Picker CIE mode

The CIE (Commission Internationale de l'éclairage) standard uses a figure that indicates the visible light spectrum.

The RGB triangle shows the colors that the specific fixture can mix.

Requirements:

Fixtures with color attributes are selected.

- To select a color, tap inside this area.

If the fixture has more than 3 mixing colors, the shaded part will become smaller.

	Hint:
	The **Align** function can be used together with the color picker.

Tap the video below to see the example.

The color picker offers the Constant Brightness Mode, which can be enabled by tapping Constant Brightness in the title bar. The default setting is off.

If Constant Brightness is disabled, the selected color is mixed with maximum brightness when the brightness fader is set to 100%. In this case, the fixture's output intensity is not kept constant but changes with the color. If the constant brightness mode is enabled, the maximum overall brightness is limited to the brightness of the fixture's darkest emitter. Changing the color in constant brightness mode does not change the output intensity of the fixture.

	Hint:
	When enabling the constant brightness mode while the brightness fader is currently positioned above the constant brightness color mixing range, the CONST B fader will become red, showing a value of > 100 %. To ensure constant brightness color mixing, the fader needs to be moved to <= 100 %.
	Hint:
	Color mixing and constant brightness mode work better the more accurate the fixture type's emitter data is.

Except for the Fixture Type color space, the gamut of the selected color space is displayed in the CIE color picker with a white line. The shaded area only depends on the fixture's emitters. It does not change with the Color Space (only the small white triangle changes with the selected color space). Color mixing in the RGB tab and the HSB Color Picker depends on the color coordinates of the RGB primaries of the selected color space.

	Hint:
	If a color is picked in the CIE Color picker outside of the gamut of the selected color space, the faders in the RGB tab will show values below 0% or above 100%.

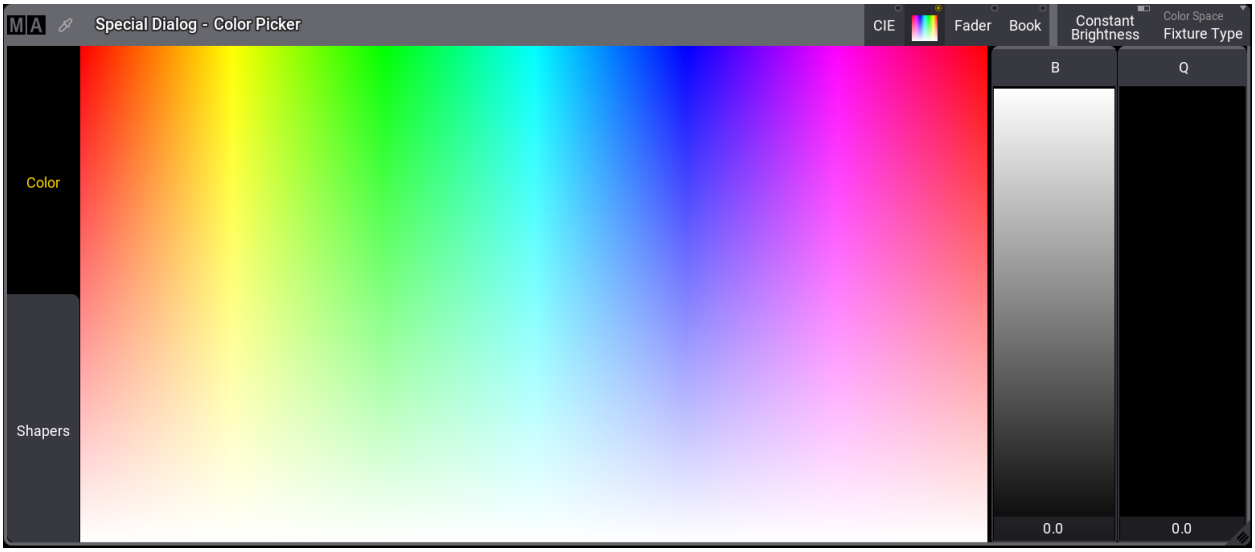
The CIE Color Picker displays the spectral profile (or **curve**) at a specific temperature corresponding to a specific peak wavelength and vice versa. As the temperature of the black body increases, the peak

wavelength decreases (Wien's Law). The intensity (or flux) at all wavelengths increases as the temperature of the black body increases. That is what we call the **black body curve**.

HSB Field

To adjust the color mix, tap the HSB field symbol in the title bar, also known as a Color Picker.

Here it is possible to tap a color in the HSB field. The x-axis (left/right) is the Hue value. The y-axis (up/down) is the Saturation value, and Brightness is the B-fader on the right side.

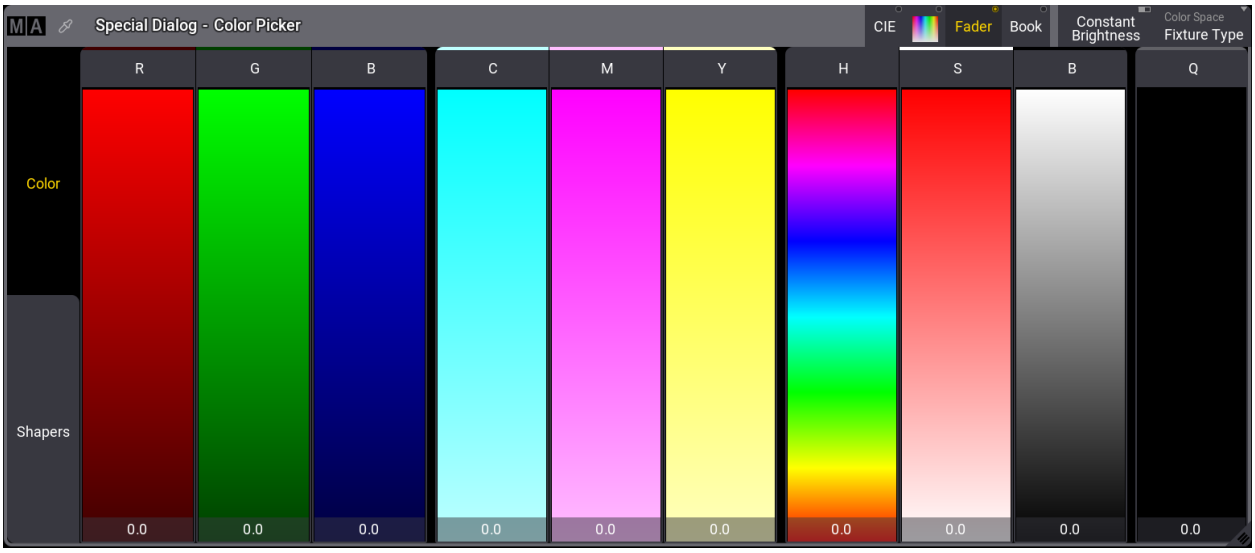


Special Dialog - Color Picker HSB mode

Fader

On-screen RGB, CMY, HSB, Brightness, and Quality faders.

Here it is possible to adjust the colors using RGB, CMY, or HSB. The Three-color systems are interlinked. This means adjusting the colors in RGB also moves the CMY and HSB faders.



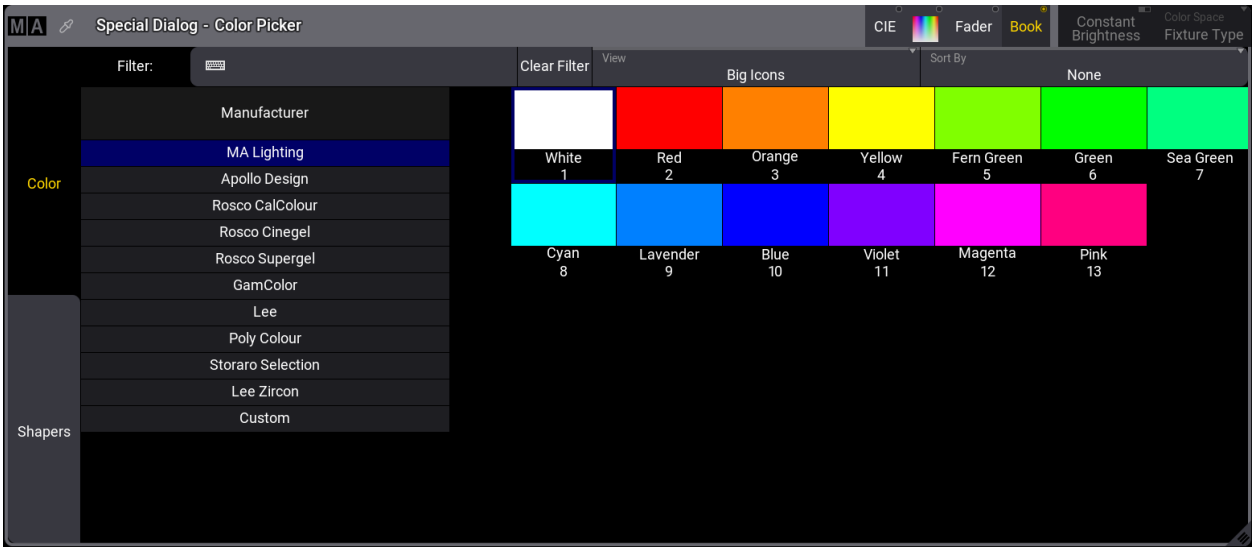
Special Dialog - Color Picker Fader mode

Book

The swatch book is a library of gel colors from different manufacturers. This can also be accessed using the **Gel keyword**.

The manufacturers are displayed on the left and the gels are displayed on the right.

For more information about gels, see the **Gel topic**.



Special Dialog - Color Picker Book mode

There are three buttons at the top of the window.

- Filter is to enter a name or a key of the color to limit the list of gels. In addition, Clear Filter resets the filter.
- View defines how the gel list is displayed. It is a swipe button with the following options:

    - **Big Icons:** This displays a big color example. Below the color is the name and color key.
    - **List:** This displays a list with Name, Key, and Color columns.
    - **Small Icons:** Displays a small color field only.
- Sort by has three different ways to sort the displayed gels. It has the following options:
    - **None:** The list is not sorted and displayed in the library's order.
    - **Key:** Sorted by the color key number.
    - **Name:** The colors are sorted by the color name.

The manufacturer list includes many of the manufacturer's gels. It also includes MA Lighting. The MA Lighting library contains all the primary and most used colors.

Color Wheels

When working with fixtures with color wheels, the software chooses the color as close as possible to the picked color from the color picker. When the fixture has two or more color wheels, the software will use all wheels (also combined) to get a color as close as possible.

To change the wheel mix mode, tap MA in the title bar. The Color Picker Window Settings opens.

There are three different Wheel Mix Modes available:

- **Mix Color Only:**
  Uses only color-mixing engine attributes, for example, RGB.
- **Color Wheel Only:**
  Uses only color wheel attributes.
- **Prefer Mix Color:**
  Uses mainly the color mixing engine attributes. If needed, the color wheels will be used as well.

	Restriction:
	When a fixture type has more than one color wheel, the color picker can only handle up to 255 color combinations across the color wheels.

The **Fixture Sheet** shows the selected color for the individual wheels. The combined color output is displayed in the sheet.

1.15.7.2. Using the Shapers Dialog

The Shapers dialog is part of the Special Dialog window and provides a graphical user interface for controlling shaper attributes in fixtures. The values that are used to visualize the shapers come from the DMX layer. For more information, see **Fixture Sheet** and **Layouts**.

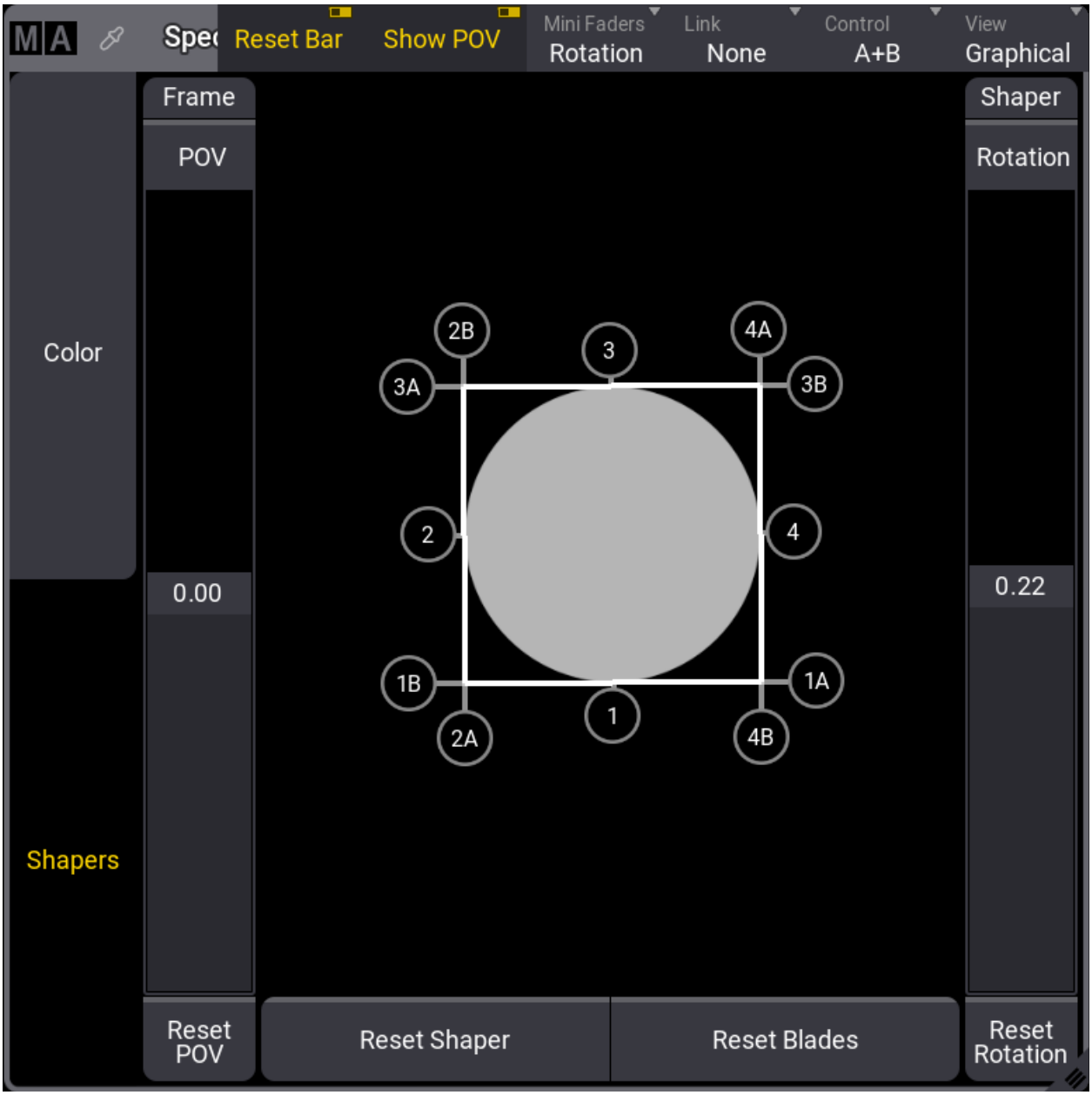
Watch the following video to get an overview of the Shapers dialog feature.

Requirements:

- A fixture type with a shaper is selected in the programmer, for example, Ayrton Eurus. For more information, see **Import Fixture Types** and **What is the Programmer**.

To open the Shapers Dialog window:

1. Open the Add Window dialog.
2. Tap Common and Special Dialog. The Special Dialog window opens.
3. Tap the Shapers tab on the left. The Shapers dialog opens.



Shapers Dialog window

The title bar of the Shapers Dialog provides the following options for controlling blades and shapers of fixtures. For more information about how to edit the title bar, see **Title Bar Configuration**.

- **Reset Bar:** When enabled, tapping Reset Shaper resets blades and rotation. Tapping Reset Blades resets the blades only.
- **Show POV:** Displays a fader to manually rotate the Point of View (POV) of the fixture. When enabled, a Reset POV button is also displayed.
- **Mini Faders:** Changes the number of faders that are displayed. Can be accessed by tapping and holding Mini Faders.
    - None: No faders are displayed.

- Full: The faders for blades and rotation are displayed depending on the value of Control.
- Blades: The faders for blades are displayed depending on the value of Control.
- Rotation: A fader for the shaper rotation is displayed.

- **Link:** Defines how the movement of one or more blades works together.
  - None: Each blade can be controlled individually.
  - All: Moving one blade moves all other blades proportionally.
  - Parallel: Moving one blade will also move the opposite blade.
  - Mirrored: Moving one blade will mirror the opposite blade.
- **Control:** Changes the control mode.
  - Ins+Rot: The depth of insertion (1-4 -"I") and the blade rotation (1-4-"R") can be defined.
  - A+B: The blade corners (A,B) can be defined.

	Hint:
	The software converts different types of shaper engines to the correct DMX values so that all modes can be used regardless of the fixture type. Blade misconfigurations are shown in the **Conflicts in Fixture Types**.
	Important:
	Depending on the fixture type, there may be inaccuracies when converting from one control mode to another.

- **View:** Changes the display mode.
  - Graphical: Shows a shaper visualisazion.
  - Faders: Shows only faders.

When the focus is on the Shaper Dialog, the encoder bar displays the Shaper Encoder Bar. Depending on the selected control mode, the encoders display the corresponding functions.

Encode Bar - Shaper Dialog

Graphical View Mode

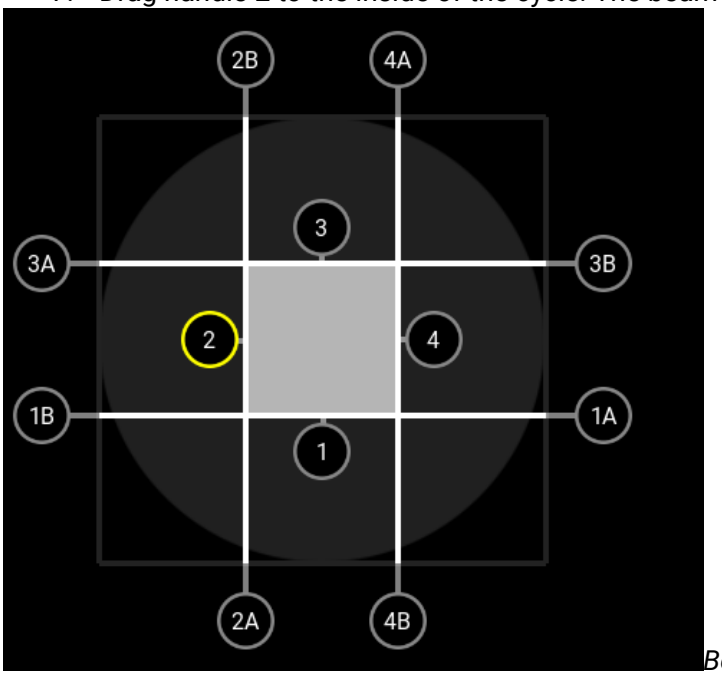
The graphical view mode shows a visualization of a fixture beam. This beam is colored light gray.

The insertion of blades can be changed with the movable handles. Handles labeled 1 through 4 are blades. Handles labeled with additional A and B are corners of the blades.

Four white lines show the actual insertion into the beam. The covered part of the beam is shown in dark gray.

Example

1. Patch a blade fixture, for example, Ayrton Eurus.
2. Select the patched blade fixture.
3. Open the Shapers Dialog window.
4. Tap and hold Link in the title bar. A dropdown menu opens.
5. Select All. The dropdown menu closes.
6. Tap and hold handle 2. Handle 2 turns yellow.
7. Drag handle 2 to the inside of the cycle. The beam becomes a square.



Beam in a squared shape.

	Hint:
	Tap and hold anywhere in the visualization to select the nearest handle.

Using the Faders

With faders, it is possible to change the insertion of the blades.

Values from 0.00 - 1.00 define the insertion of the blade. 0.00 does not cover the beam and 1.00 covers the beam completely.

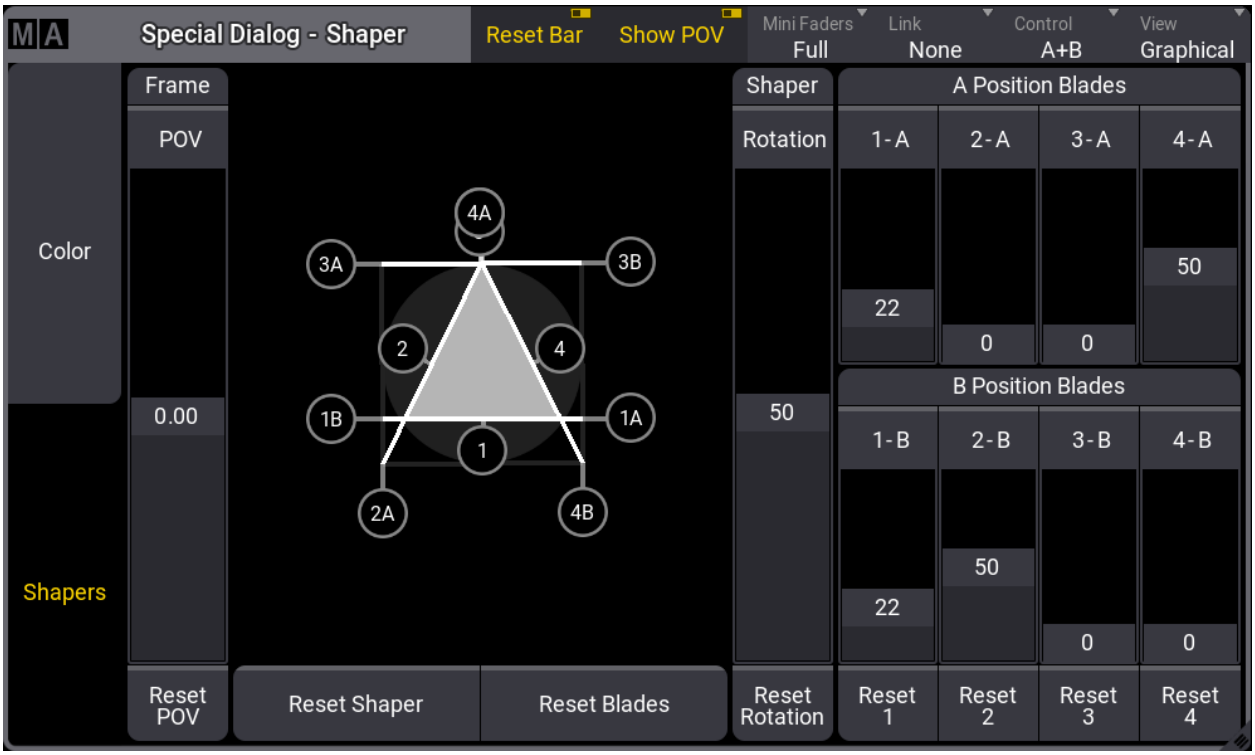
To reset single blades, Reset 1-4 can be used. To reset the rotation of the shaper unit, Reset Rotation can be used.

Example:

To create a triangle shape:

1. Select a blade fixture.

2. Tap and hold Mini Faders in the title bar of the Special Dialog window. A dropdown menu opens.
3. Select Full. The dropdown menu closes.
4. Tap and hold Control.
5. Select A+B in the dropdown menu.
6. Drag the faders to the following values: 1-A 0.22, 4-A 0.50, 1-B 0.22, 2-B 0.50.
7. A triangle shape is created.



Shaper Dialog - Mini Faders

	Hint:
	The graphical representation of the Shaper Dialog may differ from the shaper of the real fixture due to the technical conditions of the fixture.
	Hint:
	The faders are grayed out for fixtures with less than 4 blades or fixtures without a full set of attributes to control a blade. The handles for missing blades are adjusted accordingly in the graphical view.

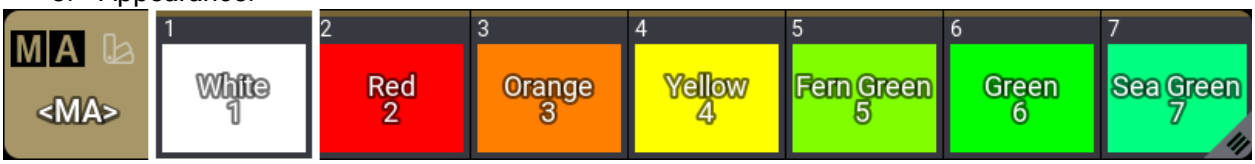
1.15.8. Gels Pool

The Gels Pool displays manufacturers and their corresponding gel series. Each gel series has different amounts of gel in its pool. Manufacturer gel pools are locked by default. Gels can be added and edited in a custom gel pool. For more general information about pools, see **Pool Windows**.

- If you want to know about the gel keyword, see **Gel Keyword**.
- The Color Picker uses a Book list with many filter options for gels. See **Using the Color Picker** for more information.

Each gel in the manufacturer pool displays:

1. Gel name.
2. Key catalog number.
3. Appearance.



Gels Pool

To open the Gels Pool:

- Tap Pools and Gels in the **Add Window pop-up**.

To select a specific manufacturer catalog:

1. Tap <MA> under the MA Logo in the top left corner. A dropdown menu opens.
2. Select a manufacturer. The dropdown menu closes and the corresponding gel pool is shown in the Gel Window.

Edit a Gel Color

The following video shows an example of editing a color using the Custom Gel Pool:

Press Edit and tap a gel pool object to open the gel editor.

The Gel Editor is divided into two areas, the **Label** area (left) and the **Gel** area (right).

Gels Pool - Editor

To edit the gel color to green:

1. Tap Color. The color editor opens.
2. Set the RGB faders to **000/100/000**. The color has changed to green.

Gels Pool - Color Editor

3. Tap Ok. The editor closes and the gel color has changed.

Custom Gel Pool

The Custom Gel Pool allows you to store gels according to your personal preferences. In the custom gel pool, you can create your own gels or copy from existing manufacturer gel pools.

To copy gels in the custom gel pool:

1. Open a manufacturer gel pool and a custom pool next to each other.
2. Use the copy function in the swipey command. For more information, see Swipey Commands in **Pool Windows**. The gel is copied into the custom gel pool.

To edit a gel in the custom pool:

1. Do the Swipey Command on the corresponding gel pool.
2. Swipe to Edit. The editor opens.
3. Edit the pool object.
4. Tap X. The editor closes and the gel has been edited.

To store the color of the last selected fixture as a new gel pool object:

1. Open the Custom gel pool.
2. Press Store and then tap an empty pool object in the gel pool. The color is stored as a new gel pool object. The pop-up closes and the gel is stored.

Gels Pool Settings

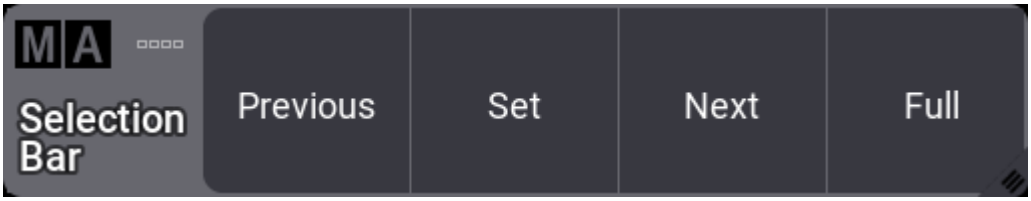
To open the Pool settings, tap MA in the top left corner. The Pool Settings pop-up opens.

- **Show Empty**: This toggle button can hide or show empty pool objects.
- **Appearance**: The appearance is applied behind the pool objects.
- **Pool Columns**: This defines the width for the pool objects. It does not change the size of the window. It defines how many columns of pool objects are in the window. If the window is wider than the number of columns, then the extra space is displayed as black (default color). If the window is smaller than the number of columns, the pool window can be scrolled horizontally. If the pool has a set width, then there is an icon (⏭) in the upper right corner of the title field.
  The **Not Defined** value dynamically sets the width to match the window size even when the window is resized.
  The **Take Current Width** sets the width to match the current size of the window. It does not dynamically change if the window is resized.
- **Font Size**: There are some different font size properties from 10 to 32. There is also a default property. This is the same as size 18. This simply changes the font size on the pool objects.
- **Pool Color**: This is the color for the title button in the pool.
- **Empty Color**: This color is applied to empty pool objects.
- **Reset Colors**: This resets the colors to the colors in the default color theme.

1.15.9. Selection Bar

The selection bar is a quick toolbar that provides access to some of the most commonly used hard keys for selecting fixtures.

It is a window that can be created like any other. Learn how in the **Add Windows topic**. It is found in the **More** tab.



Selection Bar

This window can be half field height.

To change the appearance, tap the MA logo in the title bar.

There are four buttons in the Selection Bar:

- **Previous** - Select the previous fixture.
- **Set** - Resets the MAtricks.
- **Next** - Select the next fixture.
- **Full** - Gives the selected fixtures a full intensity value on the dimmer.

These soft keys behave just like the hard keys. Follow the links to learn about them.

They are also handy when working with **MAtricks**.

1.15.10. Align

There is a dedicated Align key, an **Align Keyword**, and an **Align Bar** that can be created as a window. Align is accessible on the left side of the encoder bar.

Align is used to distribute attribute values between two or more values. There are five different align modes and Off. Read about them below.

If align is used frequently, this is a convenient window. Read the descriptions below to understand the different modes.

The default is a linear transition between the values, which can also be adjusted. There are four different **Align Transition** options:

- **Linear** (default):
  Spreads the values with the same spacing.
- **Sinus**:
  Spreads the values as if the fixtures were placed on a sinus curve. The values themselves will not represent the sinus form. Depending on the Align mode, this results in smaller value gaps at the beginning and end of the range and bigger gaps towards the center of the range, or vice versa.
- **Slow**:
  The gaps between the values will be small at the beginning of the range and increase toward the end of the range.
- **Fast**:
  The gaps between the values will be big at the beginning of the range and decrease toward the end of the range.

These can be accessed using the **AlignTransition keyword**, the **Align key**, or the **Align Bar**.

The Align function can be used for different attributes. Dimmer, position, and color are the most common. The examples below use tilt or dimmer attributes.

By default, the align mode is set to Off, and the transition is Linear. The result is the encoder will adjust all the selected fixtures equally.

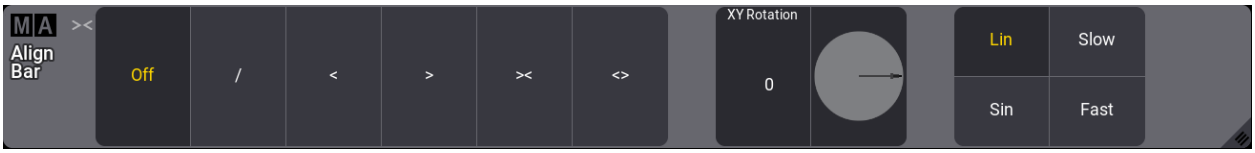
	**Important:**
	The selected order of the fixtures is important. The attribute will be adjusted proportionally to the selected order.

The align mode is active until a new attribute is adjusted.

Align Bar

The **Align Bar Window** gives fast access to all the align functions. It has both the align modes and the align transitions.

It can be created like any other window using the **Add Window pop-up**. It can be found in the **More** tab.



Align Bar window

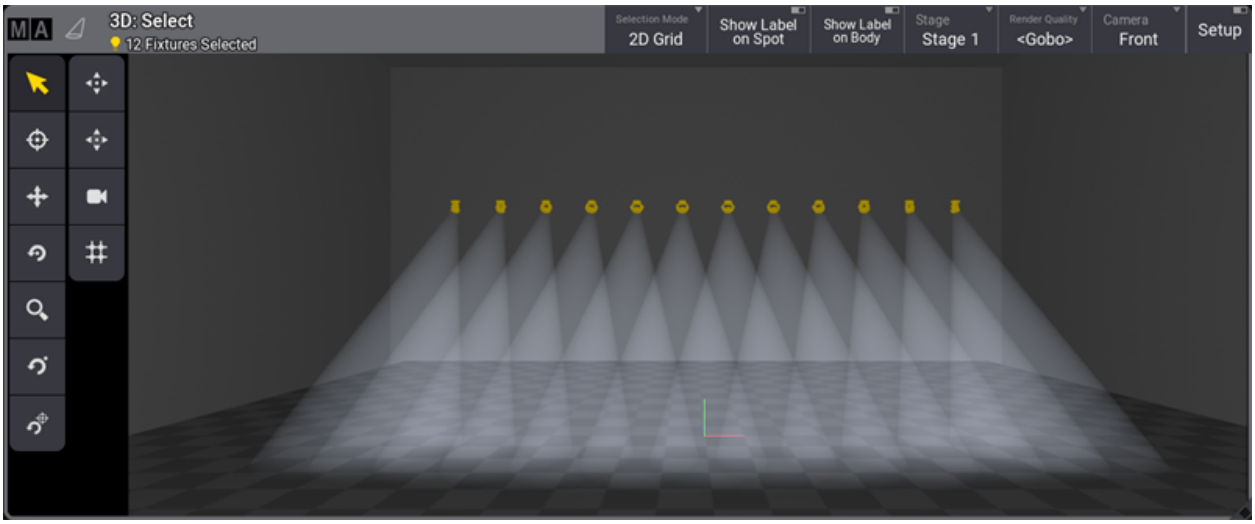
Align Bar window half-height

For the following examples, fixtures 1 thru 12 were patched from left to right. For more information, see **Patch and Fixtures Setup**.

Align /

This mode is great for fanning fixtures. Follow the steps below to create a linear fan out.

1. Select fixtures 1 thru 12.
2. Set the dimmer to full.
3. Tilt the fixtures up using the encoder wheel.
4. From the Encoder bar, tap Position.
5. Tap and hold Align on the left side of the encoder bar**.** This will open the align dropdown menu, then select **/**.
6. Turn the **Pan** encoder wheel clockwise to fan the fixtures out and counterclockwise to fan the fixtures in.



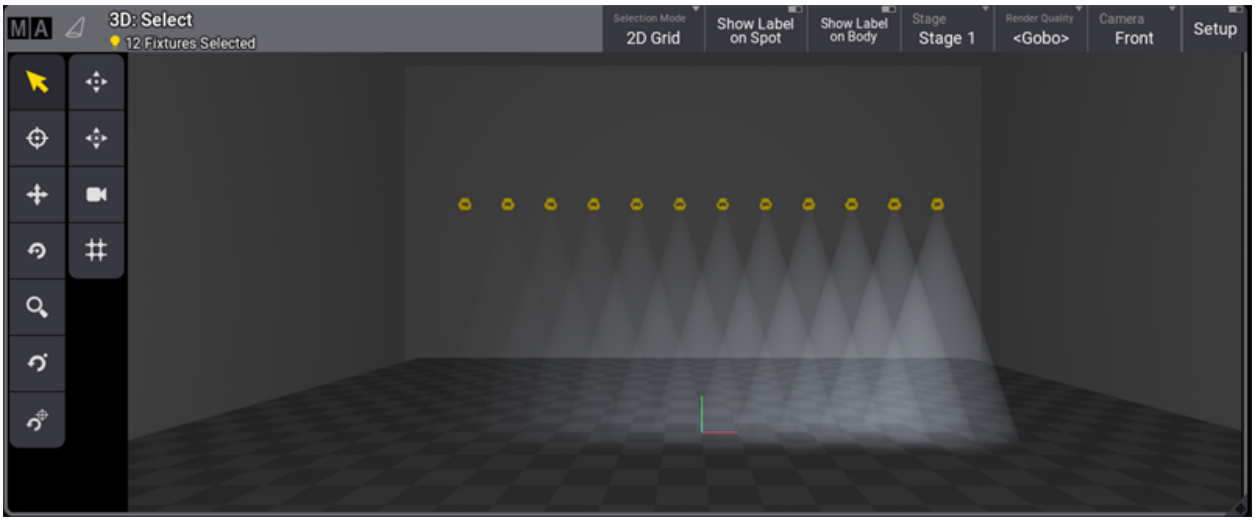
Align mode / with pan

Align <

This align mode increases the values from the first selected fixture.

1. Select fixtures 1 thru 12.
2. From the Encoder bar, tap Dimmer.

3. Tap and hold Align on the left side of the encoder bar**.** This will open the align dropdown menu, then select **<**.
4. Turn the **Dimmer** encoder wheel clockwise.

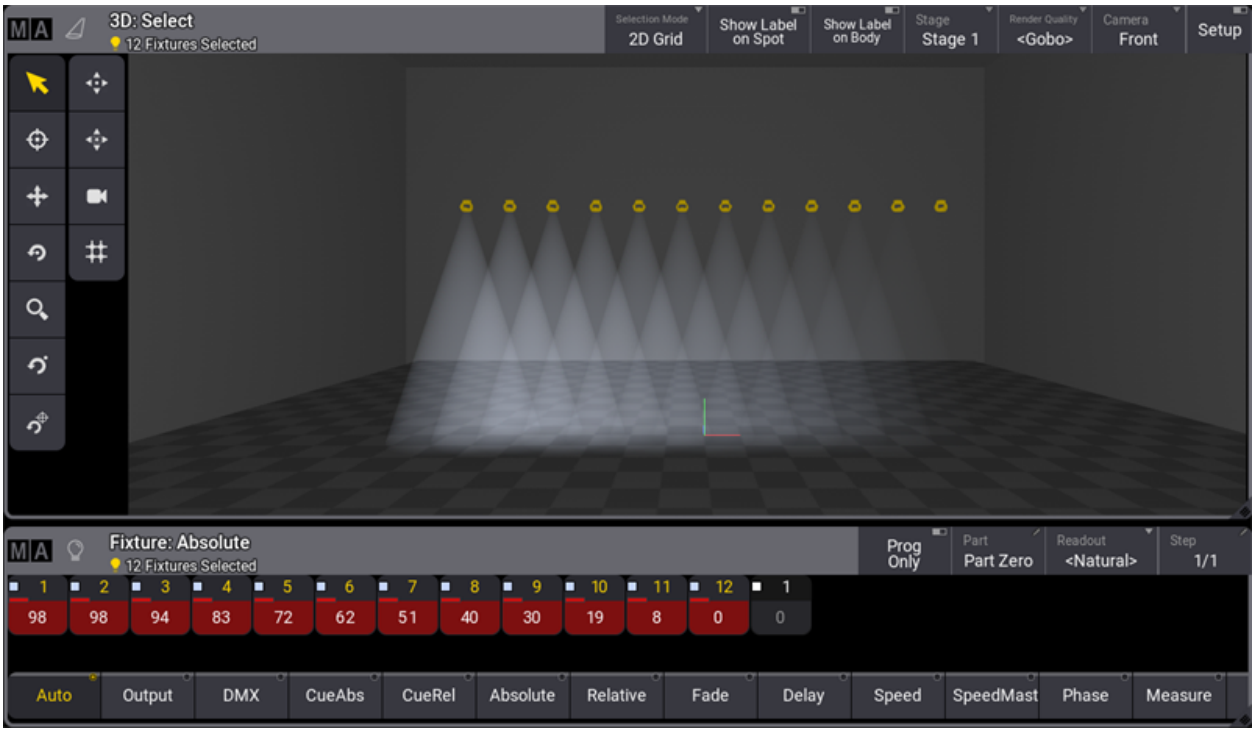


Align mode <

Align >

This align mode increases the values from the last selected fixture.

1. Select fixtures 1 thru 12.
2. From the Encoder bar, tap Dimmer.
3. Tap and hold Align on the left side of the encoder bar**.** This will open the align dropdown menu, then select **>**.
4. Turn the **Dimmer** encoder wheel clockwise.

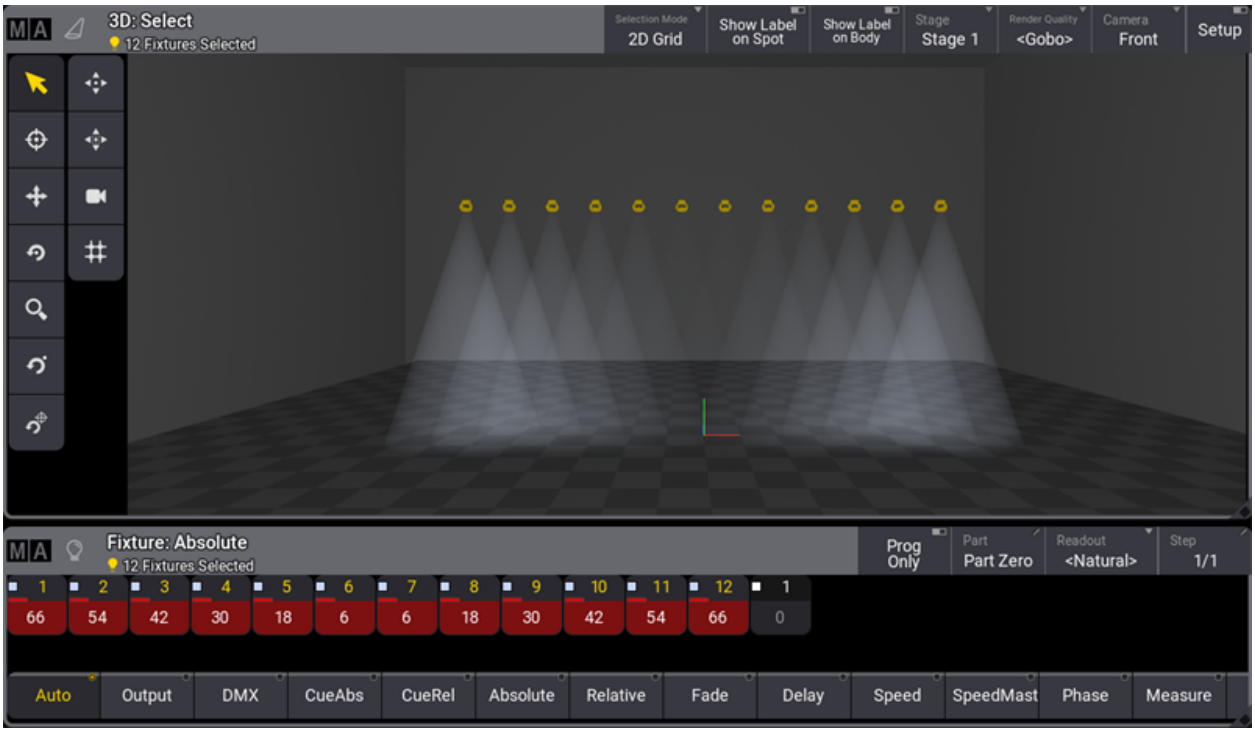


Align mode >

Align ><

This align mode Increases the values from the center of the selection.

1. Select fixtures 1 thru 12.
2. From the Encoder bar, tap Dimmer.
3. Tap and hold Align on the left side of the encoder bar**.** This will open the align dropdown menu, then select **><**.
4. Turn the **Dimmer** encoder wheel clockwise.

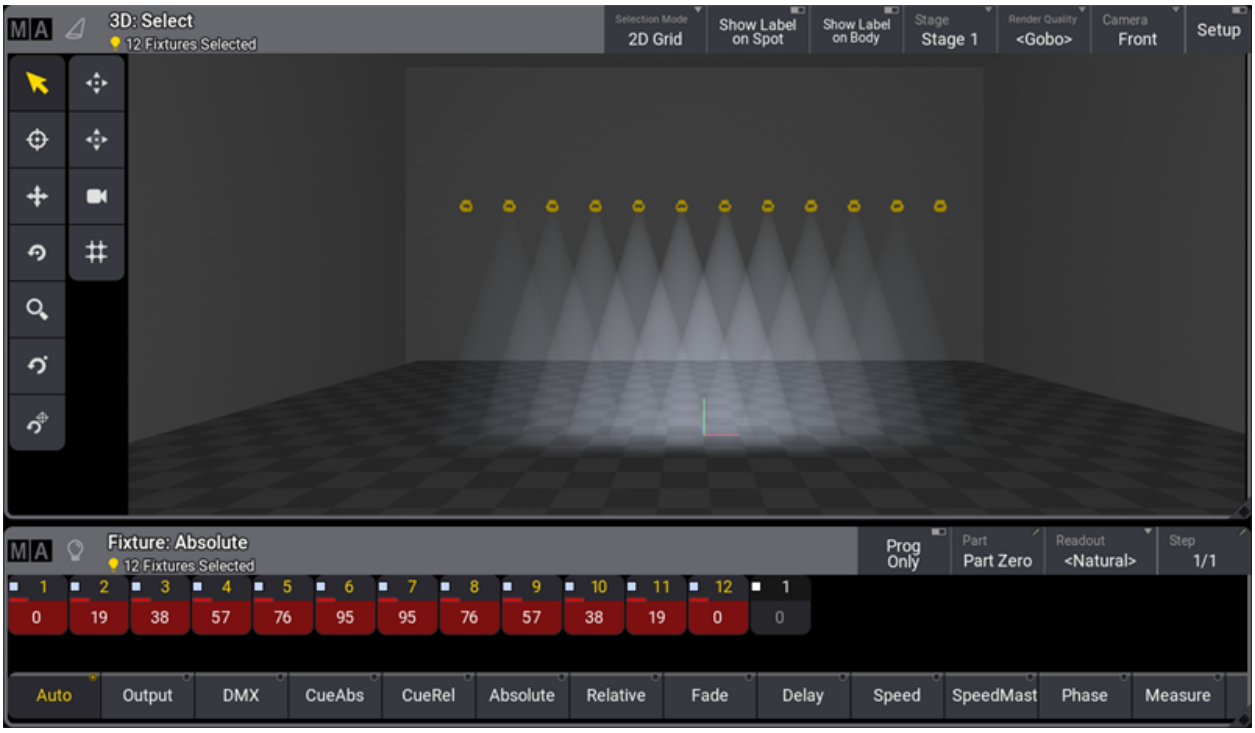


Align mode ><

Align <>

This align mode increases the values from the first end last fixture of the selection.

1. Select fixtures 1 thru 12.
2. From the Encoder bar, tap Dimmer.
3. Tap and hold Align on the left side of the encoder bar**.** This will open the align dropdown menu, then select **><**.
4. Turn the **Dimmer** encoder wheel clockwise.



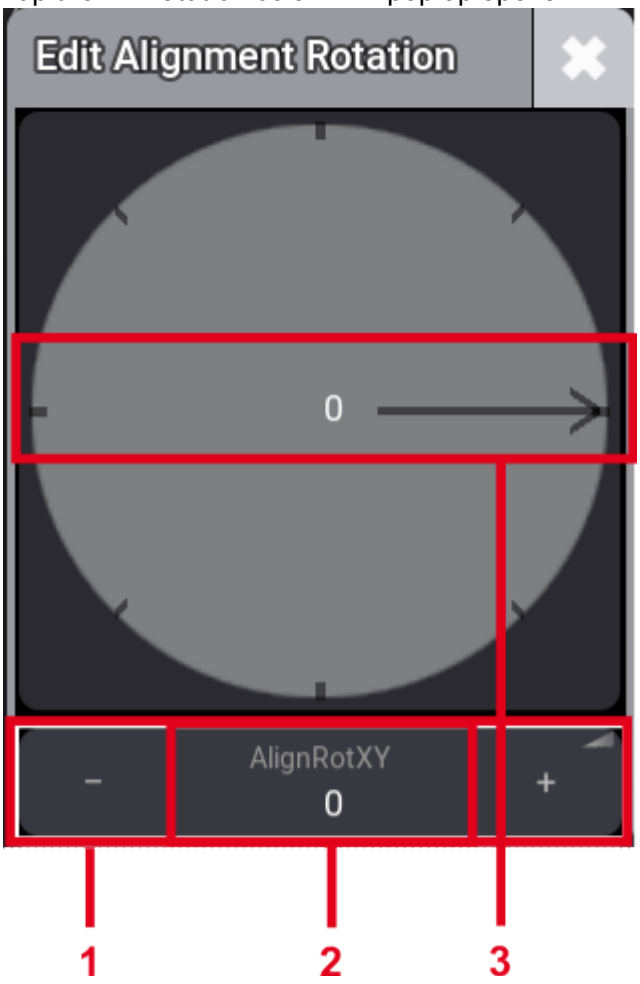
Align mode <>

XY Rotation

It is possible to set rotation values for an alignment, for example, across the selection grid. For more information, see **Selection Grid**.

To set the degree:

- Tap the XY Rotation column. A pop-up opens.



Pop-up - Edit Alignment Rotation

Choose one of the following options to change the value.

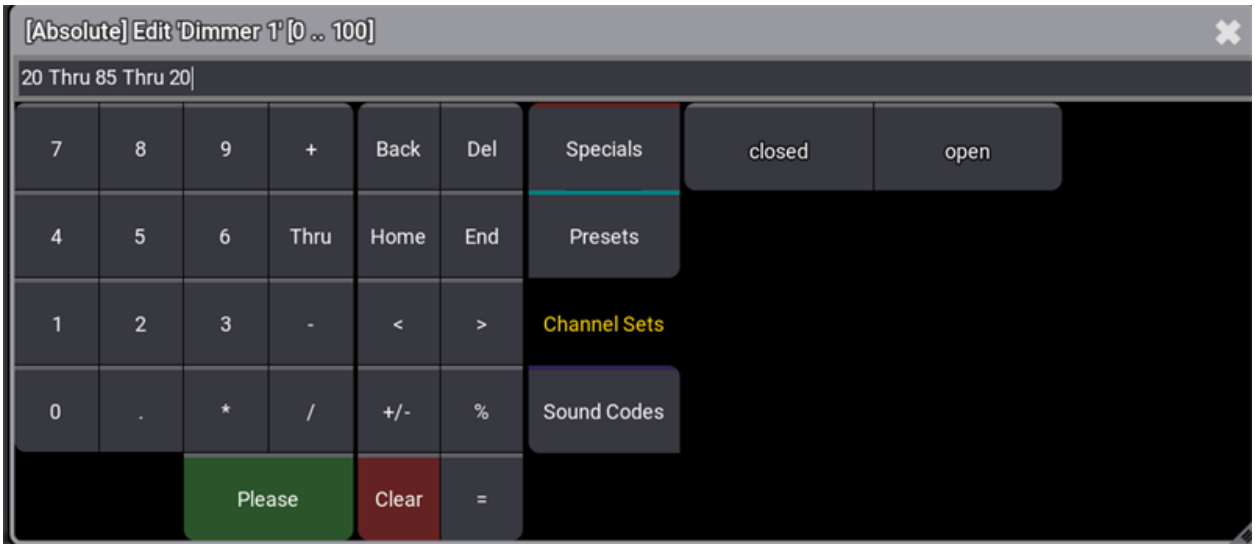
1. Drag and drop the on-screen fader.
2. Click **AlignRotXY** to open the calculator.
3. Tap, hold, and drag the arrow.

	Hint:
	Rotation can be set in 5° increments between 0° and 360°.

Align in Other Ways

Getting the same align functions with the Calculator or by the command line is possible.

Tap the encoder belonging to the attribute that shall be aligned, and the calculator appears. It is now possible to enter the align values like the example below.



Calculator

Another way to align attributes is from the command line. Select the fixtures and set the dimmer value, as in the example below.

```
User name[Fixture]>At 75 Thru 10 Thru 75
```

1.15.11. SMArt View

The **sMArt View** is a dynamic pool that gives quick access to fixture defined channel sets. The channel sets are a part of the fixture types. Depending on the fixture selection and the selected attribute in the encoder bar, the pool objects in the sMArt window will change. Each pool object represents a channel set defined in the fixture profile.

To open the window, use the **Add Windows** pop-up, then tap Tools and tap sMArt. The windows opens:

Smart View showing a gobo wheel

- Tap one of the objects to put the respective value into the programmer.

The bottom of the sMArt view shows an attribute selection bar. This makes it easy to select the desired attribute in the currently selected feature group. The example above shows the **Gobo** feature group, the selected **Gobo Wheel 1** (G1) attribute and the attributes G2, G2<>, Anim 1, Anim 1 FX, and Anim Pos.

- To enable or disable the attribute selection bar, open the **Pool Settings** and tap Show Bottom Menu.

Edit sMArt Object

To edit the channel set values for a corresponding pool object:

1. Edit a pool object. The Edit Fixture Type window opens.
2. Enable Settings in the title bar.
3. Change the values accordingly. For more information, see **Insert DMX Modes and DMX Channels**.
4. Tap ✖ to close the editor. The channel set values are changed.

1.15.12. Selection Grid

Fixtures can have information about their position in a 3D selection grid.

Each fixture is symbolized as a box using a space in the grid.

The grid organizes the fixtures in relation to each other but not necessarily their position in the 3D Viewer. A helpful tool for positioning fixtures in the selection grid according to their position in 3D space is called the 3D view to Selection Grid. For more information about this tool, see **3D Viewer**.

Tap a space on one of the screens to add the **Selection Grid** window. See **Add Window**.

Adjust the Selection Grid

The grid can be rotated by pressing the window with a single finger and moving the finger around the screen.

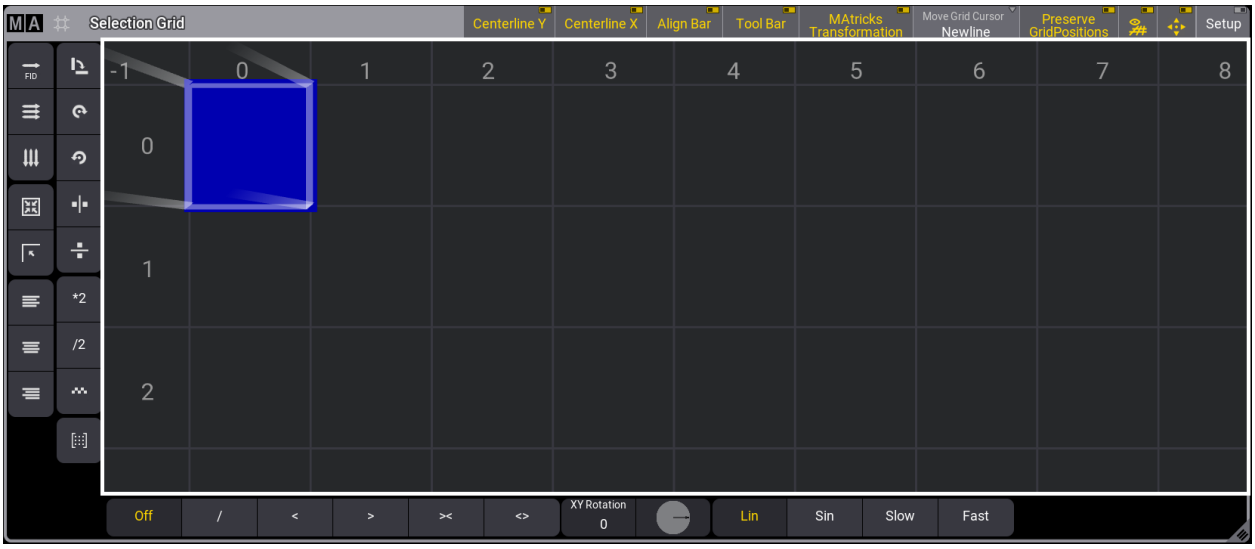
The grid can be zoomed using a pinch motion with two fingers on the touch screens or by scrolling the wheel on a mouse.

The grid can be moved around by touching the screen with two fingers and moving them around the screen. This can be done with a mouse by keeping the right mouse button pressed while moving the mouse.

Grid Cursor

Fixtures are positioned based on the position of the grid cursor. The grid cursor is the blue cell in the grid.

The grid origin, 0/0/0, is marked with a white frame.



Selection Grid window

When a cell other than the origin 0/0/0 is selected, Preserve GridPositions in the title bar will be enabled. When it's disabled, all gaps and offsets to the origin will be removed.
The state of Preserve GridPosition can be recalled if it is disabled before storing the selection into an object.

When selecting fixtures, the Move Grid Cursor setting in the title bar will work as specified in the user profile. For more information, see **User Settings**.

These are the three modes for the Move Grid Cursor.

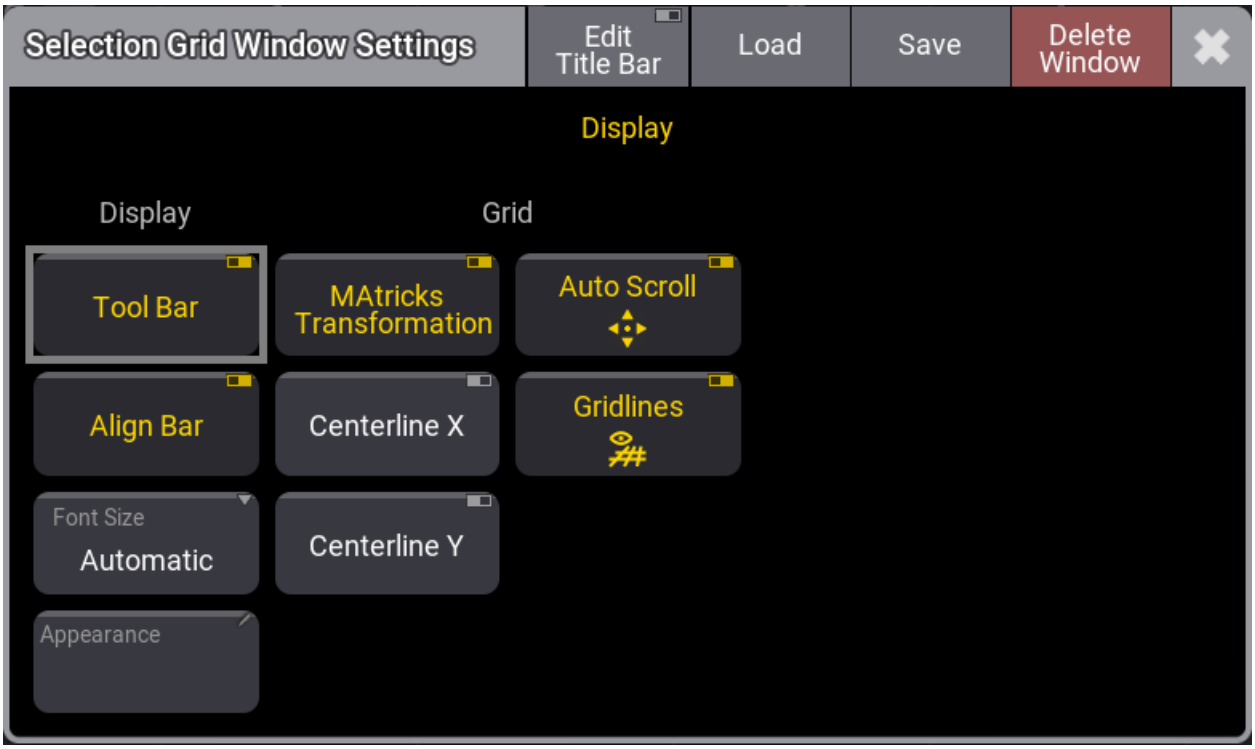
After a selection is made:

1. **None**: The cursor will stay at its position when the selection was made.
2. **Append X**: The Cursor will move to the next available X-axis cell on the grid.
3. **Newline**: The cursor will move at X=0 on the next line.

(Toggle grid lines): Toggles the visibility of the grid.

(Toggle view autoscroll): The window is reset to fit all the fixtures and the grid cursor in the selection grid window.

Tap the MA logo on the left side of the Selection Grid window's title bar to open the **Selection Grid Window Settings**.



Selection Grid Window Settings

Buttons in the display column:

- Tool Bar: Shows or hides the tool bar in the Selection Grid window.
- Align Bar: Shows or hides the align bar in the Selection Grid window.
- Font Size: This selects the font size in the window. It is a swipe button that opens a list of sizes from 10 to 32. There is also a **Default** property. The default is the same as size 18.
- Appearance: Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.
- MAtricks Transformation: Displays the fixtures stacked in the Selection Grid window when applying MAtricks. Enabled is the default value.
- Centerline X: Displays a vertical red line across the Selection Grid window at the selection center.
- Centerline Y: Displays a horizontal red line across the Selection Grid window at the selection center.
- Auto Scroll: This On/Off button activates the auto-scrolling function. This will keep the active object visible in the window by scrolling the sheet or grid.
- Gridlines: Shows or hides the grid lines.

To display or hide buttons from the Selection Grid window Title bar, tap Edit Title Bar in the **Selection Grid Window Setings** title bar:

Selection Grid Window Settings - Edit Tool Bar active.

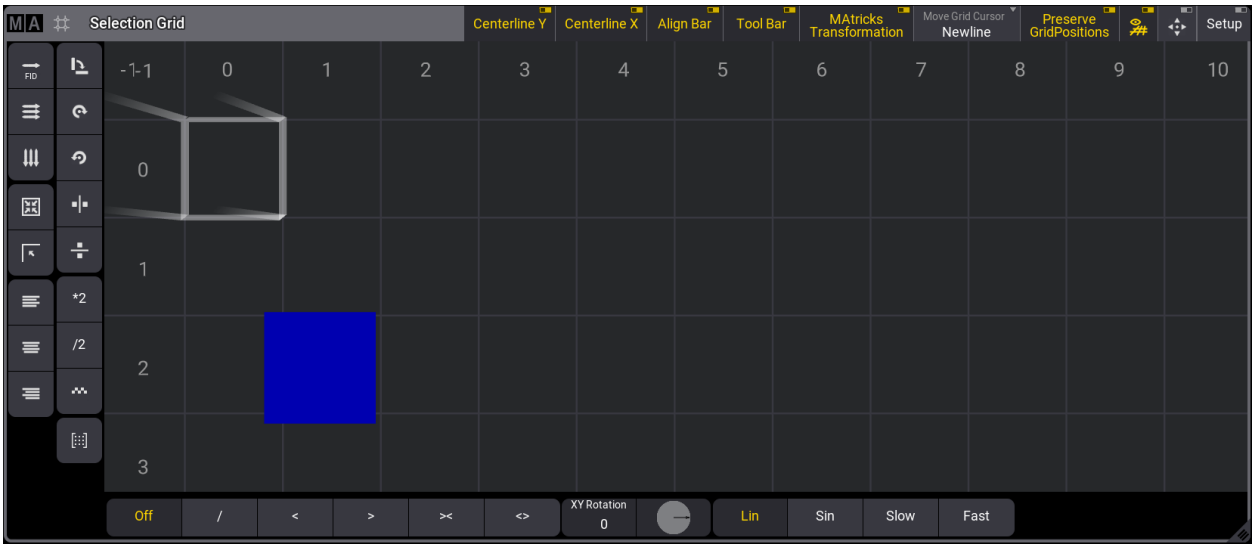
The grid cursor can be moved by tapping in the selection window, tapping the arrows on the keyboard, or using the grid keyword. For more information, see **Grid Keyword**.

Pressing and holding MA + X3 | Grid will enter the grid keyword in the command line:

For example, moving the cursor to X position 1, Y position 2, and Z position 1, press the following keys:

MA + X3 | Grid 1 / 2 / 1 Please

If the Z-axis value is 1, it can be omitted.



Selection Grid window with cursor moved to X position 1, Y position 2

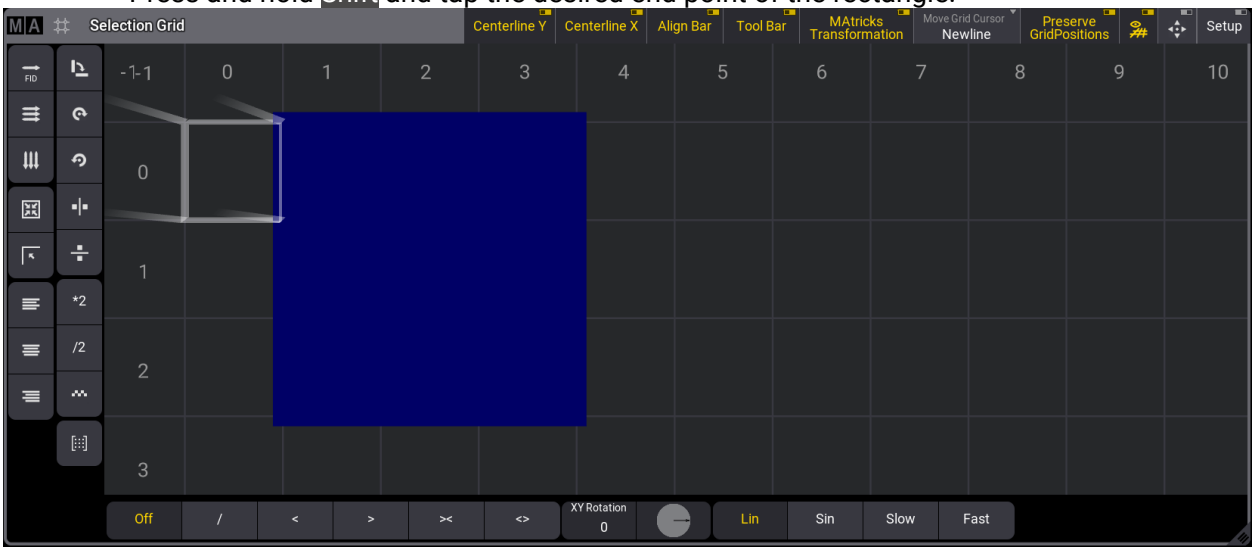
Multiple Cells

The grid cursor can be more than one cell. For example, a square area of grid cells can be selected.

To select a range of cells:

- Use the **Thru keyword** and specify the beginning cell and the ending cell.
  For example, press the following keys: MA + X3 | Grid 1 Thru 3 / 2 Please

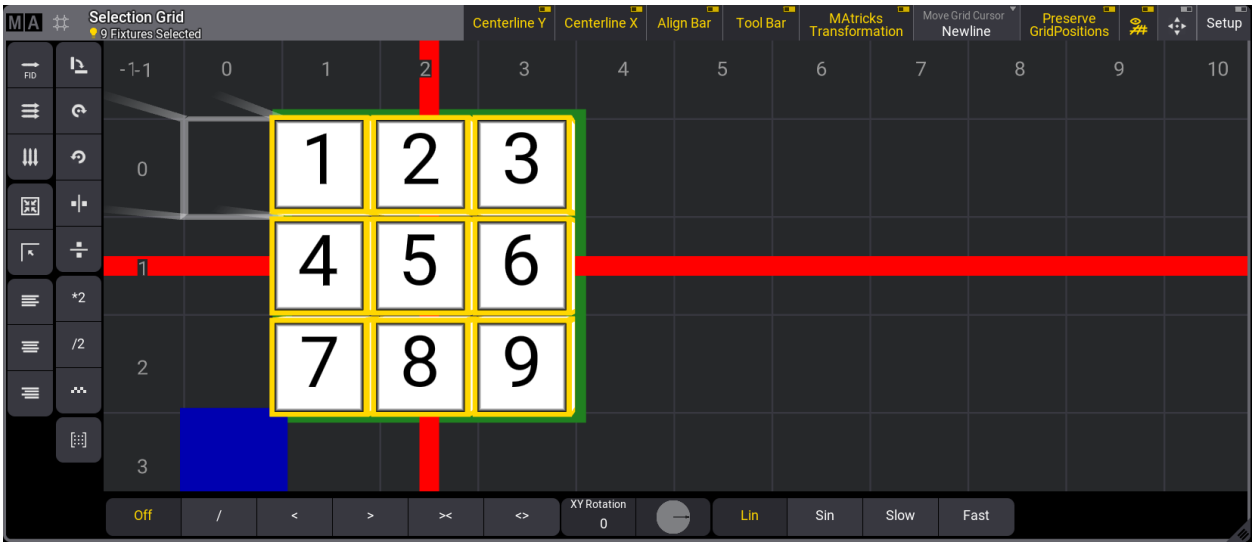
- Press and hold Shift and use the keyboard arrow keys to define the size of the rectangle.
- Press and hold Shift and tap the desired end point of the rectangle.



Multiple cells are selected in the Selection Grid window.

Selecting fixtures after defining the grid puts the fixtures in this area:

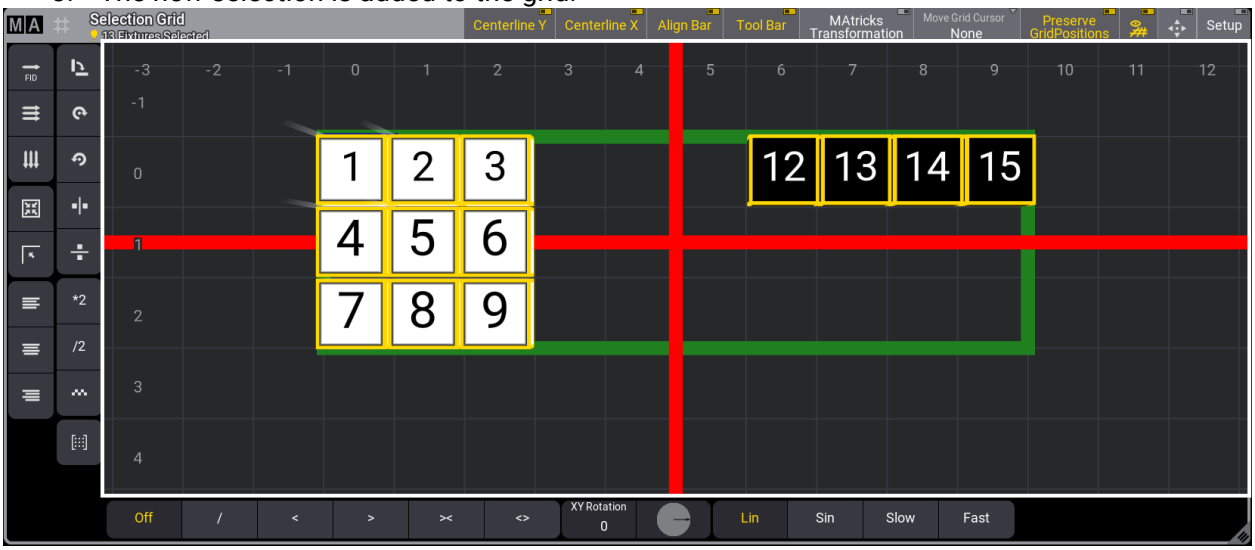
This will be the result if, for example, fixtures 1 thru 9 were selected and applied to the cell configuration above:



Selection Grid window with multiple cells selection

Fixtures can be added to the current selection:

1. Move the cursor where you want to have the next selection added.
2. Select some fixtures.
3. The new selection is added to the grid.



Selection Grid window with a new selection added

The horizontal and vertical red lines represent the center of the selection. The green outline represents the overall selection.

Fixtures can be knocked out of the programmer by pressing Off and tapping a fixture in the selection grid.
Or a range of fixtures can be knocked out of the programmer by pressing: Off Fixture 1 Thru 4 Please for example.

Store the Grid

The grid information can be stored in groups, presets, or other objects when the fixtures are positioned.

Read more about storing groups in the **Create Groups topic** or generally about groups in the **Groups topic**.

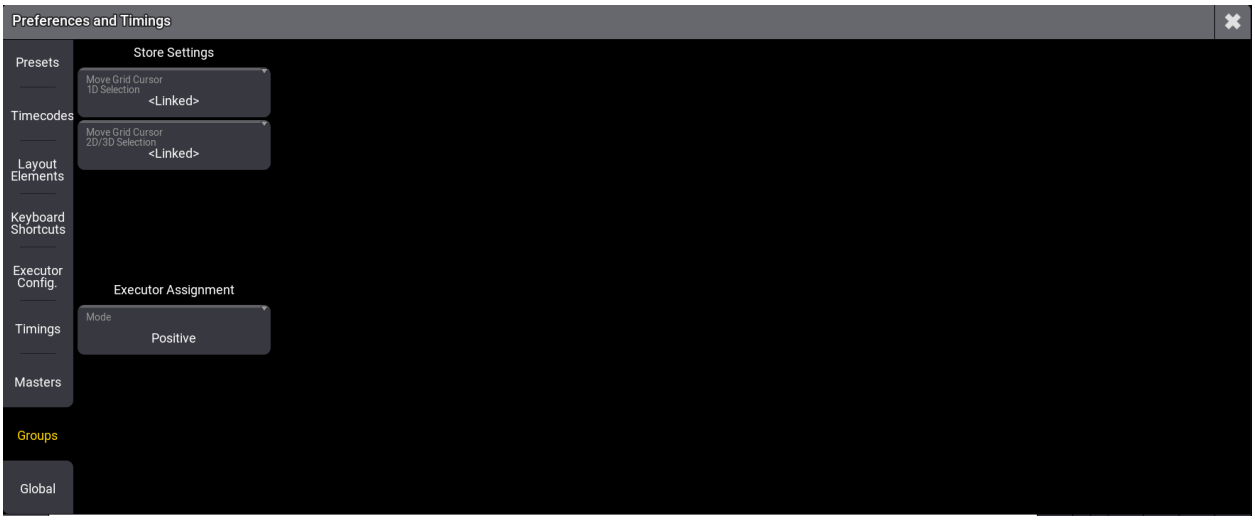
You can read about storing presets in the **Create Presets topic** or, more generally, about presets in the **Presets topic**.

Read more about storing cues in the **Store Cues topic** or the **cues and sequences section**.

Group preferences are located in the Preferences and Timings menu.

To open the Preferences and Timmings menu, press Menu, then tap Preferences and Timings.

Tap Groups on the left of the bottom of the window.



Preferences and Timing - Groups menu

- When Move Grid Cursor 1D Selection is enabled, **Move Grid Cursor** will be enabled in the stored group, given the Selection Grid has a selection only in one dimension.
- Move Grid Cursor 2D/3D Selection will enable **Move Grid Cursor** in the stored group, given the Selection Grid has a selection in at least two dimensions.
- MAtricks X, Y, and Z directions are relative to the entire range of selected fixtures.

When multi-instance fixtures are selected, pressing Down will position the sub-fixtures according to their arrangement inside the main fixture. Press Up to return to the former selection. This works for all levels, and each level can have its own MAtricks.

The position defined by the geometries inside the fixture type will determine the grid position of the sub-fixtures.

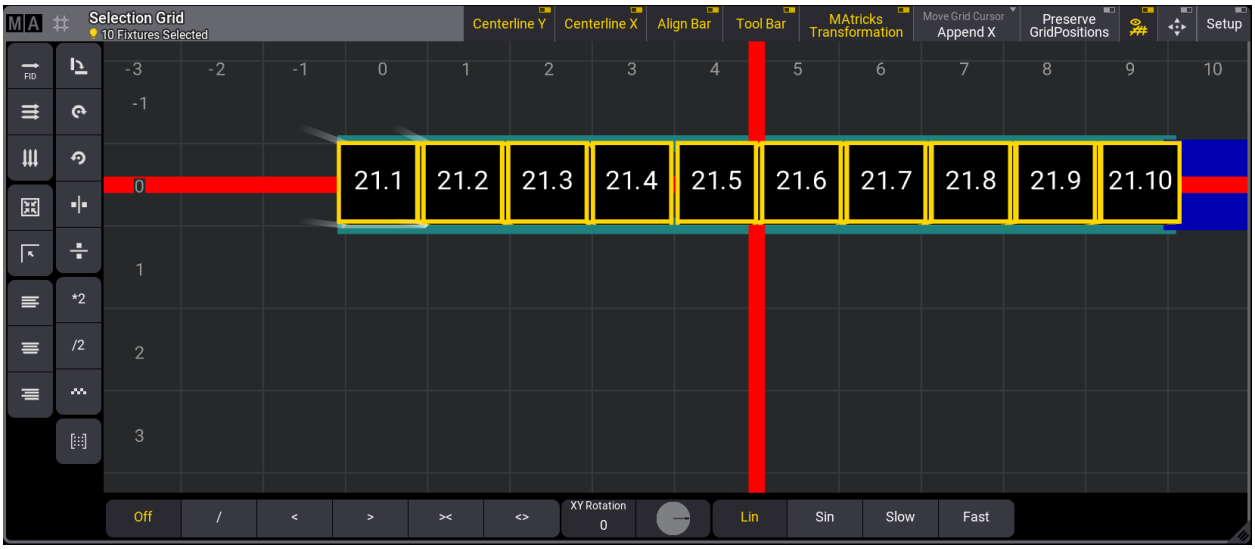
The grid position can be modified manually under the **Grid** columns of the **Geometry** tab in the fixture type editor: **Grid Auto**: Can change between Auto and Manual. In Auto mode (default value), the positions are determined automatically. When set to Manual, the user can define the grid positions independently using the following properties.

- **Grid Swap XY**: Interchanges the grid positions between the X and Y axis.

- **Grid Inv X**: Inverts the grid positions on the X-axis.
- **Grid Inv Y**: Inverts the grid positions on the Y-axis.
- **Grid Inv Z**: Inverts the grid positions on the Z-axis.
- **Grid X**: Defines the position of the geometry on the X-axis.
- **Grid Y**: Defines the position of the geometry on the Y-axis.
- **Grid Z**: Defines the position of the geometry on the Z-axis.

The Align Range functionality Rx, Ry, and Rz in the MAtricks window allows the user to define whether the values are to be aligned across the whole selection or individually per row/column. This can be defined individually per axis. When an Align Range is enabled, the frame around the selection in the Selection Grid window changes to a dark sea green color.

For more information, see **Align**.



Selection Grid with Align Range functionality applied in the MAtricks

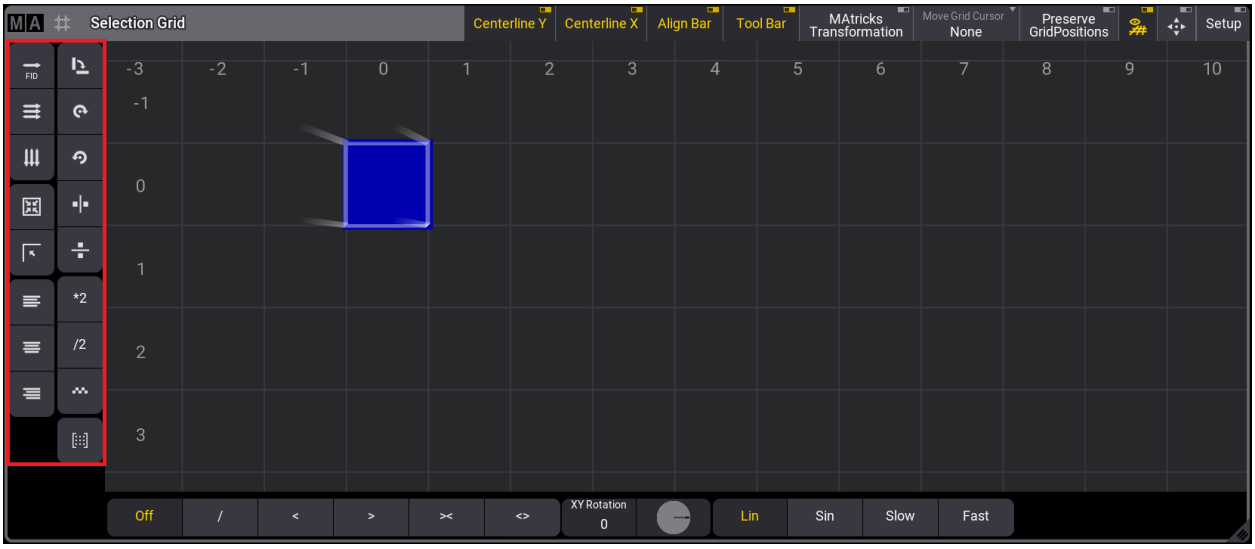
Subtopics

- **Selection Grid Tool Bar**
- **Selection Grid Setup Mode**

1.15.12.1. Selection Grid Tool Bar

Selection Grid Tool Bar

The Selection Grid Tool Bar is visible on the left side of the Selection Grid window when Tool Bar is enabled. The tools can be used to modify the arrangement of the selected fixtures.



Selection Grid toolbar

Here's a description of what the tools can do:

- **Linearize Grid Numerical**: The currently selected fixtures are rearranged into a line starting at Grid 0/0/0. The new order of the fixtures is in ascending order by their Fixture IDs (and secondarily by the CIDs).
- **Linearize Grid Left To Right then Top to Bottom**: The currently selected fixtures are rearranged into a line starting at Grid 0/0/0. Based on the current grid arrangement, the new order of the fixtures is determined to work from left to right first and then from top to bottom.
- **Linearize Grid Top to Bottom then Left to Right**: The currently selected fixtures are rearranged into a line starting at Grid 0/0/0. Based on the current grid arrangement, the new order of the fixtures is determined to work from top to bottom first and then from left to right.
- **Remove Gaps:** Removes gaps between fixtures.
- **Remove Offset:** Removes the offset between the origin and the selection**.**
- **Apply Left Alignment to Grid:** Align the selection to the left of the whole selection.
- **Apply Center Alignment to Grid:** Align the selection to the center of the whole selection.
- **Apply Right Alignment to Grid:** Align the selection to the right of the whole selection.
- **Transpose Grid**: Interchanges the X and Y grid coordinates of every selected fixture.
- **Rotate Grid by 90 Degrees to the right**: Rotate the currently selected fixtures 90 ° clockwise along the XY plane.
- **Rotate Grid by 90 Degrees to the left**: Rotate the currently selected fixtures 90 ° counterclockwise along the XY plane.
- **Mirror Grid at X-axis**: Mirrors the fixtures in the grid along a vertical mirror line.

- **Mirror Grid at Y-axis**: Mirrors the fixtures in the grid along a horizontal mirror line.
- *2 **Multiply Grid Coordinates by 2:** Relatively multiplies the space between fixtures on the x-axis.
- /2 **Divide Grid Coordinates by 2:** Relatively divide the space between fixtures on the x-axis.
- **Make Grid Symmetrical:** Transforms the current selection to be symmetrical.
- **Use MAtricks Positions and Reset MAtricks:** keep the fixtures at their MAtricks transformed coordinates and remove MAtricks.

Use the Grid keyword to trigger the grid tools from the command line. For example, to trigger the **Transpose** tool, type:

```
User name[Fixture]>Grid "Transpose"
```

These are the commands for each tool:

- Linearize Grid Numerical: **Grid "Linearize" "Numerical"**
- Linearize Grid Left To Right, then Top to Bottom: **Grid "Linearize" "LeftToRight"**
- Linearize Grid Top to Bottom, then Left to Right: **Grid "Linearize" "TopToBottom"**
- Remove Gaps: **Grid "RemoveGaps"**
- Remove Offsets: **Grid "RemoveOffset"**
- Apply Left Alignment to Grid: **Grid "Align" "Left"**
- Apply Center Alignment to Grid: **Grid "Align" "Center"**
- Apply Right Alignment to Grid: **Grid "Align" "Right"**
- Transpose Grid: **Grid "Transpose"**
- Rotate Grid by 90 Degrees to the right: **Grid "Rotate" "Right"**
- Rotate Grid by 90 Degrees to the left: **Grid "Rotate" "Left"**
- Mirror Grid at X-axis: **Grid "Flip" "X"**
- Mirror Grid at Y-axis: **Grid "Flip" "Y"**
- Multiply grid coordinates by 2: **Grid "Multiply" 2**
- Divide grid Coordinate by 2: **Grid "Divide" 2**
- Make grid symmetrical: **Grid "MakeSymmetrical"**
- Use matricks positions and reset matricks: **Grid "UseMatricksPositions"**

Grid Coordinates can be multiplied or divided by other values than 2. For example, to multiply the grid coordinates by the factor 6, type:

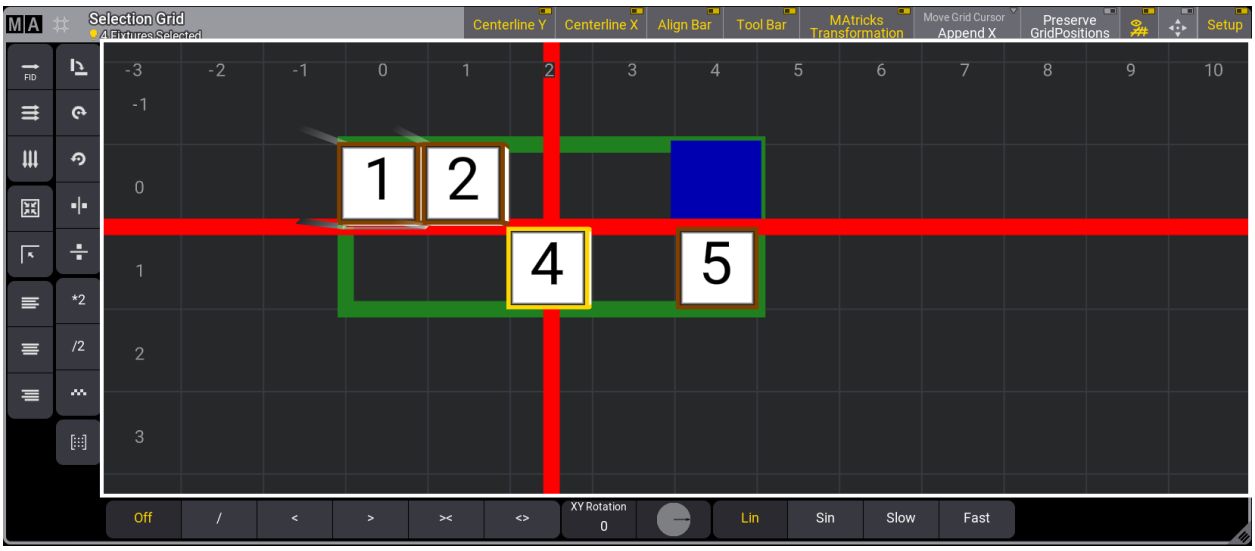
```
User name[Fixture]>Grid "Multiply" 6
```

1.15.12.2. Selection Grid Setup Mode

Selection Grid Setup Mode

When the **Setup** mode is enabled, fixtures can be moved in the selection grid.

To enter the **Setup** mode, tap Setup in the title bar of the **Selection Grid**.



Selection Grid window in Setup mode

Here's an example of how to move fixtures around in setup mode:

1. Select some fixtures.
2. Enable Setup in the title bar. Fixture boxes are now colored with dark orange frames.
3. Tap a fixture to select it.
4. Tap, hold, and drag the selected fixture to a new position.
5. When done adjusting the grid positions, tap Setup in the title bar to disable it.

More than one fixture can be selected by tapping them in the Selection grid or doing a lasso selection.

Fixtures can be placed on negative grid coordinates in the Selection Grid.

When the Setup mode is enabled, Selection in the encoder bar has a pulsating yellow indicator bar.

1.16. Scribbles

Scribbles are small drawings that are created in the software and can be assigned to objects.

All scribbles are stored in the Scribbles pool. For more information on opening the Scribbles pool, see **Add Window**.

Scribbles pool

Scribbles can be created in the Scribbles pool and then assigned to other objects or they can be created directly in the Label pop-up. For more information about labels, see **Label Keyword** and **Label Objects**.

	Important:
	Scribbles created while labeling an object are automatically saved in the Scribbles pool.

Subtopics

- **Create Scribbles**
- **Edit Scribbles**
- **Assign Scribbles**
- **Delete Scribbles**

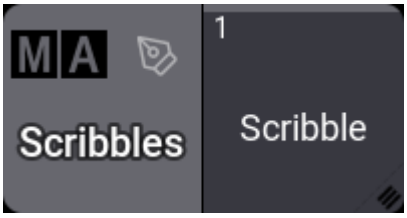
1.16.1. Create Scribbles

	Hint:
	Creating scribbles in the scribble pool or when labeling objects is possible.

Create Scribbles in the Scribble Pool

1. Access the scribble pool.
   - Open the **Add Window**.
   - Open Pools.
   - Open Scribbles.

For more information, see **Add Windows**.

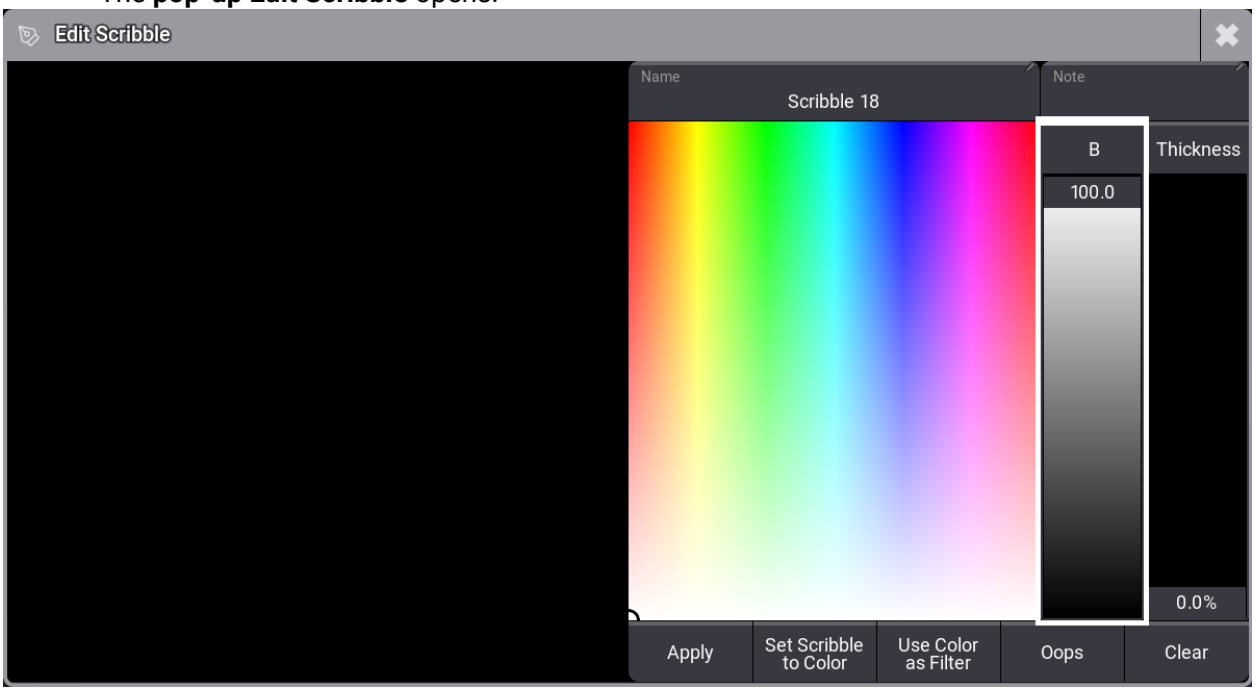


Scribble Pool

2. Store a new scribble.
   - Tap and hold an empty pool object.

   The new scribble is saved.
3. Create a scribble.
   - Press Edit and tap the scribble you saved.

   The **pop-up Edit Scribble** opens.



Pop-up Edit Scribble

4. Label the scribble:
   - Tap Name.
   - The **pop-up Edit Name** opens.
   - Give the scribble a name.
5. Select the color of the scribble.
   - Tap the color picker.
6. Set the brightness of the color.
   - Move the **B** fader.
7. Set the thickness of the scribble.
   - Move the **Thickness** fader.
8. To scribble, move your finger across the scribble pad on the left of the pop-up.
9. To delete the last step made in the scribble, tap Oops.
10. To apply the scribble, tap Apply. The scribble is applied to the scribble pool.

	Hint:
	A grandMA3 console saves the color and thickness settings of a scribble. Once the console is powered down, these settings are reset.

Create Scribbles in Other Pool Objects

This example is based on the object of a group pool.

Requirement:

- The group pool is open

1. Save a new pool object in the group pool.
2. Label the new pool object.
   - Press Assign Assign and tap the pool object.
   - The virtual keyboard opens.
3. Tap in the title bar of the keyboard.
   - The virtual keyboard expands and opens the scribble pad and its controls.
4. Follow steps **5 to 8** described in the scribble pool.

The scribble is automatically saved on the next free position in the scribble pool and applied to the group pool object.

	Important:
	Creating scribbles outside the scribble pool spams the pool because every scribble created with the label function within other pools is saved as a new scribble in the scribble pool. To learn how to avoid spamming, see **Assign Scribbles**.

1.16.2. Edit Scribbles

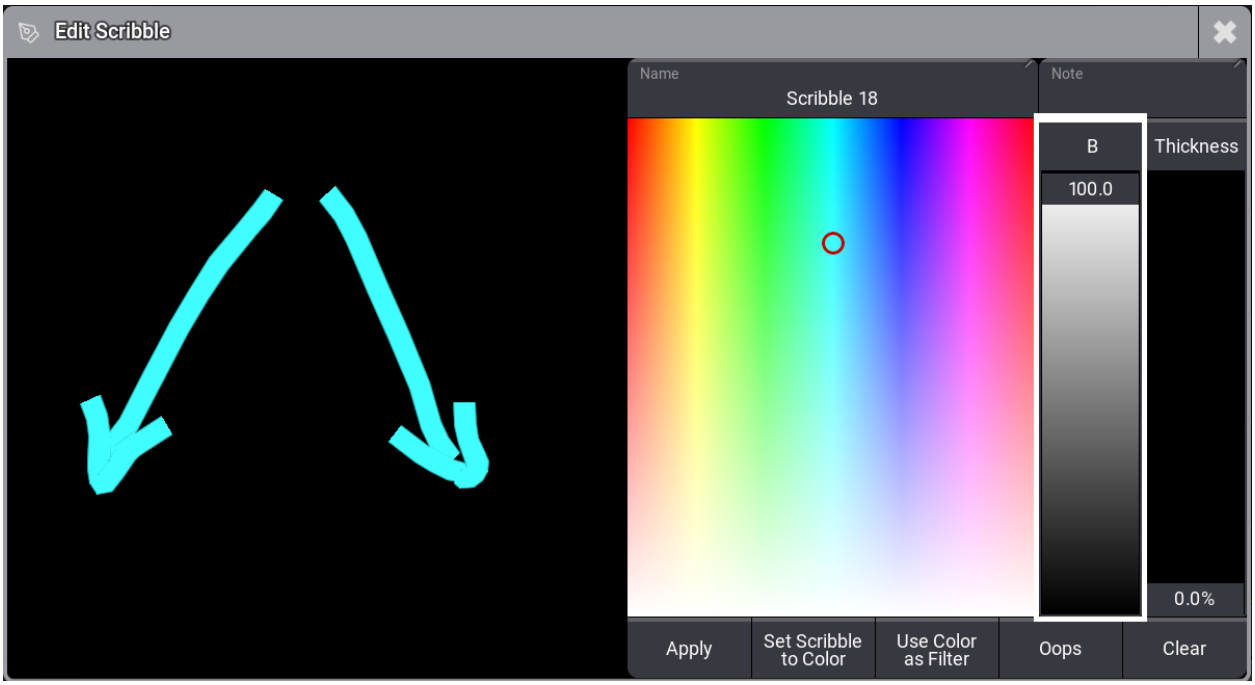
It is possible to edit an existing **Scribble**.

Edit Scribbles in the Scribble Pool

Requirement:

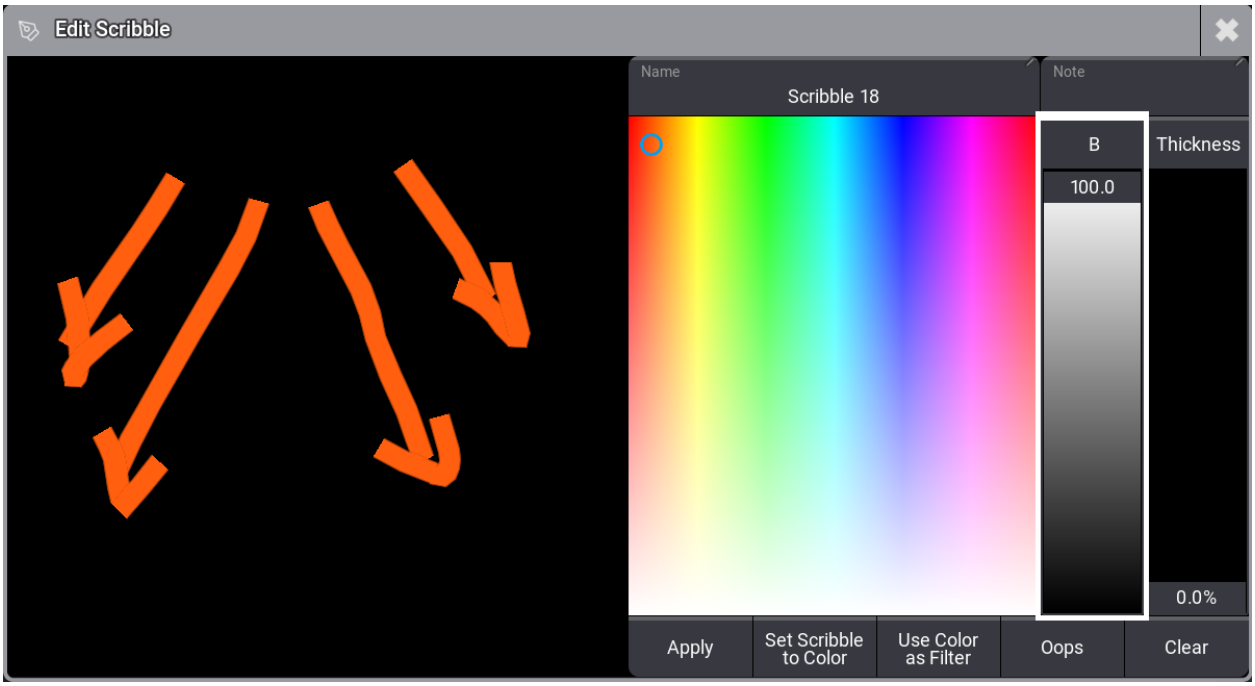
- The scribble is applied in the scribble pool

1. To edit an existing scribble, press Edit and tap it.
   -The window **Edit Scribble** opens.



Edit an existing scribble

2. Draw to add new content, or tap Clear to start from scratch.
3. Tap Oops to delete the last step if needed.



Adjustments in the existing scribble

4. Tap Apply.

The edits are saved in the scribble pool.

Edit Scribbles in Other Pools

This example is based on the group pool.

Requirement:

- The scribble is assigned to a group pool object

1. To edit an existing scribble in the group pool, press Assign Assign and tap the pool object.
   -The Window **Edit Name** opens. Tap in the title bar to open the scribble area if needed.
2. Adjust the scribble.
3. Tap Apply.

The edits are automatically saved in the scribble pool.

1.16.3. Assign Scribbles

You can assign existing scribbles in the scribble pool to other pool objects.

	**Hint:**
	To avoid spamming the scribble pool, create the scribbles in the scribble pool and then assign them to other pool objects.

This example is based on a group pool object.

Requirement:

- A scribble was created in the scribble pool.

To assign a scribble to a group pool object:

1. Press Assign.
2. Tap the scribble in the scribble pool.
3. Tap the pool object you would like to assign the scribble to.

The scribble is assigned to the group pool object.

1.16.4. Delete Scribbles

Delete Scribbles in the Scribble Pool

It is possible to delete scribbles using one of the options:

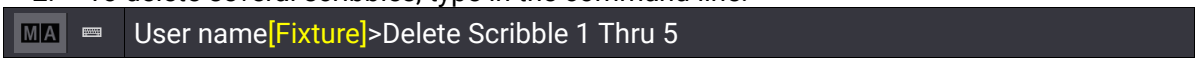
1. To delete single scribbles:
   - Press Delete and tap the scribble you would like to delete.

-or-

   - Use the Swipey.
     For information, see **Pool Windows – Swipey**.

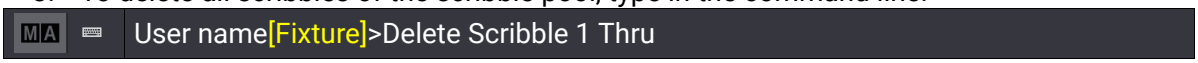
	**Important:**
	If you have assigned scribbles to other pool objects, deleting individual scribbles in the scribble pool will delete the links in all the pools.

2. To delete several scribbles, type in the command line:



Deletes scribbles 1 to 5.

3. To delete all scribbles of the scribble pool, type in the command line:



1.17. Images

Imported images are in the **Image Pool**. For more information, see **Pool windows**.

	Important:
	The overall size of the media pool is a maximum of 200 MB.

We advise that you keep the image pool as small as possible. For example, keep the maximum image size below 64 MB, and do not exceed a resolution of 1 920 x 1 080.

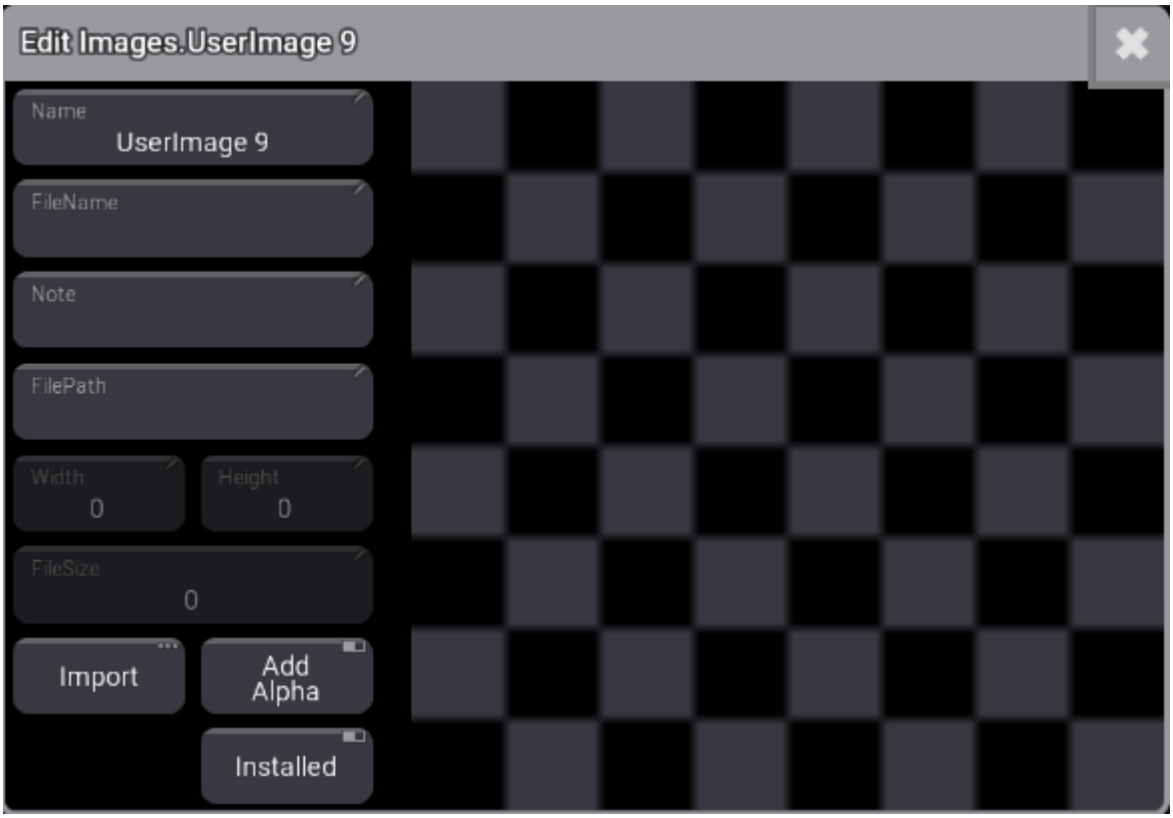
Images can be used for appearances. Read more in the **Create Appearances** topic.

Example of the images pool

Edit an image pool object using one of two options:

1. Press Edit and then tap the desired pool object.
2. Open the swipey commands on the pool object and choose Edit.

The Edit Images pop-up opens:



Edit Images pop-up window

Import Images Using the Image Pool

Workflow:

1. Edit an empty pool object.
2. Tap the Import button.
3. Tap Internal in the top-right of the title bar to change the drive to the desired source.
4. Select the desired image.
5. Tap Import.
6. Close the **Edit Image** pop-up.
7. Before closing the Edit Image pop-up, it is possible to edit the name.

Tap the video below to see the example.

By default, the grandMA3 software uses the **gma3_library/media/images** folder to import and export images. For more information, see **Folder Structure**.

Import Images from the Import/Export Menu

Workflow:

1. Press Menu.
2. Tap Show Creator

3. Tap Import on the top left corner of the window.
4. Tap Images.
5. Select the drive, then tap an empty pool object on the right side of the window.
6. Select the image to be imported.
7. Tap Import at the bottom of the window.

Delete an Image from the Pool

Deleting images is like deleting any other pool object. The image disappears in any appearance where it might be used.

There are three common ways to delete images.

Delete an Image Using the Command Line

The important keyword for this is: **Delete**.

This is the syntax for deleting a single image:

Delete Image 3.x

Delete a range of images using the standard range syntax (Thru, +, and -).

For example, if image 4 needs to be deleted:

```
User name[Fixture]>Delete Image 3.4
```

Or if images 5 to 10 need to be deleted:

```
User name[Fixture]>Delete Image 3.5 Thru 10
```

Delete an Image Using the Image Pool on a Screen

Requirement:

A visible image pool on one of the screens.

1. Press Delete.
2. Tap the image in the pool.

The image is deleted.

Delete an Image Using the Swipey Commands

Requirement:

A visible image pool on one of the screens.

1. Tap and hold the image you wish to delete.

2. Swipe out of the pool object without releasing the screen.
3. Swipe to the Delete swipey and release the screen.

The image is deleted.

1.18. Screenshots

Screenshots can be created by pressing the Print Screen key or shortcut F11 on a built-in or external keyboard. A green flash on all screens indicates that the screenshot has been taken.

An image from each screen is stored as individual PNG files in the image folder. The file names are generated using this template: YYYYMMDD_hhmmss_displayX.png

The images can be imported into the **image pool** or copied from a console using an SFTP connection. For more information, see **Networking — SFTP connection**.

	Restriction:
	There is no keyword or command to create screenshots.

Storing Screenshots on a USB Stick

Screenshots can be stored on a USB memory stick. They are stored as PNG files.

The USB needs to be the selected drive when Print Screen is pressed.

A list of the drives can be seen in the **Command Line History window** by running the following command:

It is drive number two if there are no other USB devices connected.

Select the drive using this syntax: **Select Drive [drive_number]**

Now press Print Screen.

Bring the USB to a computer. Screenshots are saved to the **/gma3_library/media/images** folder. On USB drives, the **gma3_library** folder is located in the **grandMA3** folder.

Copy Screenshots using SFTP

When accessing the console's internal drive using SFTP, the **gma3_library** folder is located at the root of the visible folder structure.

Open an **SFTP connection**, navigate to the images folder, and copy the relevant files from the console.

For more information on folder structure, see the **Folder Structure** topic.

1.19. Video

Videos can be used as a source for an appearance.

Videos can be imported into the Video Pool.

Video pool

The video pool holds the imported videos but can also have streaming video sources using the NDI format.

The video objects can be used as an image source in Appearances. Learn more about appearances in the **Appearances section**.

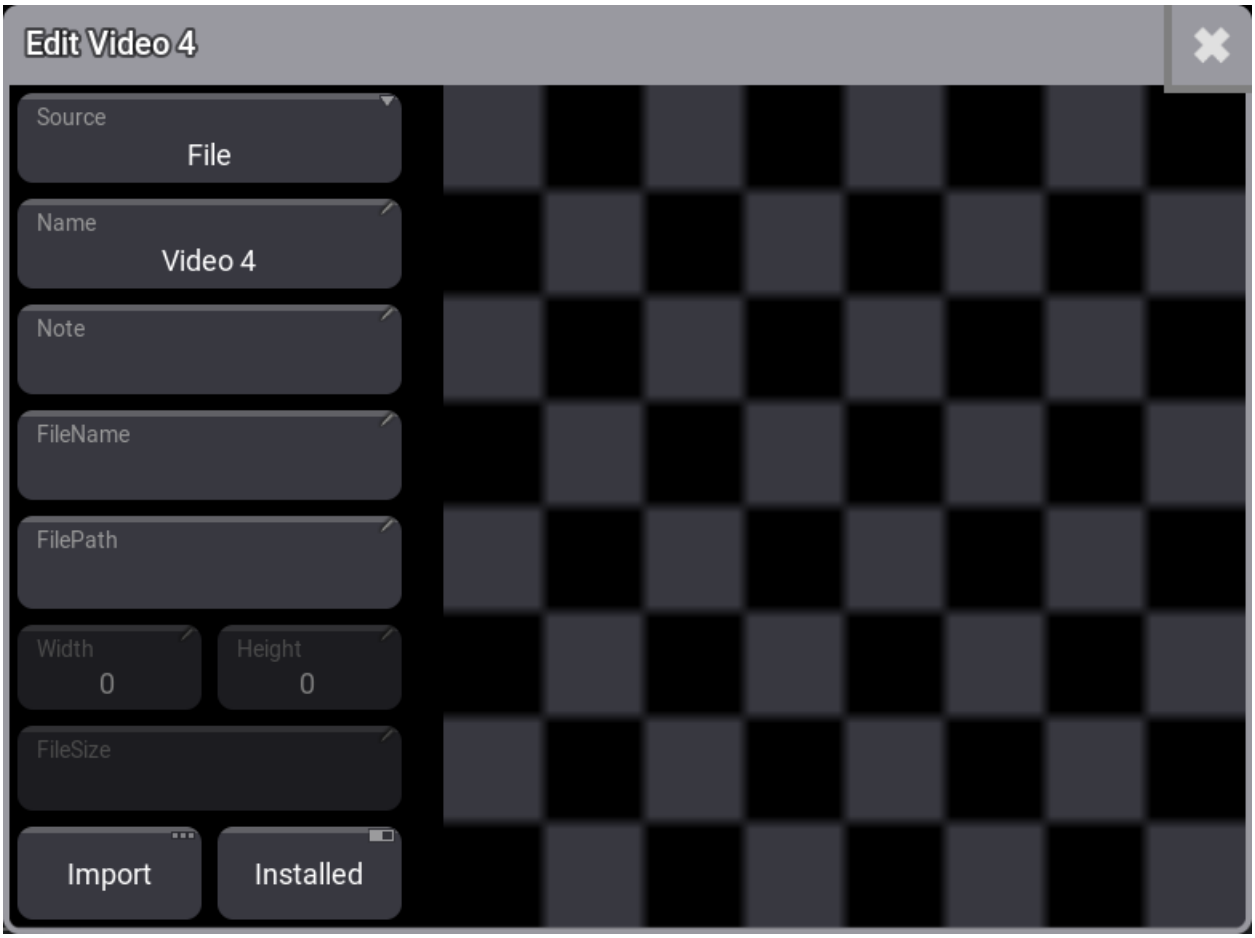
The video will then play back in a loop everywhere the appearance is used.

	Restriction:
	Third-Party Software needs to be activated and agreed to in Menu / Settings / Software Update / EULA for videos to be played back. The button is in the lower right corner. Turning it On opens a pop-up that needs to be agreed to.
	Restriction:
	The overall size of the media pools is limited to a maximum of 200 MB.

Import a Video

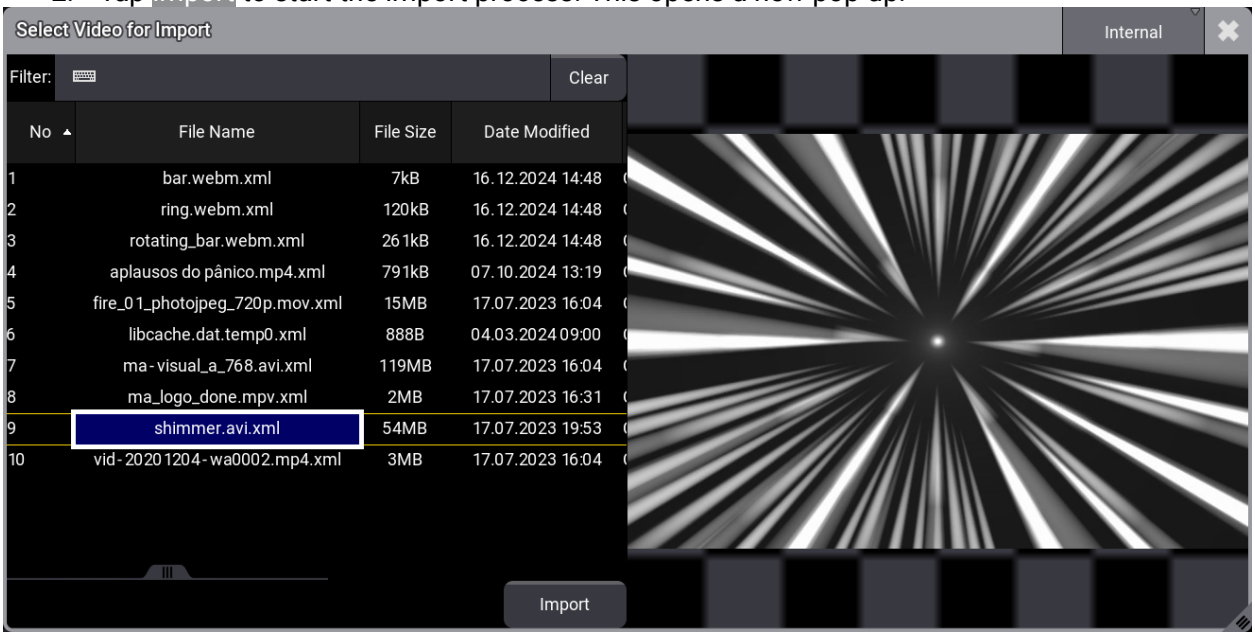
The video file needs to be in the correct folder to be able to be imported. The folder is /gma3_library/media/videos. Learn more about this in the **Folder Structure topic**.

1. Edit an empty video pool object. This opens the **Edit Video** pop-up:



Edit video pop-up

2. Tap Import to start the import process. This opens a new pop-up:

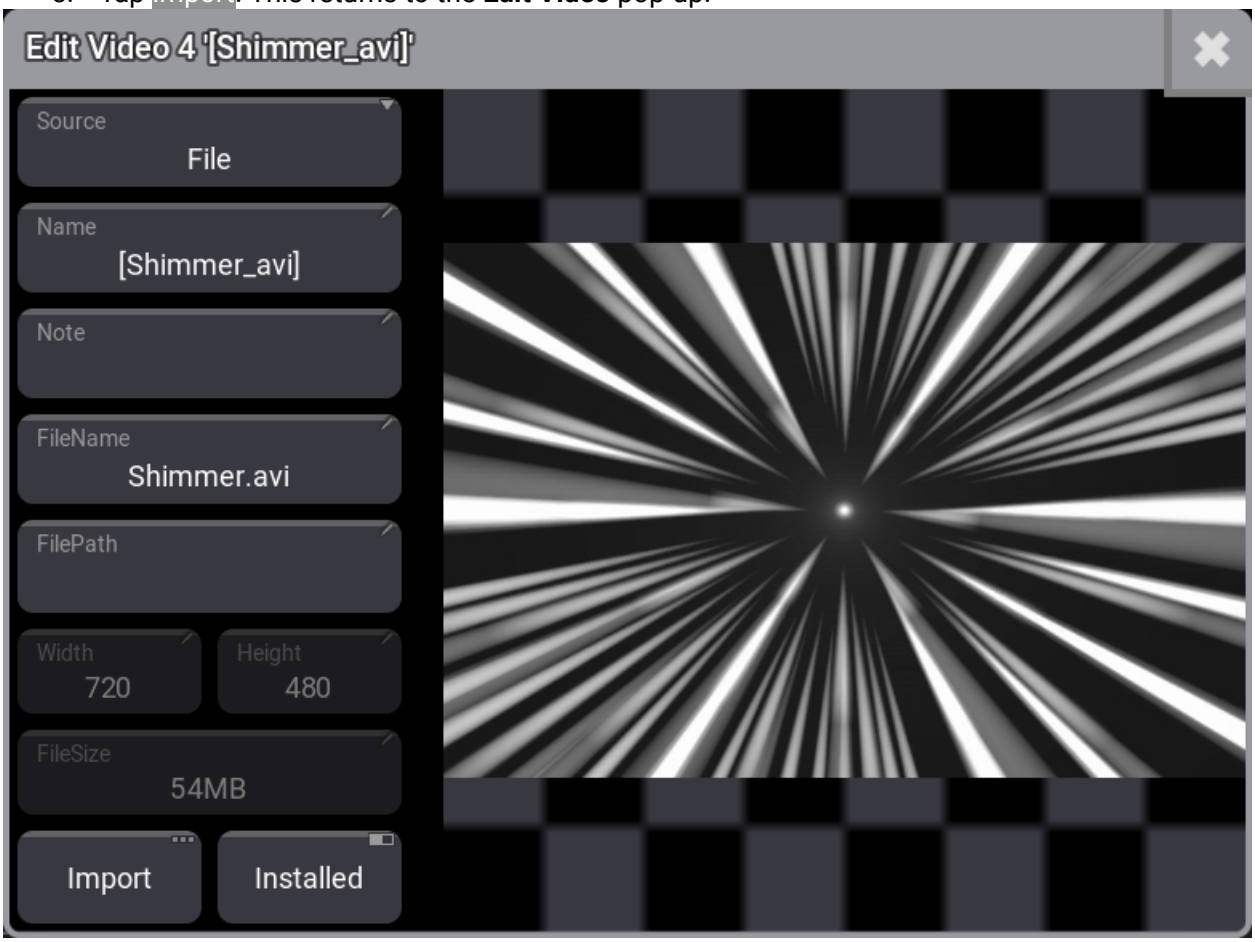


Select image for import pop-up

This lists all the video files in the video folder.

3. If needed then tap Internal in the title bar of the pop-up to change the selected drive.

4. Tap the desired video file. The video should play back in the preview area. If it does not appear then it is not a supported format or third-party software is not activated.
5. Tap Import. This returns to the **Edit Video** pop-up:



Edit video pop-up with a file selected

The editor now shows relevant data like video width, height, and file size.

6. Edit the name if desired.
7. The file is imported and the editor can be closed by tapping the .

Use an NDI Source

An NDI streaming video source can be used instead of importing a video file.

Requirement:

An NDI stream needs to be on the network.

1. Edit an empty video pool object. This opens the **Edit Video** pop-up.
2. Tap Source.
3. Tap NDI in the **Select Source** pop-up.
4. Tap Select Source. This opens the **Select NDISource** pop-up.

5. Tap the desired NDI source. If the list is empty then there are no NDI sources in the network or third-party software is not activated.
6. Select the desired bandwidth setting.
7. Close the editor by tapping the  .

Two bandwidth options are available. The bandwidth option defines the requested quality of the stream. The higher quality uses more bandwidth, and the network setup should be considered when choosing the bandwidth setting.

Create an Appearance with Video

Adding a video to an appearance is almost the same as adding an image from the image pool.

The difference is that ImageSource needs to change to "Videos" in the title bar of the **Select Image** pop-up.

Learn how in the **Create Appearances** topic.

1.20. Meshes

Meshes are stored in the Mesh **Pool**. Meshes 1 thru 9 are locked and can not be edited.

	Important:
	For more information on **Meshes**, see **Build fixture types**.

Open a **Mesh Pool** window. For information, see the **Add window** topic.

Mesh pool

In the **Add Window** pop-up, tap Pools, then in the **Media** column, tap Meshes.

Edit a mesh pool object using one of the following two options:

1. Press Edit and then tap the desired pool object.
2. Open the swipey commands on the pool object and choose Edit.

The Edit **UserMesh** pop-up opens.

	Hint:
	This icon Is displayed in the top right corner of a mesh pool object if a note exists. The note is displayed in the Note section of the Edit pop-up.

Import Meshes

To import meshes:

1. Repeat option 1 or 2 above.
2. In the **Edit UserMesh** pop-up, tap Import. The **Import mesh** pop-up opens.
3. Select the source from the title bar, for example, Internal.
4. Select the mesh to import.
5. Tap Import.

The new mesh pool object is imported.

Tap the video below to see the example.

1.21. Gobos

Gobos are stored in the Gobo **Pool**. Gobos 1 thru 4 are locked and can not be edited.

When fixtures are patched in the show file, the images of the gobos used by these fixtures are stored in the **Gobo Pool**.

Open a **Gobo Pool** window. For information, see the **Add window** topic.

In the **Add Window** pop-up, tap Pools, then tap Gobos in the **Media** column.

Gobo pool

Edit a gobo pool object using one of two options:

- Press Edit and then tap the desired pool object.
- Open the swipey commands on the pool object and choose Edit.

Assign Gobo to Appearances

To assign a gobo to an appearance:

1. Open an Appearance Pool window.
2. Open the swipey command on the pool object, select Edit or press Edit, and tap an appearance pool object. The **Edit Appearance** pop-up opens.
3. Tap Image. The **Select Image** pop-up opens.
4. From the title bar, tap and hold ImageSource and select **Gobos**. The Select Image pop-up opens.
5. Select an image and close the pop-up.
6. Continue editing the new appearance if needed and close the **Edit Appearance** pop-up.

A new appearance is created.

Tap the video below to see the example.

1.22. Symbols

Symbols are stored in the **Symbol Pool**. Symbols 1 thru 14 are locked and can not be edited.

Open a **Symbol Pool** window. For information, see the **Add window** topic.

In the **Add Window** pop-up, tap Pools, then in the **Media** column, tap Symbols.

Symbol pool

Edit a symbol pool object using one of two options:

1. Press Edit and then tap the desired pool object.
2. Open the swipey commands on the pool object and choose Edit.

Assign Symbols to Appearances

To assign a symbol to an appearance:

1. Open an Appearances pool window.
2. Open the swipey command on the pool object, select **Edit** or press Edit, and tap an appearance pool object. The **Edit Appearance** pop-up opens.
3. Tap Image. The **Select Image** pop-up opens.
4. From the title bar, tap and hold ImageSource and select **Symbols**. The Select Image pop-up opens.
5. Select an image and close the pop-up.
6. Edit the new appearance if needed and close the **Edit Appearance** pop-up.

The new appearance is created.

Tap the video below to see the example.

Subtopics

- **Import Symbols**
- **Delete Symbols**

1.22.1. Import Symbols

Import Predefined Symbols

Symbols can be imported using the command line.

1. Navigate to the symbol pool:

```
User name[Fixture]>ChangeDestination Image 2
```

2. Now import the predefined library images:

```
User name@ShowData/MediaPools/Symbols> Import Library "*.xml"
```

This command imports the symbol files from MA into the symbols pool.

3. Return to the command line root:

```
User name@ShowData/MediaPools/Symbols> ChangeDestination Root
```

For more information, see **ChangeDestination keyword**.

Using the Symbol Pool

Workflow:

1. Tap Edit, then tap an empty pool object or open the swipey commands and choose Edit.
2. From the **Edit Symbols** pop-up, tap Import. This opens the **Select Image** pop-up.
3. Select the drive, then select the symbol to import.
4. Tap Import.
5. If needed, label the new imported symbol, then close the **Edit Symbols** pop-up.

Tap the video below to see the example.

By default, the grandMA3 software uses the **gma3_library/media/symbols** folder to import and export symbols. For more information, see **Folder Structure**.

From the Import/Export Menu

Workflow:

1. Press Menu.
2. Tap Import/Export
3. Tap Import on the top left corner of the window.
4. Tap Symbols.
5. Select the drive, then tap an empty pool object on the right side of the window.
6. Select the symbol to be imported.

7. Tap Import at the bottom of the window.

1.22.2. Delete Symbols

Deleting symbols is like deleting any other pool object. The symbol disappears in any appearance where it might be used.

There are three common ways to delete symbols.

Using the Command Line

The important keyword for this is: **Delete**.

This is the syntax for deleting a single symbol:

Delete Image 2.x

For example, if symbol 4 needs to be deleted:

Deleting a range of symbols using the standard range syntax (Thru, +, and -) is also possible.

For example, if symbols 15 to 20 need to be deleted:

Using the Symbol Pool on a Screen

Requirement:

A visible symbol pool on one of the screens.

1. Press Delete.
2. Tap the symbol in the pool.

The symbol is deleted.

Using the Swipey Commands

Requirement:

A visible symbol pool on one of the screens.

1. Tap and hold the symbol you wish to delete.
2. Swipe out of the pool object without releasing the screen.

3. Swipe to the Delete swipey and release the screen.

The symbol is deleted.

1.23. Appearances

Appearances are sets of looks that can be assigned to pool objects, presets, view buttons, or windows.

All appearances are stored in the **Appearance Pool,** which can be created like any other window. Learn how in the **Add Windows topic**.

Example of the Appearance pool

Images can be a big part of an appearance. To use images, they need to be imported into the **Image Pool**. Read about the images in the **Images topic**.

Subtopics

- **Create Appearances**
- **Use Appearances**
- **Delete Appearances**

1.23.1. Create Appearances

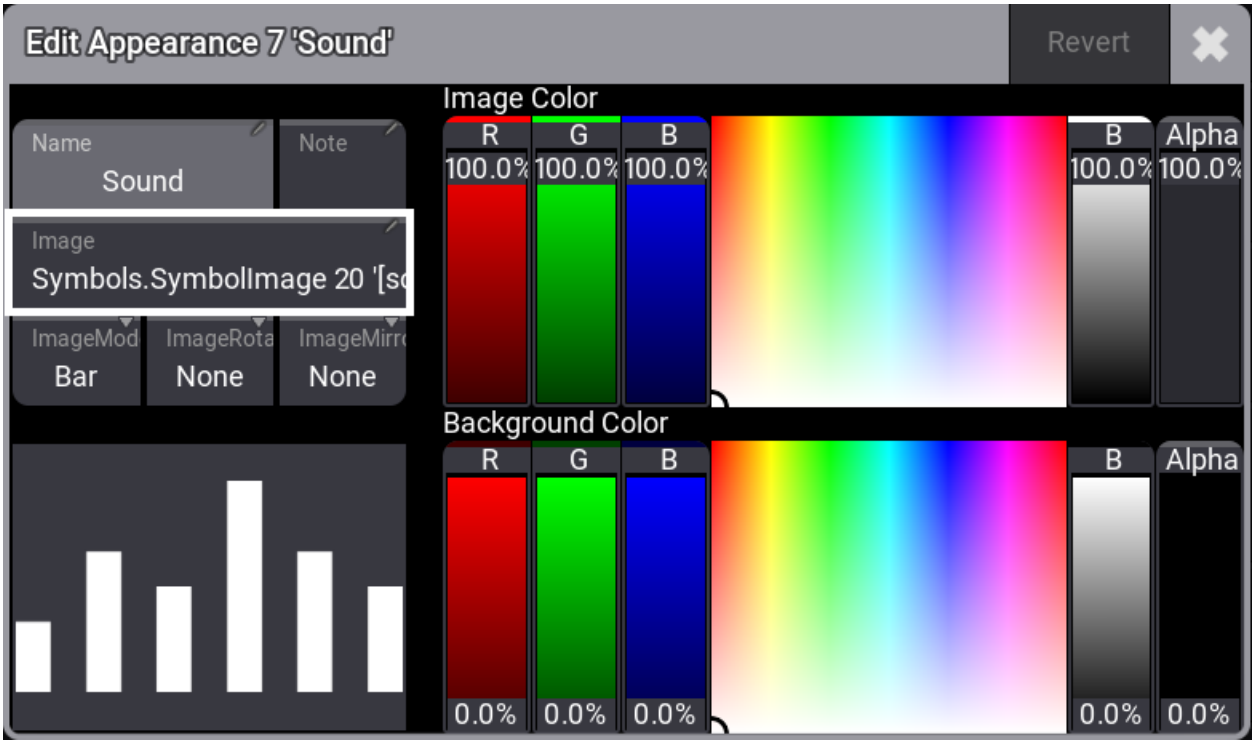
Edit an empty pool object in the **Appearance Pool** to start to create a new appearance.

Edit an appearance pool object using one of these 3 options:

- Press Edit and then tap the desired appearance object.
- Open the swipey commands on the pool object and choose Edit.
- Using the command line: **Edit Appearance [ID]**.

If an existing pool object is edited, then the editor opens with the current state of the appearance.

This is the **Appearance Editor**:



There are five input boxes, two sets of faders with a color picker to adjust color, and a preview area displaying a preview of the appearance.

Setting a Background Color

The **Background Color** fader and color picker can be used to change the background of the appearance. The default background is transparent (**Alpha** fader at 0%, **B** fader at 0%).

Turning up the alpha and using the R (red), G (green), and B (blue) faders make it possible to mix any color in the RGB range. The color picker can also be tapped to select a color.

Giving the Appearance a Name

Many editors and pop-ups have a **Name** input field. A name can be set using the **Label keyword**.

Appearances can be labeled like any **other pool object**.

It is also possible to give the appearance a name in the editor:

1. Tap the name input box.
2. Write a new name using the Edit Name pop-up.

	Restriction:
	Appearances cannot have a Scribble in the label.

Adding an Image to the Appearance

Appearances can use an image. The image is placed in front of the background but behind a label.

1. Tap the **Image** input box to add an image. This opens the **Select Image pop-up** where all the images are listed—Tap ImageSource in the title bar, to select a new source.
2. Choose the desired image source.
3. Tap the desired image in the pop-up.

	Hint:
	Appearances that use an image will not display the name on the appearance object in the Pool window if the name is the default name.

Media objects (images, videos, symbols, and gobos) can be **assigned** to an appearance. Assigning images to an empty appearance will create a new Appearance pool object.

	Hint:
	Assigning an appearance to another one creates a copy of the first appearance.

The images have different modes. Their modes define how the image is adapted to the aspect ratio.

These are the different modes:

- **Stretch**
  The image is stretched to fit the appearance area.
- **Bar**
  The image is fitted to the appearance area without changing the aspect of the image.
- **Crop**
  The image is fitted to fill the entire appearance area. The aspect of the image is kept, but it is chopped.
- **Tile**
  The image is tiled and chopped to fit the entire appearance area. The aspect of the image is kept.

Do the following to change the mode:

1. Tap the **Mode** swipe button. This toggles through different modes. You can also swipe the button to open the **Select Image Mode pop-up**.
2. Tap the desired mode in the pop-up.

The image's color can be adjusted using the top set of fader and color picker.

This means that the same image can be used on different appearances but with different color hues and transparency.

Tap the video below to see the example.

Revert Changes

To revert the changes:

1. Tap Revert. A pop-up opens.
2. Tap Ok. The changes got reverted.

	Restriction:
	Color can only be added to an image. This means that a black image cannot be changed, but a white image can.

1.23.2. Use Appearances

Appearances can be used in many places.

Almost all pool objects can have an appearance. Many windows can also have an appearance.

The appearance needs to be **created** before it can be assigned. And the object that it is assigned to needs to exist before anything can be assigned to it.

Using Appearances on Objects

Appearances can be assigned to objects using the following syntax:

Assign Appearance appear_number At object_type object_number

This can be used on pool objects.

For example, assigning appearance 1 at group 5 could be done like this in the command line:

This can also be done using the GUI.

The Appearance pool and the Groups pools should be visible on the screens for this.

1. Open the swipey commands on an Appearances pool object and choose Assign.
2. Tap a Group pool object.

The Appearance is assigned to the selected Group pool object.

Tap the video below to see the example.

Editing many objects also gives access to an appearance button that can be tapped to select one of the existing appearances.

Using Appearance on Windows

Many windows can have an appearance assigned.

The best way is to open the **settings for the window** and edit the **Appearance** setting in the **Display** tab.

1.23.3. Delete Appearances

Deleting appearances can be done like **any other pool object**.

If the appearance is used somewhere, then a warning pop-up opens; press Ok to delete.

The object that used the appearance is reset to **No Appearance**.

Oopsing the deletion will reassign the appearance.

1.24. Notes

The Notes feature allows to add useful information to objects. Each object that has Notes available, has a Note button in its editor or a **Note** cell. For example:

- Presets
- Groups
- Macros
- Cues
- Fixture Types

Notes can be added to pool objects via the label command. For more information on how to label pool objects, see **Label pool objects**.

Notes can be added via the note keyword, too. For more information, see **Note keyword**.

Notes are part of the object information. For more information, see **Info window**.

To learn more about notes in particular, continue with the following subtopics and their examples:

Subtopics

- **Notes in Pool Objects**
- **Notes in Cues**

1.24.1. Notes in Pool Objects

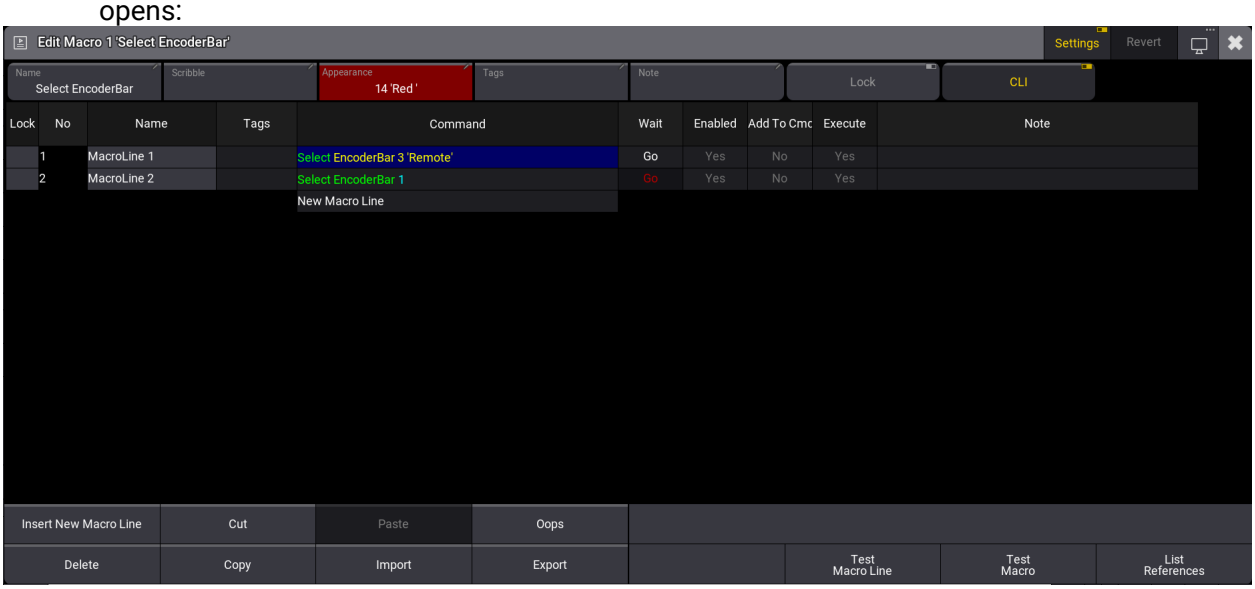
The following example shows how to add a note to a Macro. For more information about Macross, see **Macros topic**.

- **Requirement**: The Demoshow_grandMA3 is open.

For an overview of the example below, have a look at this video:

To add a note to a macro, follow these steps:

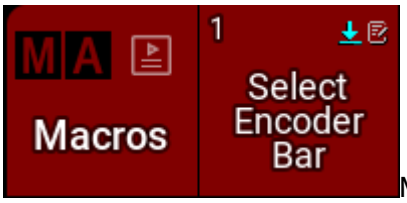
1. Open the Add Window dialog, tap the Data Pools and then tap Macros.
   The Macro Pool window opens.
2. Press Edit and then tap the first macro pool object called Select EncoderBar. The macro editor opens:



Macro Editor.

	Hint:
	Make sure Settings is enabled in the top right corner in the macro editor.

3. Tap Note in the Settings bar at the top in the macro editor. The text input editor opens.
4. Type "Macro" into the text field and then tap Save in the top right corner. The text input editor closes and the text is saved.
5. Tap , to close the macro editor.
   The pool object displays a note symbol in the top right corner:

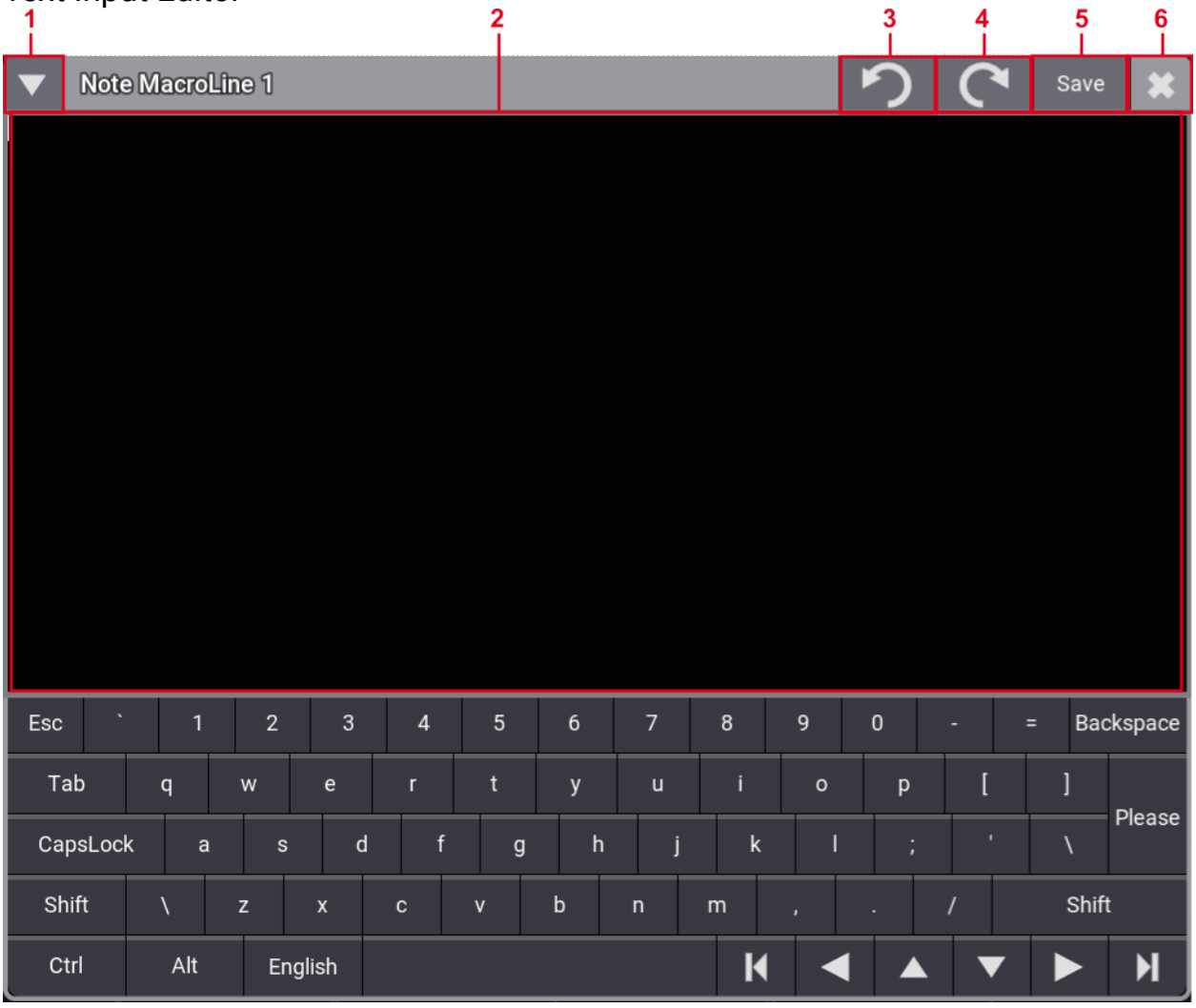


Macros - Pool object 1.

To add or edit the note in a Note cell, for example, a macro line, follow these steps:

1. Type **Edit Macro 1** into the command line and press Please. The editor opens.
2. Right-click the Note cell of MacroLine 1.
   The text input editor opens.
3. Follow the steps 4 and 5 from above.

Text Input Editor



Note Editor

The text input editor has the following features:

1. Shows or hides the virtual keyboard.

2. Text field.
3. Undoes last changes.
4. Redoes last changes.
5. Saves the changes and closes the input dialog. Alternatively, press Please on the console.
6. Closes the input dialog. A pop-up opens, if changes are unsaved.

	**Hint:**
	Press MA + 8, 2, 4, or 6 to move the cursor inside the text field.
	**Hint:**
	When editing notes, the numeric keypad on the grandMA3 hardware can be used to enter values, and Oops can be used as a backspace.

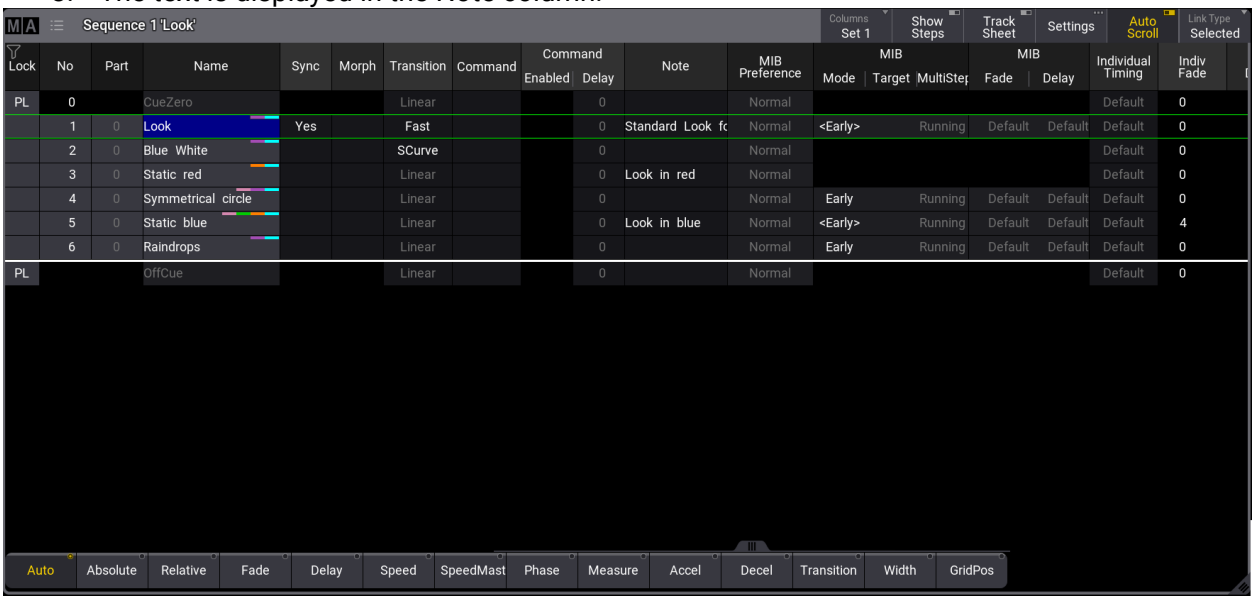
1.24.2. Notes in Cues

The following example shows how to add a note to a cue. Notes, for example, can inform the operator of primary on-stage procedures to trigger the next cue. For information about cues, see **Cues and Sequences**.

- **Requirement**: The Demoshow_grandMA3 is open.

To add a note to a cue:

1. Open the Add window menu, tap Common and then tap Sequence Sheet. The Sequence Sheet opens and displays Sequence 1.
2. Tap into an empty note cell in a cue and start typing. The Text input editor opens. After finishing typing the note, tap Save.
3. The text is displayed in the Note column.



Sequence 1 with different Notes added.

Multi-Line Editor

The text editor in notes is a multi-line editor. This means, multiple lines can be added by pressing Shift + Please when the text editor is open. For more information about the text editor, see **Text Input Editor**.

Changing the Line Height

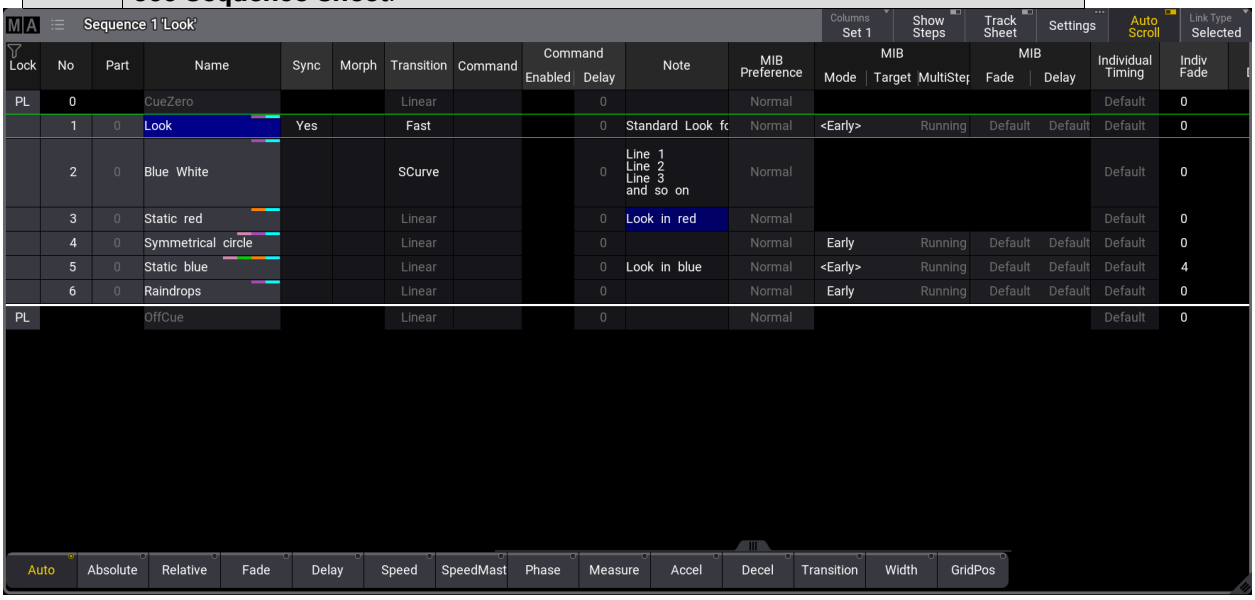
To change the line height manually:

1. Tap MA. The Sequence Sheet Settings opens.

2. Tap the Mask - tab.
3. Tap Line Height. The line height of the sequence sheet changes.

Hint:

When Line Height is set to Auto, the row adjusts its height based on the number of lines. For more information about the sequence sheet settings, see **Sequence Sheet**.



The height of the rows is adjusted dynamically when Line Height is set to Auto.

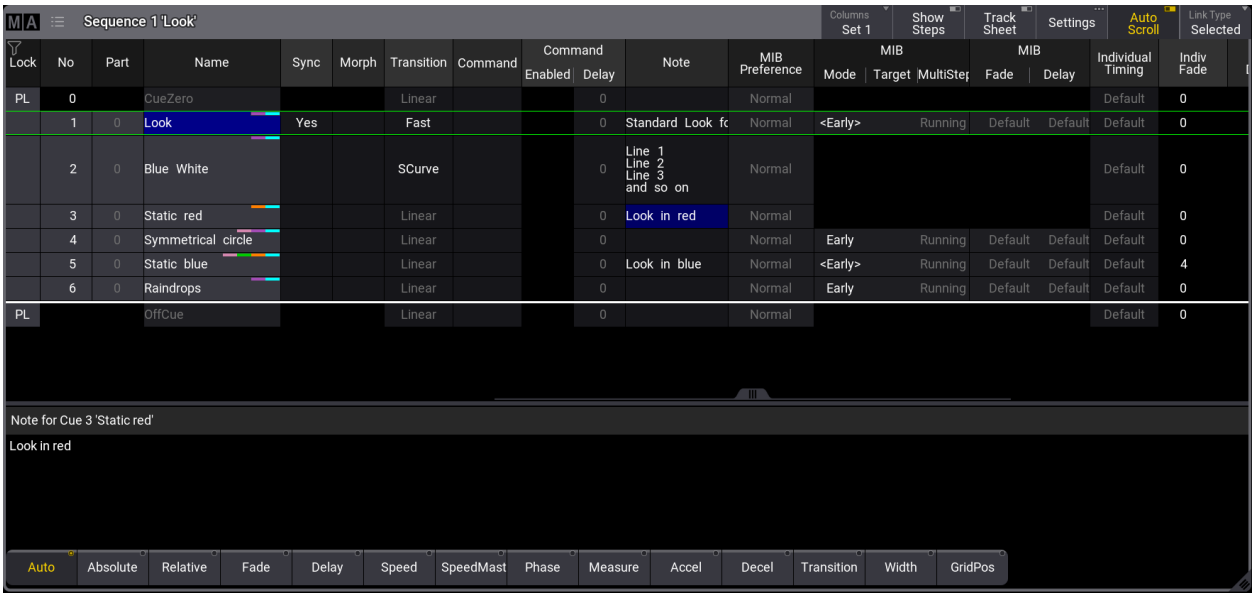
Show Notes

- **Requirement**: The Demoshow_grandMA3 is open.

Another way of displaying notes in the Sequence Sheet or Content Sheet is possible via Show Notes-tab. Show Notes can be enabled/disabled in the dedicated window settings. For more information about window settings, see **Window Settings**.

To enable Show Notes in the Sequence Sheet:

1. Open the Sequence Sheet Settings.
2. Tap the Mask-tab and then enable Show Notes. Close the window.
3. A text field with the note of the currently selected Cue is added to the Sequence Sheet as shown below:



Show Notes enabled in the Sequence Sheet

	Hint:
	When Show Notes is enabled, Show Recipes is disabled.

Important features of Show Notes:

- The text field can be used, for example, to show notes of a currently active cue, a preloaded cue, or a currently selected cue.
- The text field allows editing notes live.
- The text field shows information about the currently selected note. For example, "Note for Cue 1: Look" in the image above.

1.25. Label Objects

All objects can be labeled using the label pop-up. The label pop-up can contain a name, a note, a scribble, and an appearance. It differs depending on the type of object. For example, there are no scribbles available for labeling cues. All options are available for pool objects.

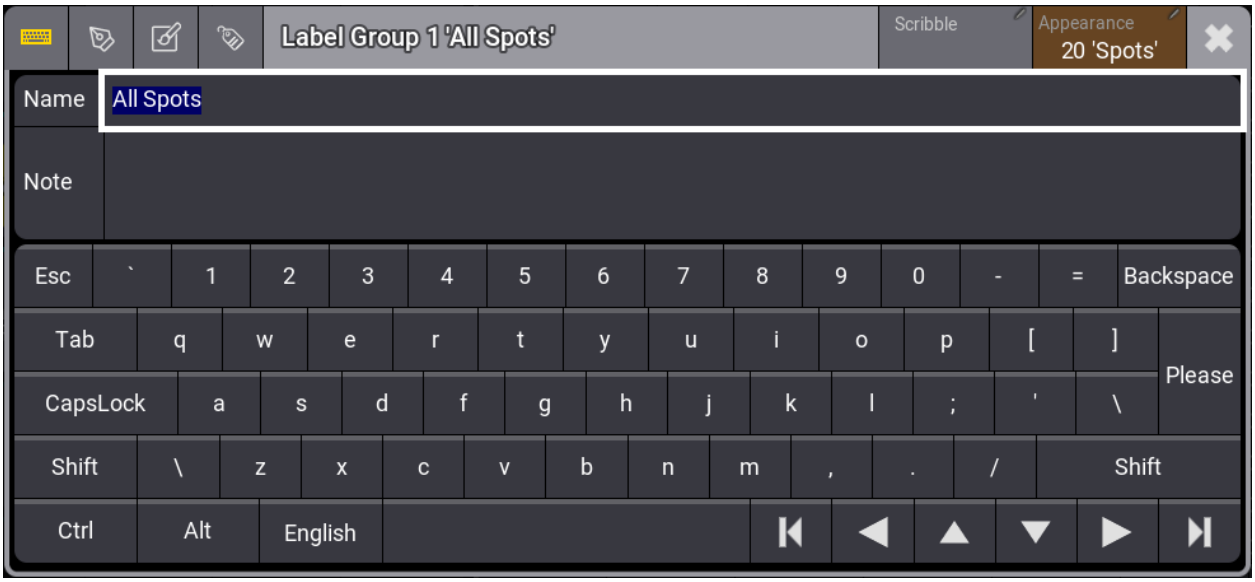
The name can be changed using the **Label** keyword. It can be written in the command line. Write **Label** followed by the object you want to name and then the name in quotation marks.

For instance, if group 1 needs to be named "Stage Right Spots" use the following command:

The label command can be accessed by pressing the Assign **key** twice. The syntax is the same no matter how the command is created:

Label [Object_Type] ["Current_Object_Name" or Object_Number] "New_Object Name"

If the object name is omitted in the command, an **Edit Label** pop-up appears:



Label pop-up for a pool object

The keyboard icon ( ) opens an on-screen keyboard. If the keyboard is visible, then the icon is yellow.

The **Scribble** icon ( ) opens the **Scribble pad**. If the Scribble pad is visible, then the icon is yellow. When starting to scribble in the blank scribble editor, a new scribble in the scribble pool will be created.

The **Appearance** icon ( ) opens the appearance editor. If the appearance editor is visible, then the icon is yellow. When starting to tap around in the blank appearance editor, a new appearance will be created in the appearance pool.

The tags icon () opens the tags pop-up to assign and unassign tags.

	**Hint:**
	On small screens all buttons behave as radio buttons. On big screens, the keyboard button is a toggle, the scribble, appearance and tags button behave as radio buttons.

Tap Scribble in the title bar to select scribbles in a dropdown menu. As soon as there is no scribble selected, Name and Note are grayed out.

Tap Appearance in the title bar to choose between all appearances in the appearance pool. The selected appearance is shown in the editor.

The label pop-up also has easy access to add a note to the object. Learn more in the **Notes topic**.

The last touched pool object has a white frame around it. If the keyboard is used, the keyboard text input is often interpreted as a new label input, which can be used to label all pool objects. This is the fastest way to label newly created objects.

Learn more about scribbles in the **Scribble section**.

Learn more about appearances in the **Appearance section**.

1.26. Groups

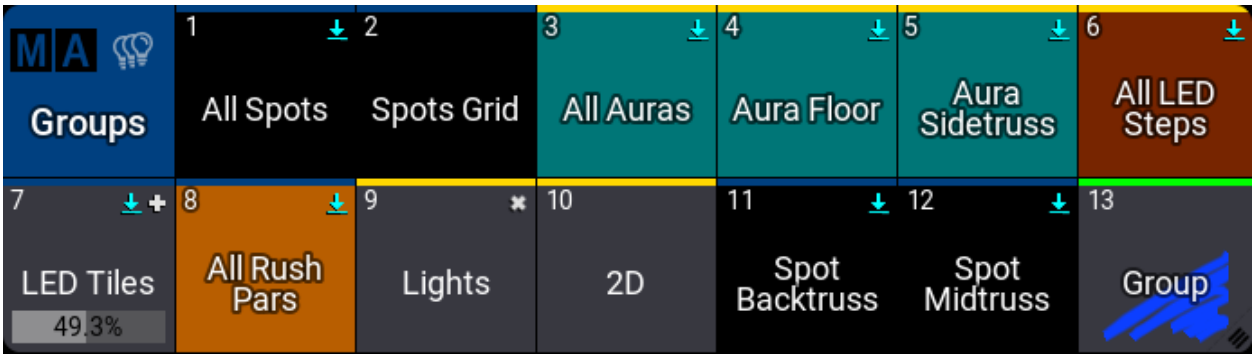
Groups contain a selection of fixtures. A group can contain one or many fixtures.

The fixtures' selection order and their grid position are also stored in a group. For more information, see **Selection Grid**.

Using groups is a fast way to select the fixtures.

Groups can also be used as masters. There are four different types of group masters. Read more in the **Group Master topic**.

Groups are organized in a pool - read more about pools in the **Pool Windows topic**.



Groups pool example

Information about the groups is not stored in cues or presets. Only the values applied to the fixtures are stored in cues and presets.

The next topics explore the creation of groups.

Subtopics

- **Create New Groups**
- **Edit Groups**
- **Delete Groups**
- **Group Masters**

1.26.1. Create New Groups

Groups are created by storing them in the **Group Pool** with a selection of fixtures.

The groups store the fixture selection, the grid information, and the fixtures' selection order (this is also a grid).

Grid information is 3D position information indicating the position relationship between the fixtures. It is not the location on the 3D stage. For more information, see **Selection Grid topic**.

The order and grid information are important for ranged value input or when creating **Phasers**.

	Important:
	Groups do not store values! - Only the fixture selection, order, and grid.

Use one of the following methods to store a group:

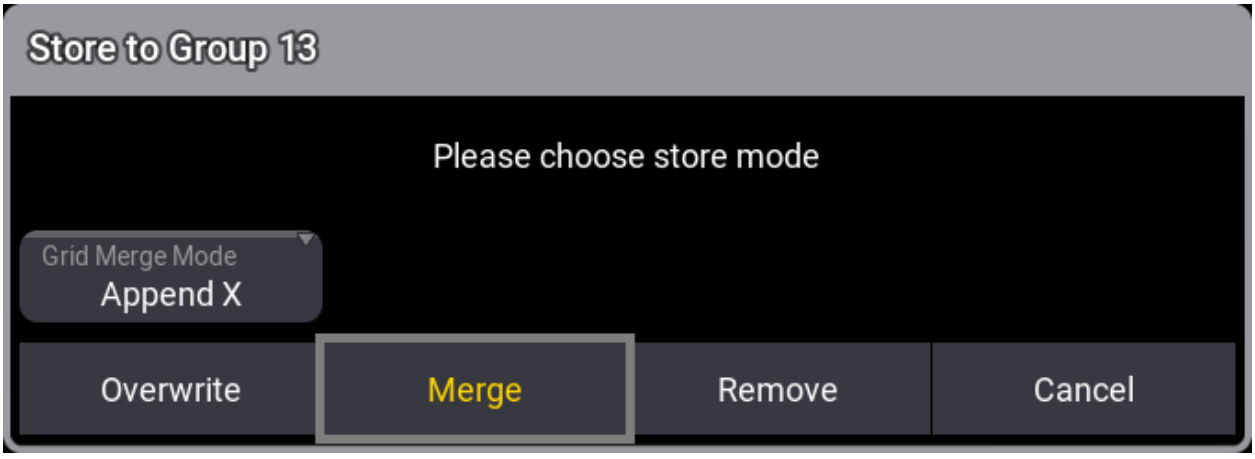
Please make sure the correct fixtures are selected, and that they are in the correct order.

1. Store the group using a syntax like this: **Store Group [my_group_number]**. The two needed keywords are **Store** and **Group**.

To store group 3, for example:

2. Press Store Group 3 Please

When storing to an existing group, the store mode pop-up opens:



Store mode pop-up.

When Grid Merge Mode is set to **Append X** (the default setting), the selected fixtures will be added to the next available X coordinate.

Read the **Selection Grid topic** for information about positioning the fixtures in a grid before storing the group.

Existing groups can be edited using the **Edit Group pop-up**.

Labeling the group is not required, but it is a good idea. You can do this using any method described in **Label pool objects** or the Edit Group pop-up.

1.26.2. Edit Groups

To edit a group, use one of the following methods:

- The **EditSetting keyword**.
- Press Edit twice and then tap a group pool object.
- Open the swipey commands on the pool object and choose Edit Setting.

The Edit Group pop-up opens.

Edit group pop-up.

The editor has two columns, **Label** and **Settings**.

Use the **Master** fader on the right to control one of the four group master types. Read more in the **Group Masters** topic.

Here is a description of the left buttons under the **Label** column:

Name:
The group name. To change a group name, use the **Label** command. For more information, see **Label keyword**.

Scribble and **Appearance:**
Can be assigned using these two buttons. For more information, see **Scribble** and **Appearances**.

Note:
For more information, see **Notes**.

Lock:
When toggled on, Name, Scribble, Appearance, Mode, and Move Grid Cursor are disabled.

Mode:
Tap to select one of the four group master modes. For more information, see **Group Masters**.

Move Grid Cursor:
For more information, see **Selection Grid** and **User Settings**.

Tap List References to open an **Info pop-up** that shows a list of the group's references and dependencies.

To add fixtures to a group, use one of the following methods:

- First Method:

1. Press Edit, then tap a group pool object.
2. Select the fixtures you want to add.
3. Press Update. The **Update** pop-up opens.
4. Tap OK to confirm the group update or Cancel to call off the operation.

Tap the video below to see the example.

- Second Method:

1. Select the fixtures you want to add.
2. Press Store.
3. Tap the group pool object to add fixtures. The **Store to Group** pop-up opens.
4. Tap Merge.

Tap the video below to see the example.

- Third method:

1. Open the swipey command on a group pool object and choose Edit Setting. The **Edit Group** pop-up opens.
2. Tap Edit Content.
3. Select the fixtures you want to add.
4. Press Update. The **Update** pop-up opens.
5. Tap OK.

Tap the video below to see the example.

To remove some fixtures from a group, use one of the following methods:

First method:

1. Open the swipey command on the group pool object. The **Edit Group** pop-up opens.
2. Tap Edit Content. The fixture sheet and selection grid frames turn green.
3. Press Off, then select the fixtures you want to remove from the group.
4. Press Update. The **update** pop-up opens.
5. Tap OK to save the changes.
6. Press Esc to exit the **Edit** mode.

Tap the video below to see the example.

Second Method:

1. Press Edit, then tap a group pool object.
2. Press Off, then select the fixtures you want to remove from the group.
3. Press Update. The **Update** pop-up opens.
4. Tap OK.

Tap the video below to see the example.

Third method:

1. Select the fixtures you want to remove.
2. Press Store, then tap a group pool object. The **Store to Group** pop-up opens.
3. Tap Remove.

Tap the video below to see the example.

1.26.3. Delete Groups

Delete a Group Pool object using one of the following methods:

- Using the **Delete** Keyword:

- Open the swipey command and select Delete.

Tap the video below to see the example.

- Press Delete and tap a pool object.

Tap the video below to see the example.

Deleting groups does not affect programmed cues or any other element. The only exception is if the group is assigned to an executor directly and used as the base for a Group Master. In that case, the executor will be empty after deleting the source group. Read more in the **Group Master topic**.

1.26.4. Group Masters

Groups can be masters of the fixtures in the group. For more information, see **Masters**.

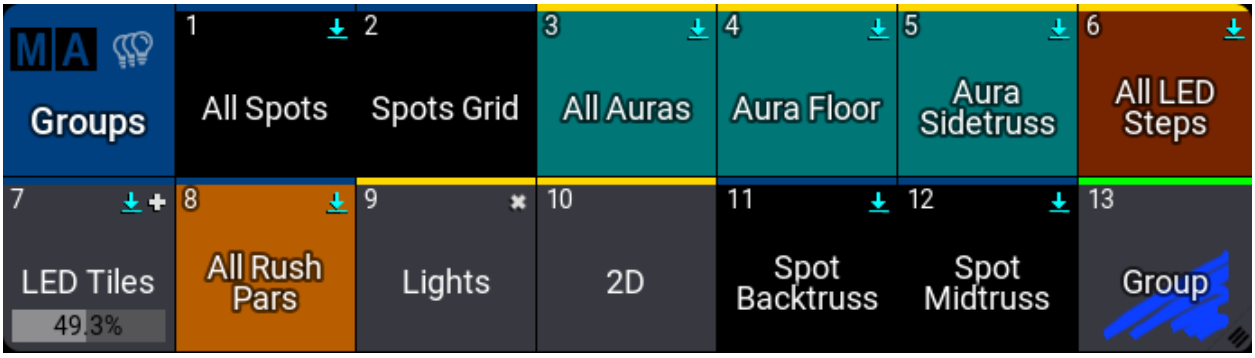
If the value is adjusted often, assigning the group to an executor can be advantageous. Learn how in the **Assign Object to an Executor topic**.

There are four different kinds of group masters:

- **Positive** - This is an HTP group master, which is indicated by + in the upper right corner of the pool object. The output of the fixtures is reduced depending on the value of the group master.
- **Negative** - This is a LoTP group master, indicated by − icon in the upper right corner of the pool object. The output of the fixtures is reduced depending on the values of the group master.
- **Scaling** - This scales the intensity output. Scaling is indicated by ✖ icon in the upper right corner of the pool object. For example, if the fixture is at 50% and the group master is at 50%, then the output is 25%.
- **Additive** - This master does not limit output but adds output as HTP merges with values from the programmer and playbacks. It is indicated by a small circle icon in the upper right corner of the pool object.

Difference Between Positive, Negative, and Scaling

The difference between the positive and negative masters is relevant when several groups contain the same or overlapping fixtures. A negative master has priority over a positive. For example, if two groups contain a fixture with 100% output, and if one group is a positive master at 80% and the other group is a negative master at 60%, then the output is 60% (limited by the negative master). If the negative master is turned up, the output stops at 80% when the positive master becomes valid. If you switch both groups to Scaling, the output will be 48% since the fixture output of 100% is multiplied by 60% and 80% of the second group master.



Group pool with masters

Groups with a mode assigned have a horizontal bar that displays the master level if it is relevant.

The master mode can be selected using the **Edit Group pop-up**. Read more about the pop-up in the **Edit Groups topic**.

The mode can also be assigned using the **Set keyword**.

Example

Using the command line to set the master mode of group 4 to positive:

```
User name[Fixture]>Set Group 4 Property "Mode" "Positive"
```

The mode type is capital sensitive. This means that writing "positive" fails, but "Positive" works.

	**Hint:**
	A pop-up appears if the mode is not specified, allowing you to choose a mode from a list.

Delete a Group Master

Deleting a group assigned to an executor does not delete the group from the pool.

If a group is deleted from the group pool, it is deleted from the show.

	**Restriction:**
	If the group is limited by the master, when it is deleted, then the fixtures stay limited. This means a new group must be created with the fixtures to remove the limitation! If the group is removed from an executor, then it still exists in the group pool, and any master level set using the executor is still valid in the pool.

1.27. Presets

A preset can hold information about attribute and timing values for a selection of fixtures and may be referenced and re-used in cues or in the programmer.

The principle of presets is to store a labeled reference in a cue, rather than the actual value itself. Or use presets for busking, where presets are called into the programmer or played back on the fly.

Updating the preset means that the cues do not need to be updated since they reference the preset and not the actual preset content.

Presets are marked with a cyan marker. This maker is visible on cues that use preset and on the values themselves, for instance, in the tracking sheet or the fixture sheet. Read more in the **Marker topic**.

	**Hint:**
	Using presets when programming facilitates the work, especially when working with a show which is used in different locations and for various customers.

Preset Pools

Presets are stored in pools. There are pools for each feature group. These pools have a default input filter that only allows the values for that feature group to be stored in the preset pool. There are also five preset pools called "All 1" to "All 5". These do not have a default input filter and can be used to store any attribute values. Learn more about the pools in the **Preset Pool topic**.

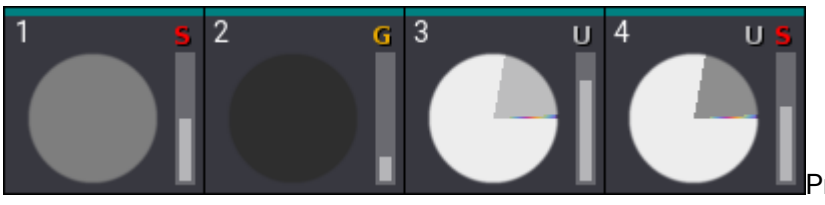
Preset Modes

Preset modes are used when the preset is stored (or updated) and when the preset is called.

There are three different preset modes: Selective, Global, and Universal.

Each preset got letters showing the preset mode.

Example:



Preset 4 has all three modes stored but only shows U and S.

The three modes are:

- **Selective (S)**:
  The data will be added as selective data for each fixture that has active data in the programmer.

- **Global (G)**:
  The data will be added mainly as global data. If there are several fixtures of the same fixture type but with different values, then the global data will be determined by average, and selective data will be added for the other fixtures which have divergent data.
- **Universal (U)**:
  Data will be added as global data, and the preset mode will be set to Universal.

When a preset is stored or updated, then there are two more options:

- **Auto**:
  When updating or storing into an existing preset, the preset mode of the preset will be respected.
  In the case of global preset mode, selective data will be added to the preset when at least one fixture that can use the preset is active with new values.
  When creating a new preset Auto mode will take the mode defined by the pool and use the rules for each mode.
- **Force Global**:
  Data will be added as global data, and untouched existing selective data will be discarded within the preset for the fixtures of the same fixture type.
  Force Global will discard the selective data when updating a preset or when storing with the merge option into an existing preset.

Each preset can store all three modes or some combinations of modes.

Each preset pool has a setting for a default mode. This mode is indicated by one of the letters in the upper right corner of the pool title field.

Position preset pool title field with Selective as the default mode for the pool

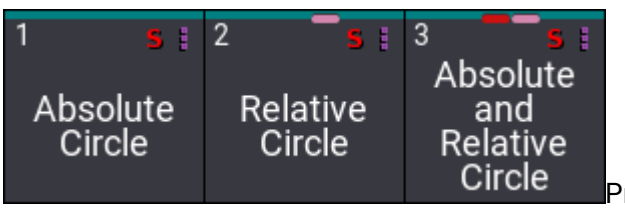
This default can be used when storing the preset or a different mode can be used.

Learn more in the **Create New Presets topic** and the **Use Preset topic**.

Absolute and Relative Values

Presets can store absolute and/or relative values. Relative values are often used with multistep presets but it is not limited to this use.

Example:



Preset with absolute, relative, and both types of values

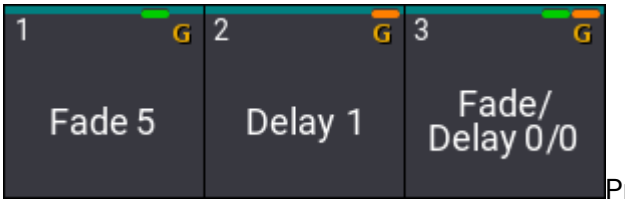
Presets with only absolute values do not have a marker. The absolute marker is a square dark red marker with rounded corners. Relative values are indicated by a dark pink square with rounded corners. Presets can contain both absolute and relative values - they will have both markers in the pool object.

Learn more about the different programmer layers in the **What is the Programmer topic**. Learn more about the markers in the **Markers topic**.

Timing Values

Timing values can be stored in presets. The timing layers are Fade and Delay.

Example:



Preset with fade, delay, and both types of values

The Fade marker is green. The Delay marker is orange.

Timing values can be combined with absolute and relative values.

The timing preset can be used when calling values into the programmer.

A timing-only preset cannot be stored or referenced in a cue. Suppose an attribute value (absolute or relative) is called together with a timing-only preset and stored in a cue. In that case, the timing information is stored as individual timing without a preset reference.

Presets that contain both attribute values and timing values are stored in cues with a preset reference.

Pool Action and Object Action

A preset pool window has a pool action for the pool objects. The individual pool objects can also have an object action. The object action is used if the **Use Object Action** is active in the **Preset Pool Settings** (link below).

The defined pool or object action is performed when the preset is tapped in the pool without a (relevant) keyword in the command line.

A small icon in the upper right corner of the pool title object indicates the selected pool action. If **Use Object Action** is activated in the settings, a plus icon will be displayed next to the pool action. See the example image below.

An icon overlay is added to the pool object when **Use Object Action** is activated. The pool object is grayed out if the object action is set to **None** (see preset 10 in the example below). There is no icon overlay displayed if the object action is set to **Pool Default** (see preset 11 in the example below).

Example of a dimmer preset pool with pool objects assigned all the possible default actions.

The pool action is set in the **Preset Pool Settings**.

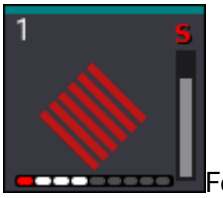
The object action is set in the object editor described in the **Edit or Update Presets topic**.

The functions are also described in the **Window Settings topic**. Follow the links to learn more.

Feature Group Indicator Bar

This bar is visible when a preset has attributes of two or more feature groups stored.

The bar is always visible on All presets.



Feature Group Indicator Bar at the bottom of the preset

The feature group indicator bar has a square for each feature group, displayed in white, gray, or red color.

- **White**:
  Indicates that values of attributes of the corresponding feature group are stored in the preset.
- **Gray**:
  Indicates, that no values of attributes of this feature group are stored in the preset.
- **Red**:
  Indicates that at least one attribute of this feature group is not accessible at the moment. For example, when having a world with a limited set of attributes active.
  The square turns also red when the currently selected world does not contain fixtures that can make use of the attribute data in the preset.

The order of the squares is similar to the order of feature groups in the encoder bar, from left to right:

- Dimmer
- Position
- Gobo
- Color
- Beam
- Focus
- Control
- Shaper
- Video

If a show file contains additional feature groups, more squares will be added automatically to the bar.

The bar can also be visible in **Worlds and Filters**.

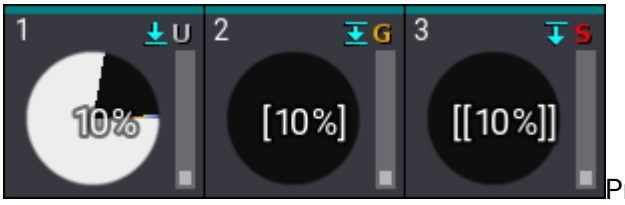
Embedded Presets

A preset can contain a link to a different preset. It is the same as cues storing a reference to a preset - just for other presets. This can be useful for instance when a show is built using a set of general presets and other presets are created using these as building blocks.

Presets can be embedded to a depth of up to 10 levels.

Example:

In this example, there is a universal dimmer preset that has values for the universal fixture. This is used to create a new global preset for fixture types. This is again used to create a selective preset for only some fixtures. If the fixture type changes, then the global preset needs to be updated. The selective preset links to the global preset values and do not need to be updated. The universal fixture is unaffected by the fixture change.



Preset 1 is used in an embedded preset. Preset 2 contains embedded data and is also used in a different embedded preset. Preset 3 contains embedded data.

The "source" preset has a downward pointing arrow that points to a line. This indicates that this preset is referenced by other presets (or cues). The preset with embedded data has an icon with an arrow and a line above to indicate that this preset uses referenced data. Presets with embedded data that is also referenced has the arrow and a line above and below the arrow.

If a new name is not defined, then the name is also referenced (referenced names are in square brackets). This means that changing the name of the source preset also updates the name of the new preset. Learn more in the **Create New Presets** topic.

Recipe Presets

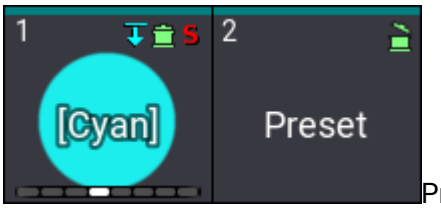
Presets can have recipe information. A recipe is one or more lines with information about a selection, value, MAtricks, individual fade, delay, speed, and phase values.

This information can be used to "cook" values into the preset or programmer. If the source information changes, the preset can be cooked again to reflect the changes. For instance, if a group is used in a recipe and it changes after the initial cooking, then the preset can be cooked again, and the changes will now reflect the group's changes.

A preset can be a **Recipe Template** that can be used to load the recipe into the programmer.

Learn more in the **Recipe Presets** topic.

Example:



Presets with recipe information

A small pot icon indicates a preset with recipes. Recipe Templates have a pot icon with an open lid.

Recipe presets can contain presets with recipes. There can be up to 10 levels of recipes within recipes.

A circular reference can be created between recipe presets. If that is the case, there is a small red infinity icon on the presets.

Example:

Presets with infinite circular references

MAgic Presets

MAgic presets are presets where value points in a range are defined. This range can then be applied to a dynamic selection of fixtures. For instance, two fixtures are stored in a MAgic preset with dimmer values of 0% and 100%. MAgic presets should be stored as a selective preset, but that does not mean that only the selected fixtures can use it - it only means that they contain values for a selection of fixtures. The MAgic preset uses the values to define the points in a range. A different fixture selection can be made and the range of values between the points is assigned to the fixtures when the MAgic preset is called into the programmer. These values are calculated based on the points and then taken into the programmer as hard values. There is no reference back to the MAgic preset.

There can be up to five defined points in the range (on each axis in the selection grid).

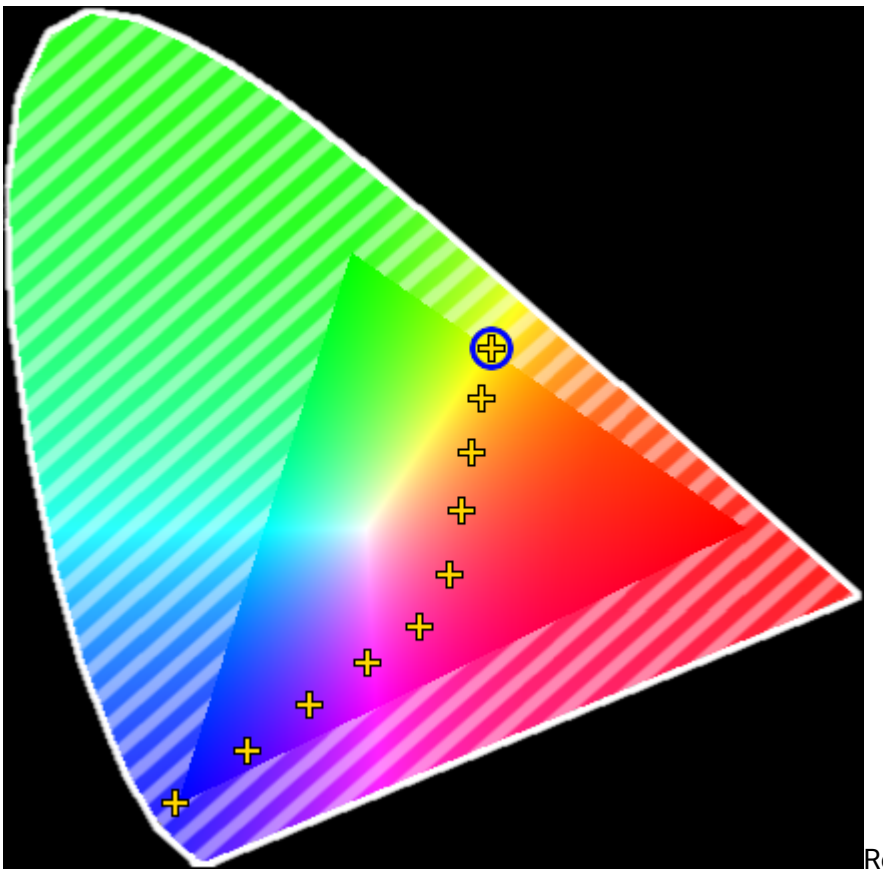
MAgic presets have a small wizard hat icon in the preset pool object

Position preset with MAgic information

Example:

A path needs to be defined to control a color range from blue to yellow. The wish is to have the range go through the red area in a CIE color picker - instead of going through white. Three fixtures are needed to define the range.

The first fixture is blue, the second is magenta, and the third is yellow. This is then stored as a MAgic preset. Now this range through three points can be used by multiple fixtures to create the desired path.


Result of a three-point MAgic color preset applied to multiple fixtures

Learn the details on how to create this MAgic preset in the **Create New Presets** topic.

Multistep Presets

Presets can contain values in multiple steps (Phasers). Presets with only one step usually have the values stored in step one. If a preset has multiple steps then they are called multistep or Phaser presets.

Example:

Multistep preset

They have a three-dot icon to indicate multistep values.

MAtricks Presets

MAtricks information can be stored in presets. Having MAtricks information in the programmer and storing a preset will add the MAtricks information into the preset (if the **Store Settings** allow it).

Example:

Preset with MAtricks information

The MAtricks icon is a green icon with nine dots in a grid.

Filtered Presets

The entire preset pool has a default input filter - except the All preset pools. The default filter is on feature group preset pools. They automatically filter the attributes in the feature group.

The input filter can be changed to a custom filter for any preset pool, including the All presets. If the input filter is different than the default, then there is an input filter icon on the preset pool title field.

Besides an entire preset pool having an input filter, it is also possible to have an input filter on individual presets. Learn more about input filters in the **Edit Presets** topic.

Example:

Preset with input filter

The input filter icon is a gray filter icon with a small right-pointed arrow.

Extract Preset Data

The values stored in presets can be extracted from the preset. This will take the values stored in the preset and put them into the programmer. There will not be a link to a preset when the values are extracted.

This can be done using the **Extract keyword**.

Different sources can be specified with the extraction.

If there is a selection of fixtures with references to presets in the programmer, then the entire selection can be extracted. The command for this would be: **Extract Selection**

Specific fixtures can be specified instead of extracting the entire selection. It can be single fixtures or a selection of fixtures: **Extract Fixture [Fixtures_Numbers or "Fixture_Names"]**

Specific presets can also be extracted into the current selection of fixtures: **Extract Preset [FeatureGroup_Number or "FeatureGroup_Name"].[Preset_Number or "Preset_Name"]**

All presets from a cue can also be extracted to a selection of fixtures: **Extract Cue [Cue_Number or "Cue_Name"]**

	**Hint:**
	When extracting by specifying a fixture, selection, cue, feature group, or attribute, presets must be active in the programmer for the desired fixtures.

When extracting embedded presets or phaser presets which have presets integrated into their steps, extract will call directly the values of the source presets.

Using the **/Single** option together with extract makes it possible to extract one level down in the hierarchy of the presets.

Example

Create 2 color presets ("red" and "blue" presets) that are embedded into a different color preset ("odd/even red/blue"). The second color preset ("odd/even red/blue") is embedded into All preset 21.1 ("cool look").

Select some fixtures and apply the All preset ("cool look") to them.

Executing **Extract Selection /Single** calls the second color preset ("odd/even red/blue") into the programmer.

Executing **Extract Selection /Single** again calls the first color presets ("red" and "blue" presets) into the programmer.

When executing **Extract Selection /Single** a third time, the result is the same as using **Extract Selection** in the first step: The hard red and blue values without a preset link.

Subtopics

- **Preset Pools**
- **Create New Presets**
- **Recipe Presets**
- **Use Preset**
- **Edit or Update Presets**

1.27.1. Preset Pools

	Important:
	If pools are new to you, please read the **Pool Windows topics** first.

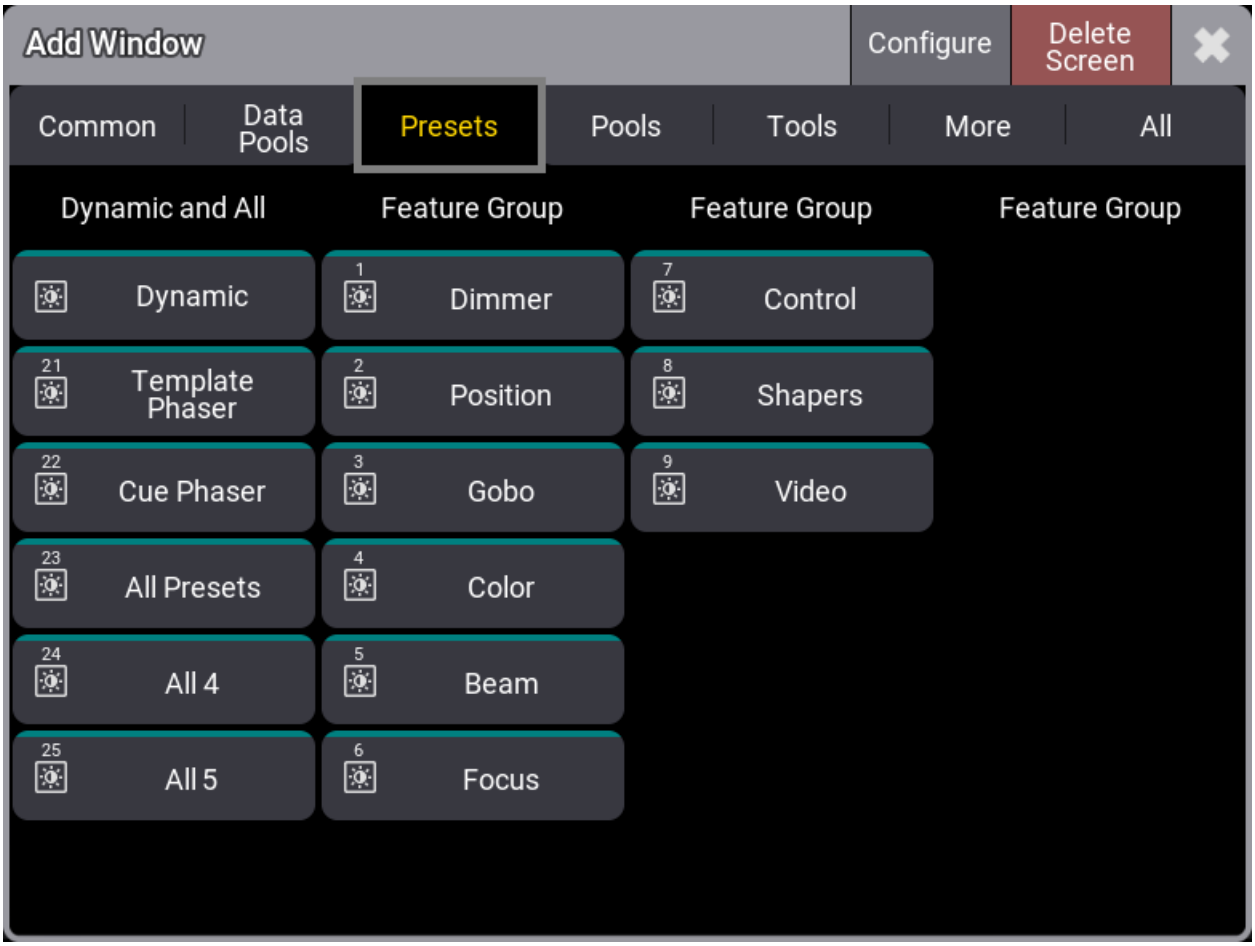
There is a preset pool for each **Feature Group** in the show. A show with no custom feature groups will have nine preset pools named after the default feature groups. For instance, Dimmer. These pools can have a feature group filter besides the standard input filter (read more below).

	Hint:
	Create custom preset pools using Feature Groups. For more information, see **Feature Group**.

Besides the feature group preset pools, there are five "All" preset pools. These do not filter based on feature groups. The "All" preset pools can be labeled to match the needs. For instance, one of the pools could be used for storing multistep phaser presets or different parts of the show.

There is a unique preset pool called "Dynamic". This is not a preset pool in itself. It automatically changes between the feature group preset pools based on the selected feature group in the **Feature Group Control Bar**.

The preset pools are created like any other window in the user-defined areas using the **Add window pop-up**. They are all under the **Presets** tab:



Open the Presets

The preset pools look and behave like many other pools, but small differences exist. Read more about pools in general in the **Pool Windows topics**.

The default preset mode for the preset pool is indicated by a letter in the upper right corner of the pool title field. Read more about the preset modes in the **Preset topic** and below for information about setting the default.

There is a colored indicator bar at the top of each pool object. If nothing is selected, it is colored like the pool color. The pool object is dimmed if there is a selection, but none of the currently selected fixtures can use a preset. If a preset can be used by all the fixtures currently selected, then the colored bar is green. It is yellow if the preset is only valid for some of the selected fixtures. These are the default colors. They can be edited in the pool settings (see below).

Presets in the All 4 pool

The cyan number below the pool number is the number of fixtures currently using the preset. See preset 2 in the example image above.

Preset Pool Settings

Enter the pool settings by tapping the MA logo in the pool title field.

This is an example of the preset pool settings.

Settings for the All 1 preset pool window

These are the settings for preset pool window:

- **Name**:
  Some pools can be named, for instance, the five All preset pools. The other preset pools are named from the feature group.
- **Input Filter**:
  This can be used to select an input filter for the entire pool. **Worlds and Filters** can be used. An input filter blocks some elements. The blocked elements cannot be stored in the pool. Tap this to open an **Assignment Editor**. Here filters or worlds can be selected, or it can be set to Empty.
- **Show Empty**:
  This toggle button can hide or show empty pool objects.
- **Appearance**:
  The appearance is applied behind the pool objects.
- **Pool Columns**:
  This defines the width for the pool objects. It does not change the size of the window. It defines how many columns of pool objects are in the window. If the window is wider than the number of columns, then the extra space is displayed as black (default color). If the window is smaller than the number of columns, the pool window can be scrolled horizontally. If the pool has a set width, then there is an icon (⏭) in the upper right corner of the title field.
  The **Not Defined** value dynamically sets the width to match the window size even when the window is resized.
  The **Take Current Width** sets the width to match the current size of the window. It does not dynamically change if the window is resized.
- **Display Mode**:
  This is used to define what the pool object displays. It has the following options:

    - **Text and Symbol**:
      Both the text (name) and symbol are shown on the pool object if they are different than the default values and not empty.
    - **Text**:
      Only the pool object's name is shown. If the pool object only has the default name, it is not shown.
    - **Symbol**:
      Only the symbol is shown. If no symbol is generated, then the pool object is empty.
    - **Auto**:
      This results in only the symbol being displayed if a symbol exists. Otherwise, the text (name) is displayed.
- **Font Size**:
  There are some different font size properties from 10 to 32. There is also a default property. This is the same as size 18. This simply changes the font size on the pool objects.
- **DataPool**:
  This defines what data pool the pool window shows data from. This makes it possible to have pools showing objects from different data pools. For instance, a group pool window from the default data pool can be shown next to a different group pool window showing groups from a different data pool.
- **Pool Color**:
  This is the color for the title button in the pool.
- **Empty Color**:
  This color is applied to empty pool objects.
- **Reset Colors**:
  This resets the colors to the colors in the default color theme.
- **Preset Mode**:
  This sets the default preset mode when storing into the entire pool. A different mode can always be specified when storing. Learn more about preset modes in the **Presets topic**.
- **For All**:
  This color is used when the preset can be used by all of the selected fixtures.
- **For Some**:
  This color is used when some of the selected fixtures can use the preset.
- **For None**:
  This color is used when the preset is not usable by any of the selected fixtures or when none of the selected fixtures are in the group.
- **Cue Part**:
  By default, presets call their values into programmer part 0. Use this setting to specify a different programmer part for presets for this pool. This setting can also be changed for individual presets in the pool. Learn more about programmer parts in the **What is the Programmer topic**.
- **Pool Action:**
  This defines the default action executed when a pool object is tapped without a (relevant) keyword in the command line.
  Pools can have some of the following actions (the available actions depend on the type of pool):
    - **At** ():
      When there is no selection in the programmer, tapping a preset does nothing. When the programmer has a selection, tapping it calls the preset into the programmer.
    - **Call**() - default action for filters:
      This action calls the tapped pool object.

- **SelFix/At** (no icon) - default action for presets:
  When there is no selection in the programmer, tapping a preset will select the fixture that can use the preset. Tapping it again calls the preset into the programmer.
  When the programmer has a selection, tapping a preset the first time will call the preset into the programmer.
- **SelFix/Extract** ():
  This value acts similarly to SelFix/At, but instead of calling the preset reference into the programmer, the values will be called extracted into the programmer.
- **Select** () - default action for sequences:
  Tapping the pool object selects it.
- **Toggle** ():
  Tapping a pool object plays it back or switches it off, depending on its current playback state.
- **Go+** ():
  Starts playback of the pool object or goes to the next cue in the sequence.
- **Flash** ():
  Flashes a pool object as long as it is tapped. Flash ignores fade times.
- **Temp** ():
  Plays back a pool object as long the pool object is pressed. Temp respects the fade times.

- **Use Object Action**:
  When enabled, the selected object action is executed instead of the selected pool action.
  Pool windows with Use Object Action enabled, are marked with a (**+**). Pool objects indicate the selected object action setting with a light grey icon in the background of the object.
- **2 Finger Edit**:
  This toggle button enables the possibility to edit objects by using the two-finger gesture or right-clicking.

The settings described above (except the preset mode) can also be set for individual presets. The individual preset settings have a higher priority than the preset pool setting.

1.27.2. Create New Presets

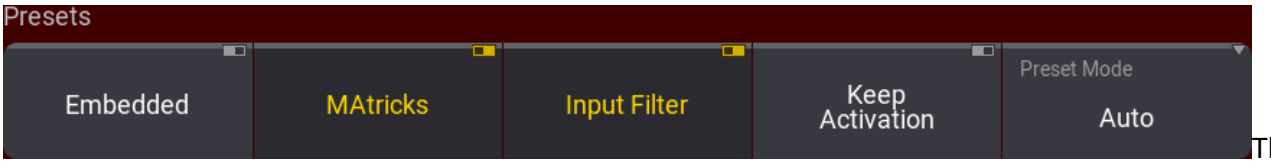
Please read the general **Preset** topic and the **Preset Pool** topic before this topic to get an understanding of the presets and how they are organized.

Creating and using presets requires some fixtures patched in a show file or the universal fixture in the patch for general universal presets.

For more information on patching fixtures, see **Add Fixtures to the Show**.

When presets are stored, some defaults are used unless other choices are actively made.

Some of these choices are accessible in the **Store Options** pop-up. This is opened by keeping Store pressed for approximately one second. The pop-up has a section regarding the presets:



The preset section in Store Options pop-up

	Hint:
	Changing a setting here without saving this as the new preference only uses the changed setting for the next store action. Learn more about it in the **Store Options and Preferences topic**.

The preset mode is one of the options. The preset pool has a default setting, but a different mode can be chosen by tapping Preset Mode until it has the desired mode. Learn about the presets modes in the **Presets** topic and the **Preset Pools** topic.

Keep Activation means that when the preset is stored, the newly created preset is active in the programmer and ready to be stored in, for instance, a cue. Turning this setting Off stores the preset but leaves the values as inactive in the programmer.

MAtricks allows storing MAtricks settings in the preset.

Read about **Embedded** and **Input Filter** below.

Input Filter

The I**nput Filter** setting in the store options determines whether input filtering in preset pools is used. If this option is active, then it is only possible to store attribute values inside the preset pools if the filter allows it. If the input filter setting is empty, a feature group filter is applied to the relevant feature group preset pools. This means that only attributes of the corresponding feature group can be stored in the preset pool. For instance, pan attributes can be stored in the Position preset pool but not in the Color preset pool.

All preset pools do not have any feature group filtering. Any attribute can be stored, regardless of the Input Filter setting in the store options.

Custom filters and worlds can be assigned to the preset pool or individual presets using the preset settings or individual preset options. Filters and worlds can also be assigned using the following syntax: **Assign Filter [filter_ID] At Preset [preset_ID]**. Substitute filter with world if a world needs to be assigned as an input filter.

Learn more about filters and worlds in the **Worlds and Filters** section. Learn about the preset pool settings in the **Preset Pools** topic.

Simple Static Preset

Storing a preset with simple static values is very easy:

1. Select some fixtures.
2. Give some of the fixture attributes a value in the programmer.
3. Store this in the relevant empty pool object.

This is the general workflow.

If the programmer values are allowed to pass the input filter (read above), they are stored in the preset.

The best way to work with presets is to have the desired preset pool available on a touch screen. The pool can then be tapped to store and call the presets.

If it only contains values in **step** one, it is considered a static preset, which means that the values do not change. Read the **Phaser** topic to learn what the steps are.

Global Preset Data

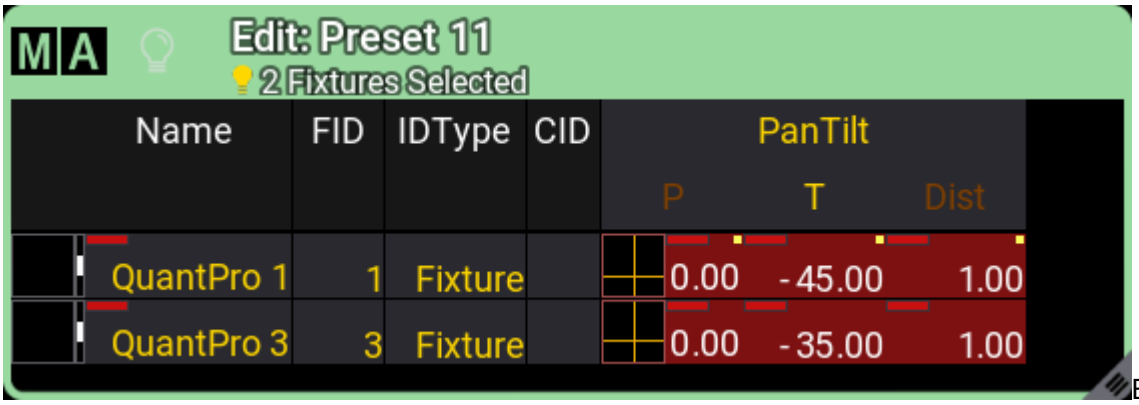
The global data is stored on real fixtures within the patch.

When storing new global presets, the global data is automatically created within the new preset. When all data for the fixture type is the same, the first fixture from the stored data is used to hold the global data within the preset.

Each fixture type gets its own global data value so that the global value can be adjusted per fixture type.

When selective data is added to the preset for one of the fixtures that holds the global data or when the fixture is deleted from the show, the global data is moved to the first patched fixture for the fixture type. If the global data was held by the first fixture before, it is moved to the next patched fixture of this fixture type.

When editing a preset with global data, the fixtures with the global data in the programmer will display a yellow square marker in the top right corner of the attribute cell within the fixture sheet.



Example showing a preset with fixture 1 holding global data and fixture 3 holding selective data.

	Important:
	When converting old show files from grandMA3 v1.5 or prior to grandMA3 v1.6 or later, the data from the global fixture type object will be migrated to the first patched fixture of the same fixture type that is not already holding selective data.

The global value is determined by the average across all values of attributes within the same activation group:
(Value of fixture 1 + Value of fixture 2 + Value of fixture 3 + ... value of fixture n) / Number of used fixtures = Average value
The value closest to the calculated average value will be the global value. The fixture with this value will then hold the global value. In the case of color, all color attributes are handled together when choosing the global value.

Furthermore, the method to determine the global data across several attributes is now considering all attributes of the same activation group. This will result in taking all attributes of the same activation group of one fixture and not taking the different attributes from different fixtures.

Universal Preset Data

Storing a universal preset using a real patched fixture will store global data for active fixture types when the preset was stored.

Universal presets are more versatile and can use the data from a real fixture with global values when the **Universal Fixture** has no data inside the preset.

This has the added benefit of creating universal presets using a real fixture from the stage if the universal fixture has these attributes.

When calling a universal preset, the software first uses global fixture type data and then uses the data from the universal fixture. When no data exists for the universal fixture, the first fixture with global data is used instead.

	Hint:
	When calling universal color, it will be transformed through the color engine. If, for instance, a 7-color LED is used to create a color and call that data universal, a similar color will be called on an RGBA fixture.

Embedded Preset

The concept of embedded preset is explained in the **Presets** topic.

This is the workflow:

1. Select the desired fixtures.
2. Recall an existing preset so the preset is in the programmer.
3. Store a new preset with the Embedded store option active.

A recipe preset could be an alternative to an embedded preset. Learn more about them in the **Recipe Presets** topic.

Store Selective Preset

1. Select one or more fixtures.
2. Give the fixture attributes some values in the programmer.
3. Press and hold Store until the **Store Settings** open.
4. Tap Preset Mode until the mode is **Selective**.
5. Tap the desired preset pool object - ensure it is valid for the attribute values.

Store Global Preset

Attribute values need to be tagged with a global flag to be stored as global data.

1. Select one fixture from one or more fixture types.
2. Give the fixture attributes some values in the programmer.
3. Press and hold Store until the **Store Settings** open.
4. Tap Preset Mode until the mode is **Global**.
5. Tap the desired preset pool object - ensure it is valid for the attribute values.

Store Universal Preset

1. Select one or more fixtures or the universal fixture.
2. Give the fixture attributes some values in the programmer.
3. Press and hold Store until the **Store Settings** open.
4. Tap Preset Mode until the mode is **Universal**.
5. Tap the desired preset pool object - ensure it is valid for the attribute values.

Values that are a part of the universal fixture type are stored in this fixture. Attributes outside the scope of the universal fixture type are stored as global values.

MAgic Presets

The concept of MAgic presets is explained in the **Presets** topic.

	Important:
	The MAgic presets only work if the stored fixtures have different grid positions on the relevant axes, and the best result is if they start at grid position 0 and the fixtures are next to each other. This does not need to match the real-world position; it is only for the individual positions between

	the fixtures.
	**Important:**
	Calling a MAgic preset in the programmer does not create a link to the preset. The fixture values are calculated based on the selection and stored as hard values. The only way to keep a reference to the MAgic preset is if the preset is used in a recipe. Learn more about recipes in the **Recipe Preset topic**.

This is the general workflow for creating a MAgic preset:

1. Select all the needed fixtures.
2. Use Next to give the first fixture values to match the first point in the range.
3. Press Next to select the next fixture and give it values for the next point in the range.
4. Repeat step 3, if needed, to have a maximum of five fixtures (on each axis in the selection grid).
5. Select all the used fixtures using Set.
6. Store the preset in an appropriate preset pool.
7. Edit the preset settings and tap MAgic to turn the MAgic function On.

The new preset can now be used by as many fixtures as needed.

Learn more about editing the **Edit Presets topic**.

	**Known Limitation:**
	Storing MAgic preset only works with selective data as multiple points cannot be defined across universal or global fixtures.

Example:

A path must be defined to control a color range from blue to yellow. The wish is to have the range go through the magenta/red area in a CIE color picker - instead of going through white or the blue/green area. Three fixtures are needed to define the range.

Requirement:

- Patch several fixtures (10 or more) with color mixing possibilities.
- Create a view with the Special Dialog - Color Picker (in CIE mode) and an All preset pool.

Follow these steps:

1. Select three of the fixtures.
2. Turn the fixture intensity to 100%.
3. Press Next to select the first of the three fixtures.
4. Tap the blue area in the color picker.
5. Press Next to select the second fixture.
6. Tap the magenta area in the color picker.
7. Press Next to select the third fixture.
8. Tap the yellow area in the color picker.
9. Press Set to select all three fixtures again.

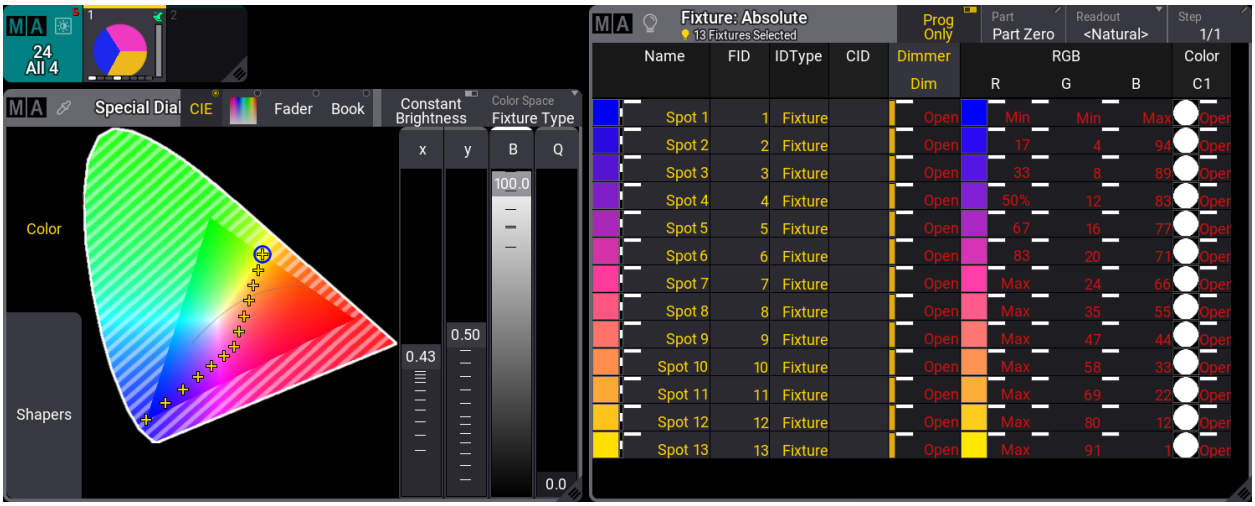
10. Press Store.
11. Tap an empty pool object in the All preset pool.
12. Use the **swipey menu** to select Edit Settings.
13. Tap MAgic to turn On the option and close the settings.

Now, there is a preset that looks something like this:

MAgic preset in the preset pool

14. Clear the programmer.
15. Select all the fixtures.
16. Tap the MAgic preset.

The result should look similar to this in the color picker:



Result of the MAgic preset used on multiple fixtures

The dimmer values are not needed for this example; it just makes it easier to see the result.

Multistep Preset

Multistep presets are presets that contain values in more than one step. It is explained in a little bit more detail in the **Presets** topic. For an explanation of steps, please read the **Phasers** section.

With Phaser information in the programmer, a preset can be stored. This will then be a multistep preset.

	Hint:
	Presets can be used for more advanced functions with Recipes. Please read the **Recipe Presets topic** for more about recipes.

1.27.3. Recipe Presets

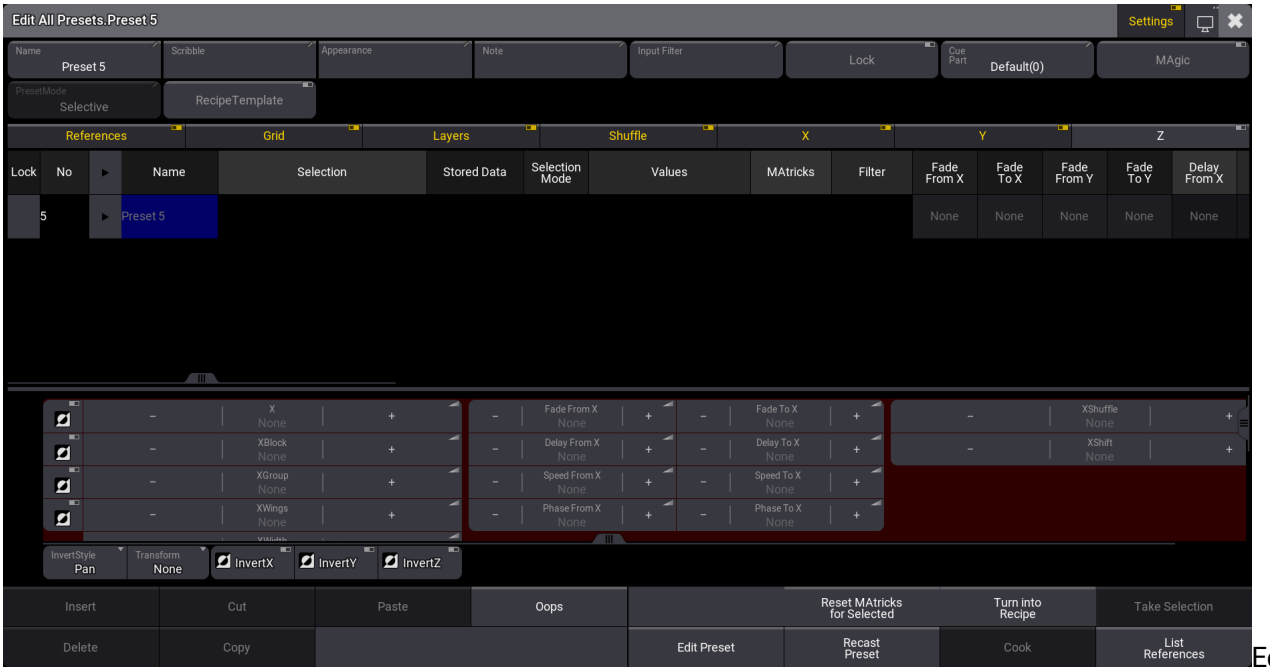
Please read the **Recipe topic** before this topic.

This topic explores using recipes in presets.

Recipes can only be added to presets that do not contain attribute values when using the user interface. Presets with attribute data offer to turn the existing attribute data into a recipe (by tapping Turn into Recipe - read more below).

Adding Recipe Lines

Recipe lines are added to presets using the **Edit Preset Object** pop-up. This can be accessed using the **Swipey** on a preset pool object.



Edit options for a preset

The middle part of this pop-up is about the recipes and, for instance, grid values.

Besides the main recipe area (described in the **Recipe topic**), the recipe editor area has a colored background. This is a different tool for editing and viewing the values in the selected recipe line.

The preset settings can be visible at the top; most of them are described in the **Preset Pools topic**. There is a setting specifically for the recipes: **Recipe Template**. This can be enabled to turn the preset into a Recipe Template. If the **Recipe Template** setting is enabled in a recipe preset, the recipe itself will be loaded into the programmer part and will be cooked there. Presets that have this setting enabled are shown with a cooking pot icon with the lid open.

The bottom of the editor has different buttons:

- **Insert New Recipe** (available when the preset is turned into a recipe preset):
  Creates a new recipe line. Adding additional recipe lines will reference the values from the first line when values are stored in the first line. When the first line contains a preset, the reference to this preset is copied. Additional lines also automatically reference the MAtricks values from the first line.
- **Cut**:
  Cuts the selected recipe line.
- **Paste**:
  Paste the copied or cut recipe line.
- **Oops**:
  Oops the last action.
- **Delete**:
  Delete the selected recipe line.
- **Copy**:
  Copies the selected recipe line.
- **Edit Preset**:
  Closes the editor and takes the selected recipe line into the programmer in edit mode.
- **Reset MAtricks for Selected**:
  Resets the MAtricks for the selected recipe line.
- **Turn Into Recipe**:
  Tap this button to turn a preset containing attribute value into a recipe preset. A new empty preset must also be turned into a recipe preset before adding recipe lines.
- **Take Selection**:
  This allows using a selection of fixtures instead of a group. Tap this button to make the current programmer fixture selection the new selection for the selected recipe line.
- **Recast Preset**:
  This will recast the preset where it is referenced. This means that if attributes are added or deleted after the preset is used in cues, then the preset might need to be recast for the cues to reflect the new content. Read about this in the **Cue Recipes topic**.
- **Cook**:
  This will cook the preset recipes. This might be necessary for the preset to reflect the recipe content.
- **List References**:
  Tapping this opens a pop-up listing all elements that reference and depend on the preset.

Create a New Recipe Preset

Open the editor for an empty preset.

Tap Turn into Recipe in the menu at the bottom.

This turns the preset into a recipe preset and creates the first recipe line.

It is almost always relevant to at least add a preset reference value. To do this, tap and hold the field in the recipe row in the values column.

This opens a **Preset Pool** selection pop-up. Here, navigating the existing presets and selecting the desired preset is possible. It does not have to be in the same feature group as the recipe preset.

Add the desired values in the other columns.

Add a selection if desired - from a group or by tapping Take Selection.

If the recipe contains a group, then the preset is automatically cooked when the edit pop-up is closed.

Example

We want a preset that can take the current selection of fixtures to a new position in two wings and with a ranged time.

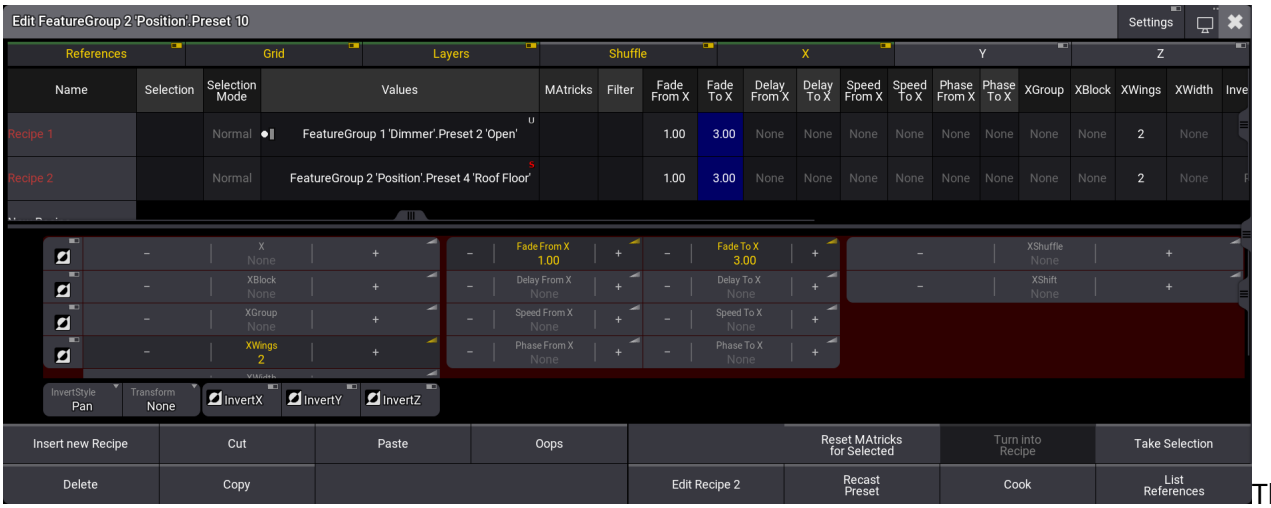
Requirement:

Have a show with some moving heads patched and placed in a row. The show also needs a position preset and dimmer preset where the fixtures are at full. This example uses the Demoshow.

Follow these steps to create the recipe preset:

1. Tap an empty position preset pool object and swipe the finger outside the preset to open the Swipey menu.
2. Select Edit Setting in the Swipey menu.
3. Tap Turn into Recipe to be able to create recipe lines.
4. Tap and hold the first field in the "Values" column.
5. Tap Dimmer and then the preset where the intensity is at full (Open).
6. Tap and hold New Recipe to create a second line.
7. Press Assign.
8. Tap the position preset (Roof Floor) in the position preset pool.
9. Tap the second recipe line to assign the preset to the line.
10. Select both recipe lines in the XWings column (if the column is not visible, then make sure both "X" and "Grid" are active in the row with toggle buttons).
11. Edit this value so it says "2" in the XWings.
12. Select both rows in the "Fade From X" column and set the value to "1".
13. Select both rows in the "Fade To X" column and set the value to "3".

It should look like this ("Settings" is turned Off in the title bar):



The finished recipe

14. Close the editor.
15. Select some fixtures (for instance, group 2).
16. Tap the new recipe preset and see the fixture move while they turn on.

If the move and fade should start from the center, then open the edit preset object pop-up again and set the InvertStyle to "All" and the InvertX to "Yes" for both recipe lines.

Working with Recipe Presets

There are three suggested workflows with presets using recipes:

- Preset containing recipe lines with a selection and Recipe Template disabled.
- Preset containing recipe lines without a selection and Recipe Template disabled.
- Preset containing recipe lines and Recipe Template enabled.

Preset Containing Recipe Lines With a Selection and Recipe Template Disabled

The recipe preset is cooked and will be referenced by the fixtures in the programmer (**Programmer Parts window**). These values can be stored or removed/overwritten in the programmer.

For example, if a recipe preset contains one line with a group as the selection and a value referencing color preset 5. The preset is cooked in the preset. Tapping the recipe preset will result in a reference to the recipe preset in the programmer.

Recipe presets with a selection can be played back from executors. Learn more bout this in the **Use Presets topic**.

Preset Containing Recipe Lines Without a Selection and Recipe Template Disabled

This recipe works with the programmers' current selection of fixtures. The recipe preset is not loaded into the programmer. Instead, the recipe value links and grid values are used with the current selection of fixtures to cook values directly to the programmer.

	Important:

	If values are stored into the recipe and then such a recipe preset with no selection stored is called, the values will be called into the programmer, but without a preset link. Recipe presets with no selection stored will call values with preset links into the programmer when a preset is defined in the preset instead of storing values in the recipe.

This is a great way to have a bunch of template objects in the preset pools that allow quickly calling complex looks based on grid values and MAtricks ranges.

Preset Containing Recipe Lines and Recipe Template Enabled

Recipe Templates load the recipe lines into the active programmer part. This means the recipe lines are cooked in the programmer when the lines contain a selection. If the recipe includes a selection then it is cooked immediately. If the recipe lines do not contain a selection, it must be added, for instance, using the **Programmer Parts Window**.

The Programmer Parts window shows the different programmer parts and it can show recipe lines in the programmer parts.
Learn more in the **What Is the Programmer topic**.

For example, if a Recipe Template contains one line with a group as the selection and a value referencing color preset 5, tapping the recipe preset will result in a recipe in the programmer part with a recipe line that references the color preset 5.

1.27.4. Use Preset

Presets are often used for live playback or as building blocks in cues.

Calling Preset Into the Programmer

The first step is to select the desired fixtures and then call the preset into the programmer.

The workflow is the same for every type of preset (MAgic, Recipe, Multistep, standard single step, etc.)

If the programmer does not have a fixture selection and a preset is tapped, then the first tap selects all the fixtures that can use the preset. In this case, it works like a group.

Tapping a preset with a fixture selection in the programmer calls the preset if it is valid for the selected fixtures.

	**Hint:**
	Please be aware that the **Default Pool Action** could be changed. This text assumes the factory default **SelFix/At** action. Learn more about default pool action in the **Windows Settings topic**.

Timing values can be stored together with attribute values in the preset, and calling a preset that includes timing information calls the values into the programmer using the preset timing.

Calling presets that only contain timing values call the timing values into the timing lavers of the programmer.

Another way to use timing with presets is the **Programmer Time** master. Learn more about the master in the **Time Control** topic.

Presets with stored timing values have a higher priority than the **Programmer Time** master, and the stored timing will be used when the preset is called.

	**Known Limitation:**
	Calling a MAgic preset into the programmer will extract the data and not reference the preset. It is recommended to use MAgic presets in combination with recipes to maintain referenceable data.

Presets Running from Executors

Presets can be assigned to executors using the **Assign keyword** or the Assign menu. Learn more about this in the **Assign Object to an Executor topic**.

If the preset uses global or universal data, the executor button should use the **At** command to call the preset values.

	**Important:**
	Using the At command on presets assigned to executors does not playback the values from the executor. The values are called into the selected fixtures

	in the programmer just like described above - if the preset is valid for the fixture selection.

Other playback commands (like Go+, Toggle, Flash, etc.) are relevant for presets with Selective data. This also means that presets with selective data can be played back from the preset pools using, for instance, a Go+ command.

Opening the Edit Setting tab of an executor's assign menu with a preset assigned offers some of the same settings as for sequences. For more information on playback settings, see the **Sequence Settings topics**.

Edit the general preset playback settings in Menu - Preferences and Timing - Preset.

The **OffFade** setting of preset playbacks determines the fade time when switching off a selective preset playback.

Store Presets in Cues

When a preset is in the programmer, it can be stored in cues. Learn more about storing cues in the **Store Cues** topic.

A reference to the preset is stored for the specific fixture attributes. This means that the value stored in the preset is not stored in the cue, but a reference to the preset is stored. So if the value stored in the preset is changed after the cue is stored, then the cue still looks in the preset to get the value when the cue is played back, and the values in the preset will be used.

Be aware that the **Absolute** and **Relative** value layers are connected to the Fade and the Delay layers. Individual fade and delay times for an attribute use the Fade and Delay layers. Storing a value in an attribute's absolute or relative layer in a preset can also add values from the fade and delay layers.

This means that if a preset contains absolute or relative values, it also connects to possible values in the fade and delay layers. If the fade or delay values are later added, changed, or updated, this change will be reflected in the cue referencing the absolute or relative value.

Fade and Delay values connected to an absolute value has a higher priority than fade and delay values connected to a relative value. For instance, if two presets are called, each with fade and delay values, but one is for absolute values, and the other is for relative values.

The phaser layers (**Speed**, **Phase**, **Measure**, **Accel**, **Decel**, **Transition**, and **Width**) are also linked. Adding one of these layers in a preset effectively adds all the phaser layers.

This is very important to keep in mind when updating the preset information.

If attributes are added to or deleted from the preset after it is used in a cue, the stored cues referencing the preset need to reflect this new change, so the preset needs to be recast. This can be done using the **Recast** keyword or the **Edit Preset Object pop-up**. Recasting a preset removes or adds attributes to the fixtures in the cue.

The recast function only recasts presets to cues where a preset reference exists for the absolute layer.

When presets are used in cues and later deleted, the preset values are transferred to the cue (hardcoded values in the cue).

Assign Presets to Attributes in the Tracking Sheet

When a value is edited in the **Track Sheet**, the available presets can be chosen in the **Calculator**.

Extract Preset Values

Preset values can be extracted to the selected fixtures. The preset needs to be valid for the selected fixtures. The **Extract** keyword is used for this.

The values stored in the preset will be pulled into the programmer without a reference to the preset. The values are then like any other typical programmer values.

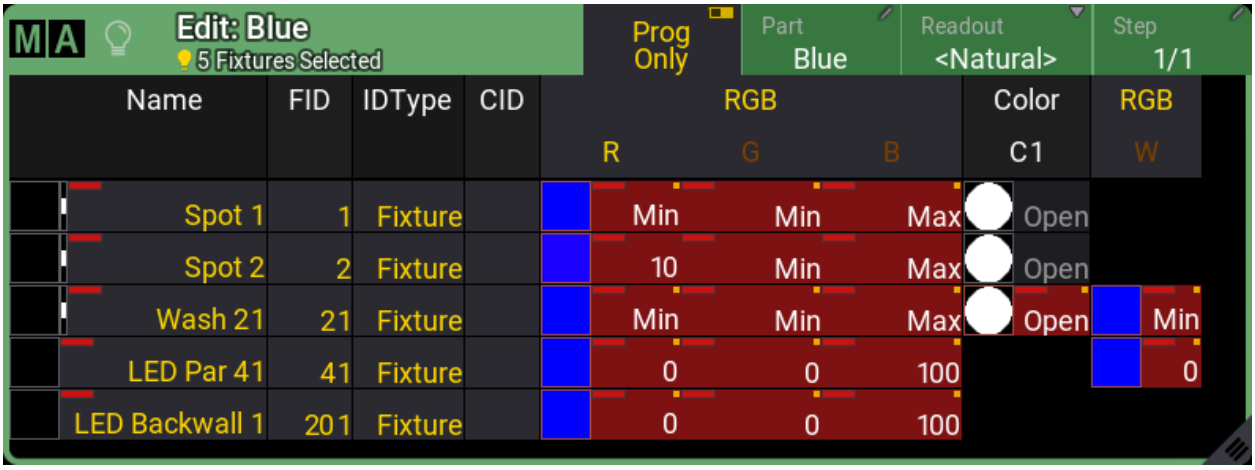
1.27.5. Edit or Update Presets

The existing presets can be edited. The values in the preset can be seen in a **Fixture Sheet** in **Fixture** mode.

	Important:
	Editing a preset calls the values into the programmer, which is sent to output unless **Blind** mode is activated first.

1. Press Edit.
2. Tap the preset to be edited - this changes the frame around the fixture sheet to a green color (yellow frame on other stations with the same user), the fixtures are selected, and the values are pulled into the programmer as active values. Edit starts blinking.

The sheet shows the fixtures that can use the preset and the attribute values.

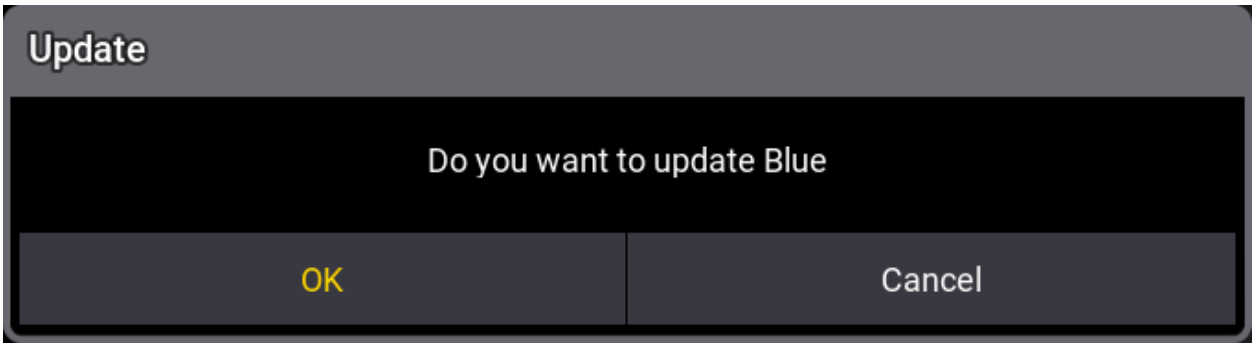


Fixture sheet in edit mode

The example above is a **Global** and **Selective** preset. The global data is marked with a yellow marker in the upper right corner of the attribute values. Any fixture with that fixture type gets the values from the global fixture type. The selective data are values without the yellow marker.

3. Edit the values in the programmer. Update lights up.
4. When the values are correct, press Update.

This opens a pop-up asking for confirmation to update the preset.



Pop-up asking for update confirmation

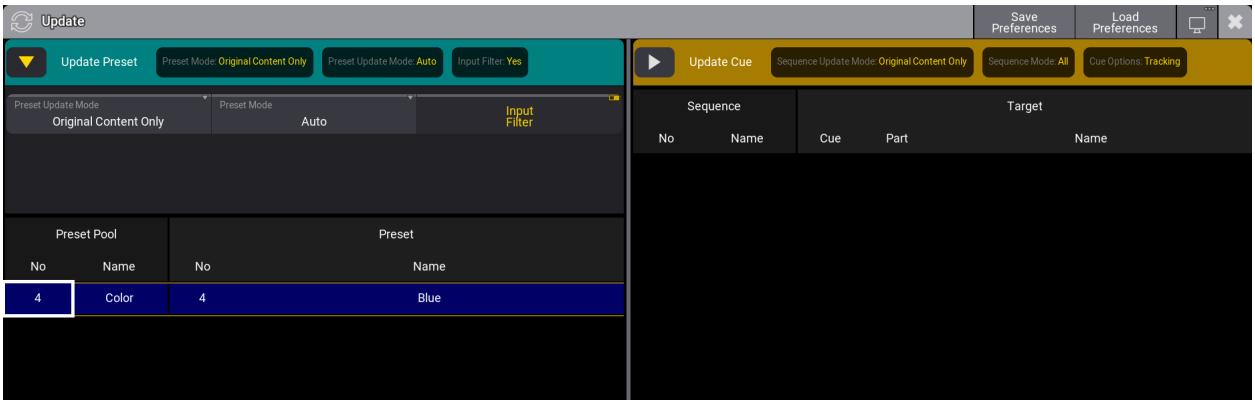
5. Tap OK to confirm the update. Cancel will return to the update menu and not update the preset.

	Hint:
	To deactivate the edit mode without saving anything, press Esc. Remember that the values are still in the programmer.

Update Existing Preset

A preset called into the programmer, where the values changed after it was called, can be updated.

Update flashes when there are values that can be updated. Tap the key to open the **Update Menu**.



Update Menu

The presets that can be updated are listed on the left side of the menu.

The active update settings can be seen in the header of the preset tab. Tapping the right-pointing arrow (see the Update Cue side in the example image above) unfolds the settings, which can be edited.

The **Preset Update Mode** is relevant when updating presets. There are two modes:

- **Original Content Only**:
  Only existing attribute values are updated if a preset is updated with this mode.
- **Add New Content**:
  This mode update existing attribute values and adds any new attribute values to the preset.

The mode can be toggled by tapping Preset Update Mode.

The **Preset Mode** defines how the preset values are updated.

This setting sets what mode will be used when updating the preset. The options are **Auto, Selective**, **Global**, **ForceGlobal**, and **Universal**. These modes are described in the **Preset topic**.

The mode set here is not linked to the default mode selected in the store settings.

The **Input Filter** setting is also used when updating presets. It can be toggled to turn On or Off the input filters when updating preset values. Learn more about input filters in the **Create new Preset topic**.

When the correct settings are selected, then the desired preset can be updated by tapping it in the menu.

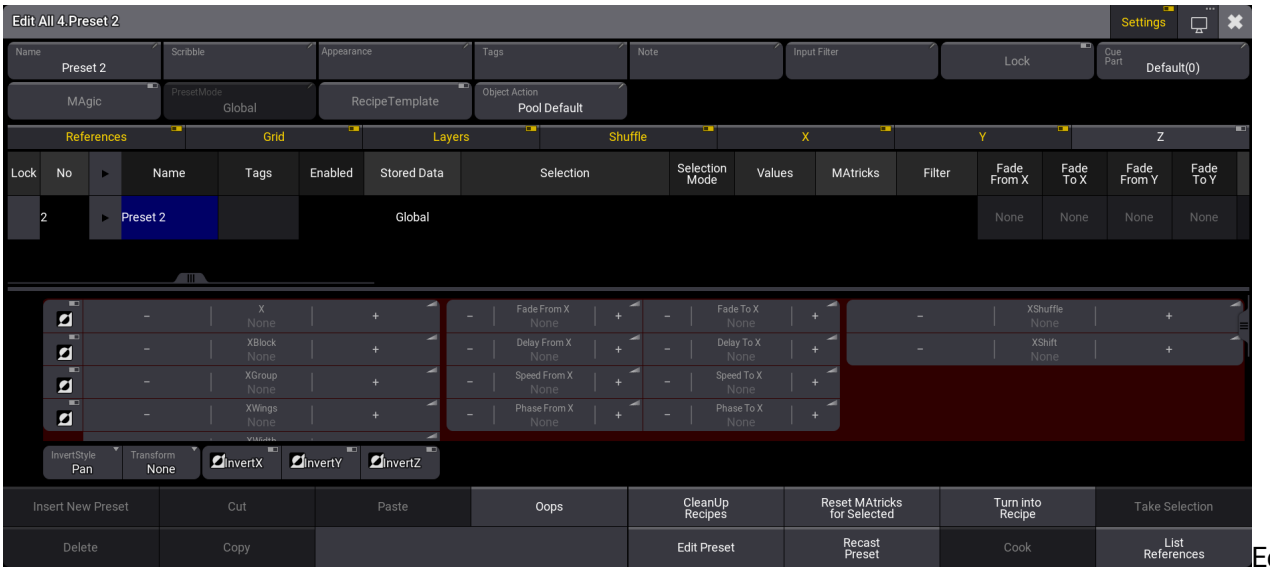
Absolute and relative layers are handled separately during preset updates.

When adding a relative layer value to a preset with an existing absolute layer value, the relative value will not automatically be added to the cue where the preset is already used. Vice versa, when adding an absolute layer value to a preset with an existing relative layer value, the absolute value will not automatically be added to the cue where the preset is already used. Recasting the preset adds information from all layers to the relevant cues. Recast can be done from the preset editor or using the command line. Learn more in the **Recast keyword**.

Recipe presets that reference the updated preset might need to be recooked to look as expected, and cues might need to be recooked for the expected result.

Edit the Preset Object

The preset object can be edited using the edit pop-up. It can be accessed using the **Swipey** menu or the **EditSetting keyword**.



Edit preset object including the settings

The settings in the top line are the same options as the ones for the entire pool, but here, they can be set for the specific preset. Settings on a preset have a higher priority than the pool settings. Learn about the settings in the **Preset Pool** topic.

There are some extra settings here.

The first is called MAgic. It is a toggle button that changes the behavior when the preset is called into the programmer.

Turning on **MAgic** is used for MAgic presets. Learn more about creating and using MAgic presets in the **Create New Presets topic**.

The other is called Recipe Template. Learn more bout this in the **Recipe Presets topic**.

The settings on the top can be shown or hidden using the Settings toggle button in the title bar.

The rest of the editor is for recipe lines. Learn about these in the **Recipe Presets** topic.

1.28. Worlds and Filters

Worlds and filters can be used as tools for programming, playback, or filtering information in some windows. Both are **Pool Windows**.

Worlds are used to limit access to fixtures and attributes. This is especially useful in a multi-user session, so every user works in a designated world with specific fixtures and attributes.

Filters are used to prevent attributes to pass a filter. Typically in store, update, and recall actions but also to filter what is displayed in some windows.

The selected world and filter always dictate what is possible using the programmer.

Worlds and filters can be assigned to some objects. For instance, a sequence or a preset. A small icon on the pool object indicates if an Input Filter ( ), an Output Filter ( ), or both ( ) are assigned. The world or filter then dictates what can be played back from or stored into the object.

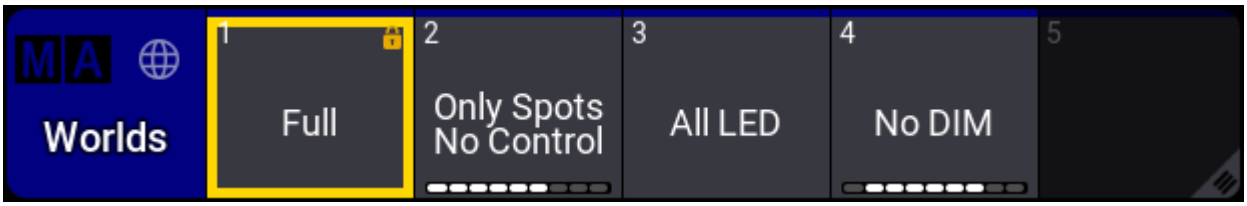
Learn more about **Input Filters** for pool objects in **Create New Presets**. For more information on **Output Filters** go to **Sequence Settings**.

Worlds

Worlds contain information about fixtures and attributes.

Worlds are used to prevent access. Fixtures and attributes not in the active world are removed in some windows and cannot be used in programmer actions.

They are stored in the Worlds pool. This can be created like any window using the **Add Window** pop-up.



World pool with world 1 selected

There is always a selected world. The selected world has a yellow frame around it.

There is a default world from the factory. It is always world number 1 and in a new show it is called "All". This automatically contains all fixtures and all attributes. This world is locked and cannot be edited.

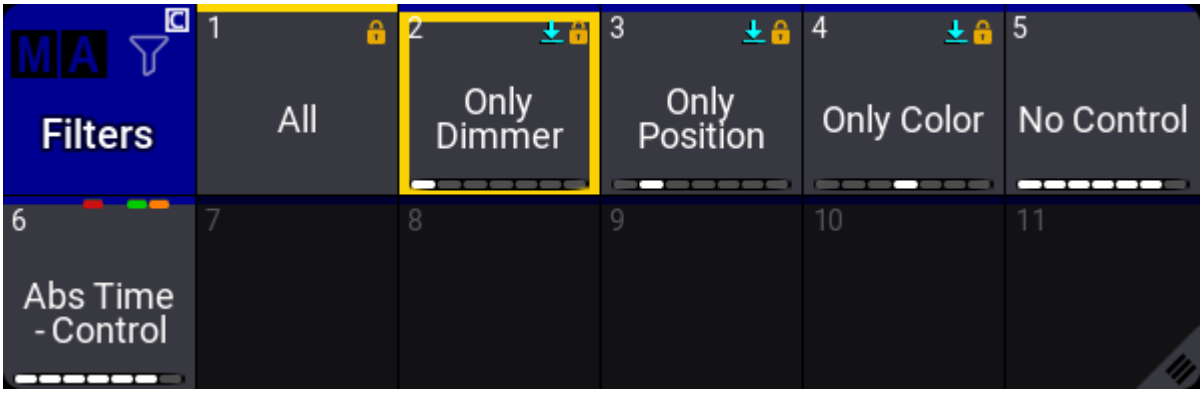
	Hint:
	Data from different worlds can still be stored in one sequence, allowing multiple users working in different worlds to program one sequence together.

Filters

Filters contain information about attributes, layers, and patch information about the fixture (Name, ID Type, Fixture Type, Layer, and Class).

Filters are used to block values from being stored or recalled. For instance, assigning a filter that blocks dimmer values as a playback filter on a sequence prevents the dimmer values from being played back from that sequence. Assigning a filter that blocks certain ID Types to a **Layout** will hide these fixtures in the layout view.

Filters are stored in the Filters pool. This can be created like any window using the **Add Window** pop-up.



Filter pool with filter 1 selected and filter 2 in a called state

There is always a selected filter. The selected filter usually has a yellow frame around it. A filter can also be called using the **Call keyword**. A called filter will have a yellow flashing frame.

When the called filter is different than the selected filter or the **At Filter** is modified, then the selected filter is marked with a yellow line above the filter. See the example image above. Filter 2 is called, and Filter 1 is selected.

The called filter will be used for the next action (Store, Update, At, and Clone) only.

A new show will have some default filters from the factory. The first one is called "All". It contains all attributes and layers. If the attribute structure changes, then this filter is automatically updated. The filter is locked and cannot be edited.

If a different filter than number 1 is the selected or called filter, then the **At key** flashes to indicate that there is an active filter.

Feature Group Indicator Bar

This bar is visible at the bottom of the world or filter pool object when a world or filter does not have all feature groups stored.

Learn more about the bar in the **Preset topic**.

Subtopics

- **Create a World**
- **Create a Filter**
- **At Filter Window**

1.28.1. Create a World

Worlds are created in the programmer and stored in the world pool.

Worlds may be thought of as matrix with rows (fixtures) and columns (attributes), and being in a world may block programmer access to rows and/or columns in sheets.

The created world will contain the rows of the current fixture selection and the columns of any active attribute. All columns/attributes are included in the created world if no attributes were active. If no fixtures were selected, then all fixtures are a part of the world.

Examples

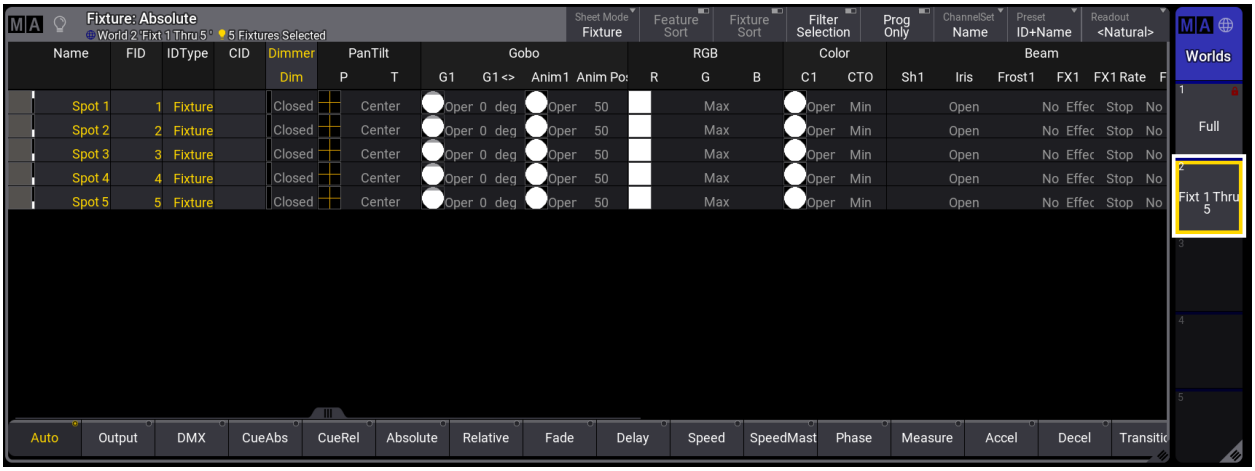
Requirement:

Have a world pool and **fixture sheet** visible on a screen.

Create a World with Fixture 1 to 5

1. Clear the programmer.
2. Select fixtures 1 thru 5 (or any 5 fixtures in your show).
3. Press Store followed by an empty world in the **World Pool**.
4. Tap the newly created world.

Now there is only programming access to fixtures 1 thru 5, reflected by the fixture sheet. All attributes are available.



Fixture sheet with World named "Fixt 1 Thru 5" active.

Create a World with Fixtures and Attributes

Worlds do not contain any values, but active attribute values are needed to store the attribute information for filtering in the world.

Example with five fixtures and only one attribute:

1. Tap World 1 (Full) in the **World Pool**.
2. Clear the programmer.
3. Select fixtures 1 thru 5 (or any 5 fixtures in your show).
4. Give the fixtures a dimmer value - the values do not matter.
5. Press Store followed by tapping an empty world in the **World Pool**.
6. Tap the new world.

Look in a fixture sheet. It only displays the selected fixtures and the dimmer attribute:

Fixture sheet with World named "Fixt 1 Thru 5 Dim" active.

Store a World with All Fixtures but Only Some Attributes

If the world is stored without selected fixtures, all fixtures can be used.

Example with only position attributes:

1. Tap World 1 (Full) in the **World Pool**.
2. Clear the programmer.
3. Select fixtures 1 thru 5 (or any 5 fixtures in your show).
4. Give the fixtures pan and tilt values - the values do not matter.
5. Press Clear once to clear the fixture selection, but keep the pan and tilt values active.
6. Press Store followed by tapping an empty world in the **World Pool**.
7. Tap the new world.

The fixture sheet lists all the fixtures but only the pan and tilt columns:

Fixture sheet with World named "PanTilt" active.

As usual, it is really a good idea to name the worlds as soon as they are created.

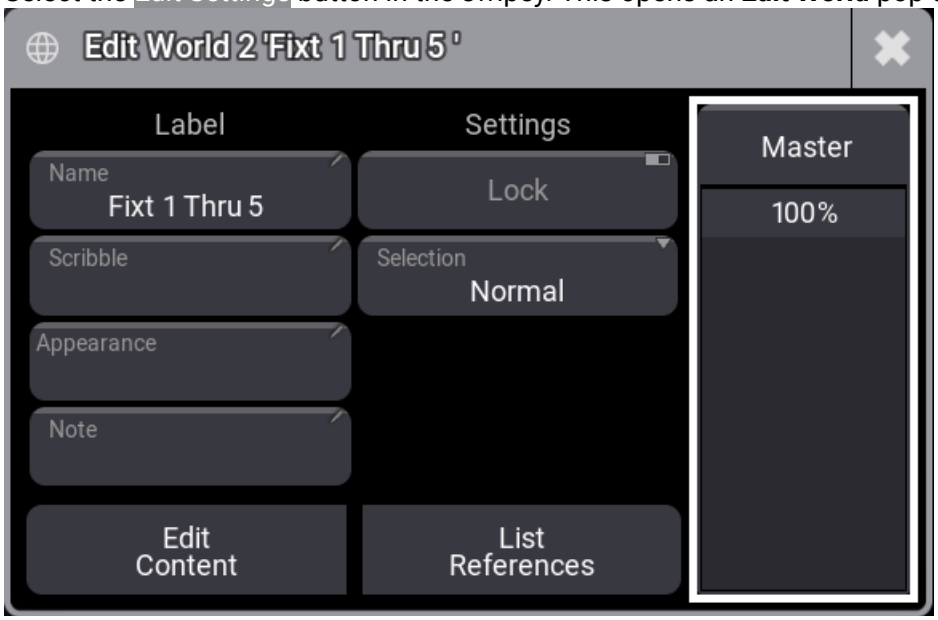
Edit a World

The content of the world can be edited using the following steps:

1. Press Edit followed by the desired world. This takes it into edit mode.
2. Change the programmer selection to match the desired fixtures and attributes.
3. Press Update followed by the world.
4. Confirm the update by tapping **OK** in the update pop-up.

The world settings can also be edited. This is done using the following steps:

1. Use the swipey menu on the desired world.
2. Select the Edit Settings button in the swipey. This opens an **Edit World** pop-up:



3. Make the desired changes.

4. Close the pop-up.

The edit world pop-up gives access to the following options:

- **Name**:
  This is the name of the world.
- **Appearance**:
  This can be used to select an appearance to the world.
- **Scribble**:
  This can be used to select or create a scribble for the world.
- **Note**:
  A note can be added to a world.
- **Lock**:
  The world can be locked to avoid changes.
- **Selection**:
  This defines if sub-fixtures should be part of the world or if it is only the actually selected fixtures that are part of the world. There are two options:
  - **Normal**:
    This is the default option. Sub-fixtures are also part of the world when they were not selected when the world was stored. This includes fixtures in a selected Grouping fixture.
  - **Strict**:
    Only the fixtures and sub-fixtures selected when storing the world are accessible.
- **Master**:
  This can limit the intensity output of the fixtures in the world.

	**Important:**
	Changes to the **Selection** are not visible immediately when changing Selection values for the currently selected world. A different world has to be selected before selecting the desired world again.
	**Hint:**
	When storing a world, the Selection property can be specified within the command: **Store World ["World_Name" or World_Number] Property "Selection" ["Option"]**

There are also two buttons. Edit Content sets the world in edit mode, as described above. List Reference opens another pop-up that lists the different elements that use the world.

Delete a World

Worlds can be deleted like most other pool objects by using the **Delete Keyword**.

Deleting the world does not delete the fixtures or attributes from the show. It is only the world that is deleted.

If a world is assigned as an input or output filter to a sequence or preset pool, then this is also removed when the world is deleted.

Oopsing the deletion brings back the world and also the filter settings for the sequence and preset pool.

The general syntax for deleting a world is:

Delete World ["World_Name" or World_Number]

It can also be a range of worlds.

World number 1 is locked from the factory and cannot be deleted.

	**Important:**
	If the currently active world where the user is in gets deleted, the world still remains active. Until a different world is selected, the user can only use fixtures and attributes of the deleted one.

1.28.2. Create a Filter

Filters are stored in the filter pool.

Filter pool with some filters

There are two main ways to create filers:

- Create a filter using the pool by editing one of the empty pool objects. This opens the **Edit Filter pop-up.**
- Store a new filter pool object using the current filter settings.

The **At Filter window** is a nice tool for seeing the current filter settings.

Create Filter from the Pool

Editing a filter pool object opens a menu that looks almost the same as the **At Filter Window**.

Edit Filter pop-up

The only difference between them is some extra input fields and the fixture patch information filter at the bottom.

The extra fields at the top can be used to edit the name, scribble, appearance, and note.

At the bottom, there is also a List References button that opens a list of elements that relates to this filter. For instance, if the filter is assigned to a preset.

The extra patch information filters at the bottom offer to add information from the patch as filters. The different information is Name, ID Type, Fixture Type, Layer, and Class. Read more about these elements in the **Patch and Fixture Setup section**.

Values can be added to each element type. For more information on Layers, see **Layer Toolbar**.

These values can be used as a positive or negative filter. Meaning that fixtures with the value can pass the filter or be blocked by the filter. Next to each of the elements is a - button. Tapping this toggles it On or Off. If this is white, then it is positive, and fixtures with the values can pass. If it is yellow, then it is negative (On), and fixtures with the values are blocked.

For example, if the **ID Type** has **Fixture** and **Channel** values and the - is white, then the filter allows fixtures with these two ID types to pass the filter. Other ID types are blocked. These filter options make it possible to have very precise filters.

The title bar has an extra button that allows this pop-up to be moved to a different display.

Follow these steps to create a new filter by editing the pool object:

1. Edit an empty filter pool object using swipeys, typing a command (**Filter keyword**), or using the keys (**Group key**).
2. Make sure the desired attributes and layers are active.
3. Add the desired patch information filter values.

4. Optionally give it a name by editing the **Name** input field.
5. Optionally assign an **Appearance** and **Scribble** by editing the input fields.
6. Optionally add a **Note** about the filter.
7. Close the editor by tapping the in the upper right corner.

The edit function is the easiest way to update or change an existing filter. Another option is using the **At Filter Window**.

Delete a Filter

Filters can be deleted like most other pool objects by using the **Delete Keyword**.

Deleting the filter does not delete the attributes from the show. It is only the filter that is deleted.

If a filter is assigned as an input or output filter to a sequence or preset, then this is also removed when the filter is deleted.

Oopsing the deletion brings back the filter and also the filter settings for the sequence and preset.

The general syntax for deleting a filter is:

Delete Filter ["Filter_Name" or Filter_Number]

It can also be a range of filters.

Filter number 1 is locked from the factory and cannot be deleted.

1.28.3. At Filter Window

The **At Filter** window is a tool to see, add, and store current filter settings of filters.

It can be added like any other **window**. Additionally tap and hold the  icon in the control bar to open the **At Filter Overlay**.

Most of the settings and functions are similar to the Edit Filter popup. For more general information see **Create a Filter**.

Create Filter with the At Filter window

To create individual filter settings, add the **At Filter** window and the **Filter** pool:

At Filter window and Filter pool with a default **All** filter enabled.
Tap on the yellow marked cells to disable the filter for this specific element. Tap on cells under the column **Feature Group**, to automatically deactivate the related **Feature** and **Attributes**.
Store a new filter by pressing Store followed by tapping an empty pool object or specify the filter using keys or commands.

1.29. MAtricks and Shuffle

MAtricks is a tool that can be used to divide a selection of fixtures into sub-selections.

The general workflow involves selecting fixtures and applying different MAtricks settings to the selected fixtures inside the primary selection.

For example, ten fixtures are selected, and you want to step through these ten fixtures one at a time to do a position correction. MAtricks is the tool used to do this.

The selection can be shuffled using a set of shuffle tools. Read more **below**.

MAtricks Tools

One of the ways to work with MAtricks is the MAtricks toolbar or window.

The MAtricks tools are available in a window that can be created like any other.

An overlay version of the MAtricks window can be opened by tapping MAtricks in the **Encoder bar**.

The two versions have the same buttons and options. For details about the different options, read the following topics.

Tap Grid, Layers, or Shuffle to display or hide the corresponding group in the title bar.

The MAtricks window is divided into three sections, one for each axis. The X-axis section has a red background, the Y-axis section has a blue background, and the Z-axis section has a green background.

Tap X, Y, or Z in the toolbar on the left side to display or hide the corresponding axis.

Each axis section has its properties grouped.

Here are the groups and their properties:

- **Grid**: Axis (X, Y, Z), Block, Group, Wings, and Width.
- **Layers**: Fade From/To, Delay From/To, Speed From/To, and Phase From/To.
- **Shuffle**: Shuffle and Shift.

The **Layers** speed readout format can be set to Hz, BPM, or Seconds in the User Profile. See **User Settings**.

For the following example, the speed readout format is set to Hertz.

To apply the speed value of 10Hz to the Speed From X property of the selection MAtricks using the command line, type:

```
User name[Fixture]>Set Selection MAtricks "SpeedFromX" 10
```

This is the result:

	Important: Now that the command respects the readout format the user sets, loading show files created in versions prior to 2.0 containing macros that set MAtricks **Speed To X** and **Speed From X** properties will have different results. Please adjust these macros accordingly.

Dimensions and Selection Grid

The MAtricks toolbar shows many settings that can be applied to the X, Y, and Z axes, which are the three dimensions used by the **Selection Grid**.

If the fixtures are in a 3-dimensional grid selection, then the MAtricks tool can be used in all three dimensions.

MAtricks Pool

The different settings that can be made in the MAtricks tool can be stored in the **MAtricks Pool**.

This pool can be created like any other window. You can find it under the **Data Pools** tab.

MAtricks pool with some store MAtricks

This pool works just like most pools in the grandMA3. The two most used functions are storing and recalling the MAtricks settings.

Store a New Pool Object

This is the process for storing a MAtricks pool object.

The easiest way to do this is by long-pressing an empty pool object. This stores the current MAtricks settings (even if no settings are stored).

The general syntax uses the **Store Keyword** and the **MAtricks Keyword**: **Store MAtricks ["MAtricks_Name" or Matricks_Number]**

	Hint:
	Tap, hold, and slide your finger across any value area to change the value.

Tap the video below to see the example.

Simple Example

This simple example uses ten fixtures.

Select the ten fixtures without any specific grid information. Press Full to turn the intensity on.

This is how it looks in the **Selection Grid** and the **MAtricks** window:

Ten fixtures without any MAtricks settings

These ten fixtures are now in one row on the X-axis.

Press Next. This is the same as tapping the + in the X setting.

This is the result:

Ten fixtures with X 1

The selection frame around the first fixture is the usual yellow. The rest have a darker yellow selection frame, indicating they are part of the bigger selection.

Try pressing Next to step through the selection. Pressing Prev (previous) goes through the selection in the opposite direction.

Notice how the X number in the MAtricks tool updates with the key presses. Tapping - and + buttons is the same as pressing Next and Prev.

A specific X number can be specified using the command line. For instance, if X needs to be 6, then the following command can be used:

```
User name[Fixture]>Set Selection MAtricks 'X' 6
```

When you are comfortable with Next and Prev, press Set.

When Set is pressed, all fixtures from the original selection are selected, and the MAtricks is deactivated.

Pressing Set again will reactivate the MAtricks.

Tap Active in the title bar to toggle the MAtricks tool On or Off without resetting the MAtricks settings.

Multiple Selection

Each user profile has two different fixture selections: selection 1 and selection 2.

Tap and hold ActiveSel at the bottom of the MAtricks window. This opens the dropdown menu, from which you can select 1 or 2.

The selection can also be changed using the **Selection Keyword**.

For example, the command needed to change to selection 2 is:

The currently active selection can be copied to the other by tapping Copy Sel.

By tapping Copy Values, the values from the currently active selection can be cloned to the other selection.

A **ClearAll** command clears both selections. **ClearSelection** only clears the active selection.

Shuffle Selection

The **Shuffle Keyword** allows for the selection order to be sorted randomly.

By default, Shuffle will randomize the selection order on all three axes in the **Selection Grid**.

Each shuffle setting per axis can be set to 0 (=None) up to 32 767. Each value represents a different shuffled selection order. When repeatedly selecting the same amount of fixtures, the same shuffle value will result in the same shuffled selection order. This can be useful when a specific shuffled selection order is desired for the same number of fixtures. In this case, apply the same shuffle value in the MAtricks when the same number of fixtures are selected. The fixtures are then shuffled the same way.

Tapping Shuffle in the MAtricks tool shuffles all three axes in the Selection Grid by adding a random number in XShuffle, YShuffle, and ZShuffle.

To shuffle on a single axis, enter a shuffle value for the desired axis in the MAtricks tool or tap the + or - in the axis until there is a wanted shuffle result.

Tap ShuffleMode on the bottom right of the MAtricks tool to access the following modes:

- **Auto**:
  When doing shuffle only for one dimension, this behaves like **Linked**. Shuffling on two or three axes behaves like **Unlinked**.
- **Linked**:
  All fixtures placed on the same position along the axis that will be shuffled but have a different position on the other axis will keep their alignment along the other axis.
- **Unlinked**:
  The fixtures placed on axes other than the axis that will be shuffled but have the same position on the shuffled axis will be shuffled independently.

	Hint:
	When deactivating or resetting the MAtricks, the original selection order will be restored.

Since the shuffle is part of the MAtricks, they are also stored in a MAtricks pool object like any other MAtricks setting.

See shuffle examples in the **Shuffle topic**.

Shift Selection

The shift setting in the MAtricks tool allows the current selection to be shifted within the selection grid positions. This can be done per axis in the grid. Therefore, change the values for XShift, YShift, or ZShift.

Positive values shift to the right (x-axis), to the bottom (y-axis), and to the front (z-axis). According to this, negative values shift in the opposite direction.

Invert Options

Invert will define which axis in the selection grid the values should be inverted when turning the encoder or applying a range of values.
The inverted fixtures are displayed with a green font in the fixture sheet, a green body color in the 3D window, and a green border in the layout window and the selection grid window.

The following buttons can be found at the bottom left of the screen:

- InvertStyle: Defines if Invert shall be applied to Pan, Tilt, Pan and Tilt, or All attributes.
- InvertX: Inverts the overall invert of the current individual inverts per MAtrick property on the X axis.
- InvertY: Inverts the overall invert of the current individual inverts per MAtrick property on the Y axis.
- InvertZ: Inverts the overall invert of the current individual inverts per MAtrick property on the Z axis.

Grid properties can also be inverted by tapping ⧈.

	**Hint:**
	When using Align in combination with Invert, the alignment is still based on the arrangement of fixtures inside the selection grid; however, the aligned values will be inverted.
	**Restriction:**
	At the moment, Align only works with the X axis.

Subtopics

- **Blocks**
- **Groups**
- **Wings**
- **Widths**
- **Shuffle**
- **Transform**

1.29.1. Blocks

The blocks function in the MAtricks creates blocks of fixtures of the specified size.

This treats blocks of fixtures as one fixture.

It is better explained with examples.

Example - Ten Fixtures One Axis

In this example, there are ten fixtures (1 through 10).

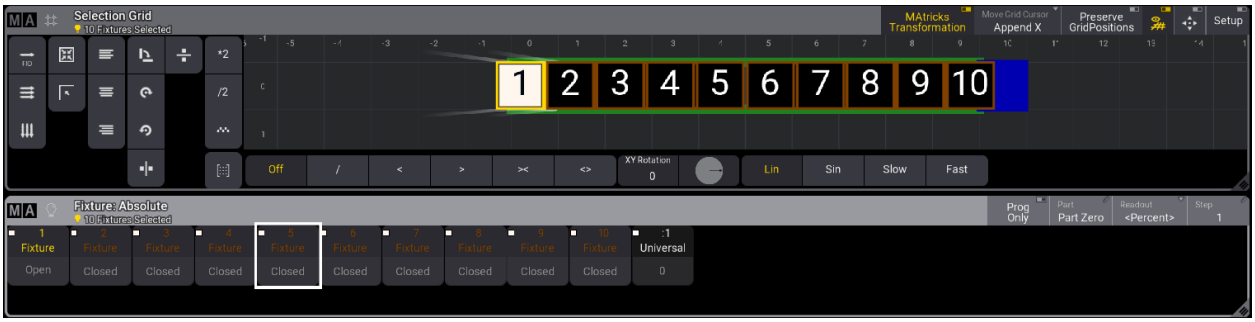
They are selected from 1 to 10 without any specific grid information. Highlight is activated.

Since there is no grid information, the fixtures are only on one axis - the X-axis.

Press Next to make the MAtricks X value 1.

Ten fixtures with MAtricks X at 1 - No XBlock

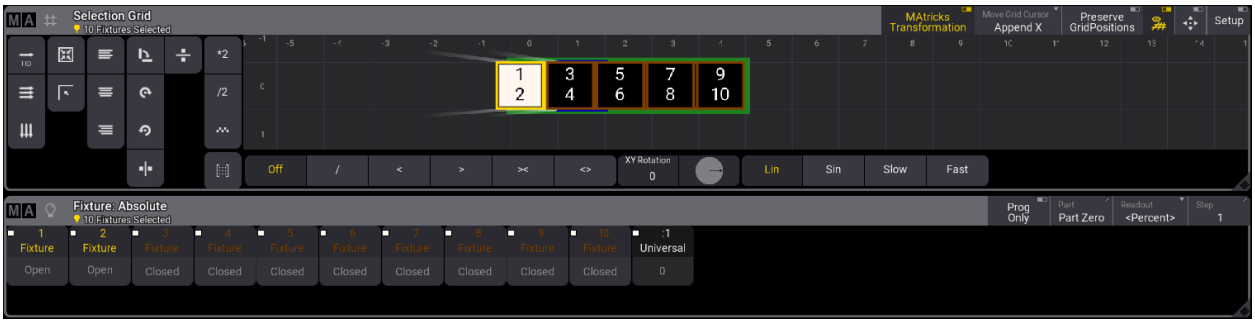
It looks like this in the Selection Grid and Fixture Sheet windows. (Fixture Sheet Mode set to Channel and Output layer selected). For more information, see **Fixture sheet**.



Ten fixtures with MAtricks X at 1 - XBlock set to None

Now tap + in the XBlock of the MAtricks window to set the value to two.

Now it looks like this:



Ten fixtures with MAtricks X at 1 and XBlock at 2

The selection grid shows five boxes. This is because fixtures 1 and 2 are blocked together and in the first position. As can be seen in the fixture sheet, they are the ones currently outputting light.

Press Next and Prev to jump through the ten fixtures blocked together two by two.

Tap + and - in the MAtricks X to jump through the ten fixtures blocked together two by two. Next and Prev can also be used.

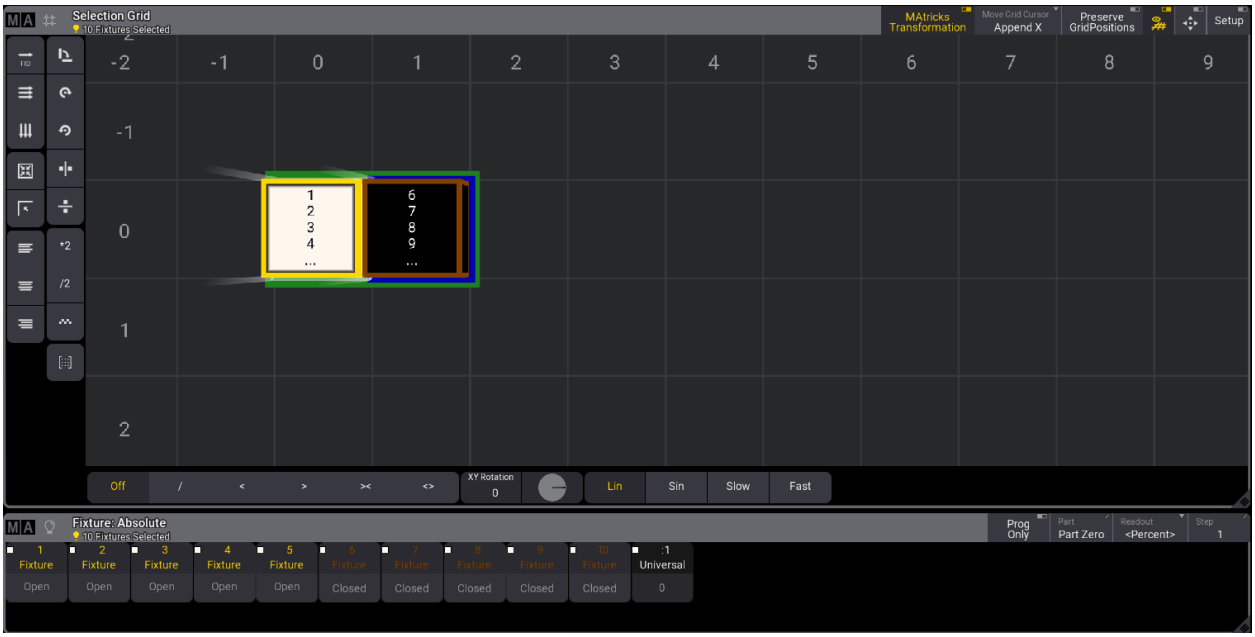
Tap the video below to see the example.

Increasing the block size increases the number of fixtures together in a block. It decreases the number of visible boxes in the selection grid and the useful MAtricks X values.

For instance, if the block is increased to five, there are only two boxes in the grid, X = 1 and XBlock = 5.

Ten fixtures with MAtricks X at 1 and XBlock at 5

It looks like this: in the Selection Grid and the Fixture Sheet.



Ten fixtures with MAtricks X at 1 and XBlock at 5
Tap the video below to see the example.

1.29.2. Groups

Groups in the MAtricks are used to separate the selection into the number of groups set.

It alternates through the selection putting fixtures into each group.

It is best explained with an example.

Example - Ten Fixtures One Axis

In this example, there are ten fixtures (1 through 10).

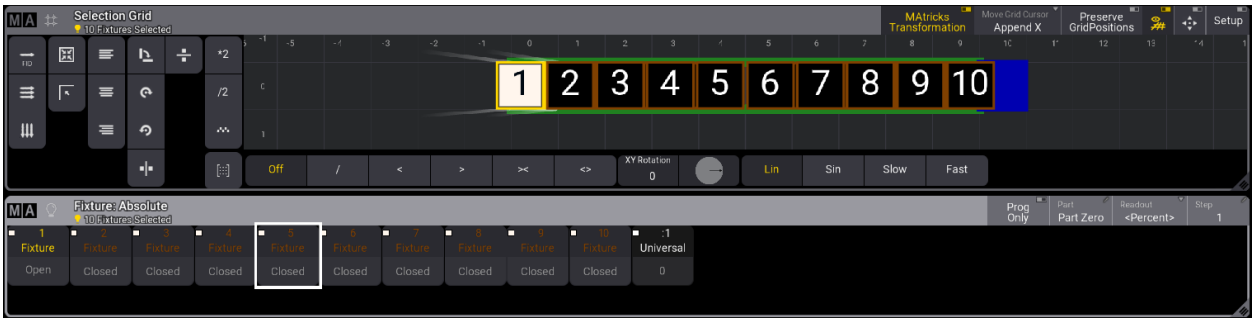
They are selected from 1 to 10 without any specific grid information. Highlight is activated.

Since there is no grid information, the fixtures are only on the X-axis.

Tap + in the X settings MAtricks window or press Next to make the MAtricks X value 1.

Ten fixtures with MAtricks X at 1 - No XGroup

It looks like this in the Selection Grid and Fixture Sheet (Channel SheetMode and Output layer selected) window.

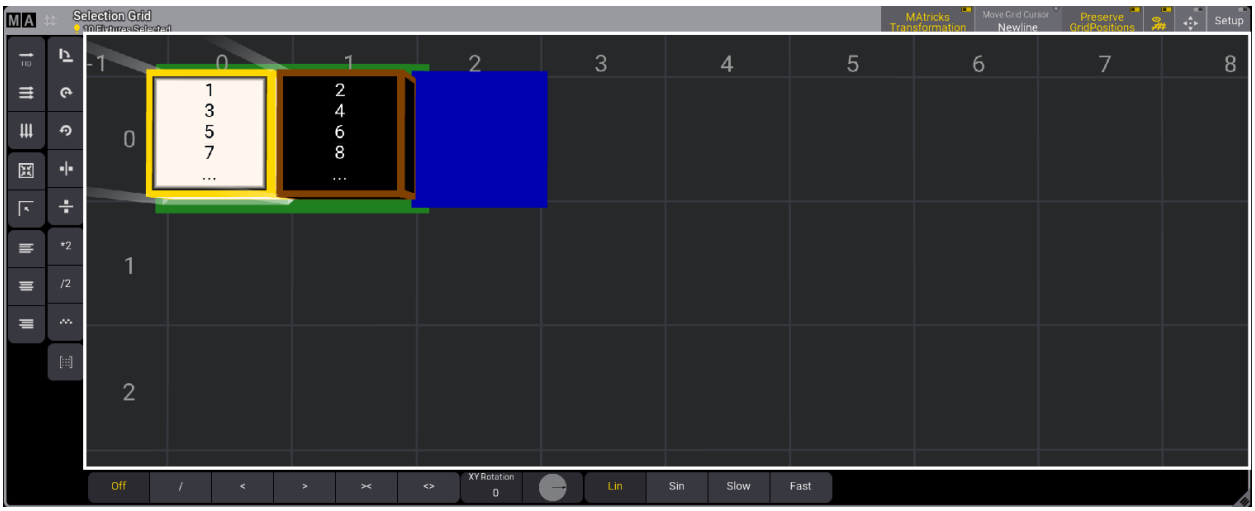


Ten fixtures with MAtricks X at 1 - No XGroup

Now tap + in the XGroup in the MAtricks window and set the value to two.

Ten fixtures with MAtricks X at 1 - XGroup at 2

Now it looks like this in the Selection Grid and Fixture Sheet.



Ten fixtures with MAtricks X at 1 - XGroup at 2

Notice how the ten fixtures are moved into two boxes in the selection grid.

The Selection shows how the fixtures are interleaved into the two groups. The first fixture goes into the first group, the second fixture into the next group, the third fixture in the first group, and so on until there are no more fixtures.

Tap + in the MAtricks X to set it to two. Tap - to set it back to one. You can also press Next and Prev to toggle between the selections.

Tap the video below to see the example.

1.29.3. Wings

Wings in the MAtricks separate the selection into the number of wings set and select devices from each wing from opposite directions.

It is better explained with examples.

Example - Ten Fixtures One Axis

In this example, there are ten fixtures (1 through 10).

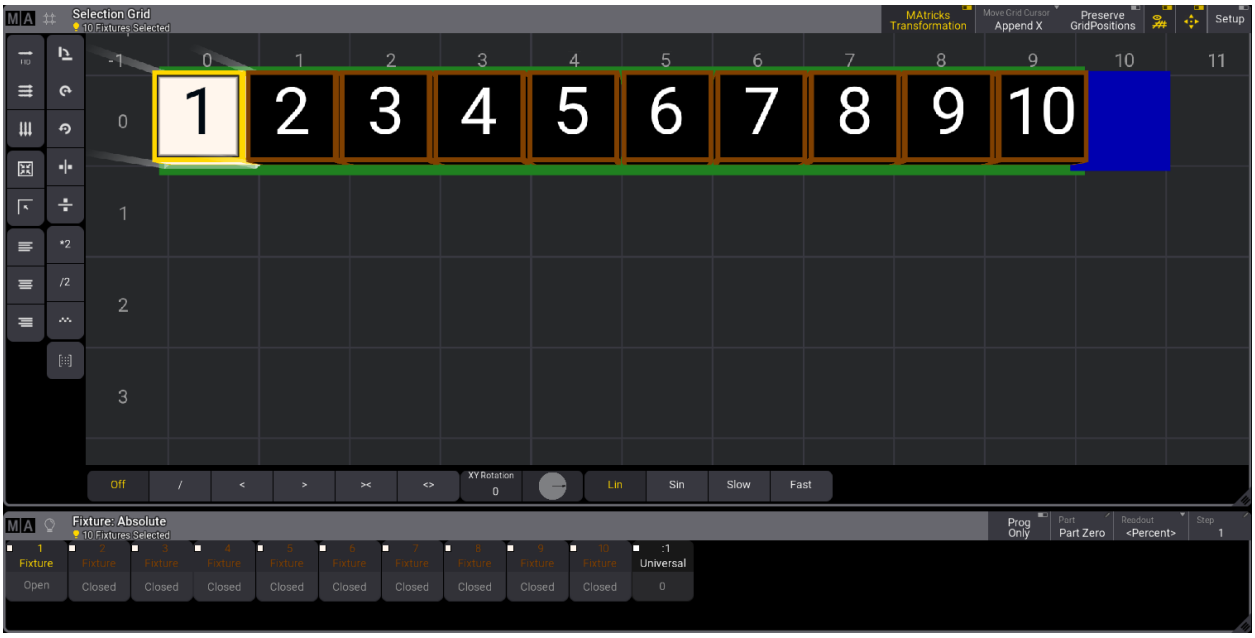
They are selected from 1 to 10 without any specific grid information. Highlight is activated.

Since there is no grid information, the fixtures are only on one axis - the X-axis.

Press Next to make the MAtricks X value 1.

Ten fixtures with MAtricks X at 1 - No XWings

It looks like this in the Selection Grid and Fixture Sheet (Channel SheetMode and Output layer selected) window.

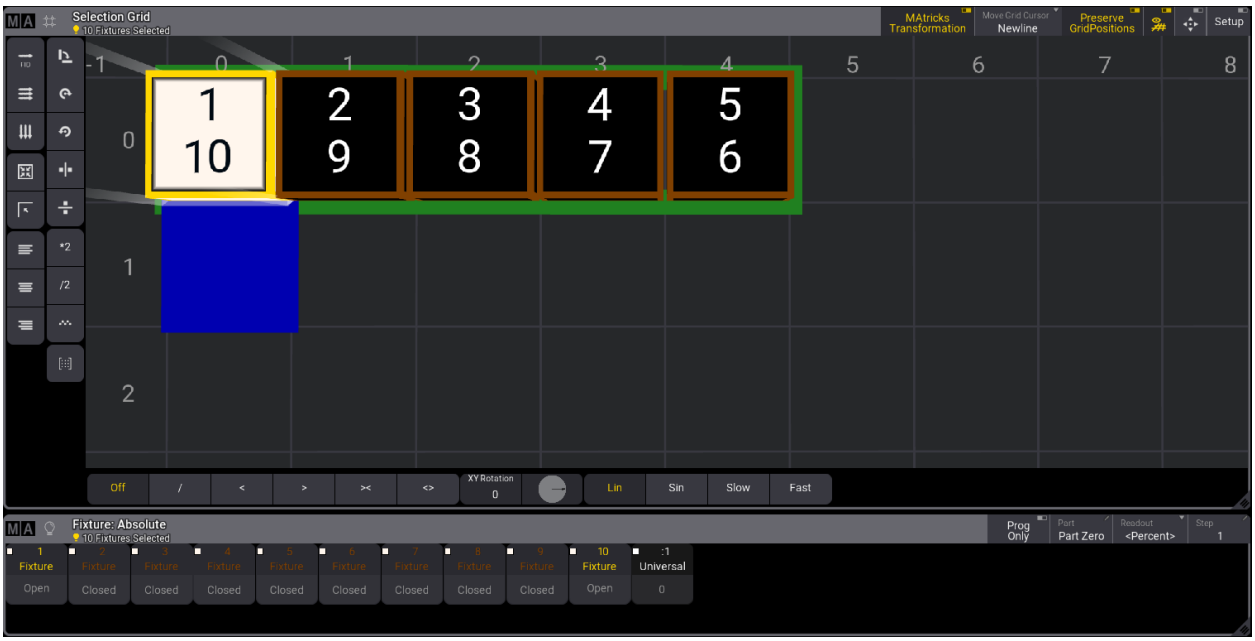


Ten fixtures with MAtricks X at 1 - No Wings

Now tap + in the XWings in the MAtricks window to set the value to two.

Ten fixtures with MAtricks X at 1 - XWings at 2

Now it looks like this



Ten fixtures with MAtricks X at 1 and XWings at 2

The selection is split into two wings, and each end of the selection is currently On.

Tap + to change the X number and move the highlighted selection towards the "center" of the selection.
Tap - to change the X number and move the highlighted selection from the inside out of the selection.
Using Next and Prev does the same.

Tap the video below to see the example.

1.29.4. Widths

Width can appear to give the same result as groups. The difference is that groups place the grouped fixtures in the same grid positions where width moves the fixtures out on the next axis.

This makes it possible to combine several axes in the MAtricks selection.

It can be a great tool for fixtures arranged in grids.

Having the fixtures in MAtricks with a width and storing a group in the **Group pool** also stores the current selection grid setup.

The MAtricks width is best explained with an example.

Example - Ten Fixtures One Axis

In this example, there are ten fixtures (1 through 10).

They are selected from 1 to 10 without any specific grid information. Highlight is activated.

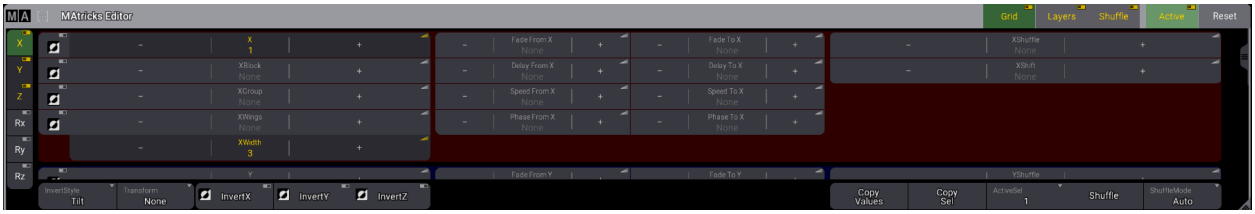
Since there is no grid information, the fixtures are only on one axis - the X-axis.

Press Next to make the MAtricks X value 1.

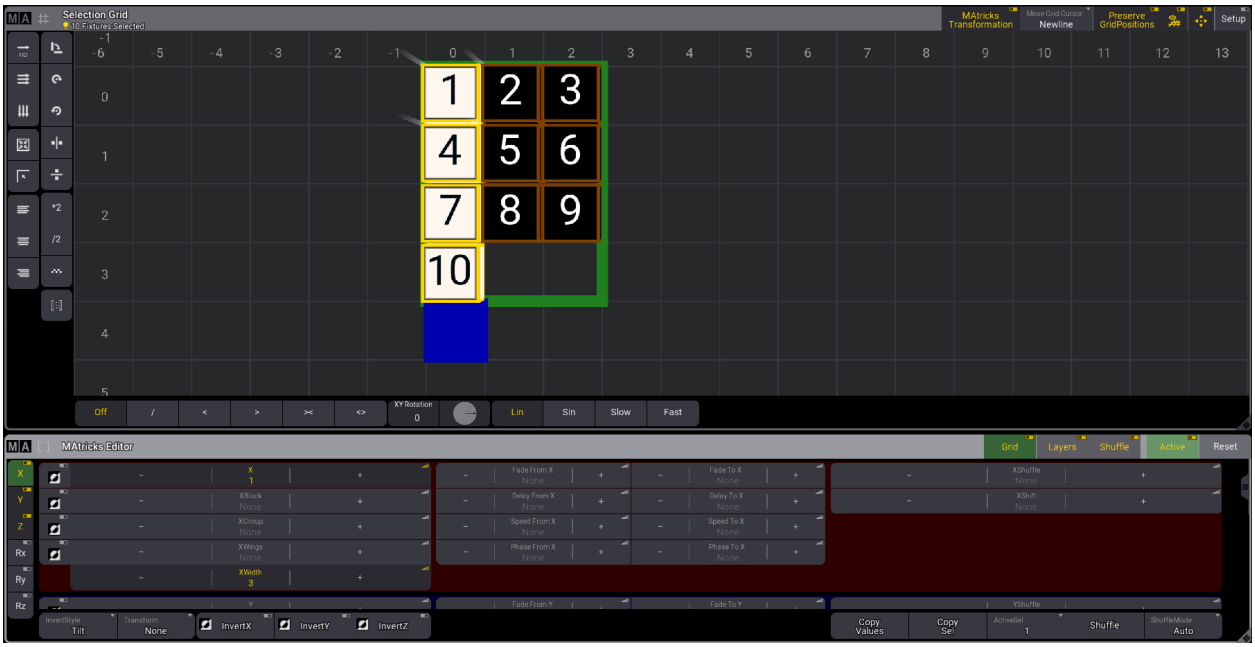
It looks like this in the Selection Grid and Fixture Sheet (Channel SheetMode and Output layer selected) window:

Ten fixtures with MAtricks X at 1 - No Width

Now tap + three times in the XWidth in the MAtricks window to set the value to three.



Now it looks like this.

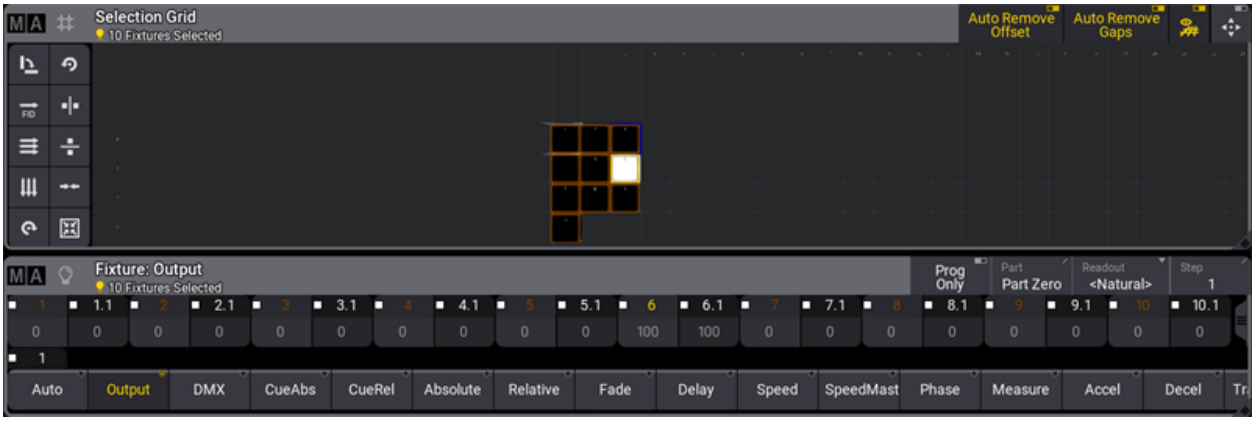


Ten fixtures with MAtricks X at 1 - XWidth at 3

All fixtures in the first X column are now highlighted.

This arrangement allows you to select a specific fixture or columns and rows of fixtures using a combination of X and Y values.

Highlighting fixture 2 in the current arrangement can be achieved by having X = 2 and Y = 1.



Ten fixtures with MAtricks X at 2, Y at 1, and XWidth at 3

Try to play around with different X and Y values.

1.29.5. Shuffle

Shuffle is used to shuffle the selection order of the current fixture selection.

The shuffle function is described in more detail in the **MAtricks and Shuffle topic**.

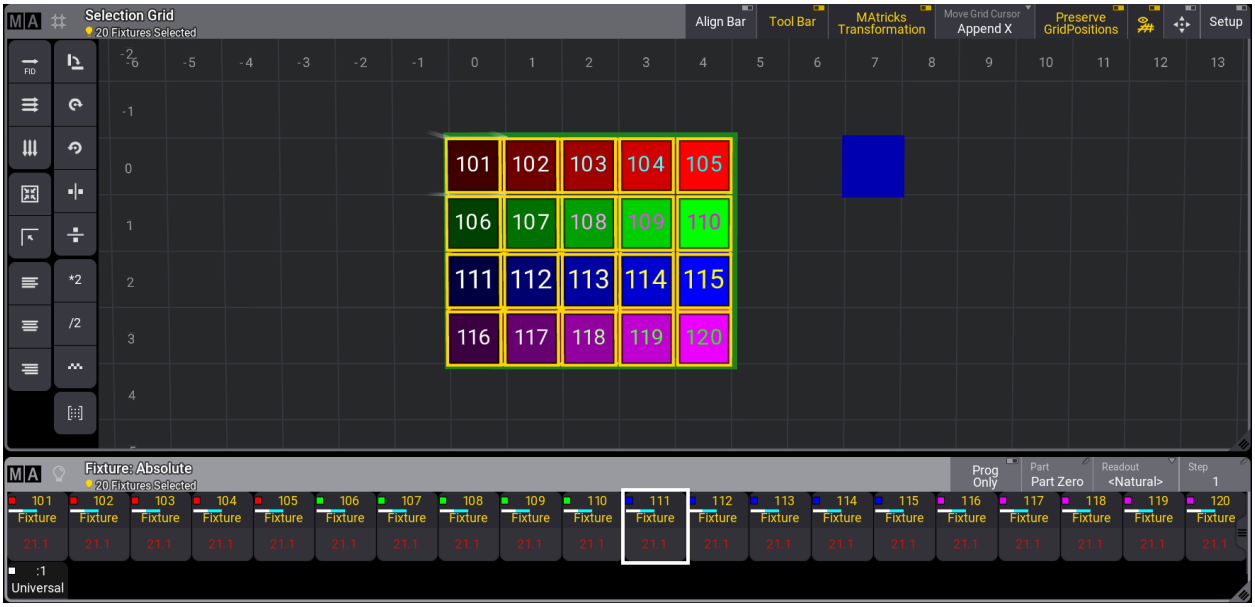
This example uses the **MAtricks window**, the **Selection Grid window**, the **Fixture Sheet window,** the **Group Pool**, the **Color Preset Pool**, and the **All 1 Preset Pool**. It is useful to have them visible on the screens.

This is an example of a shuffle on two axes.

The main setup consists of 20 fixtures with a color mix and four global color presets (Red, Green, Blue, and Magenta).

1. Select fixtures 101 thru 105 and tap the red color preset.
2. Select fixtures 106 thru 110 and tap the green color preset.
3. Select fixtures 111 thru 115 and tap the blue color preset.
4. Select fixtures 116 thru 120 and tap the magenta color preset.
5. Select fixtures 101 thru 120.
6. Use the MAtricks tool to set XWidth to 5. This arranges the fixtures nicely in a 5 x 4 grid.
7. Store a group with this grid arrangement.
8. A dimmer range of 25% to 100% is applied by pressing: At 2 5 Thru 1 0 0 Please.
9. Press Store and tap an empty All 1 preset pool object.

It looks like this:



Imagine this is how the fixtures are arranged in the real world, and this now needs to be randomly applied in both the X and Y axes.

Tap Shuffle in the MAtricks tool until a desired pattern is achieved.

Notice in the fixture sheet that this does not change the actual output of the fixtures. It only changes how they are selected.

The values need to be reapplied to get the output to look like the random selection.

To apply the dimmer, press At 2 5 Thru 1 0 0 Please.

The MAtricks Y value makes it easy to reapply the colors.

Tap + in the MAtricks Y, so the value is 1. Now apply the red color preset. Tap + again to select the next row and apply the next color preset. Continue doing this for the last two rows.

Press Clear once and tap the group stored at point seven in the initial setup.

The result could look something like this:

1.29.6. Transform

Transform is a setting in the MAtricks window. It is found in the **Invert Options** at the bottom of the MAtricks window.

If Transform is set to Mirror, the values will be mirrored depending on the other MAtricks settings, such as Blocks, Groups, and Wings.

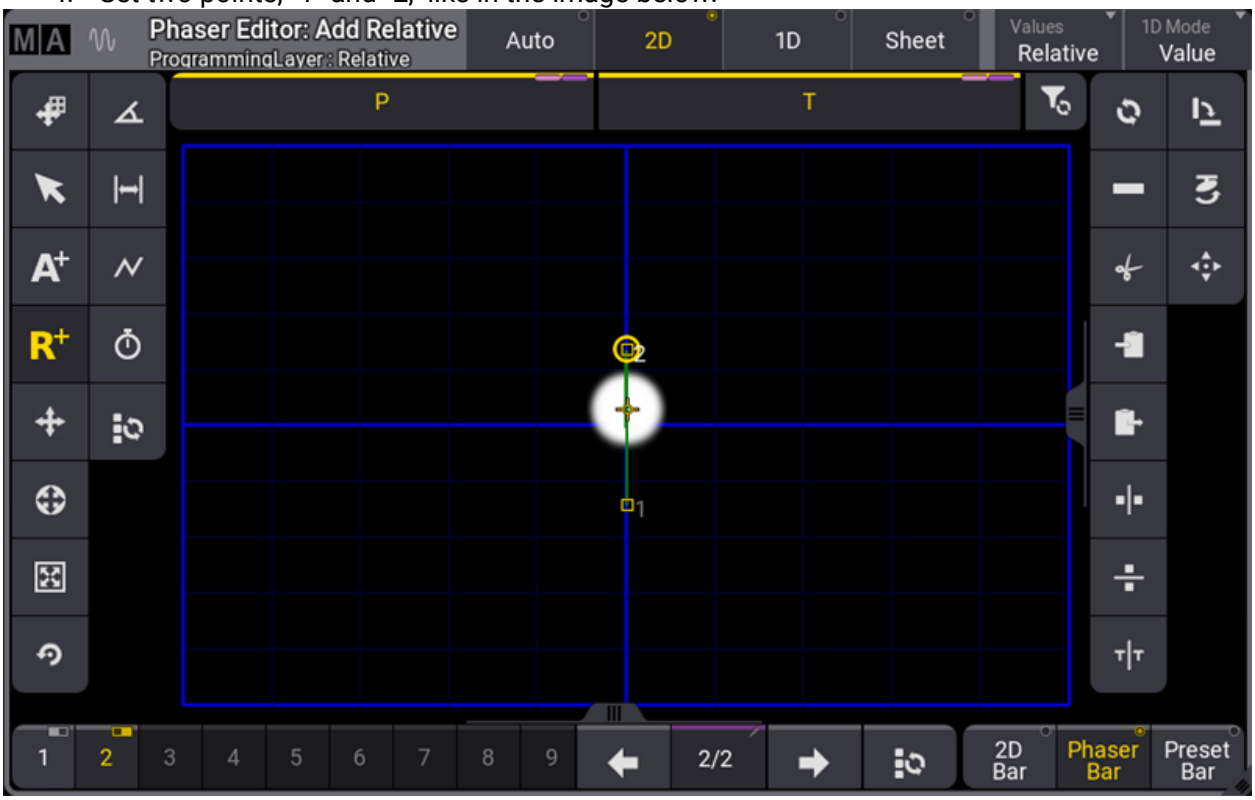
Transform allows to:

- Quickly create symmetrical objects.
- Keep a symmetrical impression on an odd fixture selection.

Example

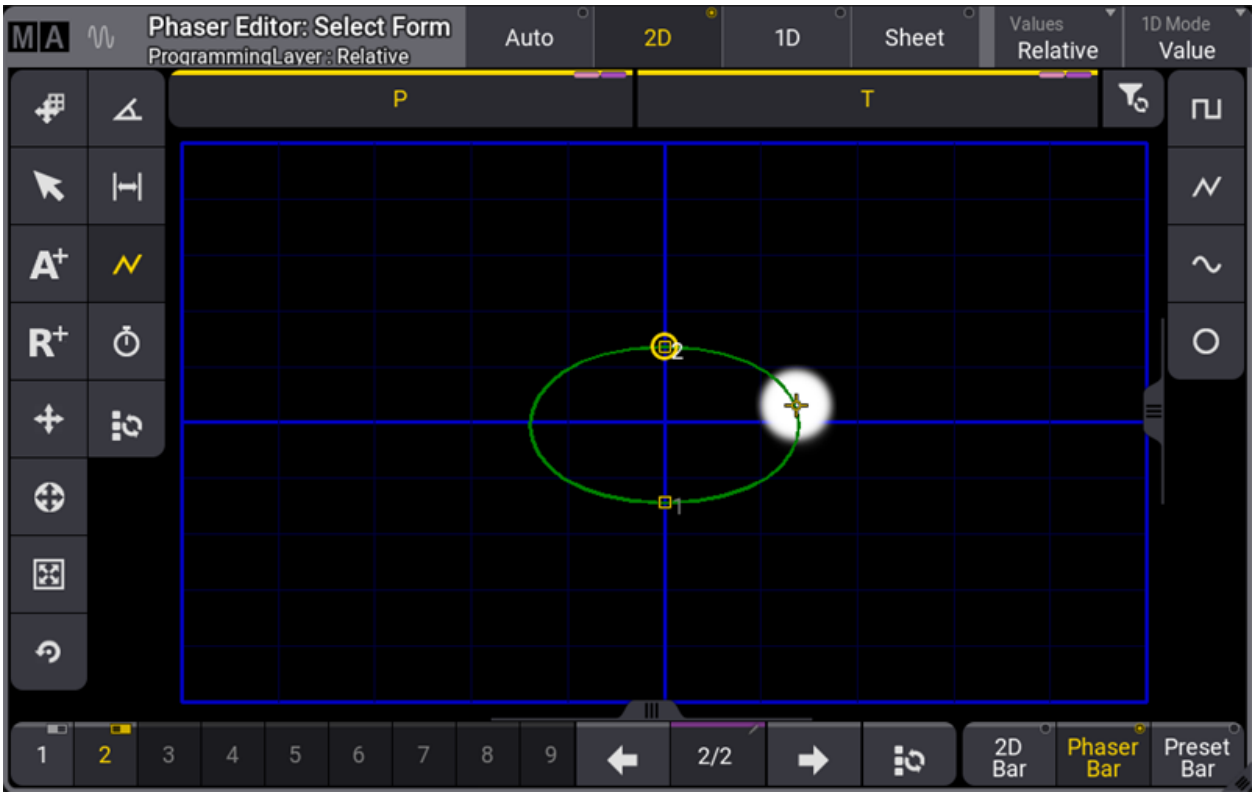
How to set Transform in a simple example:

1. Select some fixtures and set the dimmer to full.
2. Open a **Phaser Editor** window using the Add Window pop-up. Or tap the Phaser button in the Encoder Bar.
3. Select 2D and then tap the R+ button in the left tool button bar of the phaser window.
4. Set two points, "1" and "2," like in the image below:



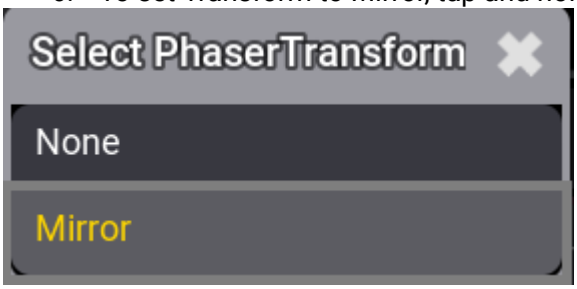
Phaser with relative position.

4. To change the phaser to a circle form, select in the left button toolbar and then select on the right button toolbar in the phaser window.



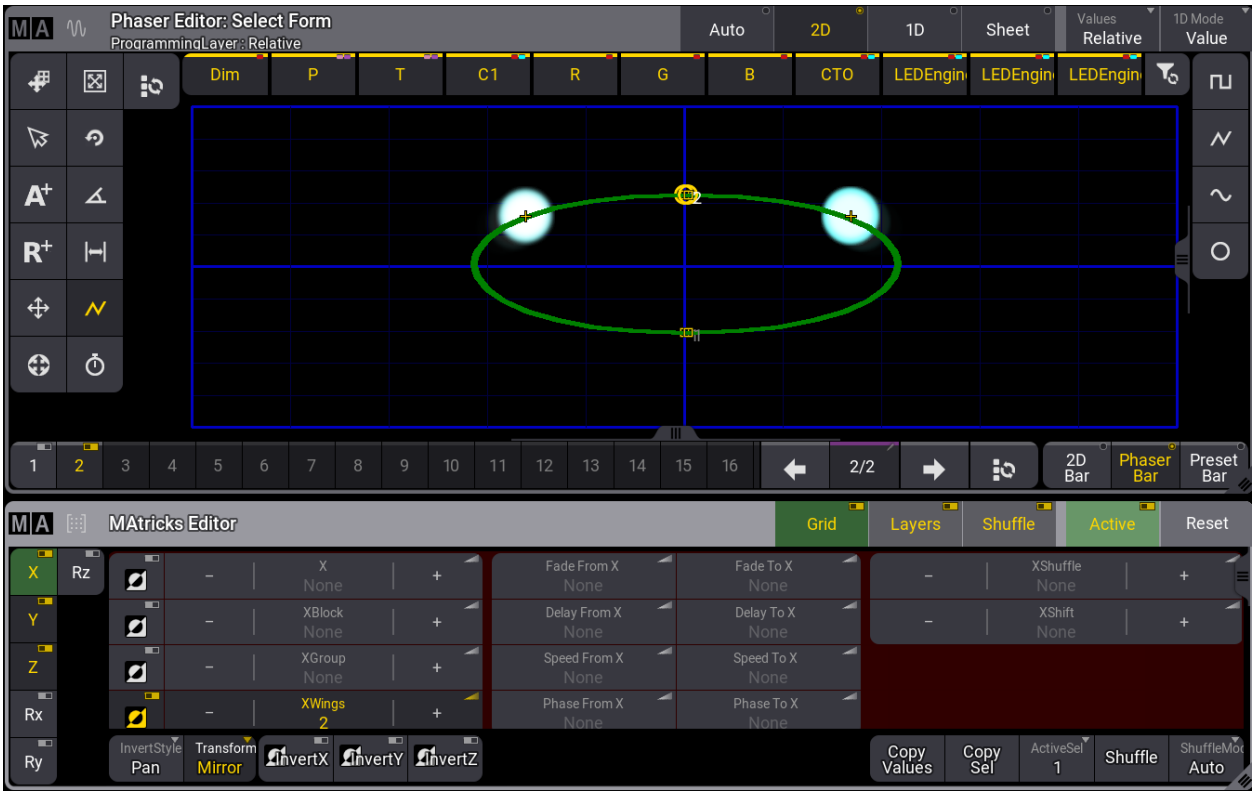
Phaser with circle form

5. Open the MAtricks window and set XWings to 2.
6. To set Transform to Mirror, tap and hold Transfom. A pop-up opens. Tap Mirror.



Transform Mirror pop-up

7. The result is shown in the image below:



If Transform is set to Mirror:

- The font color of the Transform button is yellow.
- InvertStyle changes to Pan.

If Transform is set to None:

- Inverts are switched off.

	Important:
	Setting the MAtricks, for example, XWings, to None and Transform to Mirror, disables Invert for this setting.
	Important:
	If there are only values for one step active, the values will be mirrored in this step.
	Important:
	As soon as there are attributes with two or more steps, Mirror will only mirror the values of the multi-step phasers.

Cues and Preset along with Transform Mirror

Transform - Mirror works by transforming values of the inverted fixtures to create symmetry. This requires changing the original values in the Programmer to hard values.

	Restriction:
	Storing the transformed values directly to a Cue will not store the preset reference, if the transformed values are not part of the used phaser preset.
	Hint:
	To avoid losing references, store the transformations with MAtricks to a

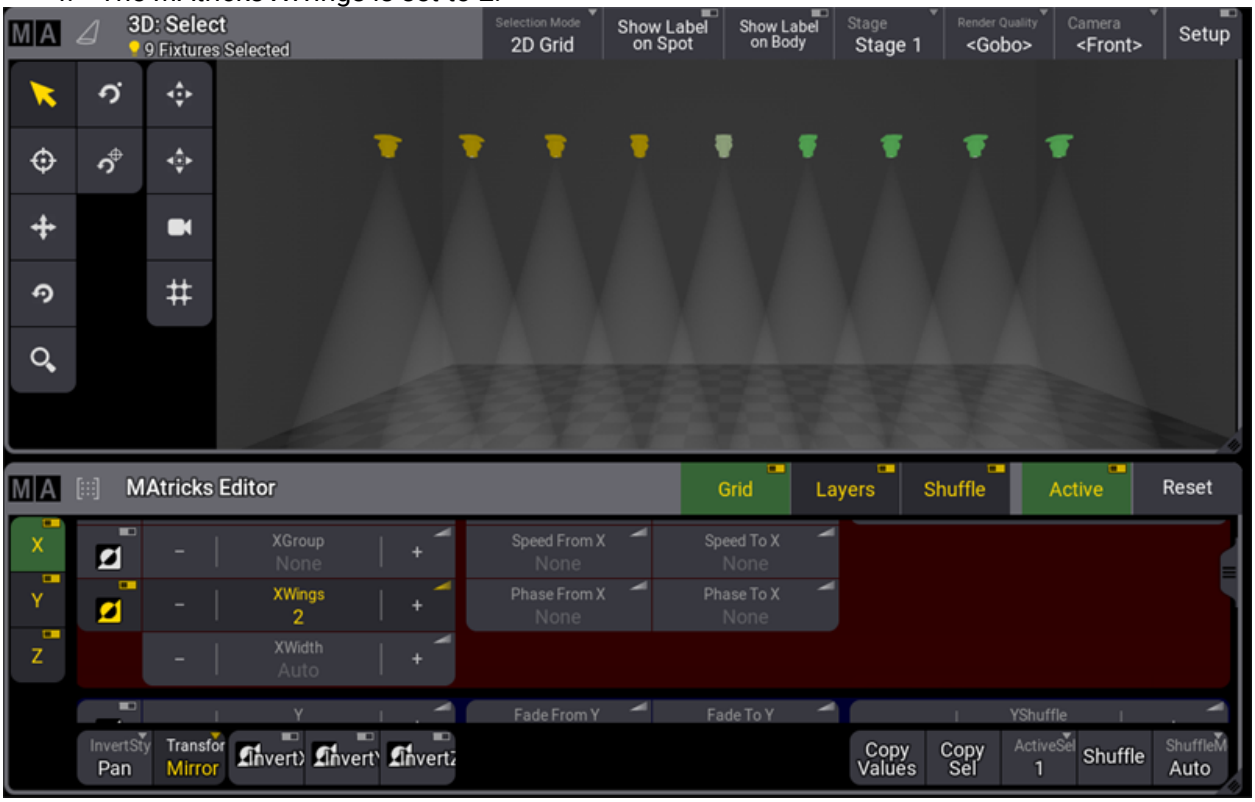
	new Preset object.

When transformed values are combined with MAtricks, the reference links are automatically saved to the preset object. Calling this preset with transformation MAtricks included will keep the links when storing it to a cue.

Odd Fixture Selection along with Transform Mirror

The image below shows the 3D window when working with Wings and an odd number of fixtures. Fixtures are split into three groups, shown in the image below:

1. Selected fixtures are generally colored in yellow.
2. The center fixture is marked as an edge fixture. It is colored light green.
3. Selected Fixtures with enabled Transform Mirror setting are colored in green.
4. The MAtricks XWings is set to 2.



Eleven selected fixtures in 3D with XWings active and transition set to mirror.

	Important:
	An edge fixture in an odd fixture selection does not follow all values to keep the symmetry.

Example: When creating a mirrored circle with the eleven selected fixtures, the edge fixture in the center will only tilt but not pan.

	Hint:
	For more information on system colors, see **System** topic.

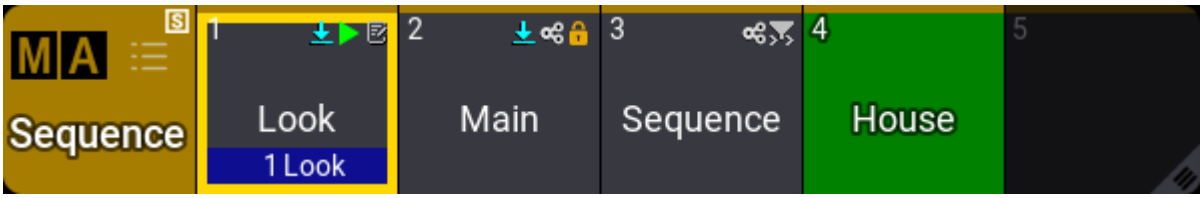
1.30. Cues and Sequences

Fixture values can be stored in **Presets** or **Cues**.

Cues are organized in **Sequences**.

Executors can control and playback sequences. The sequence is played back from the sequence pool. The executors are handles for the sequence.

The sequences are all in the **Sequence Pool**. This is created like any other window - see more in the **Add Window topic**.



Sequence pool with sequence 1 running a cue

The sequence contains information about the cues and how to transition from one cue to another.

Cues can contain fixture values (sometimes called "hard values"), references to presets, and recipes. Read the **Store Cues topic** to learn about making cues.

All cues have a minimum of one step. If there is more than one step, then it is a **Phaser**. Follow the phaser link to read about multi-step cues.

Cues have **Cue Parts**. All values are actually stored in the parts. All cues automatically have part 0. Other parts can be created, but it is not necessary. Creating several parts allows for the separation of values into different sub-cues. The parts can have different timing or properties, but parts are connected to the primary (parent) cue. This means that all the cue parts in a cue trigger together with the primary cue.

The cues in a sequence can be seen in a **Sequence Sheet**. Read more about looking at the sequence in the **Look at Cues and Sequences topic**.

Sequences have a lot of different settings. Read the **Sequence Settings topic** for more information.

Pool Action and Object Action

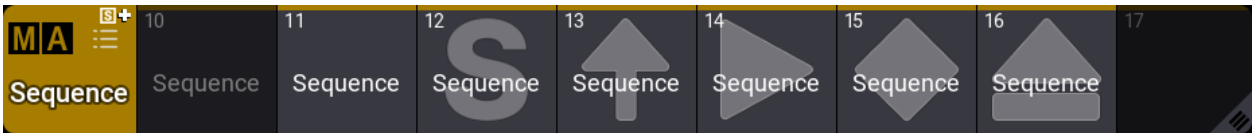
A sequence pool has a pool action for the pool objects. The individual sequence pool objects also have an object action. This action is performed when the sequence is tapped in the pool without a (relevant) keyword in the command line.

The object action is used if the **Use Object Action** is active in the **Sequence Pool Settings** (link below).

The defined pool or object action is performed when the sequence is tapped in the pool without a (relevant) keyword in the command line.

A small icon in the upper right corner of the pool title object indicates the selected pool action. If **Use Object Action** is activated in the settings, a plus icon will be displayed next to the pool action. See the example image below.

An icon overlay is added to the pool object when **Use Object Action** is activated. The pool object is grayed out if the object action is set to **None** (see sequence 10 in the example below). There is no icon overlay displayed if the object action is set to **Pool Default** (see sequence 11 in the example below).



Example of the sequence pool with pool objects assigned all the possible object actions. The pool action is set in the Sequence Pool Settings. This is described in the **Common Window Settings topic**.

The object action is set in the object editor described in the **Sequence Settings topic**.

Reference

If a sequence is referenced somewhere, it gets a small reference icon (). Referenced means that it is used or assigned somewhere. The **Info window** can show where an object is referenced.

Filters

The sequence can have a world or a filter assigned as an input filter and an output filter.

If there is an input filter, there is a small icon () in the pool object. The input filter only allows the data in the world or filter to be stored in the sequence.

If an output filter is assigned, the pool object has a small icon ( ). The output filter allows only the fixtures and attributes in the world or filter to be played back from the sequence.

See the example image above. Sequence four is called "BlueLights". It has both an input and output filter assigned, and this is a combined filter icon ().

Read more about filtering sequences in the **Worlds and Filters section**.

Shared Data

Two sequences can be connected and share the cue data. If a sequence shares data with another sequence, then it has a small icon () in the pool object. See sequences two and three in the example image above.

Shared data means that the cue content and the cue settings (for example, fade times) are the same. Changing one of the two sequences will also change the other sequence. For example, creating cues, changing cue content, or deleting cues.

The sequence settings can be different. Different executors can control the different linked sequences and playback different cues in the two (or more) linked sequences.

This can be useful if, during rehearsal, a video programmer and a lighting programmer want to work with one sequence but must be able to run different cues. The relevant input and output filters can be assigned to the two shared sequences. Two different executors can be assigned to control the two linked sequences. Now, one sequence can playback and store video data, and the other can run and store light data. All values are stored in the same linked sequence. So when the show is ready, an operator can remove the filters and just playback one sequence with all the data.

Subtopics

- **What is Tracking**
- **Sequence Sheet**
- **Content Sheet**
- **Sequence Settings**
- **Store Cues**
- **Update Cues**
- **Copy Cues**
- **Cue Recipes**
- **Store Settings and Store Preferences**
- **Play Back Cues**
- **Move In Black**
- **Cue Timing**
- **Renumber Cues**
- **Delete Cues**

1.30.1. What is Tracking

Tracking is the principle of storing only the **changes** in the cues.

If a fixture is turned On in blue, it will stay like this until it is told to change. It does not matter how many cues there are between being told to turn On and the cue where it is told to turn Off.

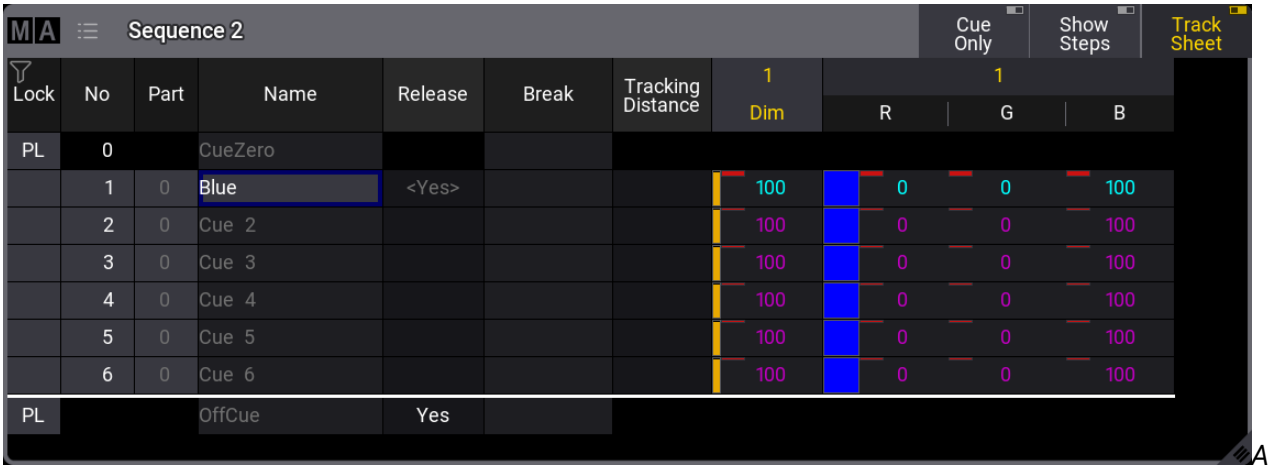
Tracking is the principle that once a parameter has a value, it stays there until it is told to go somewhere else or released from a sequence.

The following example explores the basic tracking principles and describes some of the other basic related functions after the example. Four tracking principles have their own topics: **Cue Only**, **Tracking Distance**, **Tracking Shield**, and **Break**.

Example

Fixture number 1 is stored in blue color, at 100% dimmer value in cue number 1. If there are 6 cues and fixture number 1 does not get any new information, it will stay at 100% in blue color in all 6 cues.

The best way to see this information is to turn the **Sequence Sheet** into tracking sheet mode. This can be activating the Track Sheet setting either in the title bar or the window settings.



A sequence with 6 cues - fixture 1 stored values in cue 1

Notice the difference in text color between the values in cue 1 and the others. The magenta text color indicates that the value is not stored in the cues but is a tracked value from a previous cue.

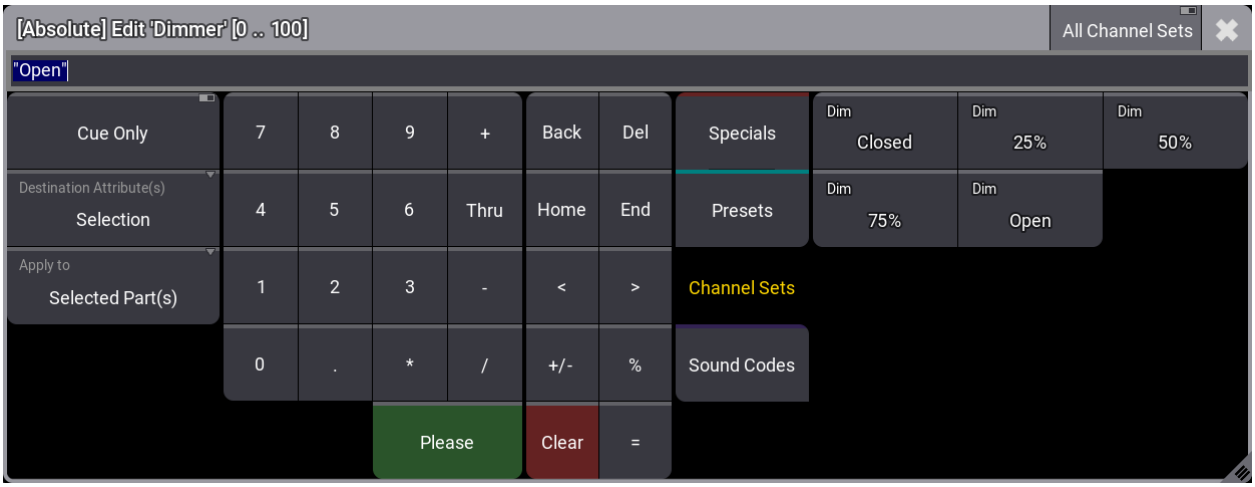
Now, change the fixture's color to red and set the dimmer to 100% again.

Store this in cues 4 **and** 5 (using the Merge option). Now it looks like this:

A sequence with 6 cues - fixture 1 has a new color in cues 4 and 5 - blocked values

The text color of the dimmer value is now white in cues 4 and 5. All the color values are also white in cue 5. This indicates that the values are stored here but are the same as the tracked value. These are called "Blocked" values. This means that if any value is changed in any previous cue, it would still be 100% in cue number 4. It blocks the tracked value from cue 1 (the same value), and the value stored in cue 5 is tracked to the end.

Values can be edited directly in the tracking sheet. Edit the dimmer value in cue 3 by right-clicking with a mouse or the two-finger edit gesture on a touchscreen. This opens the **calculator**.



Calculator editor

Select Channel Sets and tap Closed.

This closes the calculator and changes the dimmer value to 0 in cue 3 in the sequence.

sequence with 6 cues - fixture 1 has a new dimmer value in cue 3

The dimmer value is now green in cue 3. This indicates that the dimmer value is now stored with a lower value than the previous cue. Only dimmer values show this green indication for a lower value.

The dimmer value in cue 4 is now cyan, indicating a higher value than the previous cue.

The redundantly stored values in cue 5 can be removed from the cue by removing the stored values - also called "unblocking".

sequence with 6 cues - unblocked sequence

There are several ways to do this.

One option is to have the values for fixture number 1 as active values in the programmer (it does not matter what the values are) and store cue 5 with the "Remove" option. This would remove the values from the cue.

A second option is using a command like this:

This removes all redundant blocked values from the entire sequence number 2.

A third option is to unblock cue 5. This can be done by adding "cue 5" to the command above or editing a value in cue 5. It does not matter what value; all that is needed is the calculator opened with a value from cue 5.

On the left side of the calculator, change the Destination Attributes(s) value to **All Fixtures** and the Apply to to **Selected Part(s)**. Now tap Specials and then Unblock. This unblocks all attributes for all fixtures in the selected cue parts. The selection here is the selected cue when the calculator was opened.

	Hint:
	The Move In Black (MIB) function is visible in the tracking sheet if the MIB function is activated. It appears as values with deep-sea green background. Learn more about it in the **Move In Black topic**.

Cue Zero

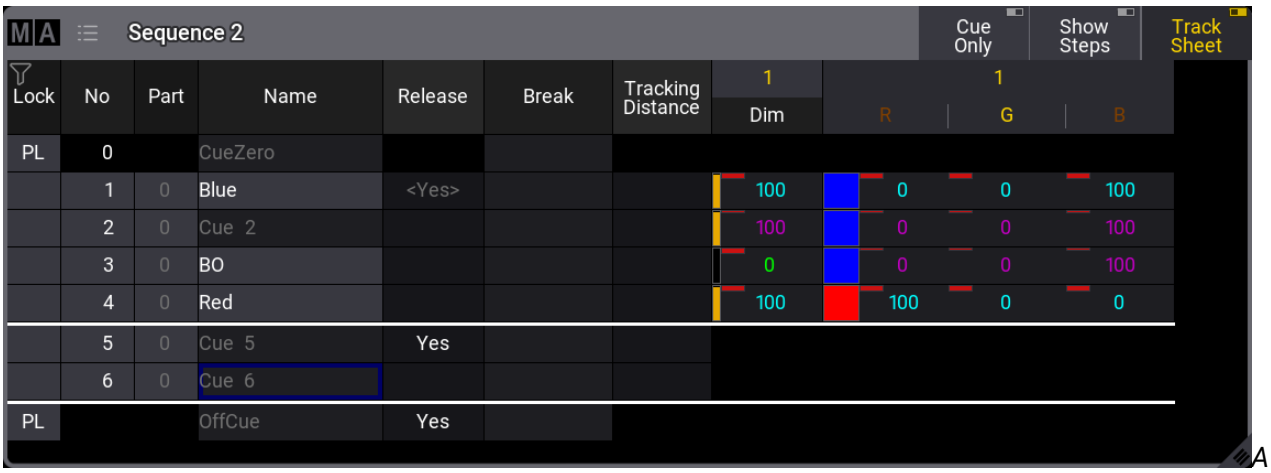
Cue Zero is an automatically created empty cue. Cue Zero can be modified or stored into using the cue name or number. Cue Zero can automatically store default values for no attributes, dimmers, or all attributes for the fixtures used in the sequence. This can be changed in the **sequence settings**.

Off Cue

The Off cue has timing information that is used when the sequence is turned Off. Release is active for the OffCue as a default.

Release

The sequence sheet has a **Release** column. Editing a field in this column changes the cell to **Yes**. This means that tracked values will be released from this cue and forward.



A sequence with 6 cues - cues 5 and OffCue with release active

It makes a white horizontal line above the released cue. This indicates that tracked values are stopped, and if there are no other values in the sequence, then the attribute is released from the sequence - empty black cells.

When an attribute is released in the sequence, then it is the same as the sequence, no longer sending any information to the attribute.

If a different sequence sends values to the attribute, these values are used. Sequence priority can be important in this case - read about priorities in the **Play back Cues topic**.

If the attributes do not receive values from any sequence or the programmer, they return to their default values.

It is also possible to store values using the **Release** option. This does not actually store the values but empties the relevant fields in the sequence tracking sheet.

Turning Tracking On or Off

In the **sequence settings**, there is the option to turn On or Off the tracking function.

When tracking is Off, the tracked values are gone, and where the value was tracked, it will be released instead.

Looking at the sequence used in the example above, it would look like this without tracking:

Sequence with Tracking turned Off

Fixtures 1 would only be On in cue 1 in blue. In cue 2 it would return the default values. In cue 4 it would turn On in red. In cue 5 the color returns to the default. The zero value stored in cue 3 is possibly redundant since it is often the default dimmer value.

Subtopics

- **Cue Only**
- **Tracking Distance**
- **Tracking Shield**
- **Break**

1.30.1.1. Cue Only

Cue Only is a **store option** for cues. When stored in a cue with Cue Only, the tracked values will be blocked in the next cue or cue part to preserve the previous look on stage. The programmer's actual values will only be stored in the target cue or cue part.

Cue Only can be used when storing into existing cues - except the last cue. There is no hard key for cue only, so it needs to be activated otherwise. These are the following options:

- As command line option keyword **/CueOnly** or /CO after the normal **Store keyword**.
- In the store options: To open the store options, press and hold Store for at least 2 seconds. Then turn On cue only by tapping Cue Only.
- In the **Store Cue pop-up**. When storing onto an existing cue, the Store Cue pop-up will appear and ask whether to Overwrite, Merge, Remove, Release, or Cancel the current store operation. It is also possible to decide if the cue is to be stored with active cue only within the pop-up. Remove and Release can be used for Cue Only. The pop-up only appears if the desired cue is not the last one in the sequence.

When storing cue only, the grandMA3 software decides, based on the 3 following rules, which cue part will block the original values:

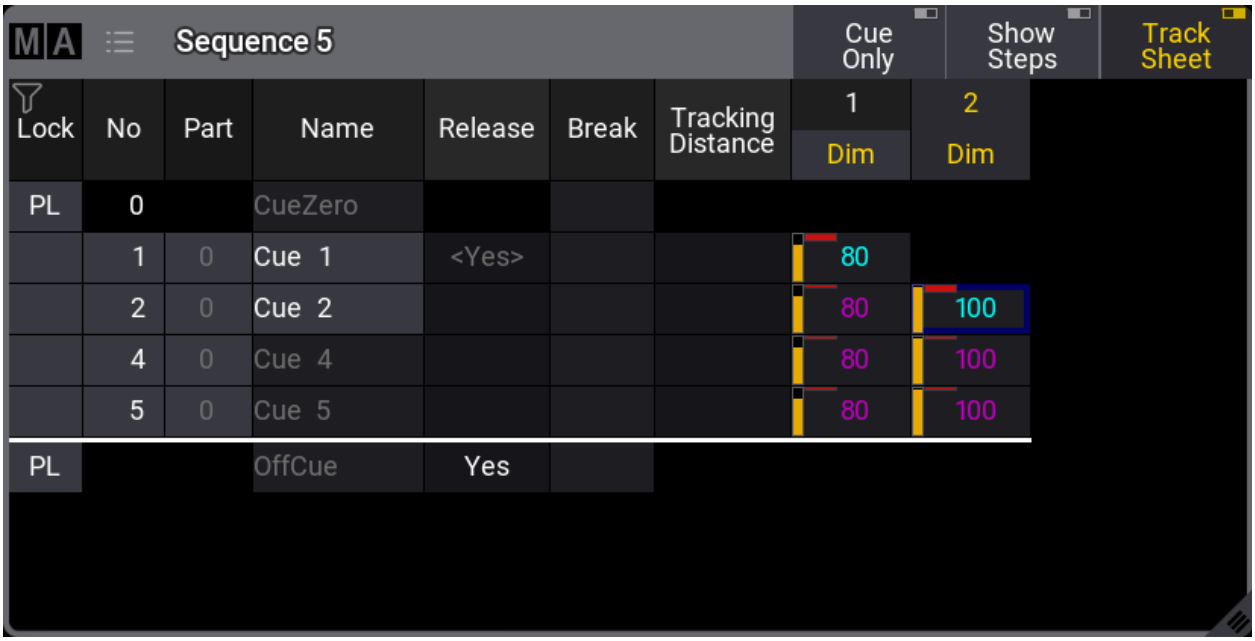
1. The default cue part is cue part 0.
2. If the next cue already contains a part with the same name as the cue part from which the original value is coming, the blocked value will be used in this part.
3. If the cue already contains attributes of the same feature group, the previous values will be blocked in this cue.

If none of the rules are true for phaser values, a new cue part will be created where the previous values will be blocked.

Example

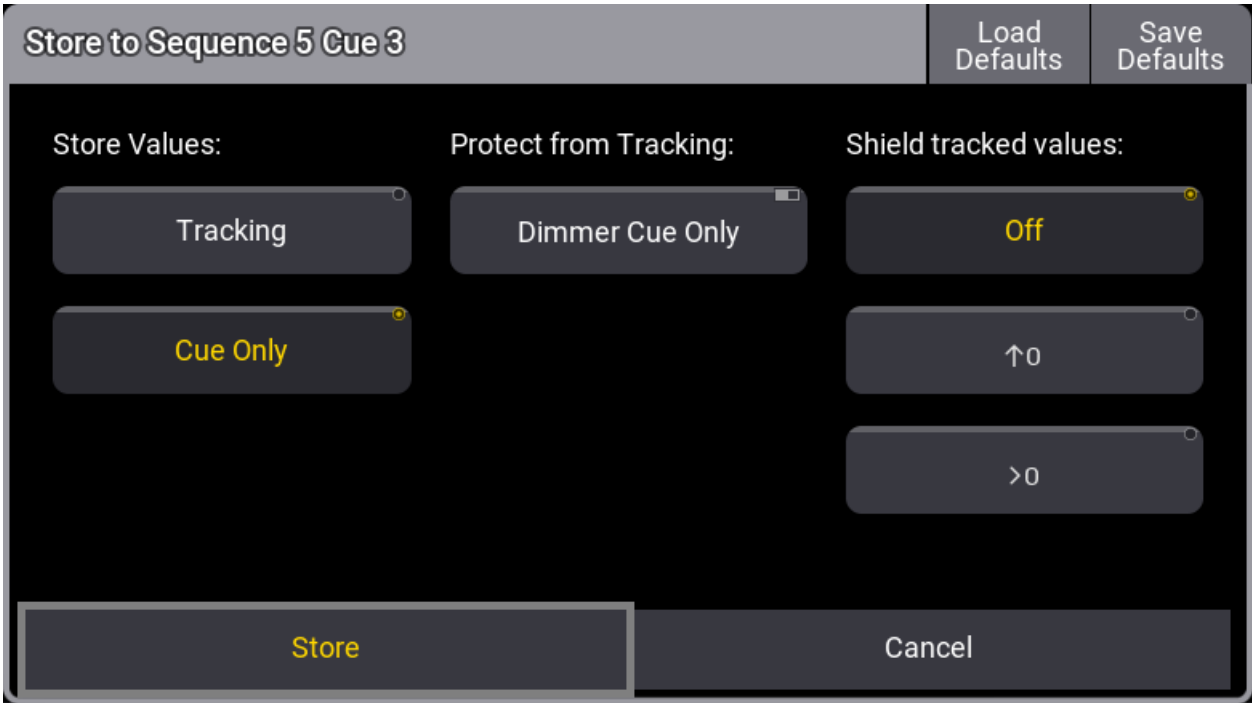
In this example, we have a sequence with the following cues and values:

Fixture 1 is at 80% in cue 1. Fixture 2 is stored at 100% in cue 2. Cues 4 and 5 exist, but they are empty. The values for fixtures 1 and 2 track into the other cues.



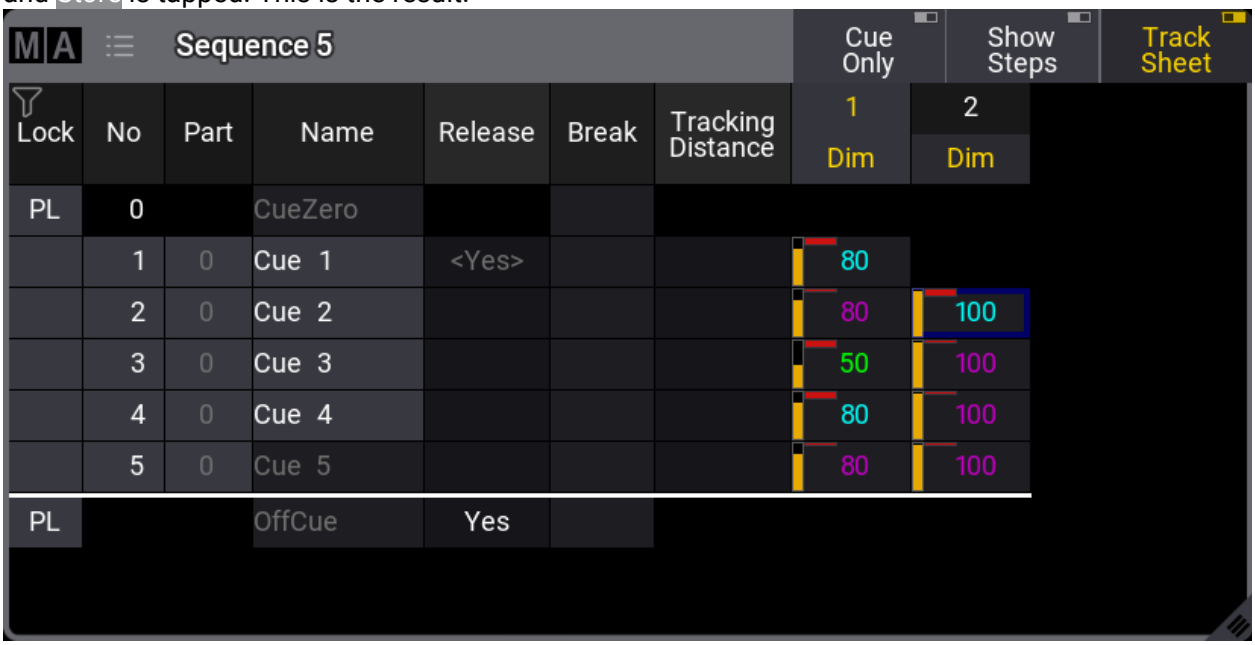
The starting sequence without cue 3. Now, fixture 1 gets an active dimmer value of 50%. This is stored into cue 3.

The store action gives this pop-up:



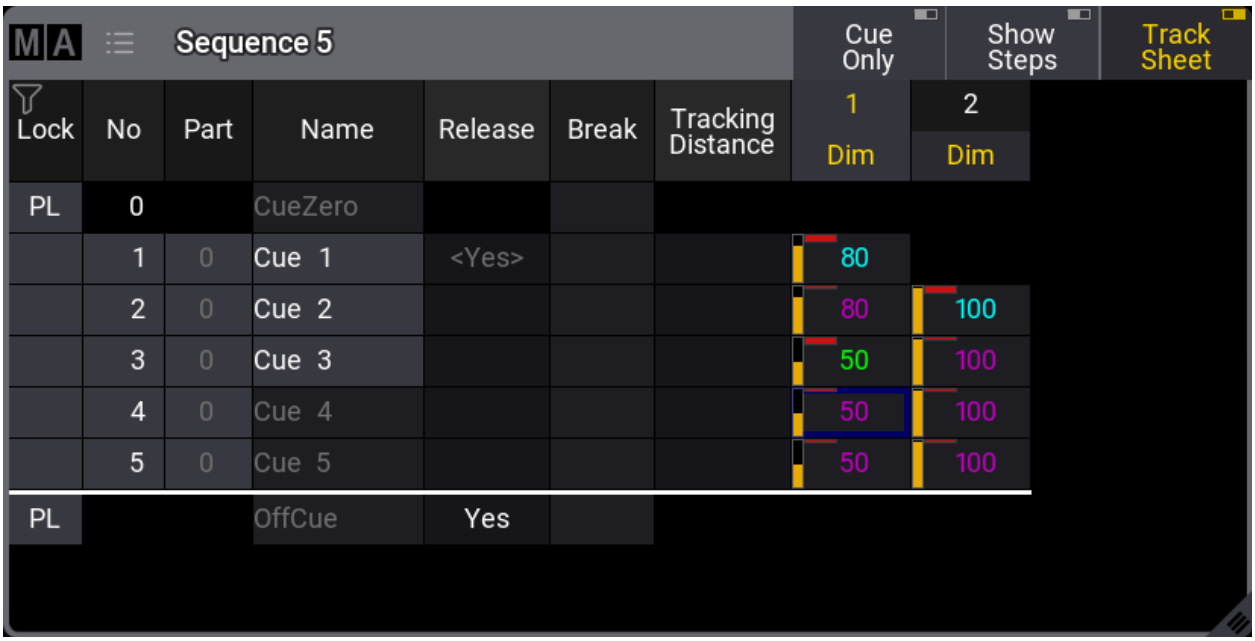
Store pop-up appears with relevant store settings. Here, Cue Only is selected in the Store Values setting,

and Store is tapped. This is the result:



The resulting sequence using Cue Only. Cue Only preserves the look of cue 4 by adding the 80% as a stored value in cue 4.

If the Tracking option is chosen instead, then the result would look like this:



The resulting sequence using Tracking. Now, the new value for fixture 1 tracks into cues 4 and 5, thus changing the look of the cues.

1.30.1.2. Tracking Distance

Tracking and common tracking functions are described in the **What is Tracking topic**.

Tracking values can have a distance. This can be changed in the **Tracking Distance** column in a **Sequence Sheet**.

There are two types of tracking distances: Normal tracking tracks until a specific cue number. Delta tracks values to a cue number calculated by the cue number, where it is set, plus a number.

Both types release the attribute when the tracking distance ends. This will return the attribute to a previously tracked value or release it from the sequence. See the examples below.

A vertical white line next to the value displays the attributes affected and the number of cues the tracking distance is valid.

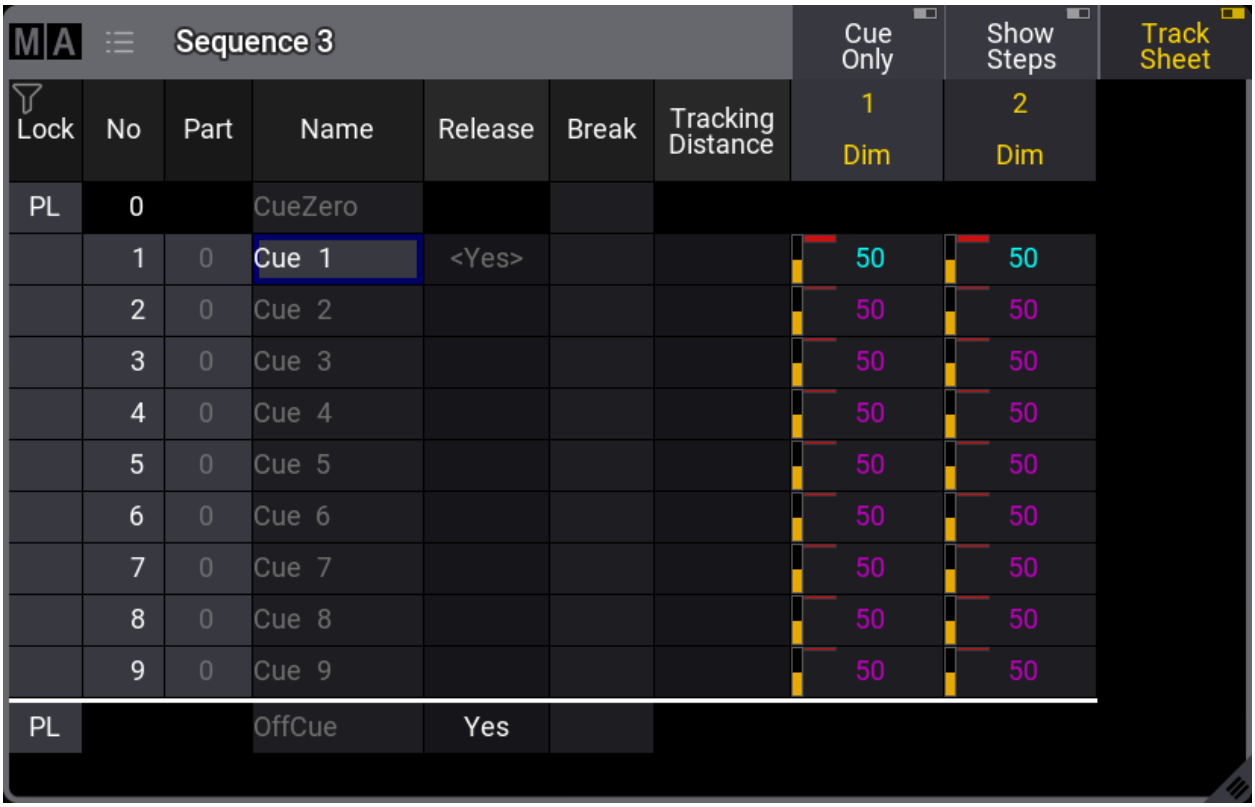
A tracking distance affects all values stored in the cue where the distance is set. The tracking distance value does not consider cue parts where the distance ends.

Tracking Using Cue Number

The tracking distance can be set to track to and include a specific cue number using the tracking distance with a normal number input. See the following example for an explanation.

Example

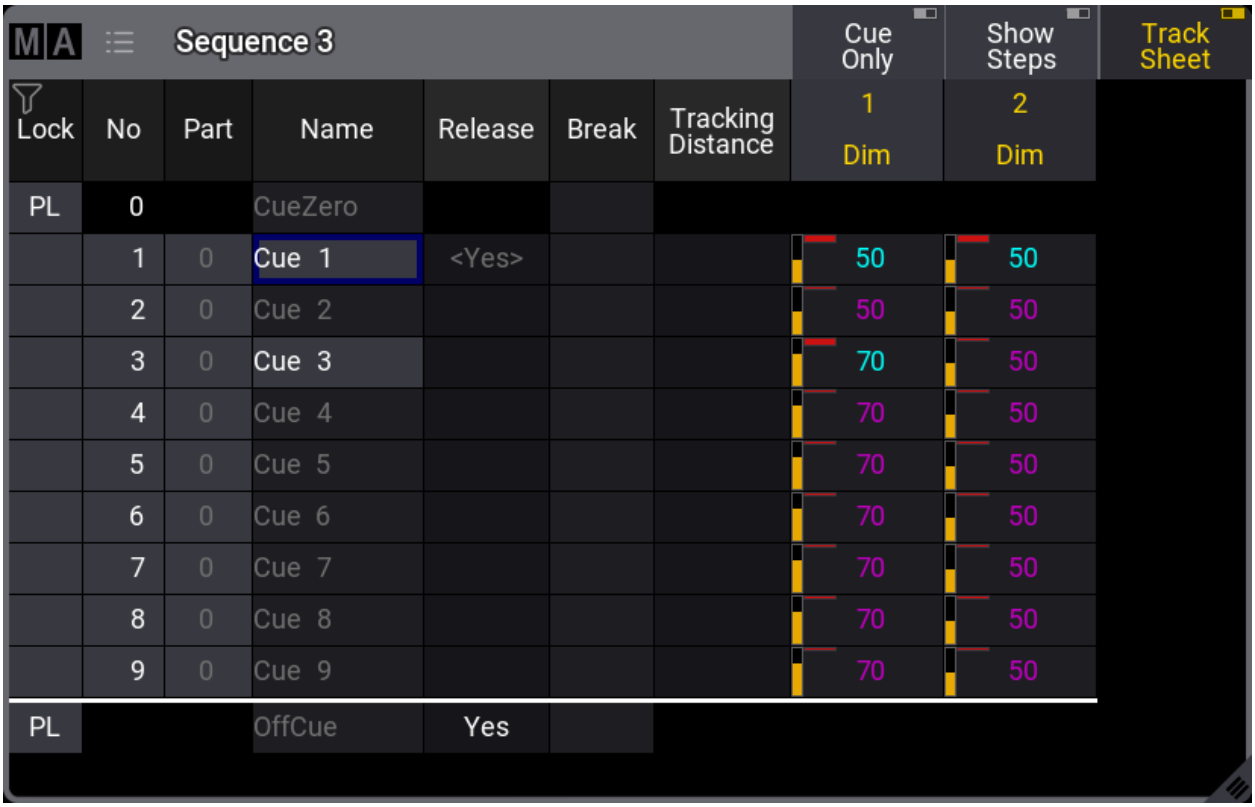
The starting point for this example is a simple sequence with nine cues and two fixtures dimmer value set to 50% in the first cue:



Base sequence for Tracking Distance using cue number

Normal tracking rules dictate that the values in cue number 1 are tracked to the end where the **OffCue** releases the value - because **Release** is set to "Yes"

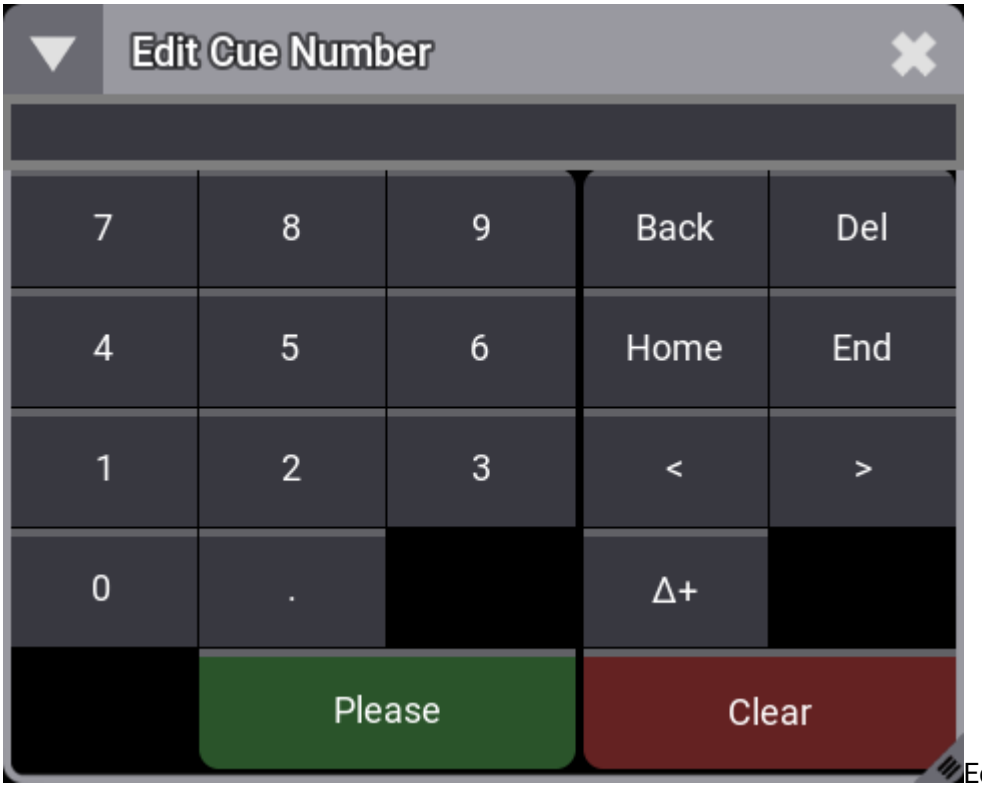
A dimmer value of 70% for fixture 1 is added to cue number 3.



Value added to cue number 3

A tracking distance is set by editing the Tracking Distance cell for cue number 3.

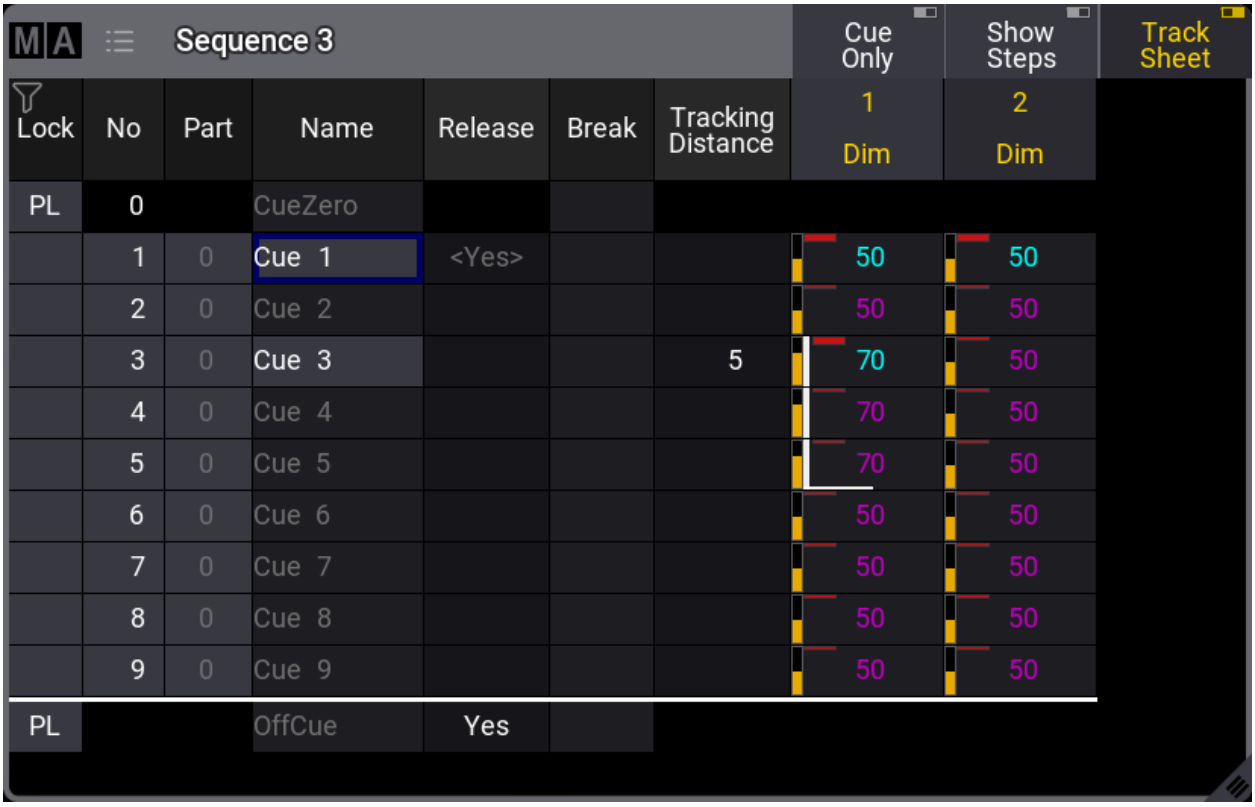
This opens the **Edit Cue Number** pop-up, where a number can be typed.



Edit Cue Number pop-up

The number typed is the cue number, after which the tracking should stop. The cue does not need to exist.

This is the result of the Tracking Distance set to a value of 5:

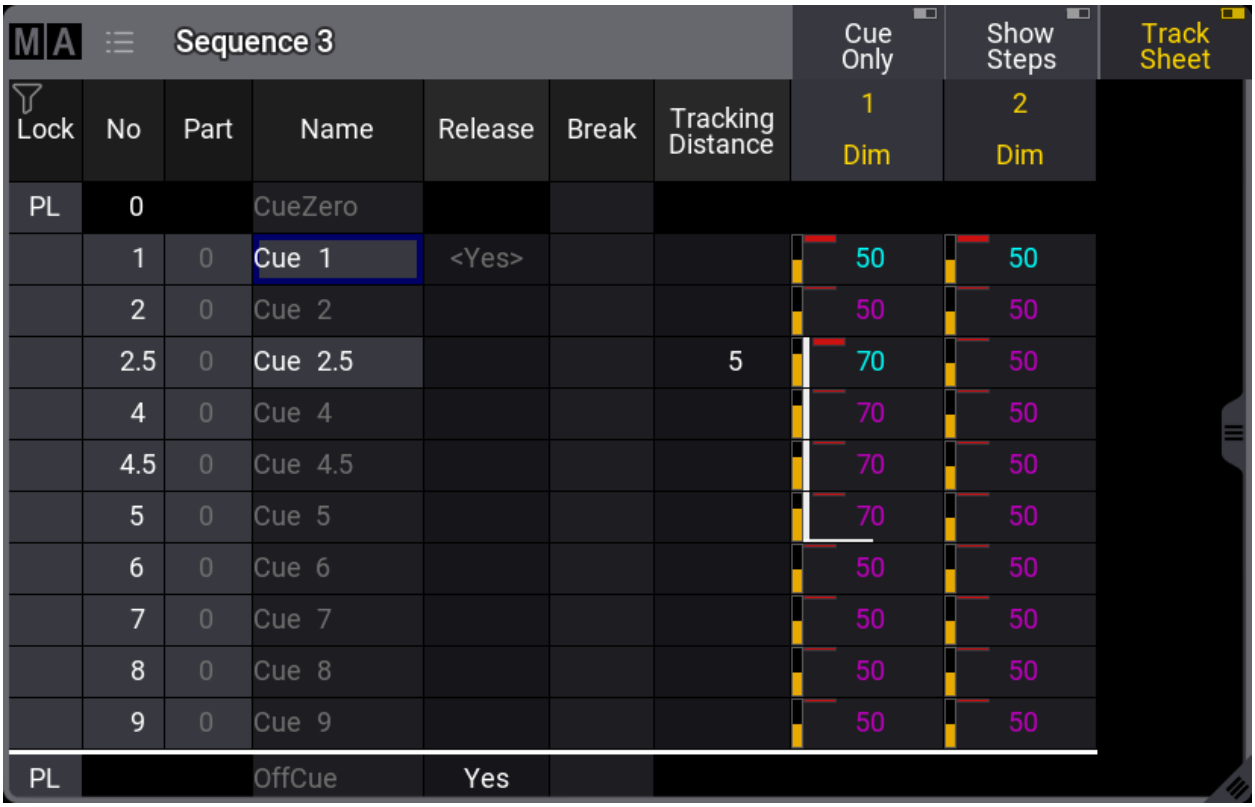


Tracking Distance set using a cue number

The tracking stops between the existing cues numbers 5 and 6 - notice the white line indicating the tracking distance. It will track to and include cue number 5. The previous tracked value of 50% continues tracking after the distance.

Adding or removing cues in between does not change that it tracks to and including cue 5.

Renumbering cue number 3 also does not change the distance.



Tracking still ends after cue 5.

Here, cue number 3 is renumbered to 2.5, and cue 4.5 is added.

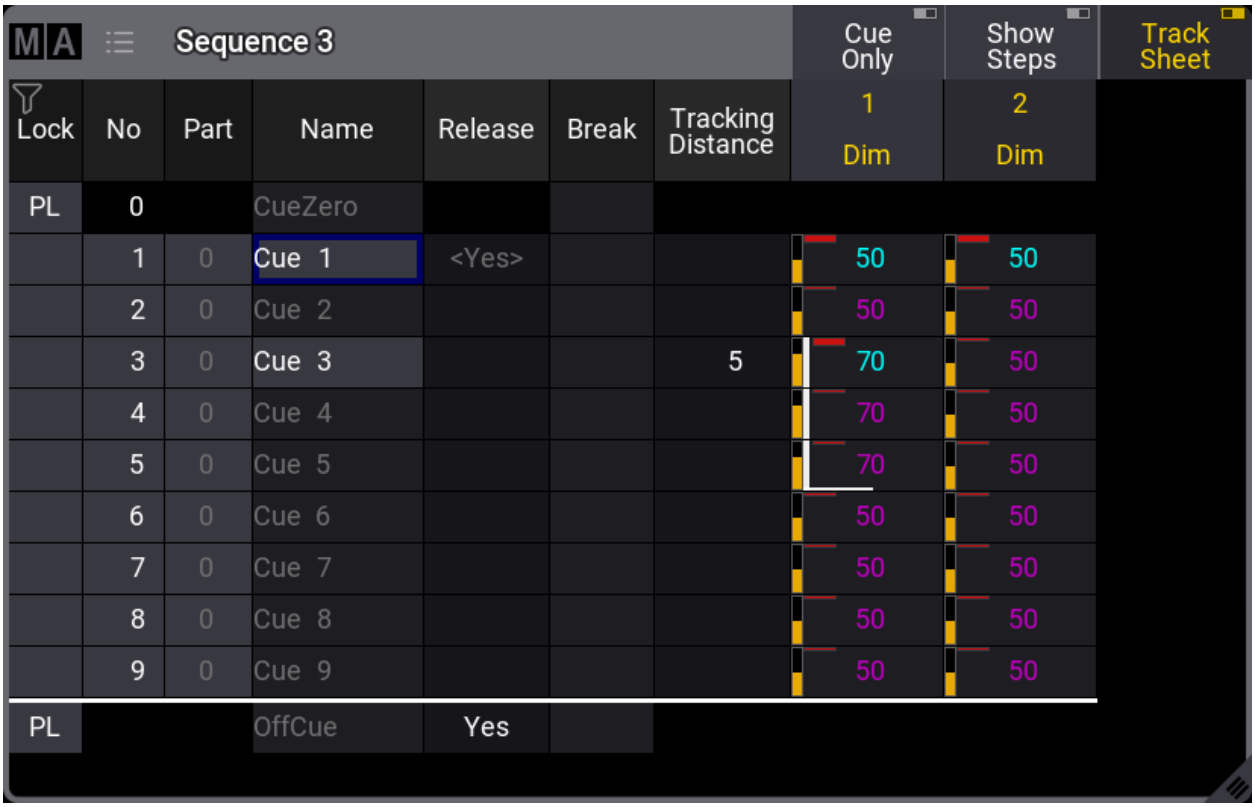
The tracking still ends after cue number 5.

Tracking Delta Distance

Setting a tracking distance with a delta number will track the value to a cue number mathematically higher than the cue where it is set. See the following example for a better understanding.

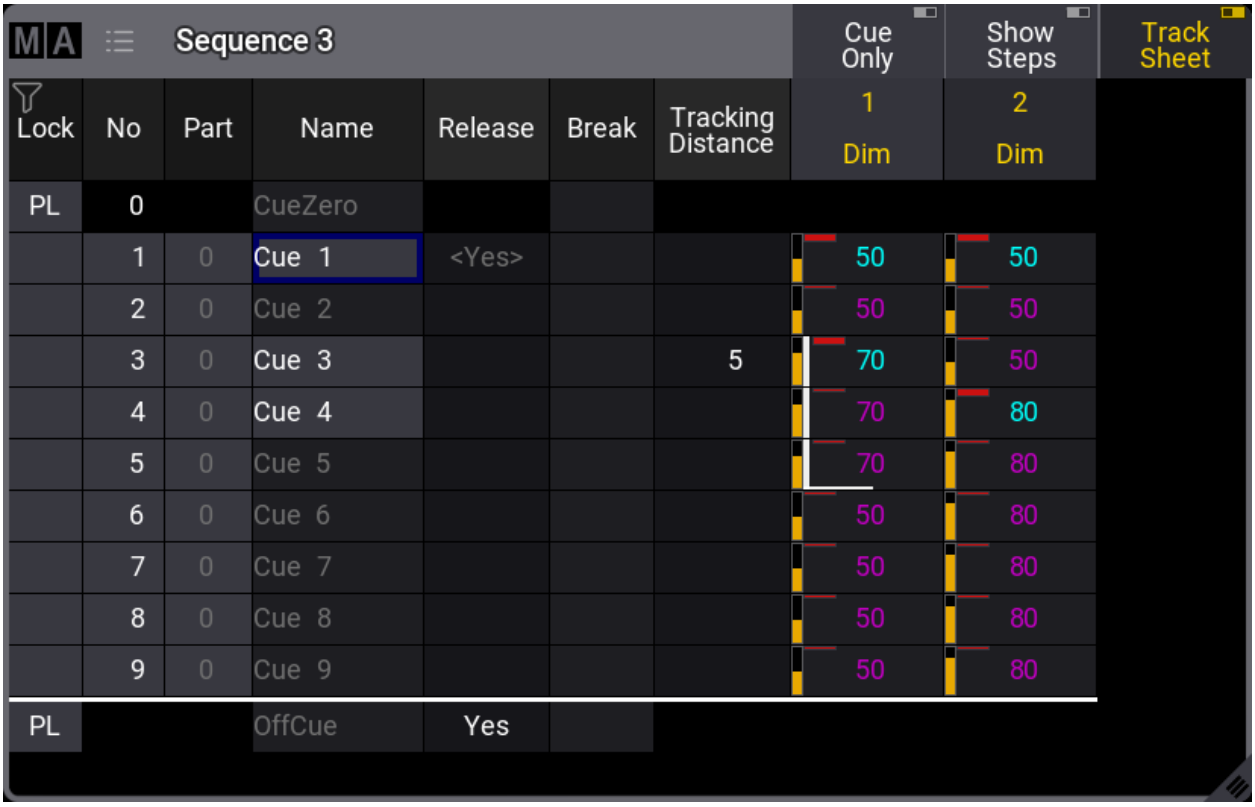
Example

The starting point for this example is the same sequence used in the previous example but cleaned up a bit:



Base sequence

Now cue number 4 is updated with a dimmer value of 80% for fixture 2:

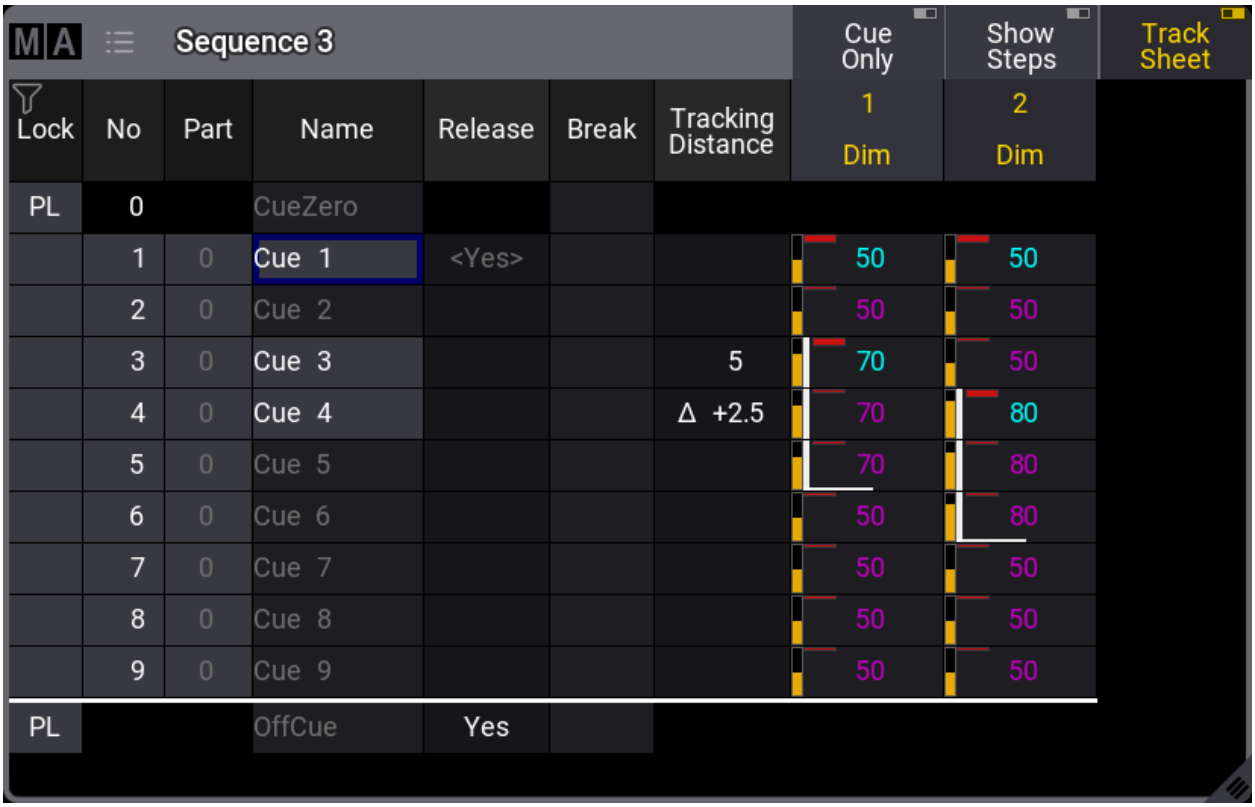


Updated cue number 4

When the Tracking Distance cell for cue number 4 is edited, the **Edit Cue Number** pop-up appears again.

Notice the Δ+ button. Use this to tap Δ+ 2 . 5 Please.

The result is that the value of 80% is tracked from cue number 4 and forward to and including cue number 6.5 (4 + 2.5 = 6.5).

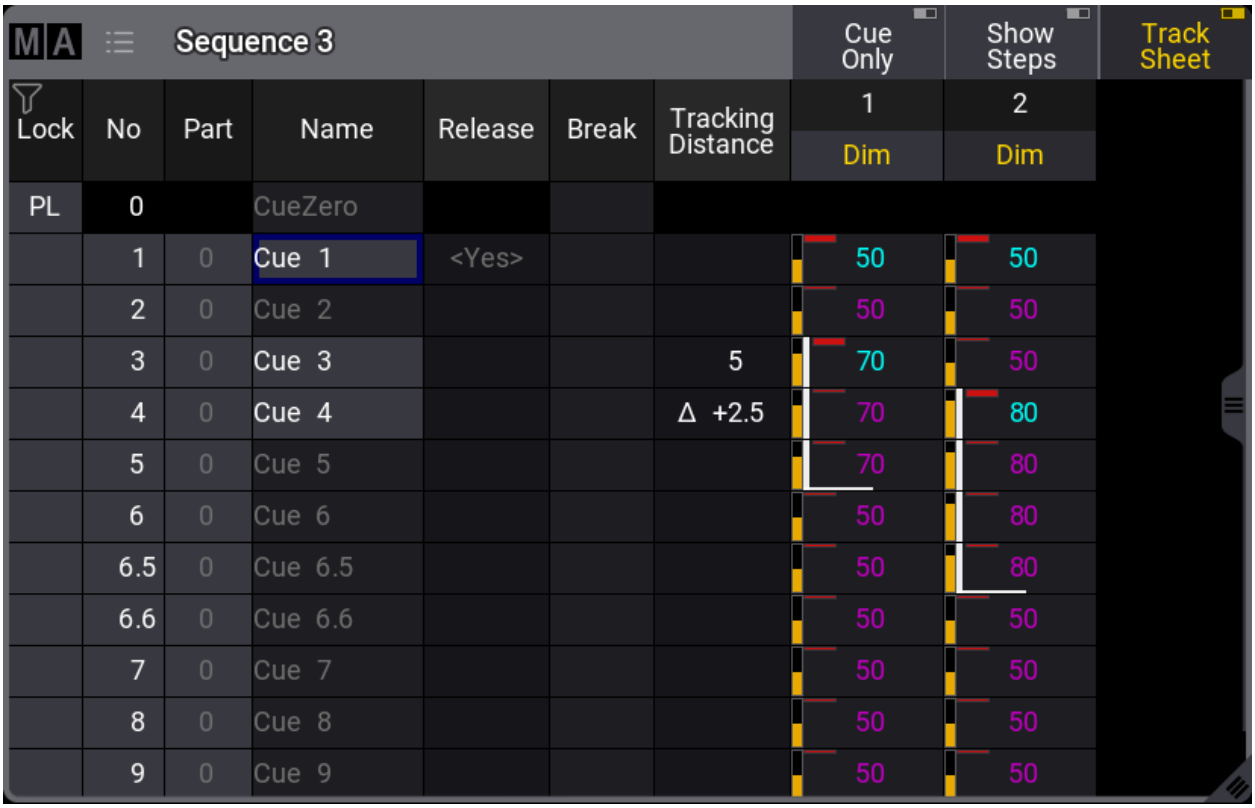


Delta Tracking Distance

Cue number 6.5 does not exist, so the current last cue for the distance is cue number 6.

Any cue added between cue number 3 and cue number 6.5 will have the 80% tracked value, but a cue number higher than 6.5 will have the 50% value.

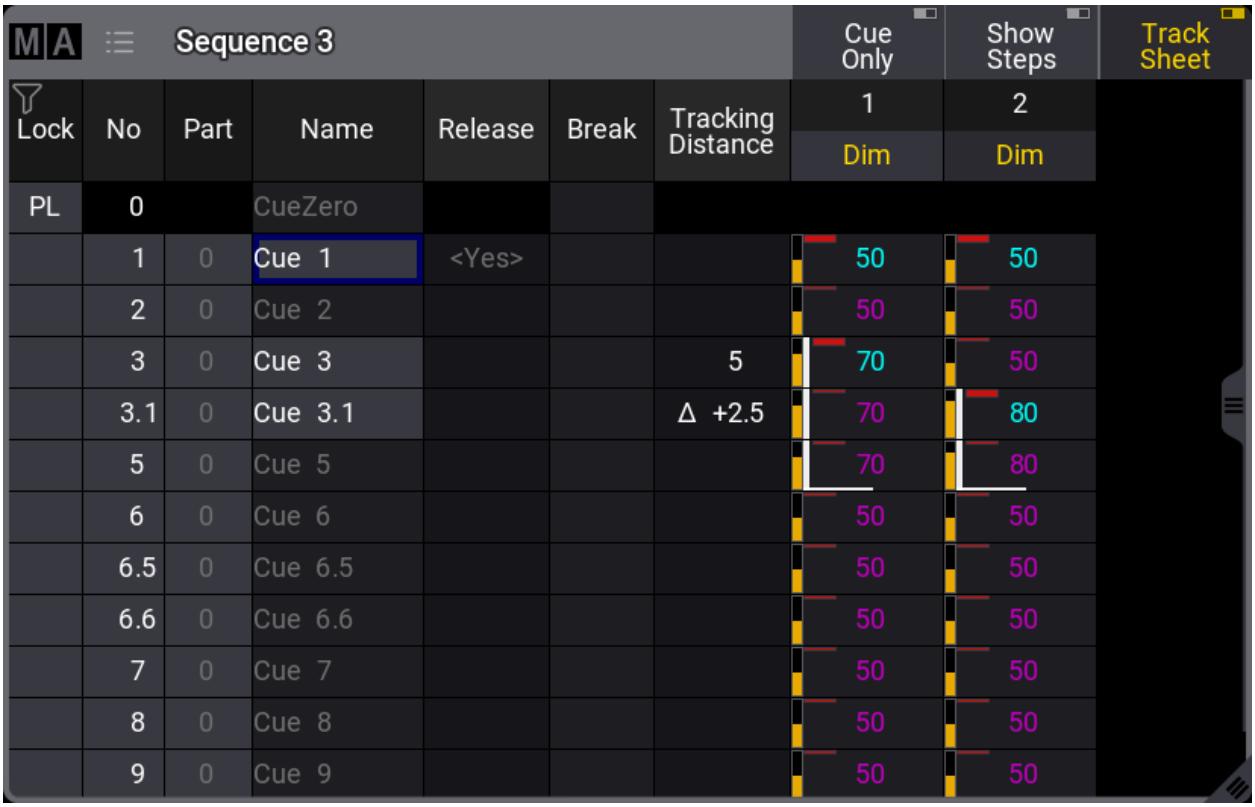
Make sure the programmer is empty and store empty cues number 6.5 and 6.6 in the sequence.



Added cues number 6.5 and 6.6

Now it can be seen that the tracking stops between cues number 6.5 and 6.6.

If cue number 4 is renumbered, this affects the distance. Renumber cue number 4 to number 3.1 by editing the cue number cell.



Updated cue number

Now the tracking stops between cues number 5 and 6. It tracks from cue number 3.1 and up to number 5.6 (3.1 + 2.5 = 5.6).

1.30.1.3. Tracking Shield

Tracking Shield is a system that can protect tracked attributes that are used in future cues. The function can be applied when storing or copying attribute values into cues. It is relevant if the new values will change the look of future cues. It uses the dimmer attribute to detect if the shield can be applied.

The function can be applied when the new attribute values are stored or copied into cues. It is a store mode available in all the pop-ups related to store and copy functions.

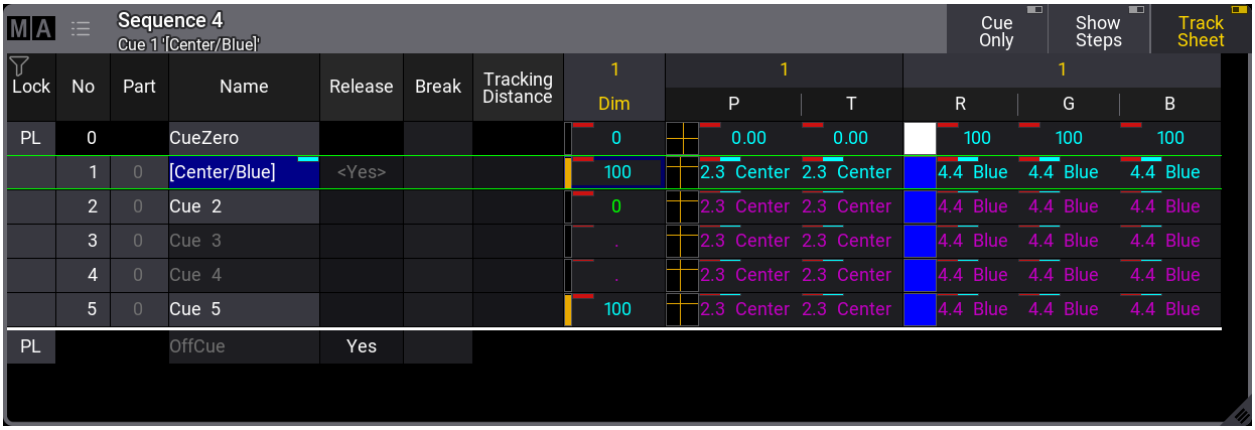
When the shield function is selected, there are two modes:

- **↑0**:
  Protects attributes in the next cue where the dimmer value increases starting from zero.
- **>0**:
  Protects attributes in the next cue where the dimmer value is above zero.

Examples

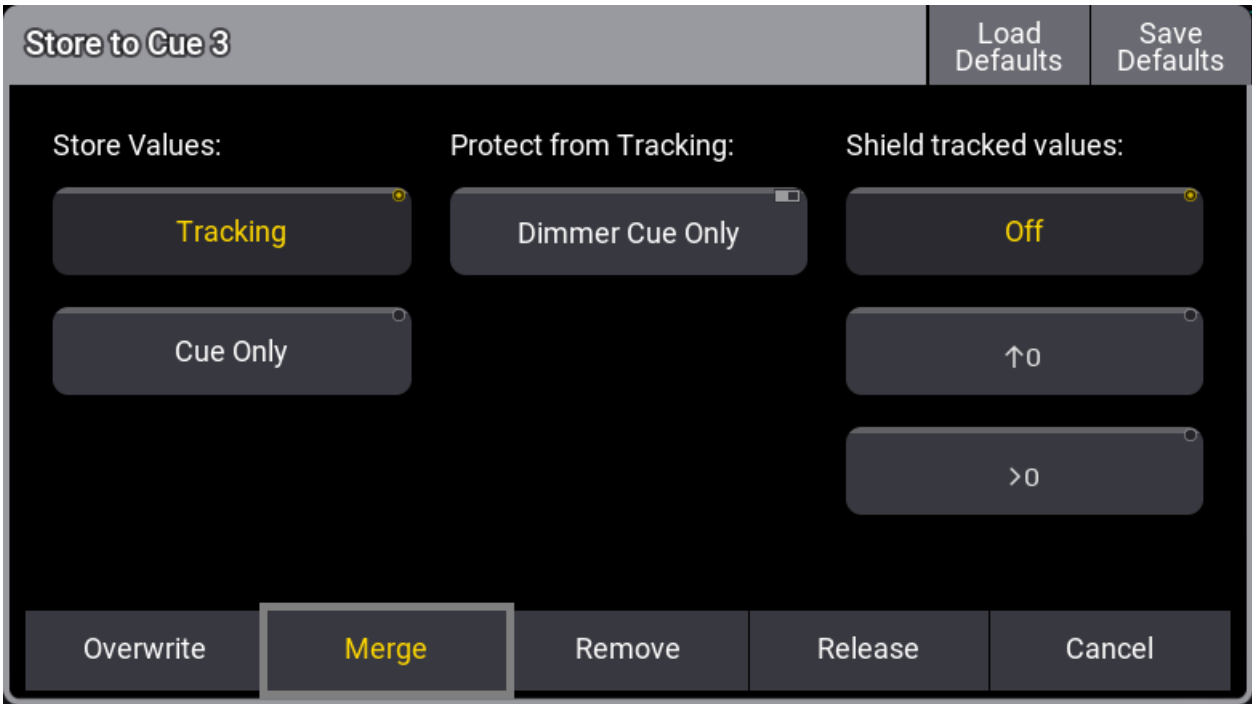
Example 1

The starting point for this example is a sequence with five cues. Fixture 1 is stored at 100% on a center position with a blue color in Cue 1. The fixture turns off in Cue 2 and On again in Cue 5. The position and color are not stored in Cue 5. They are tracked values, and the desire is that the fixture has the same position and color in Cue 5 as the currently tracked values.



Sequence Sheet showing the starting point, for example, 1

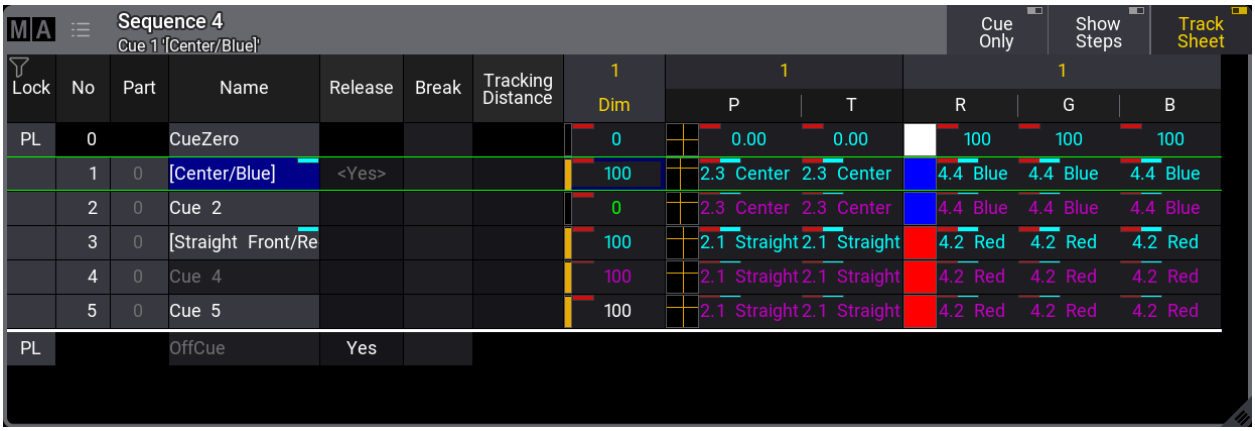
Now, the fixture is selected and given a new position, color, and dimmer value. These new attribute values are stored in cue 3. This opens a pop-up like this:



Store Cue pop-up

Here, we have the "Shield tracked values" options.

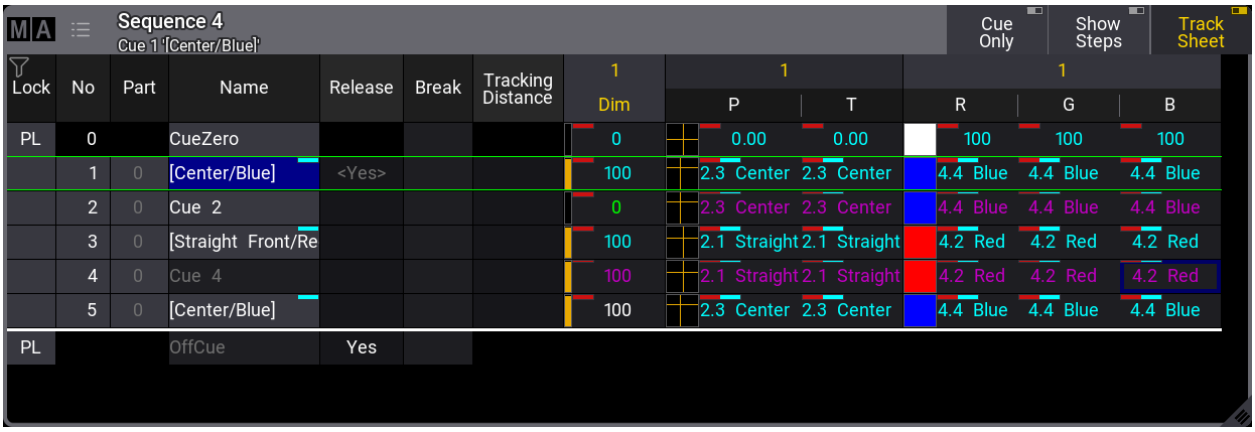
This is the result if the cue is stored using merge with "Off" selected:



Sequence Sheet showing result using normal Tracking

The new attribute values are tracked into Cue 5, and the result is that Cue 5 will look different.

This is the result if the cue is stored using merge with the "↑0" selected:

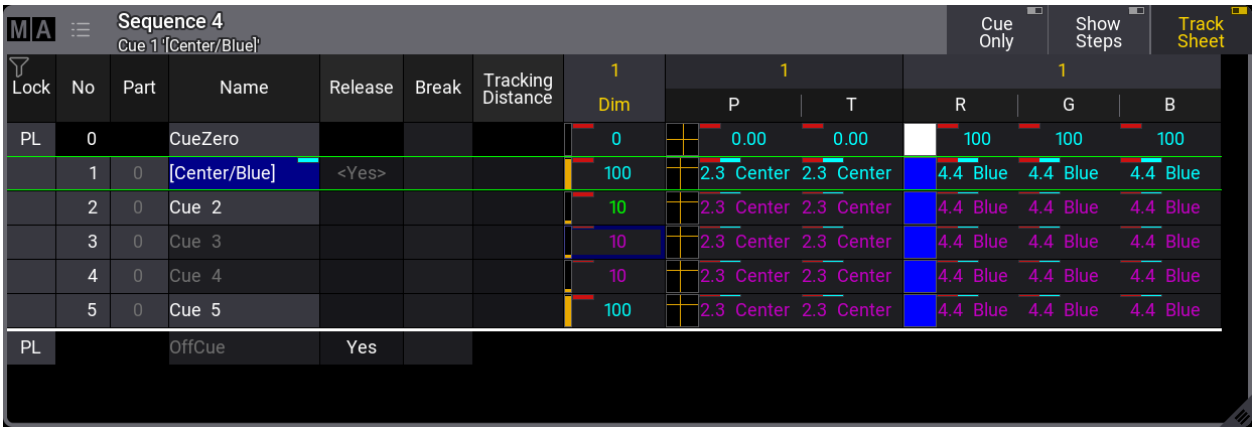


Sequence Sheet showing result using Tracking Shield in "↑0" mode

The Tracking Shield function saves the previously tracked attribute values in Cue 5, and the cue looks the way it should. The result in this example is the same if the ">0" option is selected.

Example 2

This second example uses almost the same sequence as a starting point.

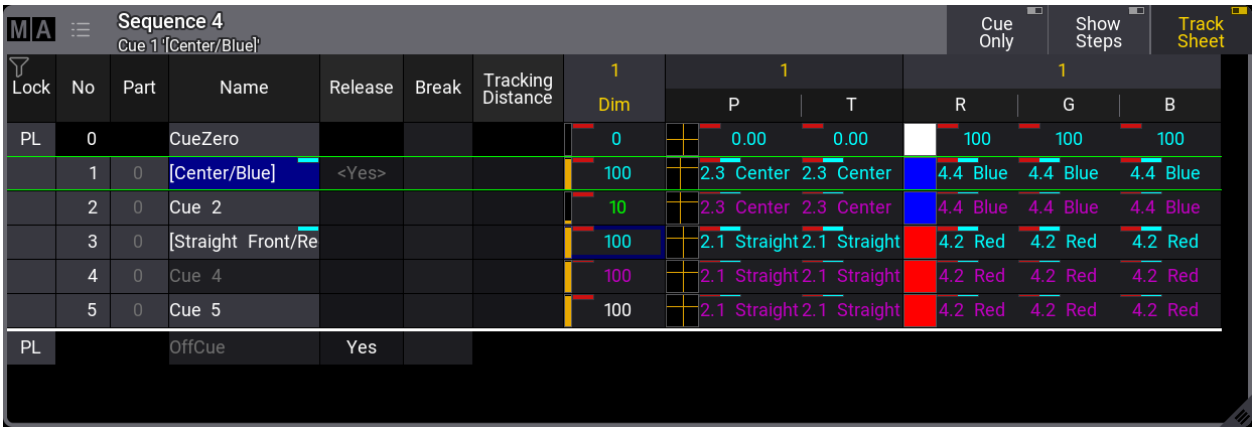


Sequence Sheet showing the starting point for example 2

The difference is that the fixture is at 10% in Cue 2 instead of 0%.

This example uses the same new attribute values as example 1, and the new attributes are also stored into Cue 3.

This is the result if the cue is stored using merge with normal "Tracking" and the shield setting "Off":

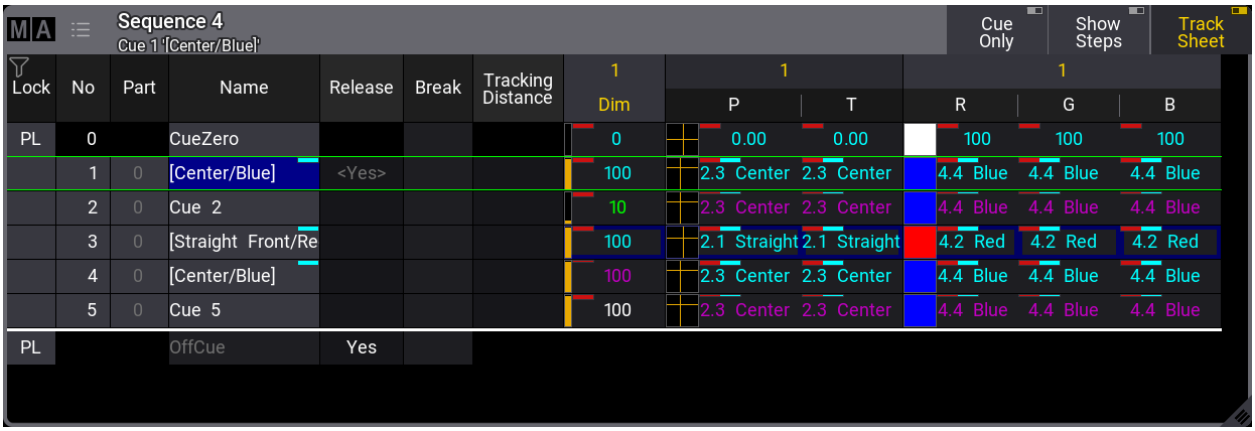


Sequence Sheet showing result using normal Tracking

The new attribute values are tracked into Cue 5, and the result is that Cue 5 will look different.

The result shown above is the same result if the cue is stored using merge with the "↑0" mode. This mode is only effective when the dimmer attribute comes from 0% and goes to a value above 0%.

However, this is the result if the cue is stored using merge with the ">0" mode:



Sequence Sheet showing result using Tracking Shield in ">0" mode

Here, the new attributes are stored into Cue 3, and the previous tracked values are stored in Cue 4, where the dimmer attribute was above 0%.

Similar results could be achieved using **Tracking Distance** and **Break**. Tracking Shield is another useful programming tool applied with the store action.

1.30.1.4. Break

Tracking and common tracking functions are described in the **What is Tracking topic**.

Breaks in cues are a filtered version of the **Release** function. Where the release stops all tracking values, the break will stop tracking of attributes based on a selected filter or world.

The main difference between the release function and setting a break, filtering all attributes, is that the release will release currently tracked attributes from the sequence immediately when the release is activated. Setting a break does not affect existing stored and tracking values. An existing break can only affect tracking values added or changed that pass the cues where the break is already set - if the selected filter or world includes the attribute.

Editing the **Break** cell in a cue opens an **Assignment Editor** pop-up.

Assignment Editor pop-up

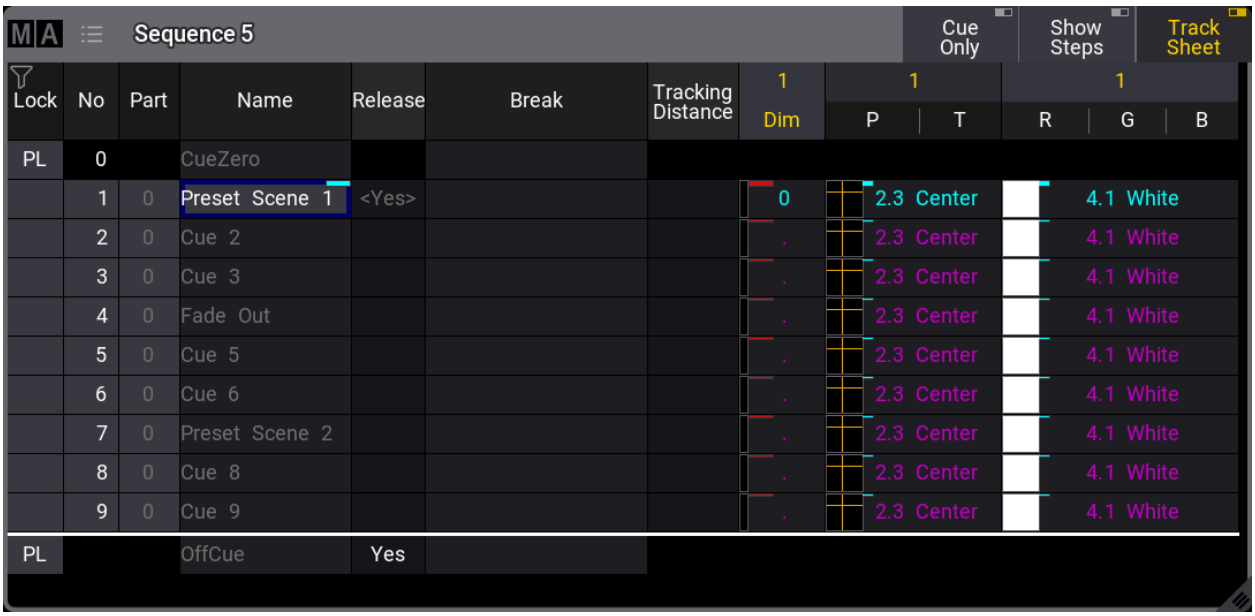
This has three tabs: Empty, Filter, and World.

Empty will remove any break setting. Filter and World allow the selection of an existing filter or world. Selecting one of these will stop tracking changes or additions, passing the cue where the break is set based on the content of the selected filter or world. Learn more about filters and worlds in the **Worlds and Filter section**.

Follow the example below for a better understanding of the break function.

Example

The base sequence for this example looks like this:

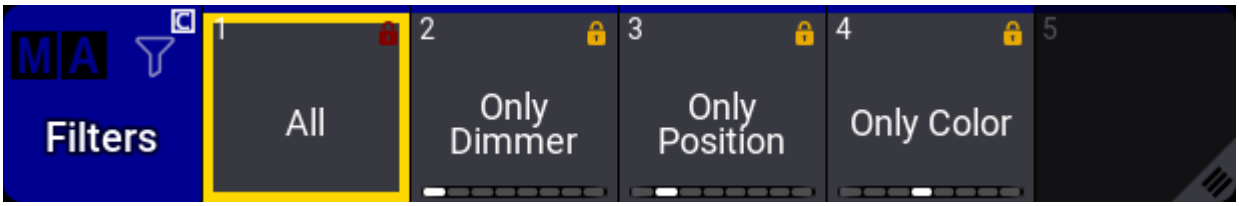


Base sequence for Break example

Here we have nine cues. Cue number 1 has some stored values (Dimmer, Position, and Color) for fixture 1.

The values stored in cue number 1 are tracked from 1 to the end. The values are released in the **OffCue** because there is a **Yes** in the **Release** column.

Two filters in the **Filters** pool will be used: "All" and "Only Dimmer".



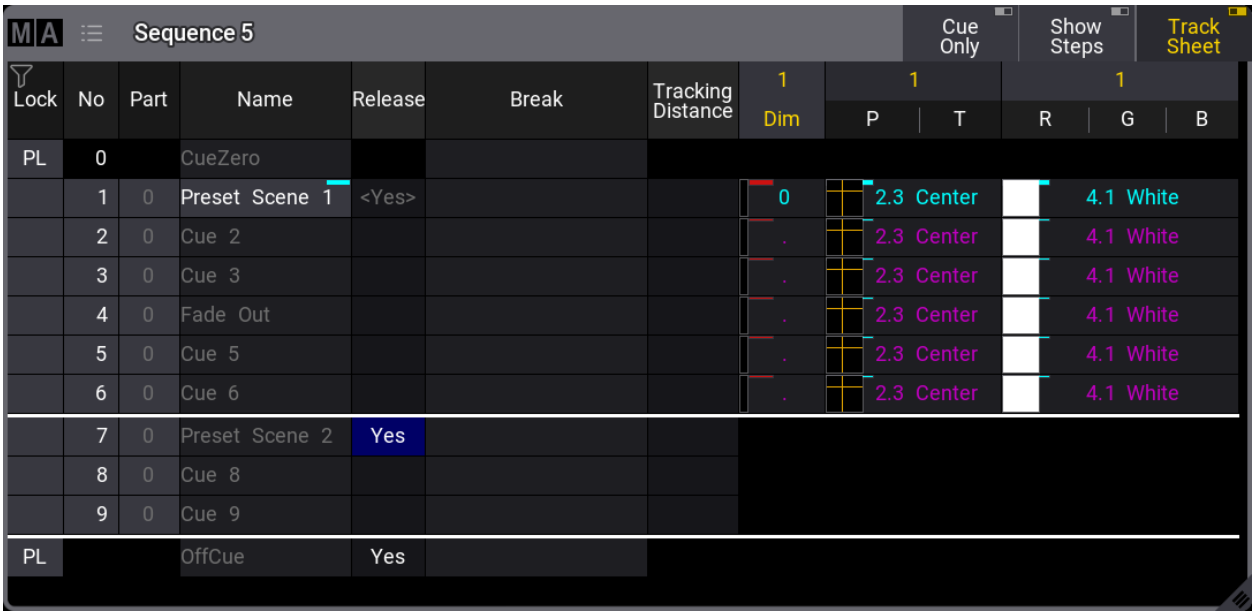
Filter pool with filter objects

The filters are one filter that contains all attributes and layers and one that only contains the dimmer attributes and all layers.

In this scenario, it is known that cue number 4 is a fade out to black, and cue number 7 is used to pre-set fixtures for the second scene in the show.

The break function can protect these two cues from tracking value changes.

Before moving any further with the example, let us explore the release function by editing the **Release** cell for cue 7, so it changes to "Yes".

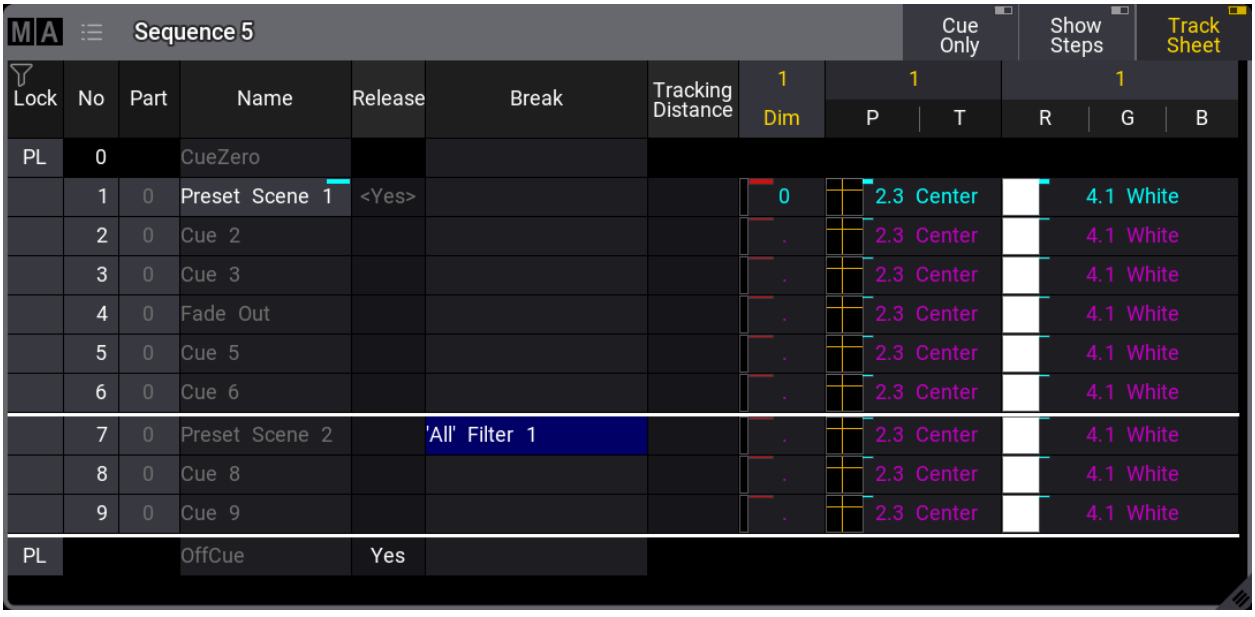


Release is out of the scope of this topic (It is explained more in the **What is Tracking topic**), but it is included here to show that a release stops all future and existing tracking. The tracking values are released from the sequence.

Release is not the goal for cue 7. Remove the "Yes" in the **Release** column again.

Editing the **Break** cell for cue number 7 opens the **Assignment Editor**. Select Filter and then tap All to select it.

Now the sequence looks like this:

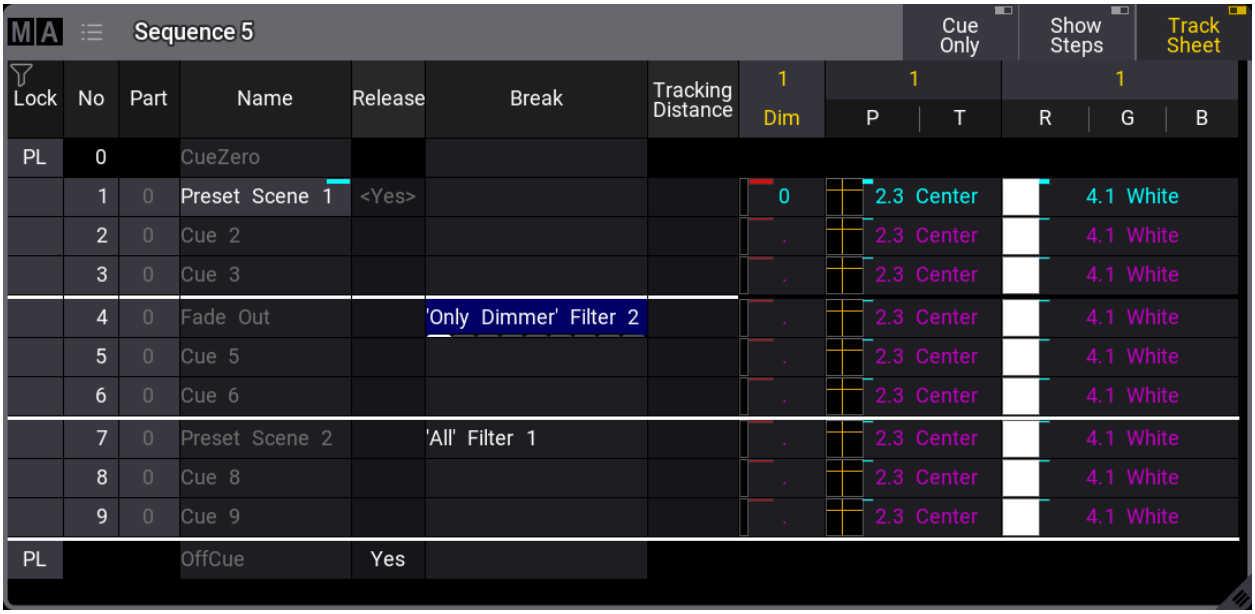


Break is set for cue 7

Notice the difference between this and the release. The existing tracking values are not released but are protected from future changes.

A break adds a white line to indicate the tracking break. If the break affects all attributes - typically by using Filter 1 or world 1 - it will extend across the entire row (just like the release).

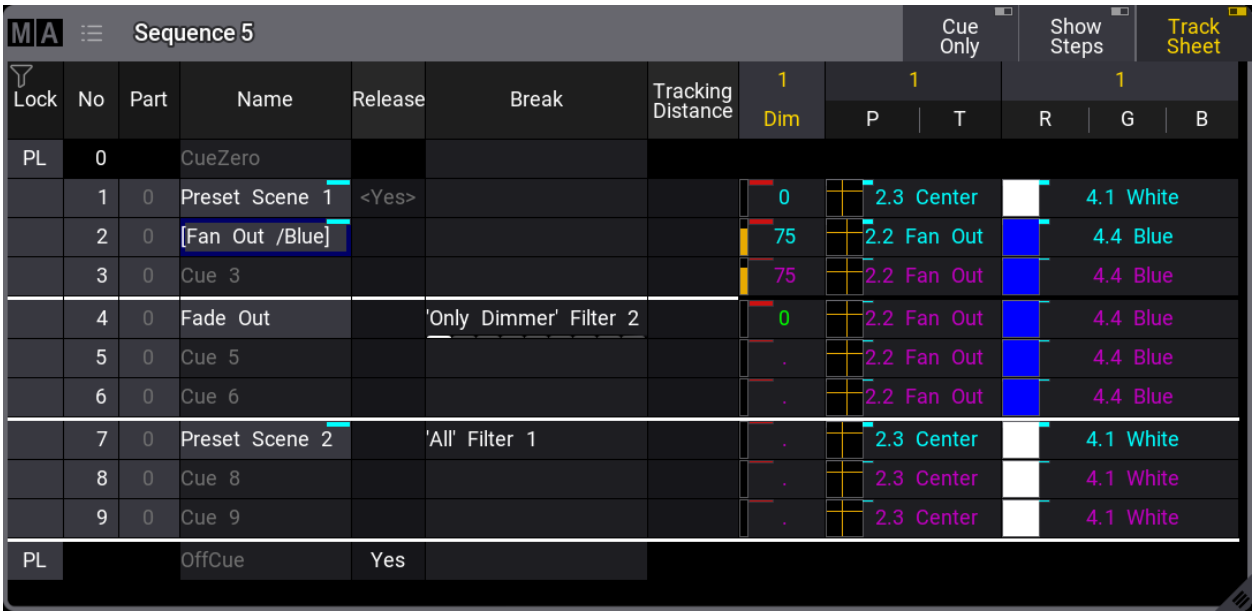
Cue 4 is a fade out. The current dimmer value of 0% can be protected by adding a second break in cue 4, but this time using the "Only Dimmer" filter:



A break is set for cue 4

The break in cue 4 looks a bit different. The white line is black in the attribute columns. This indicates that some attributes are filtered. The Feature Group Indicator Bar is visible at the bottom of the break cell. It shows the features active in the filter.

Now, all the breaks are set, and the values can be changed. A new dimmer, position, and color values for fixture 1 are stored in cue number 2.



Result

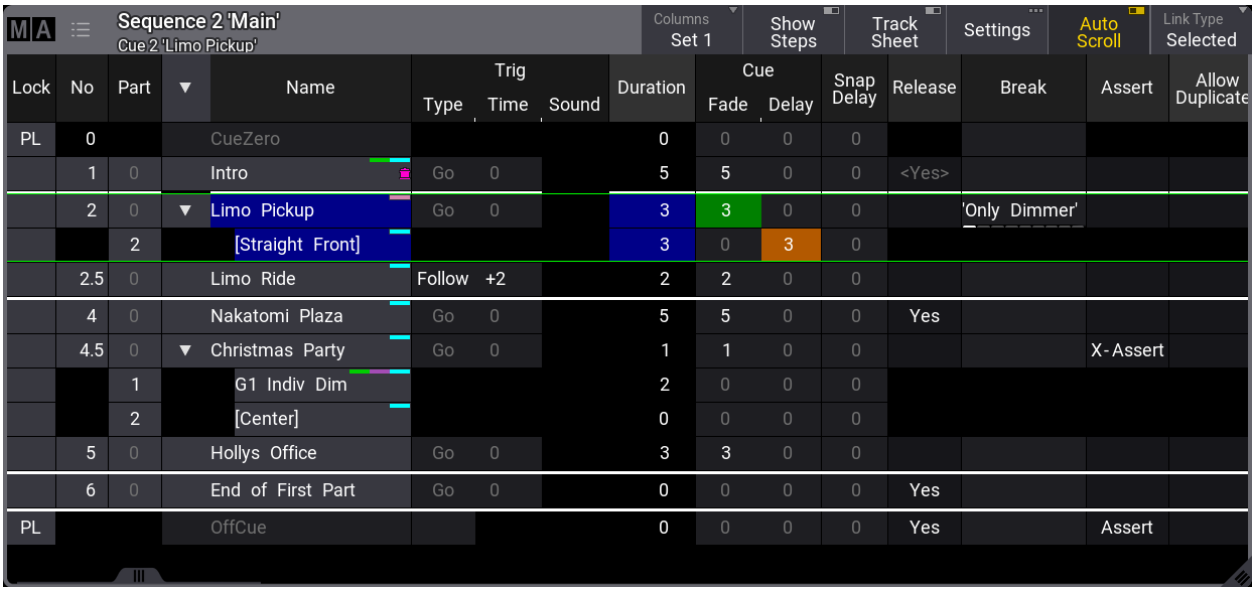
The result is that the dimmer, and only the dimmer, value is returned to 0% (the previous tracked value) in cue number 4 because the filter in the break stopped the newly added values from tracking past the cue.

All the values in cue 7 were protected from changes by the filter in the break, and the old tracking values are added (if needed) in cue number 7.

1.30.2. Sequence Sheet

The different sequences are best seen in the **Sequence Pool**.

The best way to see the cues inside a sequence is the **Sequence Sheet**:



Sequence sheet for a sequence called "Main" - Cue 2 is active

The sheet shows the cues and cue parts in rows. The different cue settings are in columns. Read below for a description of the different columns.

The purpose of this sheet is to see the cues in a sequence. It also shows the active cue with a green frame. Cue fades, and delays are visualized with moving bars while the fades are running.

The sheet can be created as a window in a view using the **Add Window pop-up**, or displayed as a temporary pop-up by editing a sequence pool object.

The sheet can show a lot of different markers and colors - read more about all these in the **Markers topic**.

The sheet can display a value tracking section. In this section, all the stored attribute values are displayed. This can be a very handy tool for seeing the flow of an attribute through the cues. These values can also be edited directly in the sheet.

Most other fields in the sheet can be edited directly. This can affect the look of the show. For instance, the cue fade and delay times are stored when the cue is created. The default timing is used if nothing else is defined. The cue timing can be edited in this sheet.

Tapping this sheet to give it focus changes the encoder toolbar to make it easy to edit the cue timings. Read more about the toolbar **below.**

When a trigger type is set for the OffCue of a sequence, Wrap Around will be disabled. Enabling Wrap Around for a sequence removes the trigger type for the OffCue.

Title Bar

The left side of the title bar has the MA logo. Tap this to open the settings for the sheet. Read about them **below**.

Next to the logo are the sequence number and name. Information about the active world (if different than world 1) is displayed below the Sequence number and name. Next to the world information is the number and name of the currently active cue.

Some buttons can be on the right side of the title bar. This can be defined by editing the title bar in the window settings. Learn more in the **Title Bar Configuration topic**.

Main Part of the Sheet

The main part of the sheet is below the title bar. Here is the sheet with rows and columns.

A sequence sheet setting defines how the cue timing is displayed in the sheet. The option is called **Condensed Timing**. The cue fade and delay are actually four different times because there are both fade and delay for values fading in (or up) and for dimmer values fading down. The condensed view shows this in two columns where the time can be separated by a slash (/). The value on the left is the InFade. the value on the right is the OutFade. The list below shows the uncondensed timing columns. Read more in the **Cue Timing topic**.

The order of the columns and which columns are visible can be changed and stored in **Columns** sets. It can also be edited dynamically. Learn more in the **Adjustable Columns topic**.

This is a short description of each of the possible columns in the sheet:

- **Lock**:
  Editing this cell adds or removes "UL". This means **User Locked** and protects the cue content and settings from being changed.
- **No**:
  This is the cue number.
- **Part**:
  This shows the cue part number.
- **Name**:
  This is the cue (part) name. If the cue contains part cues, an arrow allows them to fold and unfold the parts. In the example above, there are cue parts in cues 2 and 4.5. It is unfolded so the parts can be seen.
- **Trig Type**:
  There are five different trigger types. Editing this cell opens a small select pop-up with the five different options:
  - **Go**:
    The cue needs a valid command (Go+, Goto, Go-, >>>, <<<, Kill) to be triggered.
  - **Time**:
    The cue is triggered a set time after the previous cue is triggered. The time is set in the **Trig Time** column.

    - **Follow**:
      A follow cue is triggered when the previous cue is completely done with the cue transition (which includes all individual timing).
    - **Sound**:
      This will trigger the cue using a sound as the trigger. Choosing one of 22 different frequency areas in the Trig Sound column is possible. Learn more about sound input in the **Sound Window topic**.
    - **BPM**:
      This will trigger the cue using the beats in the sound input. This can become useful with several cues triggered by the BPM (beats per minute).
- **Trig Time**:
  The values stored here are only used if the trigger is **Time**.
  If the trigger is **Time**, then the time in the cell will be used. The time starts counting down when the previous cue is triggered. If the previous cue uses a "Follow" trigger, then the cue's countdown is started when the follow cue is triggered.
- **Trig Sound**:
  This setting defines the sound used to trigger the cue when the **Trig Type** is sound.
- **Duration**:
  This is the overall cue time transition time. It is a combination of the longest fade time and any delays. This is the time used with the Follow trigger. The cell cannot be edited. It always shows the complete transition time.
- **CueIn Fade**:
  This is the fade time for all non-snap attributes and dimmer values that increase in value.
- **CueIn Delay**:
  This is the delay or wait time between the trigger and the actual cue in fade begins.
- **CueOut Fade**:
  This is the fade time for dimmer values that go down in value. The default values for this are the same as the CueIn Fade time - it is linked to the cue in fade with the **None** value.
- **CueOut Delay**:
  This is the delay for the **Out Fade** (only dimmer values). Its default value is the same as the CueIn Delay value.
- **Snap Delay**:
  This can be used to control when "snap" attributes change values.
- **Release**:
  Changing the value to **Yes** in this cell makes the cue release tracked values. Learn more in the **What is Tracking topic**.
- **Break**:
  A break blocks new values of attributes being tracked. Editing the cell opens an **Assignment Editor** pop-up where a filter or world can be selected. The selected filter or world defines what is blocked by the break. Learn more in the **Break topic**.
- **Assert:**
  Assert can be used to make tracked values take precedence as if they are stored values in the cue. There are three options:
    - **None**:
      Tracked values are not asserted
    - **Assert**:
      The tracked values take precedence as if they are stored values in the cue. Values are asserted using the timing from the cue where the values are originally stored.

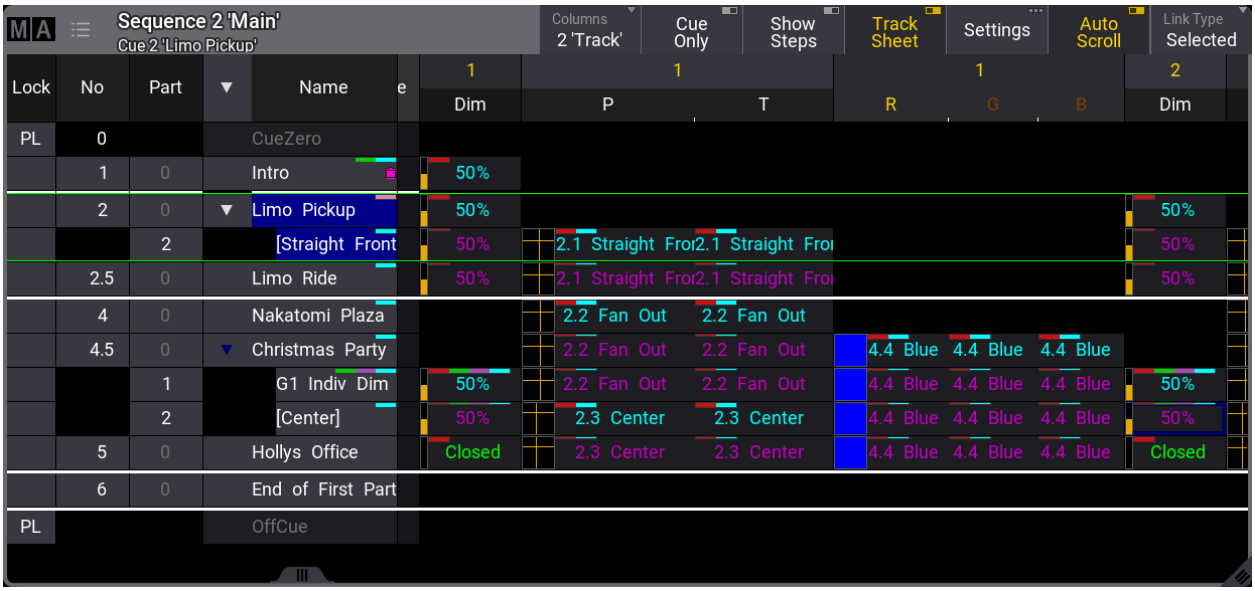
    - **X-Assert**:
      Like assert, but the tracked values are asserted using the timing from the cue where the X-Assert is set.
- **Allow Duplicates**:
  If several parts of the same cue are to contain values for the same attributes, use **Allow Duplicates** to enable this function. Absolute and relative values in multiple parts will use the value with the highest cue part number.
- **Tracking Distance**:
  The tracking distance sets how many cues a value should track. If the cell is empty, it tracks until changed. Read more about tracking distance in the **What is Tracking topic**.
- **Sync**:
  Synchronizes the fixtures of the phaser. For example, if fixtures join the already running phaser, they will be synchronized with the fixtures already running. Learn more about sync in the **Phaser topic**.
- **Morph**:
  If the property is enabled and the phase of the fixtures changes from one cue to the next, they will stay on track and morph into their new phase value. If the option is disabled, the fixtures will take the direct way to their new position in the phase.
- **Transition**:
  This can be used to select a transition path for the fade. The different options are described in the **Cue Timing topic**.
- **"Preset type" Fade**:
  Each preset type has columns called the name of the preset type followed by "Fade". This uses the cue in fade as a default. It can be changed to give all values in the preset type a different fade time. All fixtures with new values in this preset type will use this timing for the attributes in the preset type.
- **"Preset type" Delay**:
  Each preset type has columns called the name of the preset type followed by "Delay". This uses the cue in delay as a default. It can be changed to give all values in the preset type a different delay time. All fixtures with new values in this preset type will use this timing.
- **Command**:
  Commands (like the ones written in the command line input) can be written in the cues. When the cue is triggered, they are executed on the GlobalMaster, IdleMaster, or Standalone station if Command Enabled allows it. Editing the cell opens the **Command Editor**. Learn more about this editor in the **Create Macros topic**.
- **Command Enabled**:
  This can be toggled **Yes** or **No** if there is a command. If exclamation marks surround the option, then the **sequence settings** overwrite the setting. The overwritten option can be seen in the header of the column.
- **Command Delay**:
  This will add a delay between the triggering of the cue and the execution of the command. See **Command** just above.
- **Note**:
  This is a multiline text field where a note can be added to the cue. Learn more in the **Notes topic**.
- **MIB Preference**:
  This defines whether the cue is good for the MIB function. Read more in the **Move In Black topic**.

- **MIB Mode**:
  This sets the MIB mode. This can only be edited if MIB is possible. Read more in the **Move In Black topic**.
- **MIB Target**:
  Defines a cue where the MIB should be performed if possible. This can only be edited if MIB is possible. Read more in the **Move In Black topic**.
- **MIB MultiStep**:
  Defines what should happen with phasers in the MIB. This can only be edited if MIB is possible. Read more in the **Move In Black topic**.
- **MIB Fade**:
  Sets the MIb fade time. This can only be edited if MIB is possible. Read more in the **Move In Black topic**.
- **MIB Delay**:
  Sets the MIB delay time. This can only be edited if MIB is possible. Read more in the **Move In Black topic**.
- **Indiv Fade**:
  This is "Individual Fade". It displays the time for attributes that have individual stored fade times. This cell cannot be edited.
- **Indiv Delay**:
  This is "Individual Delay". It displays the time for attributes that have individual stored delay times. This cell cannot be edited.
- **Indiv Duration**:
  This is "Individual Duration". It displays the overall time for attributes that have individual stored fade and delay times. This cell cannot be edited.
- **Speed Master**:
  A Speed Master can be assigned to the cue or cue part by editing this cell. The speed of the cue or cue part is only controlled by the assigned speed master. A speed master assigned to the sequence has a lower priority and does not influence a cue or cue part with a different assigned speed master.
- **Speed Scale**:
  This can be used to scale the speed of the cue or cue part. A speed scale assigned to a cue or cue part has a higher priority than a speed scale assigned to the sequence.
- **Appearance**:
  An Appearance can be assigned to the cue or cue part. It is connected with a sequence sheet setting called **CuePart Appearance**, which defines how the appearance is displayed. Read more **below**.

All cells with a light or dark gray background color can be edited, and the field's value can be changed. Fields with a black background cannot be edited.

Track Sheet Mode

The sequence sheet can be in **Track Sheet** mode. This can be changed in the window settings - and the setting can be a button in the title bar.



Sequence Sheet in Track Sheet mode

This mode can use a different selection in the **Columns** setting in the sheet settings. The example image above also has the **Columns** setting in the title bar.

The track mode shows attribute data in columns on the right side of the sheet.

The attribute values and markers have different colors, indicating different statuses, like the tracking status. Read about the colors and markers in the **Colors topics**. Read about tracking in the **What is Tracking topic**.

The values can be edited in the sheet. The Cue Only setting can be activated in the settings, and it can be a button in the title bar that appears when **Track Sheet** is On. This makes edited values follow the **cue only rules**.

When cues with multiple parts are expanded to show all the parts, it is easy to see exactly what parts have stored which values. When the cues are collapsed to show only one row, the values from the parts are shown in this row with a small text in the lower right corner telling what part the values come from.

Collapsed cue with multiple parts

This example is the same cue 2 and 4.5 as the image above. The only difference is that the cue is collapsed to show only one row. Notice the small text showing that the pan and tilt values are from part 2 and in cue 4.5 the dimmer is from part 1.

The **Layer Toolbar** can be turned On in the settings. This can be useful when there is a desire to edit or look at values in other layers.

Editing a value opens the Calculator, where a new value can be selected from Presets, Channel Sets, Specials, or simply typed.

The **calculator** has some special functions in the track sheet that define what is changed, and there are special buttons that give access to block, unblock, and extract presets.

Extract presets will remove the link to a preset and store the current preset values directly in the cue.

Sequence Sheet Settings

The sheets have a lot of settings. They are accessed by tapping the MA logo in the upper left corner of the window.

Sequence Sheet Settings - Display tab

Some of them are general settings that are shared with other windows. Read about them in the **Window Settings topic**.

The settings have three tabs: **Display**, **Mask**, and **Columns**.

This is a short description of the display settings in the sequence sheet.

- **Layer**:
  It selects which layer is displayed in the window. It is a swipe button that opens a list of the layers. A special property is **Auto**. This property makes the window follow the selected layer in the **Encoder Bar**.
- **Cue Only**:
  It defines if the cue only function is On/Off when editing values.
- **Fixture Sort**:
  This On/Off button activates the sorting of fixtures. The fixtures are sorted in the selection order to the top or left hand side of the sheet showing the fixtures.

- **Appearance**:
  Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.
- **Step**:
  It selects which step to display. Steps are used with **Phasers**. It is a property input button that opens a calculator pop-up.
- **Settings**:
  This is only visible when editing the Title Bar. This setting determines whether there is a Settings button in the title bar or if it is hidden.
  The actual Settings button can be tapped to open the settings for the object. For instance, tapping the Settings button in the title bar of the sequence sheet opens the settings for the sequence.
- **Feature Sort**:
  This On/Off button activates feature sorting. The selected feature is moved before the other features in the sheets showing features.
- **Font Size**:
  This selects the font size in the window. It is a swipe button that opens a list of sizes from 10 to 32. There is also a **Default** property. The default is the same as size 18.
- **Readout**:
  This selects the value readout for fixture attributes. It is a swipe button that opens a list of readout types with the following options:
  - **Auto**:
    This makes the sheet follow the selected readout in the **Encoder Bar**.
  - **Natural**:
    Each attribute has a defined Natural readout. This is defined in the **Attribute Definition**. Selecting this option will show the different readouts defined for the attributes.
  - **Percent**:
    This is a range from 0 to 100.
  - **PercentFine**:
    This is a range from 0.00 to 100.00.
  - **Physical**:
    This uses the physical range defined in the fixture type definition.
  - **Decimal8**:
    This is a decimal range from 0 to 255.
  - **Decimal16**:
    This is a decimal range from 0 to 65 535.
  - **Decimal24**:
    This is a decimal range from 0 to 16 777 215.
  - **Hex8**:
    This is a hexadecimal range from 00 to FF.
  - **Hex16**:
    This is a hexadecimal range from 0000 to FFFF.
  - **Hex24**:
    This is a hexadecimal range from 000000 to FFFFFF.
- **#Columns**:
  This input button sets the number of columns a sheet should display (the settings **Transpose** and **Adjust Columns** must be switched On except in the **DMX Sheet**).

The DMX Sheet shows all the DMX channels and their output values. Learn more in the **DMX Sheet topic**.

- **ChannelSet**:
  This setting defines the readout of values that are part of channel sets. It has three options:
  - **Value**:
    Displays only the value.
  - **Value + Name**:
    Displays the value and channel set name.
  - **Name**:
    Displays only the channel set name.
- **Layer Toolbar**:
  This On/Off button shows or hides a **layer toolbar** at the bottom with the different Layers.
- **Condensed Timing**:
  This toggles if the cue timing columns are displayed condensed or if all four cue timing columns are visible.
- **Adjust Columns**:
  This On/Off button makes a sheet adjust the column width to match the window size and the number of columns.
- **Time Format**:
  This defines the time format for the windows. This can be used to select a different format than the default set in the **user profile**.
- **Countdown**:
  A cue timing countdown can be displayed while the fade is running. This setting has three options:
  - **Off**:
    There is no countdown in any of the timing columns. They always display the set times.
  - **Duration**:
    The duration column displays a countdown while the cue transition is running.
  - **All**:
    The duration and cue timing columns display a countdown while the cue transition is running.
- **Preset**:
  This defines how the preset information is displayed in the sheets. There are six properties which are different combinations of these three elements:
  - **ID**:
    Shows the ID number of the preset.
  - **Name**:
    Shows the name of the preset.
  - **Value**:
    Shows the values stored in the preset.
- **Transpose**:
  This On/Off button flips the columns and rows in windows.
- **Frame Readout**:
  This defines the frame readout for this window. It can be used to overwrite the default set in the **user profile**.
- **CuePart Appearance**:
  This defines how the cue part appearance is displayed in the sheet. The options are:

    - **Off**:
      Cue part appearance is not displayed.
    - **Number**:
      The appearance is only shown on the cue number column.
    - **Num+Name**:
      The appearance is displayed in the number and name columns.
    - **All**:
      The appearance is displayed on all columns.
- **Merge Cells**:
  Cells can be merged to show a value only once if the adjacent cell has the same value and belongs to the same feature or feature group. For instance, if all red, green, and blue values are "100", then "100" are only shown once.
    - **None**:
      Cells are not merged.
    - **Feature**:
      The values of a feature are merged to only be shown once if the two or more adjacent values are the same.
    - **Feature Group**:
      The values of a feature group are merged to only be shown once if the two or more adjacent values are the same.
- **Auto Scroll**:
  This On/Off button activates the auto-scrolling function. This will keep the active object visible in the window by scrolling the sheet or grid.
- **Link Type**:
  This setting defines which sequence is shown in the sheet.
  There are three different link types. The options are:
    - **Fixed**:
      The sheet displays the information from a specific sequence. The selection is made in the Sheet Settings. Read about the **Fixed Target** setting above. It can also be set using the **Assign** and **Sequence** keywords and tapping the sheet's title bar.
    - **Selected**:
      The sheet displays information from the selected sequence.
    - **LastGo**:
      This automatically shows the latest sequence to receive one of the trigger commands (<<<, >>>, Go+, Go-, Goto, Load, On, Select, Top, Temp, Flash, Toggle On, Pause). This includes if the sequence is triggered from a running timecode recording. A sequence can be excluded from LastGo by turning Off the **Include Link Last Go** setting in the **Sequence Settings**. LastGo only shows sequences triggered by the same user profile.
- **Fixed Target**:
  This setting defines the sequence a sheet displays if the **Link Type** is **Fixed**. Tapping this setting opens an **Assignment Editor** pop-up where a sequence can be selected.

The mask settings for the sequence sheets are:

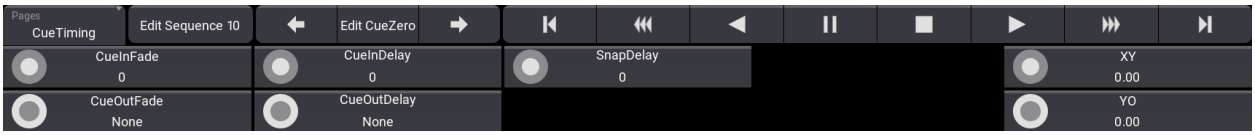
- **Color Mode**:
  This switches the color readout between Auto (following the User Profile setting), RGB and CMY. The default value is to follow the setting in the **User Profile**. The user profile setting is shown between "<>".
- **Feature Graphic**:
  Shows or hides a small graphic next to each feature in the sheets showing the features.

- **Filter**:
  An existing filter can be chosen to filter the content of the window.
- **Filter Toolbar**:
  This setting shows or hides a filter toolbar that can be used to filter the content of the sheet.
- **Line Height**:
  This defines the line height for the rows in the sheet. Valid options range from 1 to 12. There is also an **Auto** option, which selects the smallest height possible to show the content.
- **Selection Only**:
  This On/Off button is a mask function that hides fixtures not currently selected in the programmer. This is valid when the **Track Sheet** mode is On.
- **Show Command Test**:
  This makes it possible to test a command stored in a cue without activating the cue. Toggling this setting On will display a play icon (▶) in the command cells that contain commands. Tapping the play icon executes the command of the cue.
- **Show Notes**:
  This On/Off button shows or hides the notes for the selected cue at the bottom of the sheet. Read more about it in the **Notes topic**.
- **Show Recipes**:
  This On/Off button shows or hides the cue recipes at the bottom of the sheet. Read more about it in the **Cue Recipes topic**.
- **Show Steps**:
  This On/Off button shows or hides the cue steps in the sheet. This is useful in a sequence sheet with **Track Sheet** On.
- **Track Sheet:**
  This On/Off button shows or hides the values and tracking information for each attribute in the sheet.

Finally, there is a tab called **Columns**. This allows editing of the columns and column sets in the sequence sheet. Learn more about this in the **Adjustable Columns topic**.

Sequence Edit Toolbar

The encoder toolbar changes when the sequence sheet has focus. The sheet can get focus by tapping the sheet.



Sequence Edit Toolbar - Basic Timing page

There are several pages with many different settings for the cues. The pages can be changed using the swipe button in the upper left corner of the toolbar (see the image above).

The top row in the toolbar gives access to select a cue. There are also playback controls that can be used to run cues. Read more in the **Play Back Cues topic**.

Turning the two rings on the encoders changes the respective values for the selected cue in the sheet.

The lower row is the outer ring of the dual encoder. The middle row is the inner ring of the dual encoders.

1.30.3. Content Sheet

The Content Sheet is used to see the fixtures and values stored in cues. It looks much like the **Fixture Sheet** but has a masking function that only displays what is stored in cues and cue parts. It is like a combined **Sequence Sheet** in tracking mode and a **Fixture Sheet**.

The Fixture Sheet is a window that shows all the patched fixtures that have an ID. It has different modes that can use different versions of the attribute values for each fixture.
Learn more in **Fixture Sheet topic**.
The Sequence Sheet shows the cues in a sequence and all the settings related to cue transition. It also has a mode called Track Sheet that shows the attributes values in the cues.
Learn more in the **Sequence Sheet topic**.
The Fixture Sheet is a window that shows all the patched fixtures that have an ID. It has different modes that can use different versions of the attribute values for each fixture.
Learn more in **Fixture Sheet topic**.

It could look like this:

Content Sheet

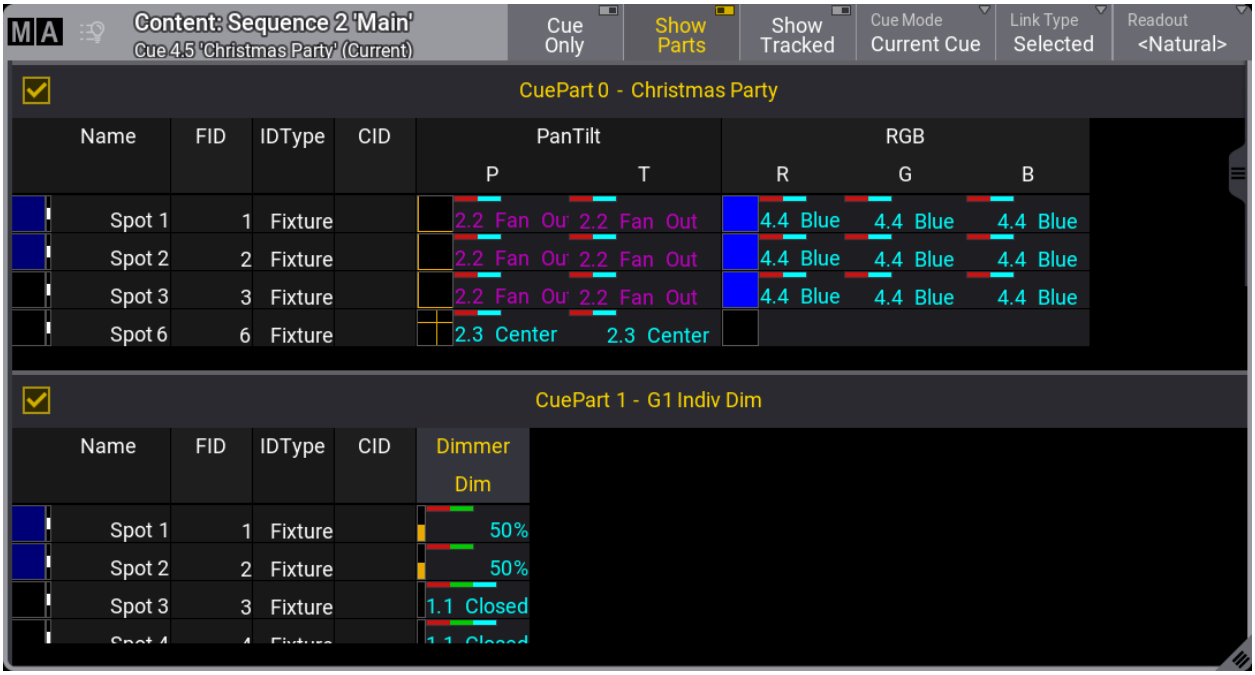
The Title Bar shows the sequence ID and the cue ID of the cue being shown. If a world other than the default Full world is selected, it is also displayed in the title bar (Small World icon with a name and number next to it).

The values displayed in the sheet can be edited directly in the sheet, just like in the Sequence Sheet in **Track sheet mode**.

The look of the content sheet is very customizable. The content sheet settings control this. **Read more about it below**.

The example image above shows most of the pan and tilt values with a small note saying "Part 2". This indicates that the values are stored in cue part 2. A red "DUP" in the lower right corner means duplicate data is stored in a different part of the cue.

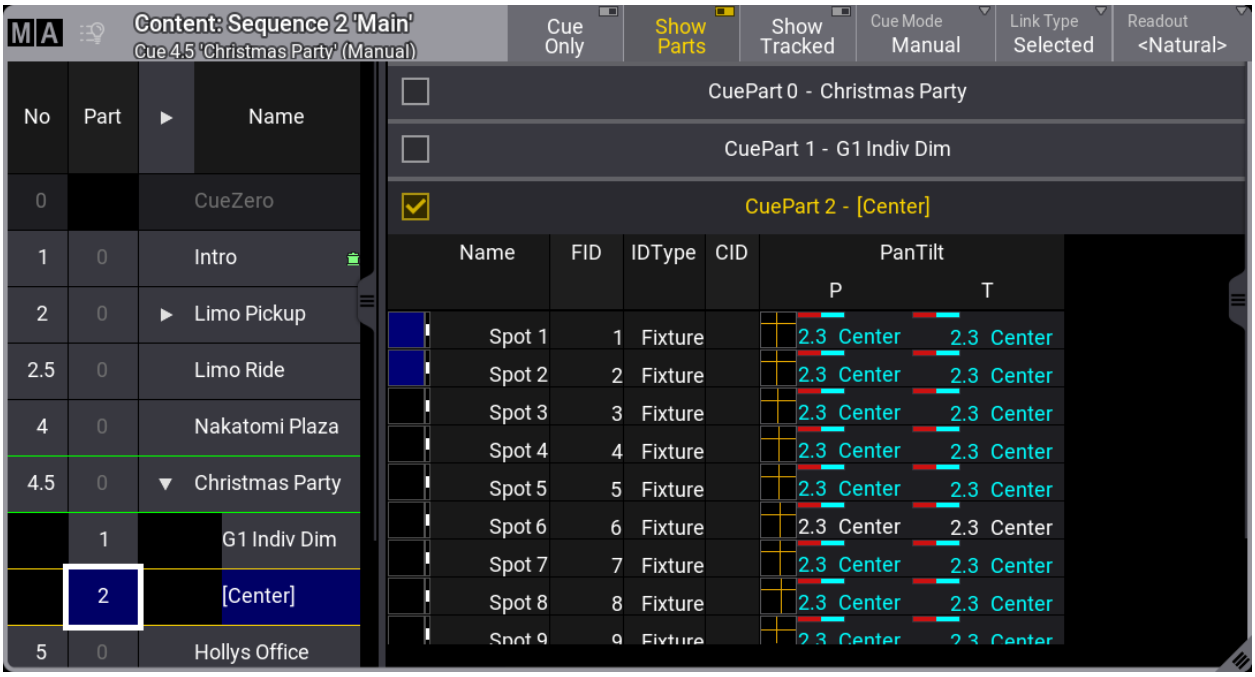
If Show Parts is turned On, the parts are separated into different frames. The same example cue would look like this:



Content Sheet with multiple part frames

Each frame with a cue part can be toggled On or Off by tapping the checkmark in the frame.

The content sheet can automatically follow the active cue being played back. The setting **Cue Mode** defines which cue to show in the sheet. Setting the mode to **Manual** makes it possible to select a specific cue instead of following the active cue. In manual mode, an extra section on the left displays a list of all the cues in the sequence.



Content Sheet in Manual mode

The green frame around a cue indicates the active cue being output. The blue background indicates the selected cue being displayed in the content sheet.

Tapping a cue in the list makes it the selected cue.

Tapping the part number in the cue list automatically deselects the other parts if Show Parts is active.

Content Sheet Settings

The content sheet has a lot of settings. They can be accessed by tapping the MA logo in the upper left corner of the sheet.

The settings are divided into two different tabs - Display and Mask.

This is a short description of each of the display settings:

- **Show Parts**:
  Turning this On separates the cue parts into individual frames. Each frame can be turned On or Off by tapping the upper left corner of each frame. If this setting is Off, the cue parts are combined into one frame showing the cue part number in the lower right corner of each relevant attribute.
- **Cue Only**:
  It defines if the cue only function is On/Off when editing values.
- **Fixture Sort**:
  This On/Off button activates the sorting of fixtures. The fixtures are sorted in the selection order to the top or left hand side of the sheet showing the fixtures.
- **Sheet Mode**:
  The sheet mode changes how the sheets look. There are four different modes:
  - **Fixture**:
    This shows a matrix with the fixtures in rows and the attributes in columns.

    - **Channel**:
      This shows the fixtures as tiles with the dimmer attribute.
    - **Dimmer+**:
      Looks similar to the **Channel** mode. However, it additionally displays the attributes of the selected feature group. Vertical gray separators are displayed when there is a jump in IDs and when the IDType changes for fixtures that do not have a fixture ID. This mode can display the **Fixture Graphics**, but does not display the **Feature Graphics**.
    - **Sheet/Filter**:
      Similar to **Dimmer+**. However, it displays all attributes unless there is a defined filter in the **Mask** tab of the sheet settings.
- **Cue Mode**:
  Cue Mode has three different options:
    - **Current Cue:**
      This will make the sheet display the information related to the current active cue.
    - **Previous Cue:**
      This will display the values from the previous cue. This is the last active cue, even if the sequence order is jumped.
    - **Next Cue:**
      This displays the values for the next cue if a triggering action is performed on the sequence. If a cue is "Loaded", the loaded cue will be displayed.
    - **Manual**:
      This mode allows selecting a specific cue. When this mode is selected, a list of the cues appears on the left side. Tap the desired cue in the list to see the content.

  If the setting is available in the title bar, the different options can be toggled by tapping the button in the title bar.
- **Feature Sort**:
  This On/Off button activates feature sorting. The selected feature is moved before the other features in the sheets showing features.
- **Font Size**:
  This selects the font size in the window. It is a swipe button that opens a list of sizes from 10 to 32. There is also a **Default** property. The default is the same as size 18.
- **Readout**:
  This selects the value readout for fixture attributes. It is a swipe button that opens a list of readout types with the following options:
    - **Auto**:
      This makes the sheet follow the selected readout in the **Encoder Bar**.
    - **Natural**:
      Each attribute has a defined Natural readout. This is defined in the **Attribute Definition**. Selecting this option will show the different readouts defined for the attributes.
    - **Percent**:
      This is a range from 0 to 100.
    - **PercentFine**:
      This is a range from 0.00 to 100.00.
    - **Physical**:
      This uses the physical range defined in the fixture type definition.
    - **Decimal8**:
      This is a decimal range from 0 to 255.

    - **Decimal16**:
      This is a decimal range from 0 to 65 535.
    - **Decimal24**:
      This is a decimal range from 0 to 16 777 215.
    - **Hex8**:
      This is a hexadecimal range from 00 to FF.
    - **Hex16**:
      This is a hexadecimal range from 0000 to FFFF.
    - **Hex24**:
      This is a hexadecimal range from 000000 to FFFFFF.
- **#Columns**:
  This input button sets the number of columns a sheet should display (the settings **Transpose** and **Adjust Columns** must be switched On except in the **DMX Sheet**).

  The DMX Sheet shows all the DMX channels and their output values. Learn more in the **DMX Sheet topic**.

- **Layer Toolbar**:
  This On/Off button shows or hides a **layer toolbar** at the bottom with the different Layers.
- **Fixture Appearance**:
  This defines how the appearance of the fixtures is shown in the sheets. There are three options:
    - **None**:
      The fixture appearance is not shown.
    - **Enabled**:
      The appearance of the fixture type is shown.
    - **Graphic**:
      The appearance is shown with a colored background to match the output.
- **Preset**:
  This defines how the preset information is displayed in the sheets. There are six properties which are different combinations of these three elements:
    - **ID**:
      Shows the ID number of the preset.
    - **Name**:
      Shows the name of the preset.
    - **Value**:
      Shows the values stored in the preset.
- **Color Mode**:
  This switches the color readout between RGB and CMY. The default value is to follow the setting in the **User Profile**. The user profile setting is shown between "<>".
- **Time Format**:
  This defines the time format for the windows. This can be used to select a different format than the default set in the **user profile**.
- **Fixture Graphics**:
  This defines which graphics are displayed in front of the name column in sheets showing the fixture graphic. Resizing the name column to a very small size will hide the graphic.
  This setting has the following options:
    - **None**:
      No graphic is shown.

    - **Flip**:
      Adds the flip indicator for fixtures with position attributes on the left side of the **Name** column.
    - **Simple**:
      Adds a simple square graphic indicating combined color and dimmer values next to the flip indicator in the **Name** column.
    - **Gobo**:
      Adds a gobo image on the simple graphic. It only displays the gobo of one gobo wheel at a time. Gobo wheels in ascending order define which gobo is displayed. For example, when Gobo 1 is set to open, then the gobo of Gobo 2 is displayed.
- **Fixture Select**:
  When this is On, fixtures can be selected by tapping the name or ID in the sheet.
- **Show Tracked**:
  This On/Off setting shows or hides all the values tracked from previous cues. Turning it Off hides the tracked values and only shows the values stored in the cue.
- **Frame Readout**:
  This defines the frame readout for this window. It can be used to overwrite the default set in the **user profile**.
- **Feature Graphic**:
  Shows or hides a small graphic next to each feature in the sheets showing the features.
- **Channel Set**:
  This setting defines the readout of values that are part of channel sets. It has three options:
    - **Value**:
      Displays only the value.
    - **Value + Name**:
      Displays the value and channel set name.
    - **Name**:
      Displays only the channel set name.
- **Merge Cells**:
  Cells can be merged to show a value only once if the adjacent cell has the same value and belongs to the same feature or feature group. For instance, if all red, green, and blue values are "100", then "100" are only shown once.
    - **None**:
      Cells are not merged.
    - **Feature**:
      The values of a feature are merged to only be shown once if the two or more adjacent values are the same.
    - **Feature Group**:
      The values of a feature group are merged to only be shown once if the two or more adjacent values are the same.
- **Link Type**:
  This setting defines which sequence is shown in the sheet.
  There are three different link types. The options are:
    - **Fixed**:
      The sheet displays the information from a specific sequence. The selection is made in the Sheet Settings. Read about the **Fixed Target** setting above. It can also be set using the **Assign** and **Sequence** keywords and tapping the sheet's title bar.
    - **Selected**:
      The sheet displays information from the selected sequence.

    - **LastGo**:
      This automatically shows the latest sequence to receive one of the trigger commands (<<<, >>>, Go+, Go-, Goto, Load, On, Select, Top, Temp, Flash, Toggle On, Pause). This includes if the sequence is triggered from a running timecode recording. A sequence can be excluded from LastGo by turning Off the **Include Link Last Go** setting in the **Sequence Settings**. LastGo only shows sequences triggered by the same user profile.
- **Fixed Target**:
  This setting defines the sequence a sheet displays if the **Link Type** is **Fixed**. Tapping this setting opens an **Assignment Editor** pop-up where a sequence can be selected.

This is a short description of each of the mask settings:

- **Fixture**:
  This On/Off button defines whether fixtures using the Fixture ID Type are shown or hidden in the sheet.
- **Channel**:
  This On/Off button defines whether fixtures using the Channel ID Type are shown or hidden in the sheet.
- **Universal**:
  This On/Off button defines whether fixtures using the Universal ID Type are shown or hidden in the sheet.
- **MArker**:
  This On/Off button defines whether fixtures using the MArker ID Type are shown or hidden in the sheet.
- **Houselights**:
  This On/Off button defines whether fixtures using the Houselights ID Type are shown or hidden in the sheet. This is the default name for this ID type, it can be changed in the **Patch**.
- **NonDim**:
  This On/Off button defines whether fixtures using the NonDim ID Type are shown or hidden in the sheet. This is the default name for this ID type, it can be changed in the **Patch**.
- **Media**:
  This On/Off button defines whether fixtures using the Media ID Type are shown or hidden in the sheet. This is the default name for this ID type, it can be changed in the **Patch**.
- **Fog**:
  This On/Off button defines whether fixtures using the Fog ID Type are shown or hidden in the sheet. This is the default name for this ID type, it can be changed in the **Patch**.
- **Effect**:
  This On/Off button defines whether fixtures using the Effect ID Type are shown or hidden in the sheet. This is the default name for this ID type, it can be changed in the **Patch**.
- **Pyro**:
  This On/Off button defines whether fixtures using the Pyro ID Type are shown or hidden in the sheet. This is the default name for this ID type, it can be changed in the **Patch**.
- **Show Recipes**:
  This On/Off button shows or hides the cue recipes at the bottom of the sheet. Read more about it in the **Cue Recipes topic**.
- **Show Notes**:
  This On/Off button shows or hides the notes for the selected cue at the bottom of the sheet. Read more about it in the **Notes topic**.

- **Show ID Type**:
  It shows the ID Type of the fixtures on the sheet. This setting is only relevant when the **Sheet Mode** is different than **Fixture**.
- **Show Name Field**:
  It shows the name of the fixtures on the sheet. This setting is only relevant when the **Sheet Mode** is different than **Fixture**.

1.30.4. Sequence Settings

Each sequence has a lot of different settings. The settings can be accessed from the title bar of each **sequence sheet** or the **executor assign menu**. Each of these two locations has a button called Edit Settings or Settings. Tap this to open the settings.

Sequence settings pop-up

The settings are divided into different sections. Label, Start, Playback, Speed, Protect, and MIB. Read about the different settings in each section below.

Each user profile has default sequence settings. These settings are used when a new sequence is created. The default can be loaded and stored from this pop-up.

The title bar has two relevant buttons. The left one is Load from Default. Tapping this loads the stored sequence defaults. Next to this is Save as Default. Tapping this stores the current settings as the new default.

Label

This group is about the name and look of the sequence.

Name

Edit this input field to change the name of the sequence.

Scribble

This scribble field can be used to select or create a scribble for the sequence. This scribble is visible in the sequence pool and on the executor label.

Appearance

This appearance field can be used to select or create an appearance for the sequence. This appearance is visible in the sequence pool and on the executor label.

Note

This input field can be used to add a note to the sequence.

Prefer Cue Appearance

When this option is enabled, and the current cue has an appearance, the cue appearance will be displayed on the executor or in the layout instead of the sequence appearance.

Executor Display Mode

The executor display mode defines how the sequence will be displayed on an executor:

- **Data only**:
  Only the cues with their appearances will be displayed. The cue appearance is only displayed in the line of the cue and not in the background.
- **Appearance only**:
  Only the sequence or cue appearance will be displayed. No cue names, fade bar, etc., will be displayed. This can be handy for sequences with only one cue, color cues, or gobo cues.
- **Both**:
  Each cue line displays its cue appearance, and the sequence appearance or the appearance of the current cue will be displayed in the background of the executor.

Start

The settings in this section are about starting and stopping the sequence.

Auto Start

The **Auto Start** feature switches the executor 'On' when the master is moved from zero.

Auto Stop

The **Auto Stop** feature switches the executor 'Off' when the master is moved down to zero.

Master Go Mode

The **Master Go** function is active if auto stop is turned Off. Tapping this button opens the **Select Master Go Mode** pop-up.

There are four options here. They all take effect when the master fader is moved from zero and up. The options are:

- **None**:
  The cue is still running.
- **Go**:
  It executes a Go.

- **On**:
  The current cue is reloaded (fading in again).
- **Top**:
  The first cue is activated.

Auto Fix

The **Auto Fix** feature can be activated for each executor. It will automatically **Fix** active executors and keep them visible even when pages are changed. The executor is automatically unfixed when it is switched off.

Cue Zero Mode

This setting defines what is automatically stored in Cue Zero. There are three options:

- **Off**:
  No attributes are automatically stored in the cue. This is the default option.
- **All Used Attributes**:
  This adds default values for all attributes of the fixtures used in the sequence.
- **Only Used Dimmers**:
  This adds default dimmer attributes of the fixtures used in the sequence.

Auto Stomp

If auto stomp is On, an **absolute** value from a cue will stomp a phaser running from a different playback.

Playback

The playback settings are about running the cues. Running or playing back cues is described in the **Play Back Cues topic**.

Tracking

This turns On or Off value tracking in the sequence. Read more in the **What is Tracking topic**.

Wrap Around

Wrap around allows the sequence to return to the top/first cue if a Go (forward) command is performed after the last cue in the sequence is reached.

Release First Cue

This setting defines if the first cue releases tracking values. These tracking values can come from the last cue if **Wrap Around** is active. If **Release First Cue** is On then it adds a **<Yes>** to the **Release** column in the first cue of the sequence. Learn more about the different columns in the **Look at Cue and Sequences topic**. The first cue can manually be set to release by editing the field in the sequence.

Restart Mode

There are three different restart modes:

- **First Cue**:
  This always restarts the sequence with the first cue.
- **Current Cue**:
  This restarts the sequence with the cue where it was when the sequence was turned Off.
- **Next Cue**:
  This restarts the sequence with the next cue based on where it was when the executor was turned Off.

Cue Command

This option defines how the cue commands can be executed. It can be set to follow the value set for each cue in the sequence's **Command** column, or it can force the execution or disable the execution of the commands in the **Command** column in a sequence. The stored commands are not deleted or removed; they are just disabled.

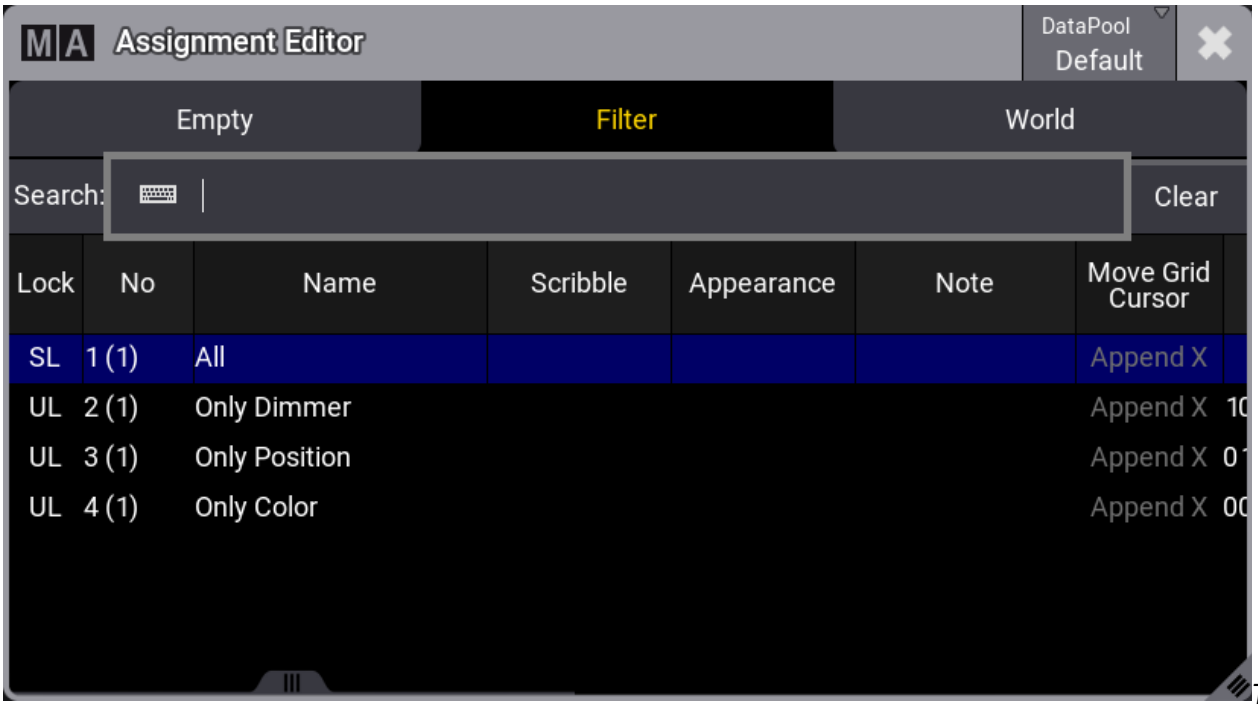
- **Enabled**:
  Each cue command can be enabled or disabled in the **Command Enabled** column. Disabling the command does not delete or remove the command. They are just disabled.
- **Force No**:
  The cue command execution is disabled for the entire sequence. The **Command Enabled** column header will display "Force No" if this option is selected. The user-defined settings in the Command Enabled column are not changed to show "No". It shows the user-set value with exclamation marks before and after.
- **Force Yes**:
  The cue command execution is forced to be enabled for the entire sequence. The **Command Enabled** column header will display "Force Yes" if this option is selected. The user-defined settings in the Command Enabled column are not changed to show "Yes". It shows the user-set value with exclamation marks before and after.

XFade Reload

When this option is enabled, the Xfader needs to be pulled back to 0 after completing a crossfade to do the next crossfade.

Output Filter

The output filter can have a **Filter or World** assigned. Tapping the button opens the **Assignment Editor** pop-up:



Tap a world or filter in the list to apply an output filter

The editor has three tabs. **Empty** can be used to select no filtering. **Filter** and **World** are lists of each type. Each lists the possible choices of their type. Tap the desired filter or world to apply it.

The fixtures and attributes in the world or filter are allowed to pass the output filter and be output from the sequence. The same sequence can be played back from several executors, and each executor shares the output filter settings. If different output filters are needed, the sequence should be linked or shared with another. The other sequence can have different output filter settings. Read about linked sequences in the **Cues and Sequences topic**.

If the sequence has an output filter applied, there is a small output filter icon ( ) in the sequence pool button.

Priority

This is the priority of the sequence. The priorities are described in the **Play Back Cue topic**.

Soft LTP

The Soft LTP function is described in the **Play Back Cue topic**.

Playback Master

Here it is possible to select a **Playback Master**. It functions as a sub-master that multiple sequences can share.

XFade Mode

This is used to set how the two CrossfadeA/XFadeA and CrossfadeB/XFadeB faders work. There are two crossfade modes:

- **Split**:
  The dual crossfaders work as masters for the current/next cue.
- **AB**:
  The dual crossfaders work as crossfaders for increasing/decreasing values.

Speed

Speed and rate can be used to adjust the stored times without reprogramming the show. For instance, a cue can have a fade time of 5 seconds. A rate master can adjust the fade time live while playing it back. All sequences have their own individual rate and speed, but they can be linked to a global master. This master can then adjust the timing for multiple sequences simultaneously. Read more details in the **Speed Masters topic**.

Rate Master

The sequence has a rate master. It can be linked to a shared global speed master, or it can have an individual rate master. Tapping this button will open the **Select Rate Master pop-up**.

In the pop-up, it is possible to select the **None** option for having an individual rate master for the sequence or select one of the global speed masters.

Rate Scale

Enabling this binds the rate to defined steps instead of a variable value. Tapping this button opens the **Select Rate Scale pop-up**.

Choosing one of the steps in the pop-up, selects the multiplier or divider. This multiplies or divides the rate by the selected factor.

Speed Master

The sequence has a speed master. It can be linked to a shared global speed master, or it can have an individual speed master. Tapping this button will open the **Select Speed Master pop-up**.

In this pop-up, it is possible to select the **None** option for having an individual speed master for the sequence or select one of the global speed masters.

Speed Scale

If a sequence is assigned to a global speed master (read above), it can be useful to adjust a speed scale. Tapping this button will open the **Select Speed Scale pop-up**.

In the pop-up, selecting one of the multipliers or dividers is possible. This multiplies or divides the speed by the selected factor.

Speed from Rate

This links the speed to follow the rate.

Protect

This group of settings is used to protect the sequence from different actions.

Input Filter

The playback filter is described in a little bit more detail in the **Cues and Sequences topic**. Tapping this button opens the **Assignment Editor pop-up**: See above about the output filter.

The fixtures and attributes in the world or filter are allowed to pass the filter and can be stored in the sequence.

Swap Protect

Activating this option protects this sequence from the Swap playback action. Learn more about this action in the **Swap Keyword topic**.

Kill Protect

Activating this option protects this sequence from the Kill playback action. Learn more about this action in the **Kill Keyword topic**.

Include Link Last Go

This setting is On as default. When it is set to Off, playing back a sequence will not trigger the LinkLastGo functionality in the **sequence sheet**.

Use Executor Time

This makes the executor playback cues using the stored timing. If this is turned on, it is affected by the **Exec Time** master fader, who overwrites the timing.

Off when Overridden

The Off when Overridden function allows a sequence to be automatically turned Off if another sequence has taken control with all the attributes in the sequence = this executor does not control any attributes.

Lock Sequence

The sequence is locked against changes when this is On. It can still be played back.

MIB

This group is about MIB settings for the sequence. Read more about MIB in the **Move In Black topic**.

MIB

Enable, disable, or force MIB for the sequence. The options are:

- **Enabled**:
  MIB will be performed according to the cue and cue part MIB settings.
- **Never**:
  MIB will never be performed for this sequence. All cue and cue part-specific MIB settings will be ignored.

- **Force Early**:
  Forces the early MIB for all cues that can perform MIB as soon as the dimmer is closed. Further MIB settings specified per cue or cue part will be ignored.
- **Force UponGo**:
  MIB is forced to be executed with the next cue transition after the dimmer is closed. For all cues that can perform MIB. Further MIB settings specified per cue or cue part will be ignored.
- **Force Late**:
  Forces the MIB latest in the cue before the dimmer opens again. For all cues that can perform MIB. Further MIB settings specified per cue or cue part will be ignored.

MIB Mode

This setting defines which MIB mode will be executed when doing a MIB. The **MIB Mode** per cue or cue part needs to be set to **Default**. The options are:

- **None**
- **Early**
- **UponGo**
- **Late**

Learn about the mode types in the MIB topic (link above).

1.30.5. Store Cues

Storing a cue is the default **Store** action in grandMA3.

This means that if nothing else is defined and Please is pressed right after Store, a new cue is stored in the selected sequence.

Or press Store, followed by pressing the executor button where the sequence, with the cue, should be stored.

If active values are in the programmer, they are stored in the cue (with the default store settings), but programmer values are not needed to store cues.

Of course, there are more details about storing cues, so please keep reading.

Store a New Cue on an Empty Executor

If a cue is stored on an empty executor, the grandMA3 software automatically stores the cue in a new sequence and assigns this executor to control the sequence using the default settings.

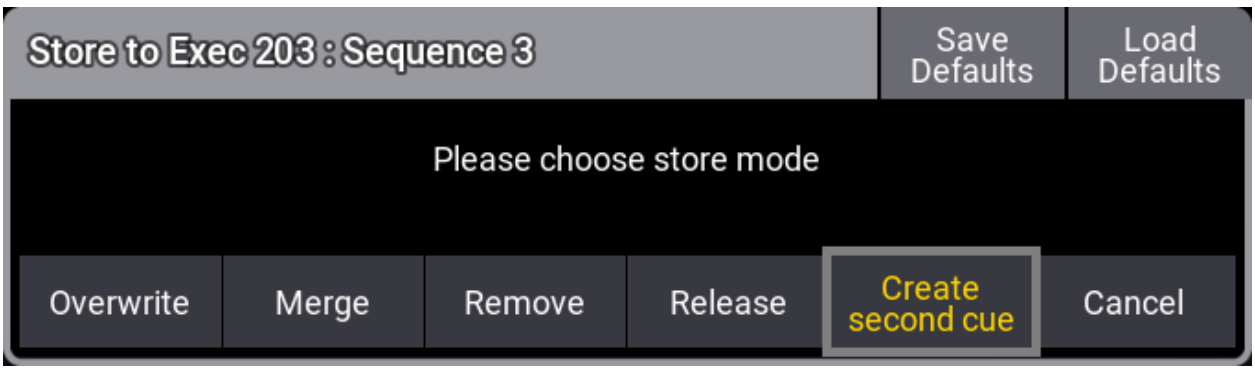
Returning to the second example at the top of this topic, just pressing the Store button and then an executor button on the empty executor is enough. The software assumes the desired action is storing a cue.

This will automatically be cue number 1 - nothing else was specified.

Storing cues obey **Worlds and Filters**, which enables control of what is stored. Worlds and filters can also be assigned to the sequence both as an input and also as an output filter - independently of each other. This will function as an input or output filter, allowing only the elements in the world or filter to be stored in or played back from the sequence.

Store the Second Cue

If the store function is used again on the same sequence - without adding cue number details - then the grandMA3 does not know what should happen, and a pop-up appears, giving different choices.



Please choose store mode pop-up with "Create second cue" option

Tapping Create Second Cue will store a cue with the next whole number.

Overwrite and **Merge** options are explained below. **Remove** and **Release** are described in the **What is Tracking topic**. **Cancel** does not store anything.

Cue Numbers

When a cue is stored, it is possible to specify a cue number. This is done using the following syntax: **Store Cue [Cue_Number]**. It is also possible to specify a sequence or an executor using the keys in the command section while storing:
Store Cue [Cue_Number] Sequence ["Sequence_Name" or Sequence_Number] or
Store Cue [Cue_Number] Executor [Executor_Number].

Notice that the cues are stored in the sequence. Using the executor number will store the cue in the sequence the executor is controlling.

Cue numbers have three decimal numbers. If all are zero, then they are not displayed. But cue number "42" is the same as cue number "42.000" - it is not "42 thousand", it is "42 point 0 0 0". The currently highest cue number that can be stored is "999 999.999". The lowest number that can be stored is "0.001".

	Restriction:
	Storing nearly 1 billion cues will completely fill the memory and make the show file VERY big. The software will cancel the store process before the system crashes, but almost any operation after this will make the software shut down!!

The software hides the trailing decimal zeros, but they are still there. This means that cue "5.2" is after cue "5.11" because they are actually cues "5.200" and "5.110".

It is not limited to storing only a single cue number at a time. It can just as easily be a range of numbers - this means it is possible to use Thru, +, and - keys to create number ranges to store.

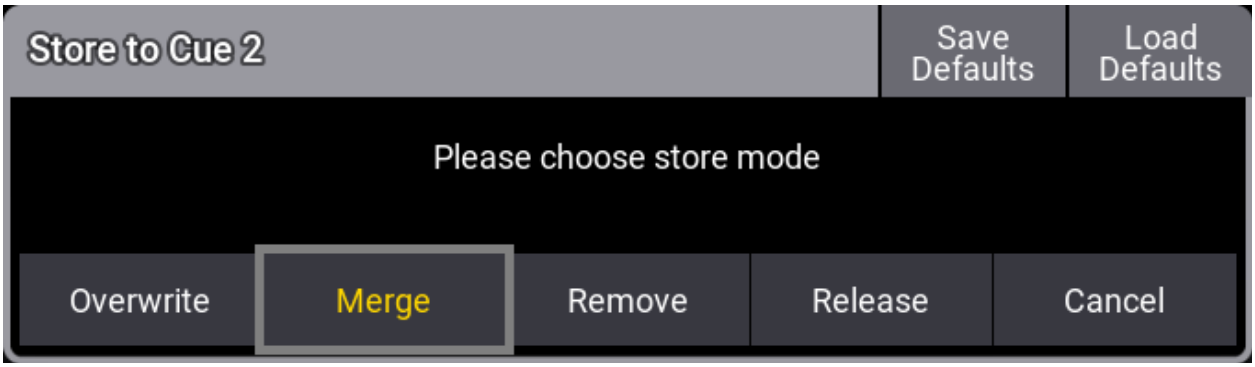
Selecting the sequence can be a good idea if you are working with or adding a lot of cues to the same sequence.

This can be done easily by pressing Select and then one of the buttons associated with the executor controlling the sequence or tapping the sequence in the sequence pool.

The selected sequence is used if no sequence or executor is defined in the store command.

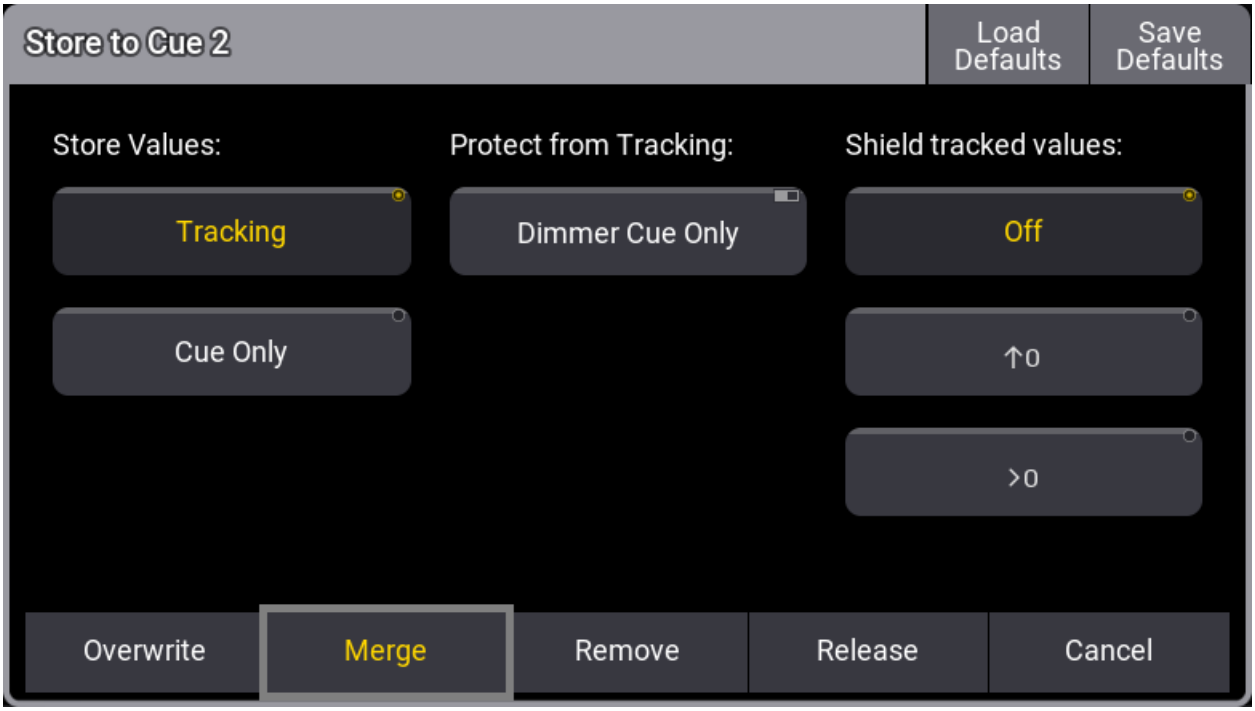
Store Into Cues That are Not Empty

If the store operation is used to store into already existing cues, then a pop-up like this appears:



Please choose store mode pop-up

or if there are cues following the cue stored into, it looks like this:



Please choose store mode pop-up with following cues

The pop-up only appears if the store options do not specify what should happen.

Look at the **Store Options and Defaults topic** for information about specifying this while storing.

Cue Only is described in the **Cue Only topic**. The shield tracking options are described in the **Tracking Shield topic**.

Remove, **Release**, and other tracking information are described in detail in the **What is Tracking topic**.

The two remaining options are:

- **Overwrite:**
  This will remove what is already stored in the cue and only store the new values using the tracking settings in the pop-up.

- **Merge:**
  This will merge the new values into the existing values. New values have a higher priority and will overwrite existing values.

Examples

In the following examples, we have a sequence with the following two cues:

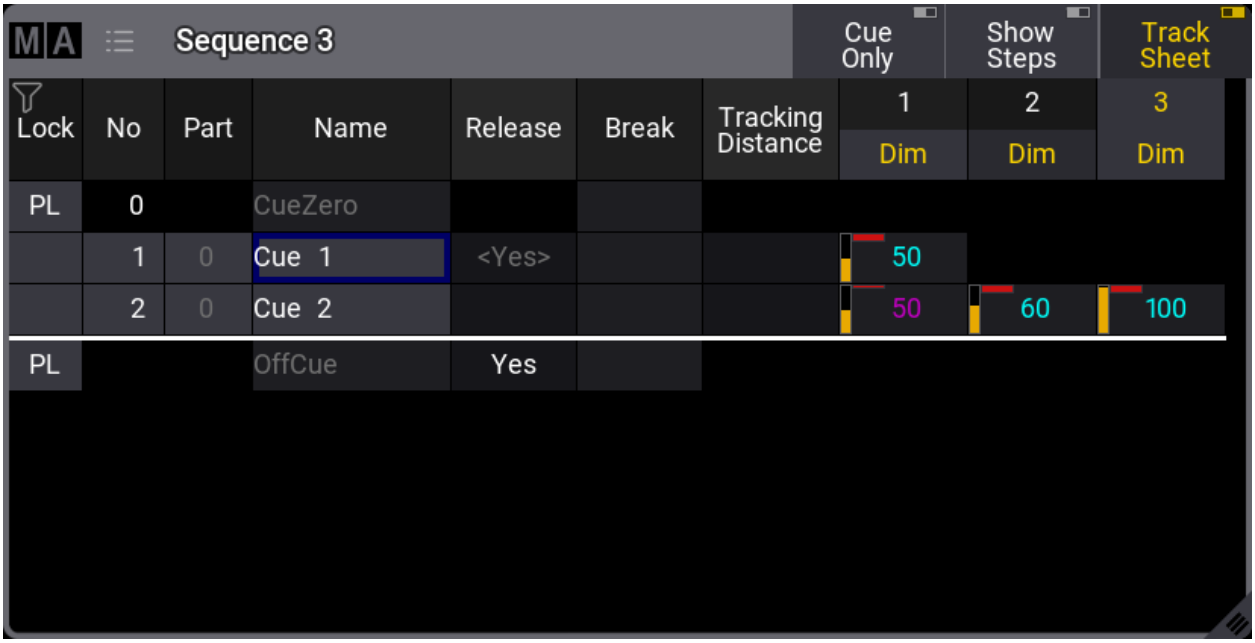
The magenta value for fixture 1 in cue 2 is a tracked value.

Now we turn on fixture 3 at 100 % and store this into cue 2.

This is the result if **Overwrite** is chosen:

Now fixture 2 is gone. This is because it only had values stored in cue number 2. The dimmer value of fixture 1 is not affected because it is a tracked value.

If we had chosen **Merge** instead, it would have looked like this:



Now, the value from fixture 3 is added to the existing values.

Store Cues with Timings

When storing a cue, it is possible also to store the different cue timings. This is described in detail in the **Cue Timings topic**, but here is the short version.

The Time key will add different timing keywords to the command when storing.

For instance, storing cue 4 with a fade time of 6 seconds and a delay of 1 second, the following keys can be pressed:

Store Cue 4 Time 6 Time 1 Please

This is the result in the command line feedback:

```
OK :  Store Cue 4 CueFade "6" CueDelay "1"
```

Pressing the Time button repeatedly will change what timing keyword it adds.

Adding and Using Cue Labels

A cue can be given a name - using the **label keyword** - while it is stored. This is the syntax: **Store Cue [Cue_Number] "My Cue Name"**.

The keyboard is needed for writing this. The quotation marks are needed to tell the software that this is text - then, it is not interpreted as a command.

Labels can also automatically be enumerated while storing. Have a look at this command:

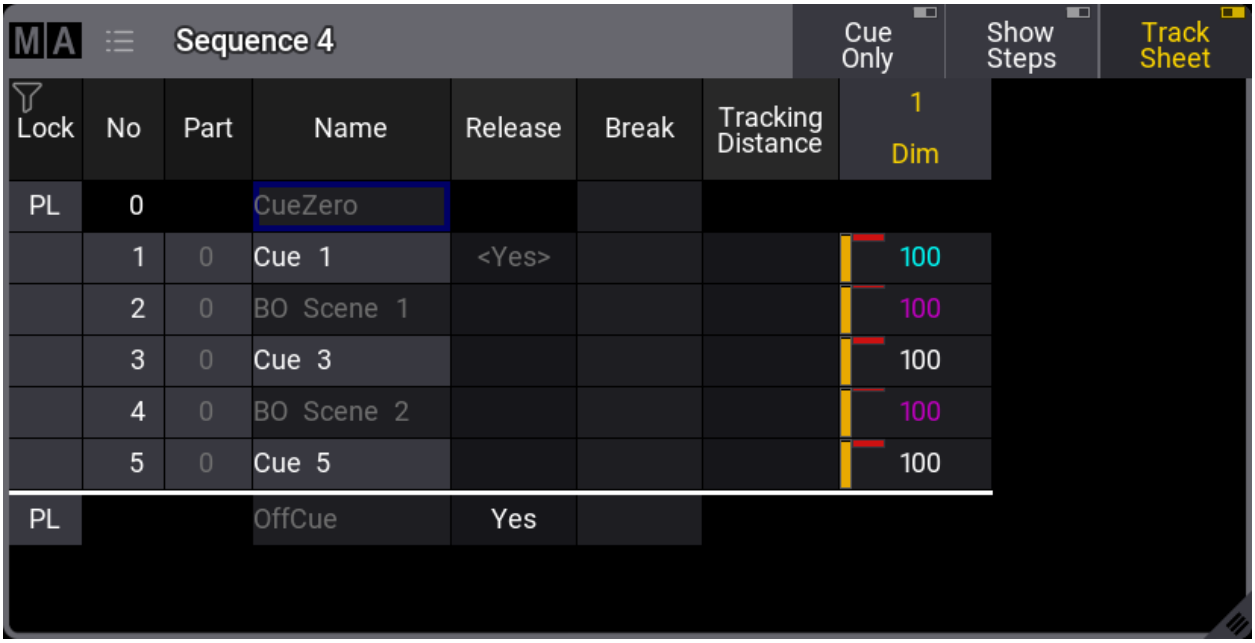
```
User name[Fixture]>Store Cue 2 + 4 "BO Scene 1"
```

This will not label both cues 2 and 4 the same - it will add 1 to the number for each cue. The result is that cue 2 is called "BO Scene 1" and cue 4 is "BO Scene 2". This enumeration only works if the number is the last part of the label and if there is a space between the last word and the number.

Cue labels can be used when storing. This means that if there are several cues whose labels start with "BO", it is possible to store into all these cues in one operation using BO plus an asterisk. See the following example.

Example

This is the cue sequence and content before storing:



Notice that cues 3 and 5 are **blocked**. This means that 100% is stored in the cues even though it is currently not necessary.

With an active value of 0% for fixture 1, use the keyboard to type the following command:

```
User name[Fixture]>Store Cue "BO*" /Merge
```

This is the result:

The two cues whose label begins with BO now got the new value.

Store Cue Parts

Besides the many cues that could be stored in each sequence, it is also possible to store 256 **Cue Parts** to each cue.

Values are always stored in a cue part, and often when we talk about a cue, we are actually talking about all the parts in a cue.

Cue parts are a division of the cue. Cue part 0 is always created with the cue, and this part has all the values unless a different part is created or specified when storing.

Have a look at this example:

Notice that the value for fixture 3 is stored in cue 2, part 10. It does not have a value in cue 2, part 0. And the values from fixtures 1 and 2 are in part 0 and therefore do not have any value in cue 2, part 10. It appears as tracked values.

Storing something in a part is almost as easy as storing the main cue. Using the example above, the keypresses would be:

Store Cue 2 Cue 1 0 Please

The second press on the cue key will result in the **Part keyword**, and the command line feedback looks like this:

```
OK
:   Store Cue 2 Part 10
```

As a default, it is only possible for an attribute to be stored in one cue part per cue. But this can be changed so an attribute can have values in multiple cue parts in the same cue. If **Allow Duplicates** is turned On for a cue then attributes can be stored in all parts of the cue.

In this example, **Allow Duplicates** is set to Yes for cue 2, and fixture 1 has values in both part 0 and part 10:

The cue part with the highest number takes precedence, and in the example above, fixture 1 will end at 40% output when cue 2 is triggered.

Allow Duplicate is a column in the **Sequence Sheet**.

Using Command Line Input to Add More While Storing

In the example using the cue labels to store, a command showed some of the other possibilities while storing cues.

The command line gives access to all the store options in the GUI Store Options - read about them in the **Store Options and Defaults topic**.

All the different elements are described in the **Store Keyword topic**.

Examples

The following are just a few extra command line examples showing some of the possibilities while storing.

Stores cue number 1.2 in sequence 4.

It does not matter if you write sequence or cue first. So this could also have been:

The commands can often be written shorter in the command line input. See some examples in the **General Syntax Rules topic**.

Read the topics about each keyword to see the short version of the keyword.

This will store the cue with a name, merged, and stored with an in-fade of six seconds and an out-fade of three seconds. Read more about the store options in the **Store options and defaults topic**.

Store Remove

A version of storing is the **Store Remove**, where Remove is selected in the store pop-up (described above).

This will remove the stored values for the attributes that currently have active values in the programmer.

The actual values in the programmer are irrelevant in this case. They simply indicate what attributes should be removed from the cue.

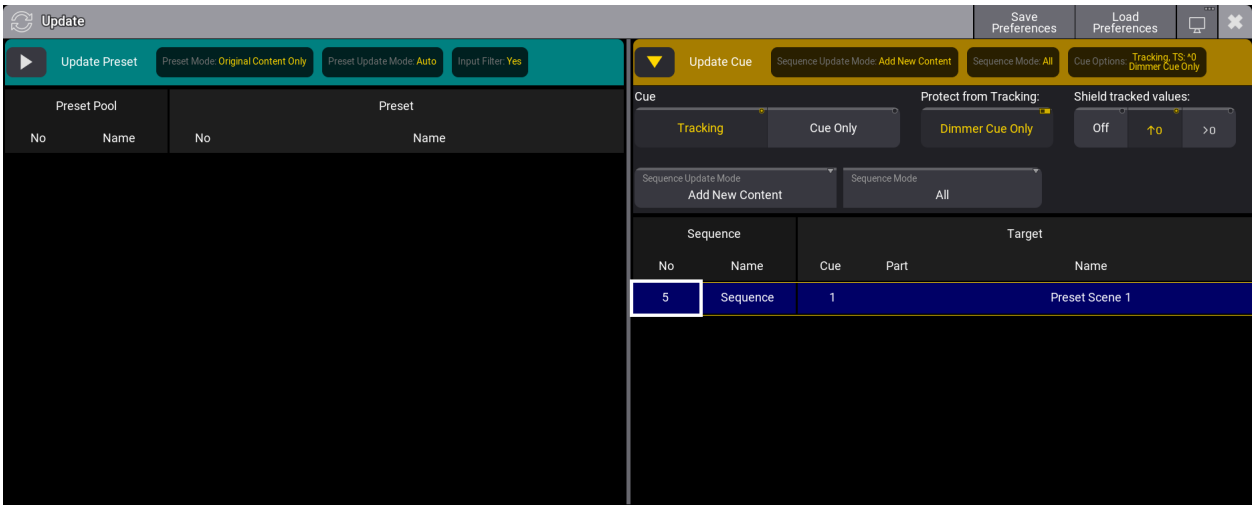
1.30.6. Update Cues

The stored values in a cue can always be changed by storing new values in them.

But if there are active cues and active values in the programmer, it is also possible to update the cue or cue part.

The **Update key** flashes when this scenario is valid.

Pressing Update opens the **Update Menu**.



The Update Menu with possible update locations.

The main update menu is split into a left and right section. The right side displays a list of possible cue update targets. If there are multiple data pools in the show, columns with the data pool information are shown.

To update the cue part with the current settings, just tap the desired cue part on the list on the right side. This immediately updates the stored values in the selected location.

Entire cues can also be selected as the update target. If values already exist in single parts and are updated with the **Sequence Update Mode** setting to **Add New Content**, the values will be updated in their respective parts.

Some options are relevant to the cue update process. The current selected values for these options are displayed in the dark yellow area on the right side. These settings can be changed by tapping the right-pointing triangle (▶). This opens the settings - the example image above shows the settings for the cues.

The top row of the cue settings are the tracking settings. These settings define how and if the updated values are tracked through the following cues. Learn more about these settings in the **What is Tracking topic** and its sub-topics.

The two settings on the second row are **Sequence Update Mode** and **Sequence Mode**.

Sequence Update Mode:

The mode toggles between two options. **Original Content Only** updates the already existing values at their original location - possibly a previous cue. **Add New Content** updates the active cue with all the programmer values (filter and worlds are still respected), possibly adding new values and new content to the cue.

Sequence Mode:

The sequence mode is a filter that can be used to filter the list of possible cue parts. The options are:

- **All**:
  This shows all possible cue parts from all active sequences.
- **Selected**:
  This only shows the cue part from the selected sequence.
- **Last Go**:
  This shows the latest cue to receive one of the trigger commands (<<<, >>>, Go+, Go-, Goto, Load, On, Select, Top, Temp, Flash, Toggle On, Pause). This includes if the cue is triggered from a running timecode recording. A sequence (and its cues) can be excluded from LastGo - read about the **Sequence Settings**. Last Go only shows cues triggered by the same **user profile**.

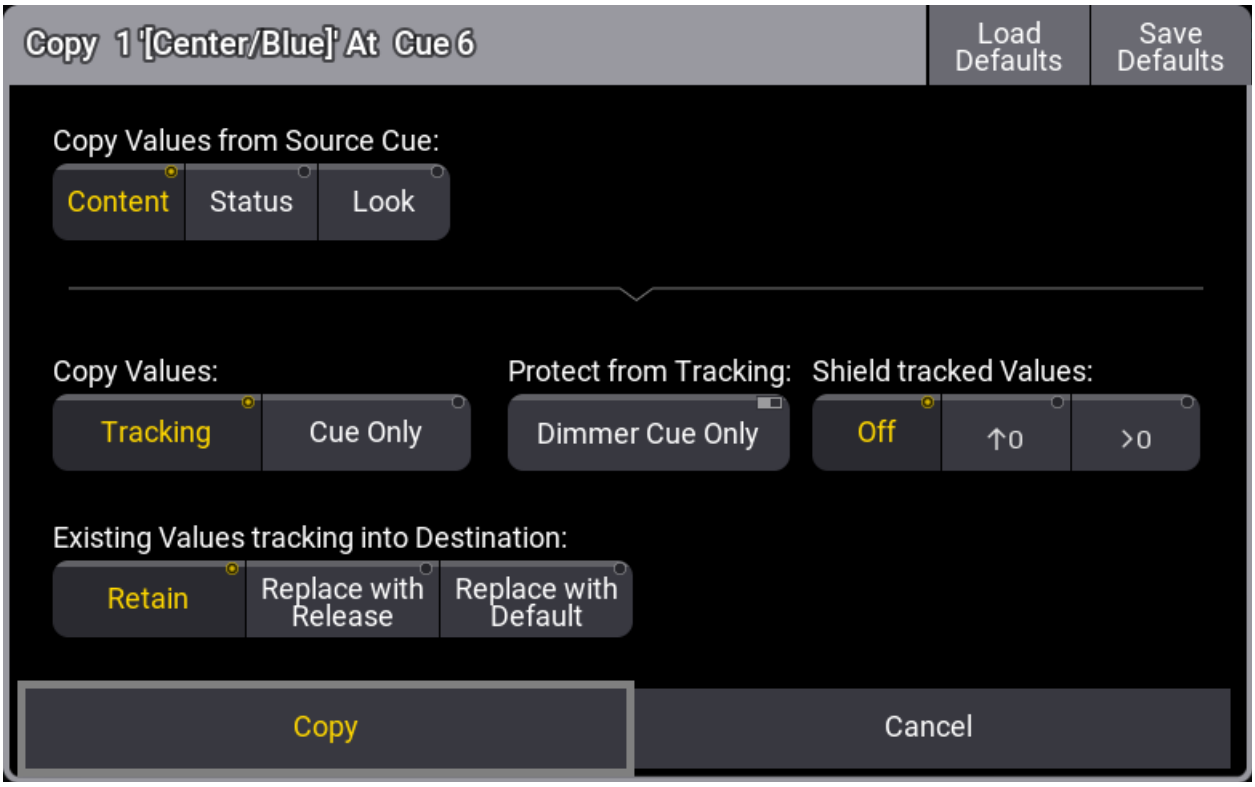
The left side of the **Update Menu** is about presets. Learn more about presets in the **Preset section**.

1.30.7. Copy Cues

Cues can be copied to a new or existing cue.

The **Copy keyword** is needed for this operation. The general syntax is **Copy Cue ["Cue_Name" or Cue_Number] At ["New_Cue_Name" or New_Cue_Number]**.

Different options define what and how the cue is copied. These options appear in a pop-up when the copy command is executed.



Copy cue pop-up

Copy values from Source Cue have three different radio buttons that are mutually exclusive:

- **Content**:
  Copies only the values stored in the source cue.
- **Status**:
  Copies the status of the source cue. Status includes the values stored in the cue and all tracked values from former cues.
- **Look**:
  If the fixture's dimmer attribute has a value status above 0% in the source cue, then the status values of all attributes in the source cue are copied.
  If the fixture's dimmer attribute has a value status of 0% in the source cue, then only the dimmer attribute is copied from the source cue.
  This means that the current status of the dimmer value is always copied.

Copy Values define how values should track through the sequence. There are two radio buttons:

- **Tracking**:
  The new values are added as normal tracking values and will track onward.
- **Cue Only**:
  The new values are added using Cue Only. Attributes that do not have a previous value it can return to will store the default value in the following cue.

  Cue Only is a function that stores values at the destination, but also stores the previouslly tracked values in the following cue. The result is that following cue keeps the same look. Learn more about **Cue Only** in the **Store Cues Topic**.

A single option in the **Protect from Tracking** section is **Dimmer Cue Only**. Turning this On will store the dimmer attributes as Cue Only, but the other attributes can be stored as tracking using the other settings.

The **Shield tracked Values** have three options. It is about the Tracking Shield function. Learn more in the **Tracking Shield topic**. The options are:

- **Off**:
  Tracking Shield is not used.
- **↑0**:
  Protects attributes in the next cue where the dimmer value increases starting from zero.
- **>0**:
  Protects attributes in the next cue where the dimmer value is above zero

Existing Values tracking into destination also has three mutually exclusive radio buttons. This setting defines what should happen with values that track into the new copied cue from previous cues.

- **Retain**:
  The tracking values are kept and tracked into the new cue.
- **Replace With Release**:
  Tracking values are replaced with the "Release" special value.
- **Replace With Default**:
  Tracking values are replaced with the default value.

Finally, there are four different possible execution buttons:

- **Copy**:
  If the destination is an empty location, then there is a copy button. Tapping this executes the copy action using the settings in the pop-up.
- **Overwrite**:
  If the destination already contains values, then tapping Overwrite will copy the cue. The copy action overwrites the destination cue with the source cue data and deletes values not defined by the source cue.
- **Merge**:
  If the destination already contains values, then tapping Merge will copy the cue and the copy action merges the data of the source to the destination. Existing data in the destination cue will be kept as long as both the source and destination do not have data of the same attribute. If existing data in the destination is also part of the source, the data from the source wins and overwrites the destination.
- **Cancel**:
  Always available. Cancels the cue copy operation and closes the pop-up.

Copy of Ranges

When copying ranges of cues, these rules apply:

- When a natural order of cues is selected using **Thru**, the software attempts to maintain all gaps from the source range at the destination.

Example:

Precondition: Cues 1, 2, and 4 exist.

Result: New cues 11, 12, and 14.

- If a cue exists already in the range of the destination spot (even if it would fit into a gap), the software suppresses the range at the destination so that the original order of cues is uninterrupted. The software tries to set whole cue numbers. If this is not possible, it appends dotted cue numbers. The software tries to use as few decimals as possible (1 -> .1 -> .01 -> .001).

Examples:

Precondition: Cues 1, 2, 4, and 12 exist.

Result: New cues 11, 11.1, and 11.3. The gap between cues 2 and 4 is maintained but shifted, as the start of the destination is 11 and not 11.1.

Precondition: Cues 1, 2, 4, and 11.1 exist.

Result: New cues 11, 11.01, and 11.03. The gap between cues 2 and 4 is maintained but shifted, as the start of the destination is 11 and not 11.1.

- When the source range of cues is selected in a reversed order by using **Thru**, the software ignores the gaps from the source range when creating the destination range. The same rule for the resulting cue numbers at the destination from above applies here as well.

Example:

Precondition: Cues 1, 2, and 4 exist.

Result:
The content, status, or look of cue 4 is copied to cue 11.
The content, status, or look of cue 2 is copied to cue 12.
The content, status, or look of cue 1 is copied to cue 13.

- In the case of defining the source range by using **+**, the arising gaps between the single cues can be maintained if the cues are selected in ascending order. If the order is reversed or mixed, then the gaps are suppressed. The same rule for the resulting cue numbers at the destination from above applies here as well.

Examples:

Precondition: Cues 1, 2, and 4 exist.

Result: New cues 11 and 14

Precondition: Cues 1, 2, and 4 exist.

Result: New cues 11, 12, and 13.

- When the source list is generated by using a combination of **Thru** and **+**, the above rules apply. The thru-part of the source list uses the rules of thru, while the +-part uses the +-rules.

Copy Paste

The copy function places data in temporary memory. This can be pasted to a destination using the **Paste keyword**.

The principle is that a cue is copied into the memory and pasted to a destination.

This process does not provide the same options as described above in the normal copy syntax. Paste uses content as the source and will do a standard copy or merge.

Cut

Cues can be cut and pasted as a means to move a cue. Use the **Cut keyword** instead of copy. This does not give the same options as the copy function.

A cue needs to be cut first and then pasted into a destination.

1.30.8. Cue Recipes

Recipes can be used in cues and presets. See the **Recipe** topic to learn the basics of recipes and the **Recipe Preset** topic to learn about using recipes in presets. It is a good idea to read the recipe topic before this one.

This topic is about recipes in cues.

A recipe can contain multiple lines describing what should happen based on a set of information. The recipe "cooks" values into the cue.

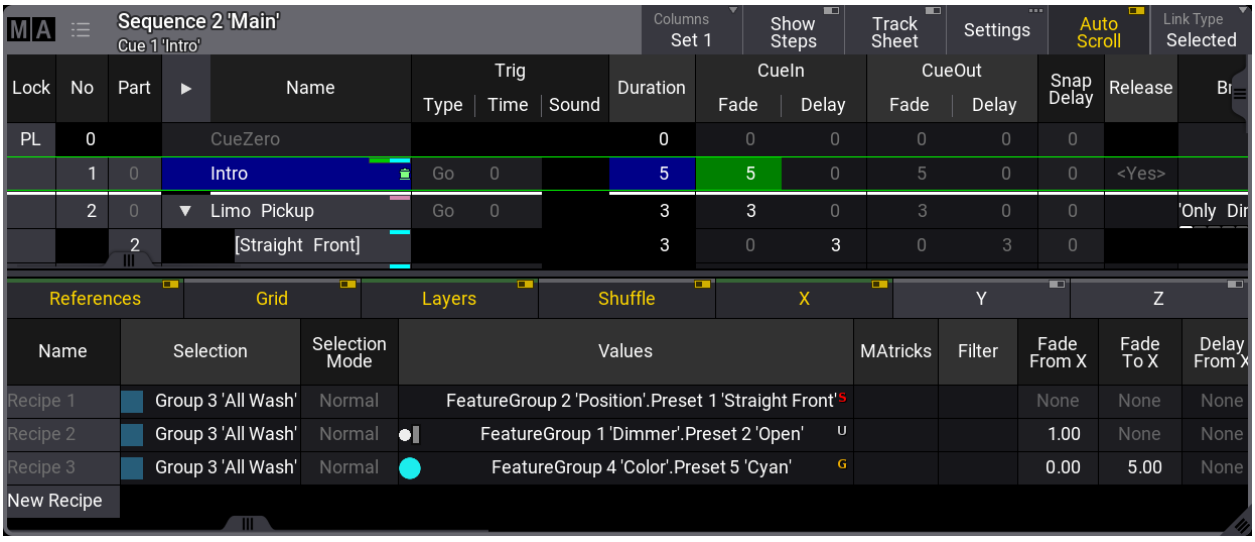
A recipe line can contain information about a selection, values, MAtricks reference, filter, individual fade, delay, speed, phase values, and grid values.

Values from recipes can be combined with conventionally stored values.

Adding Cue Recipes

Recipes are added to each cue part in a sequence.

The best way to access the recipes is by turning On the Show Recipes mask in a **Sequence Sheet**:



Sequence sheet with cue recipes showing

This gives access to adding, editing, and deleting recipe lines.

Showing the recipe lines in the sequence sheet adds a filter line that allows filtering of the different elements in the recipe - not to be confused with the Filter column on the recipes. This line can be moved up or down by tapping and holding the line and sliding it up and down. Release the screen at the desired location.

Learn more about this line and the different columns in the **Recipes** topic.

The cells in the recipe lines can be edited and more lines can be added by tapping and holding the New Recipe.

Each line can contain a set of information.

The MAtricks reference column and the individual MAtricks columns only take effect when there is ranged data from, for instance, a **MAgic preset** or a timing range.

Making changes to the recipe line automatically cooks the line using the merge option. Recipe lines without a group do not auto-cook.

Deleting an already cooked recipe line does not auto-cook. The recipe needs to be manually cooked to reflect the new result.

Cooking can be done using the **Cook keyword**. An entire sequence can be cooked in one command.

The general syntax for cook is: **Cook [object] (/option)**

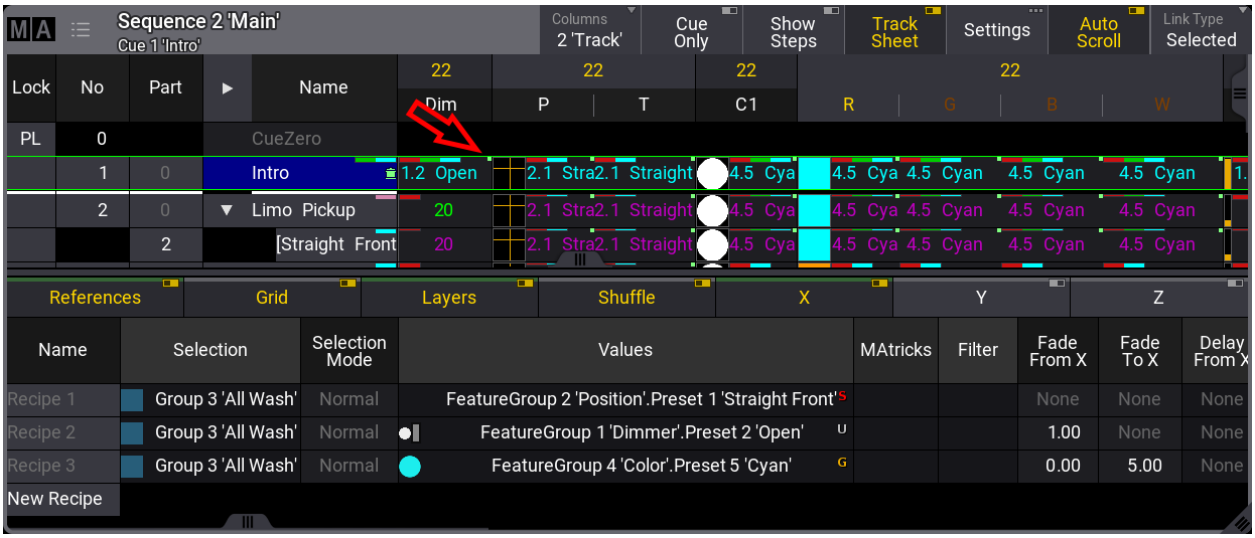
There are four options:

- **Merge** - Default if nothing else is specified. Removes values that are flagged as cooked, then cooks the values from the recipes and also replaces values at the destination based on the recipe ingredients.
- **MergeLowPriority** - Replace existing cooked data and add new data based on the recipe ingredients, but do not change non-cooked data.
- **Overwrite** - Delete all contents of the target object and cook new data based on the recipe ingredients.
- **Remove** - Delete any data from the target object that has been cooked but never updated.

Executing a cook command without an option keyword opens a pop-up prompting for which option to use. It also includes a cancel option that cancels the cook command.

A cue with recipe information gets the small pot icon in the name column. It does not indicate whether there are cooked values or not; it only indicates if there are recipe lines and if the lines are valid.

Activating the Track Sheet mode in the Sequence Sheet makes the attribute values visible for each cue. Values that come from a recipe have a small green marker in the upper right corner.



Sequence Sheet in Track Sheet mode showing the Recipe marker

1.30.9. Store Settings and Store Preferences

There are some settings used when storing objects like cues, presets, and groups.

Each user has their own preferred settings.

During the store, it is also possible to add option commands that will use a specific set of store settings.

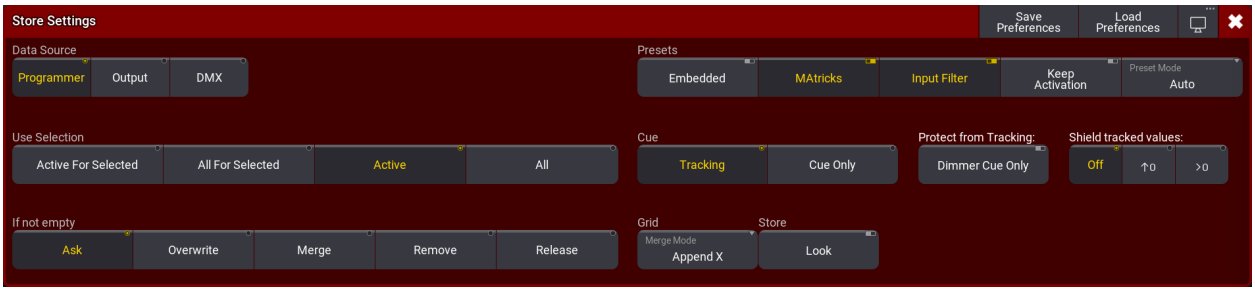
In this topic, the settings regarding storing cues are described. Some of these settings will impact how other cues will respond. Elements regarding tracking are described in the **What is Tracking topic** and sub-topics.

Temporary Store Settings

The store settings can be opened as a temporary version.

This is done by pressing and holding the Store key for approximately one second.

It appears as a pop-up on screen 1 (default location).



Store Settings pop-up showing the current settings

There are different areas with different settings: Data Source, Use Selection, If not empty, Presets, Cue, Grid, and Store.

In the title bar, there are three buttons. Tap Save Preferences to store the current settings as the new default user store preferences. Tap Load Preferences to load the stored user store preferences values into the store settings pop-up. The last button is used to **change the menu location**.

Data Source

This defines the data source for what will be stored.

There are three different sources:

- **Programmer** (Default):
  This will use the values in the programmer as the source. It takes the **Use Selection** and **Store** settings into account (read below).

- **Output**:
  Uses the status of the current output from the console as the source. The output can be affected by masters and DMX profiles. The **DMX sheet** shows what the output actually is. The **Use Selection** setting is limited to **All For Selected** or **All** (Default).
- **DMX**:
  This uses incoming DMX values as the source. It is only valid for DMX channels that have fixtures patched. The **Use Selection** setting is limited to **All For Selected** or **All** (Default).
  When storing incoming DMX, only the values are stored that are not at the default of the fixture type. In addition, values that are at fixture type default will be stored into the sequence when the sequence contains already this attribute with a different value.

Use Selection

This decides what values, from the source, will be used when storing.

- **Active For Selected**:
  Stores all active attributes but only for the fixtures selected in the programmer. This is only available to the **Programmer** data source.
- **All For Selected**:
  Stores all attributes of the selected fixtures.
- **Active** - (default for programmer source):
  Stores the values that are active in the programmer. This is only available to the **Programmer** data source.
- **All** - (default for Output and DMX source)**:**
  Stores all attributes for all fixtures.

If Not Empty

This defines the store method used if values are stored into an existing cue.

- **Ask** - This is the factory default setting:
  This means that none of the other options below are used and a pop-up asks the user what to do.
- **Overwrite**:
  All existing data is deleted and the current source and selection are used to store new values.
- **Merge**:
  This will merge the new values into the existing values. New values have a higher priority and will overwrite existing values.
- **Remove**:
  This will remove the stored values for the attributes using the current **Use Selection** setting.
- **Release**:
  A special release value is stored. The actual values in the source are not relevant, but the selection is used to define where the release value is stored.

Presets

These are the settings used when storing presets.

They are described in the **Create Presets topic** or in the other **Preset topics**.

Cue

There are two radio buttons here: Tracking and Cue Only.

Values can be stored using the tracking principles or as Cue Only.

Learn more about **Tracking** in the **What is Tracking** topic.

Learn more about **Cue Only** in the **Cue Only topic**.

Protect from Tracking

This setting is called **Dimmer Cue Only**. It is relevant when the Cue mode is Tracking (see above). Toggling this setting On will store the dimmer values using Cue Only principles and other attributes as tracking (link about Cue Only and Tracking topics above).

Shield tracked values

The shield function can be used to protect tracked values in future cues. This setting is only relevant when the Cue mode is Tracking. The three options are explained in the **Tracking Shield topic**.

Grid

There is one setting in this area. It is Merge Mode. This is relevant for all elements that store the fixture grid position - Learn more about the grid in the **Selection Grid topic**.

The **Merge Mode** is used when an object (including fixture grid position) is merged into another object of the same type. For instance, a group merged into another group or a preset merged into another preset.

There are two options that the button toggles between:

- **Append X**:
  The fixtures are merged into an offset X position in the selection grid. The merged fixtures (source fixtures) are positioned after the last existing fixture (destination fixtures). They are appended to the existing fixture's X position.
- **Off**:
  The fixtures are merged into the grid positions where they are originally stored.

Store

This area only has one toggle button: **Look**. This can be turned On or Off.

When this is On, then all attributes for fixtures with a dimmer value above 0 are stored. If **Active For Selected** is selected and there are fixtures that have a dimmer level actively at 0% in the programmer, then only the dimmer value will be stored for these fixtures, even if other attributes are active in the programmer.

This can be combined with various settings for **Data Source** and **Use Selection**.

Cue Preferences

There is a set of timing preferences used when cues are created. They can be seen and edited in the Menu -> Preferences and Timings.

Here are the general cue timings and the preset timings. Read more about these different timings and what they do in the **Cue Timing topic**.

1.30.10. Play Back Cues

Cues are stored in a Sequence. All sequences are stored in a sequence pool.

When a cue is active, the sequence is active in the pool.

It is not possible to run a cue without playing back the sequence.

Executors are controls that allow for easy hands-on playback of the sequences. Executors can playback or trigger other objects than sequences as well.

Sequences do not need to be associated with an executor to playback cues.

Relevant Playback Commands

Many keywords can be used for playback operations. They are all listed using the **Help** command.

For sequence playback, there are some very common keywords (read details about them by following the links):

- **Go+**
  Use this to trigger the next cue with a "Go" trigger or to specify a cue (**Go+ Cue ["Cue_Name" or Cue_Number]**). This command triggers subsequent cues using a follow or timed trigger.
- **Go-**
  Use this to trigger the previous cue or a specific cue using **Go- Cue ["Cue_Name" or Cue_Number]**. Using the syntax that specifies the cue number does not trigger subsequent cues.
- **Goto**
  Use this command to go to a specific cue. The specified cue is triggered when the command is executed. Please note that this command also asserts tracked values so that the visual result can be different than a normal Go command. The **Assert** mode for the cue is relevant. "None" and "Assert" will assert tracked values using the cue timing from the cue where the value is stored. "X-Assert" will assert the tracked values using the cue timing of the "Goto" cue. Learn more about the Goto keyword in the **Goto Keyword topic**.
- **Load**
  This is used to preload cues. The cue is then ready to be triggered via a Go+ command. Several cues in different sequences can be loaded and triggered together with the **Go+ Loaded** syntax.
- **Pause**
  Executing this command will pause all running fades, delays, and phasers, effectively halting all values where they currently are. The fade and delay are also resumed using the Pause command.
- **Top**
  This keyword is used to trigger the first cue
- **<<< (GoFastBackward)**
  If a target cue is not specified, the previous cue in the sequence is triggered, ignoring any cue timing and jumping to the cue using the timing defined in the **Playback Timings**. The default timing is 0 seconds. A cue can be specified, and this will trigger the specified cue only

(**<<< Cue ["Cue_Name" or Cue_Number]**). This keyword does not trigger follow and timed cues.

This timing is set in the **Preference and Timings** menu under **Timings**. Learn more in the **Cue Timing topic**.

- **>>>(GoFastForward)**
  If a target cue is not specified, the next cue in the sequence is triggered, ignoring any cue timing and jumping to the cue using the timing defined in the **Playback Timings**. The default timing is 0 seconds. A cue can be specified, and this will trigger the specified cue only (**>>> Cue ["Cue_Name" or Cue_Number]**). This keyword does not trigger follow and timed cues.

  This timing is set in the **Preference and Timings** menu under **Timings**. Learn more in the **Cue Timing topic**.

But there are many more that can be useful.

These functions can be assigned to executor keys for easy access (See how in the **Assign Object to an Executor topic**), but they can also always be sent directly to the sequence. For instance, the following command can be used if you want to go to cue 4 in sequence 6:

```
User name[Fixture]>Goto Cue 4 Sequence 6
```

The order of sequence and cue does not matter. So it could also be:

```
User name[Fixture]>Goto Sequence 6 Cue 4
```

The system interprets it as in the last example. The Command Line History window shows this response:

```
OK
:  Goto Sequence 6 Cue 4
```

If an executor handles a sequence, the playback commands can also be sent to the executor.

```
User name[Fixture]>Go+ Executor 101
```

This will send the go command to the object assigned to Executor 101. If this object is a sequence, the next cue will be triggered.

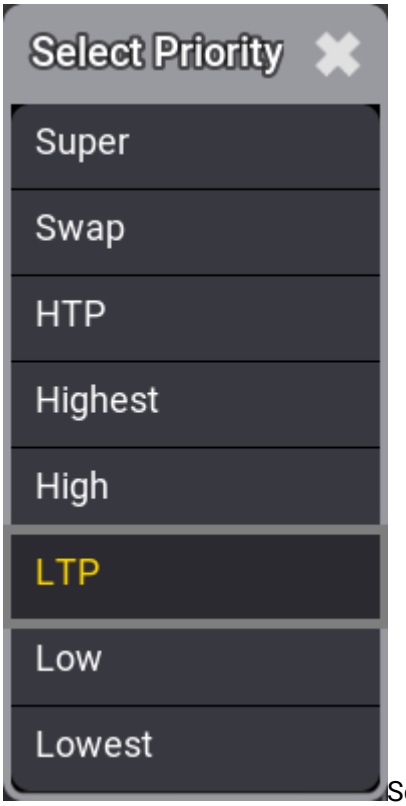
Sequence Priorities

When several sequences affect the same fixture attributes, then priorities become important. The priority setting is a property of the sequence.

These settings can be opened by pressing the Assign key and then one of the executors keys where the sequence is assigned. This opens the **Assign Menu**. On the right side of the menu, there are some buttons. Please tap Edit Settings. These are all the settings available for a sequence. The settings can

also be opened by tapping Settings in the title bar of a sequence sheet. All the settings are discussed in the **Sequence Settings topic**.

The priority setting can be found in the **Playback** group of settings. Tapping it toggles through the different properties, swiping it opens a small select pop-up like this:



Select Priority pop-up

This is the list of possible priority properties. This is a short explanation of the priorities:

- **Super:**
  This priority is the LTP priority above any other playbacks and even above the programmer.
- **Swap:**
  Intensity is working as LTP with higher priority than HTP.
- **HTP (Highest Takes Precedence):**
  The highest intensity value will be used. Other parameters will use LTP.
- **Highest**:
  Highest LTP priority - Like LTP but with the highest possible LTP priority.
- **High:**
  High LTP priority - like LTP but a higher priority than normal LTP.
- **LTP (Latest Takes Precedence):**
  This is the normal LTP priority. The newest attribute value is prioritized over the old value.
- **Low:**
  Low LTP - This is a lower LTP priority.
- **Lowest**:
  Lowest LTP - This is the lowest possible LTP priority.

LTP is one of the most used priority settings. There are five different levels of LTP priority to give different sequences different levels of LTP priorities.

The list in the pop-up is also prioritized, where Super at the top has the absolute highest priority. This also means that HTP sequences have a higher priority than LTP sequences.

Soft LTP

This option relates to how intensities change from one sequence to another when the values transfer from the original sequence to the new one.

When an attribute has a value from a cue and another cue in a different sequence with the same LTP priority is beginning to send new values to the same attribute, then Soft LTP might influence how the transition from the old value to the new value happens. This is only relevant when the new value is applied by moving the master of the new sequence. In this case, the attribute can jump from the old value to the new value and immediately have the value it should have according to the master position (Soft LTP Off), or it can start to fade from the old value to the new value using the master position as a crossfader (Soft LTP On).

	Important:
	Both sequences need to be active, and both need to send value to the same attributes. **Auto Start** and **Auto Stop** are turned On by default for sequences. If these properties are turned Off, the sequences must be turned On manually for the SoftLTP function to affect the transition. If the Master faders are both at 100%, then SoftLTP does not have a function.

Example

1. Patch a fixture with a dimmer attribute.
2. Give the dimmer a 50% value and store this in a new sequence - call this sequence "Original".
3. Give the dimmer a value of 100% and store this as a new cue in a new sequence. Call this sequence "New".
4. Assign an executor fader as the master for sequence "New".
5. Ensure **Auto Start** and **Auto Stop** are active for sequence "New".
6. Open a fixture sheet and make sure it is set to show the **Output** layer.
7. Run the cue on sequence "Original" and have the master at zero for sequence "New". The dimmer value should now be 50%.
8. Now move the master slowly up from zero and see the output value change.

Soft LTP On: The dimmer will fade from 50% to 100% while the master moves up to 100 (fade from the "Original" value to the "New" value).

Soft LTP Off: The dimmer will jump down to the value dictated by the master position and fade up together with the master (jump from the "Original" value to the "New" value based on fader position).

9. Now try to move the fader slowly down to zero.

Soft LTP On: The dimmer fades from 100% to 50% (from the "New" value to the "Original" value).

Soft LTP Off: The dimmer value will fade down to 0% with the master, and when the sequence turns Off then the dimmer will jump to 50% (Fade down with "New" master position - jump to "Original" value when "New" is turning Off).

1.30.11. Move In Black

Move in black (MIB) is a function where tracking sequences look ahead and preposition attributes of fixtures that are fading the dimmer in from zero to automatically prevent transitions where the fixture would move the attributes into position while the fixture is fading in.

MIB is enabled on a cue part basis by giving the MIB property of the cue part a value that tells the console when and how it should do the prepositioning.

MIB settings are applied to cues and affect all fixtures stored in the cue. A special **Hold** value can be stored on the Dimmer attribute to prevent a MIB action for specific fixtures. **Hold** is similar to giving the dimmer attribute a value of 0%, but it does not trigger the MIB function for the fixture.

Hold can found in the calculator for the dimmer attribute, in the **Specials** tab.

It can also be applied using the **Hold Keyword**.

While the MIB is in progress, the MIB indicator in the Fixture Sheet indicates this:

- Fast flashing: Fixtures intensity fade to 0% in order to do an MIB.
- Slow flashing: Fixtures do the MIB.

Several options and properties modify MIB behavior. Two of them are **MIB Fade** and **MIB Delay** timing properties.

MIB Fade is the fade time of attributes that will be positioned by MIB. It is available in different places. There is an order of importance in which MIB fade time to apply:

1. Per cue part
2. Per attribute in a fixture type
3. Global in the show file

When a MIB fade time is specified in the cue part, the individual attribute MIB fade time will be ignored. When the MIB fade time of the cue is set to default, the global MIB fade time will be applied unless an attribute has an individual MIB fade time set.

MIB Delay is the delay time the attributes wait from having the dimmer closed until the MIB fade is performed. The MIB delay time can be specified per cue part or global in the show file. The same ruleset as for MIB fade applies: When the MIB delay time of the cue is set to default, the global MIB delay time will be applied.

When the MIB fade and/or MIB delay is performed between cues, the MIB times specified in the (future) cue part where the dimmer opens again will be applied. For example, a fixture is moving in cue 3 to be ready for cue 5. The MIB times specified in cue 5 is used for the MIB.

Cue MIB Settings

The sequence sheet can display several columns for the different MIB settings. Read more about the sequence sheet in the **Look at Cues and Preferences**.

- **MIB Preference**:
  This specifies a cue's suitability for MIB. It is a percentage number from 0(never) to 100(best). The MIB modes Early, UponGo, and Late prioritize the cue with the highest rated suitability and choose this cue for executing the MIB. The MIB mode Defined does not respect the MIB preference.
  Edit the cell to type a number or select one of the following preference options:
    - **Never(0)**:
      An MIB will never be performed.
    - **Worst(1)**:
      If there are no other options, then this cue will be used.
    - **Bad(25)**:
      It is not optimal, but it is better than the two others.
    - **Normal(50)**:
      This is the default value.
    - **Good(75)**:
      This is a better cue than normal.
    - **Best(100)**:
      This is the optimal cue to perform the MIB.
- **MIB Mode**:
  Defines how early or late the MIB shall be performed per cue part.
    - **Default**:
      Performs MIB corresponding to the MIB mode setting of the sequence setting **MIB Mode**.
    - **None**:
      MIB will not be performed for this cue.
    - **Defined**:
      A specific cue can be defined in the **MIB Target** column where the MIB is to be performed. The MIB is performed when the specified cue is active.
    - **Early**:
      Performs the MIB as soon as the dimmer is closed. Typically, after the cue transition has finished.
    - **UponGo**:
      Performs the MIB with the next cue transition after the dimmer has closed. The MIB executes with the cue after **Early** would have triggered the MIB.
    - **Late**:
      Performs the MIB latest in the cue before the dimmer opens again.
- **MIB Target**:
  A specific cue where MIB is performed for this cue part, see MIB Mode above. When setting an MIB target, the **MIB Mode** will be changed to **Defined**, and vice versa. When changing an MIB Mode that is not **Defined,** the **MIB Target** will be removed.
- **MIB MultiStep**:
  It is possible to decide whether a phaser where the fixtures are already prepositioned shall keep running with the closed dimmer or if they shall be paused. This can prevent unwanted noise and movement of stepper motors for prepositioned fixtures running a phaser, especially when it would disturb the audience. The two options are:
    - **Running**:
      A phaser is running with a closed dimmer.

    - **Paused**:
      A phaser will be prepositioned but does not start running until opening the dimmer.
- **MIB Fade**:
  The MIB fade time per cue part. It can be a set time or default. Default takes the global show file MIB fade time, or the attribute MIB fade time.
- **MIB Delay**:
  The MIB delay time per cue part. It can be a set time or default. Default takes the global show file MIB delay time or the attribute MIB delay time.

The sequence sheet hides those MIB cells that are not considered for the different combinations of MIB settings or if a cue or cue part is not suitable for MIB.

Global MIB Settings

To change the global MIB Preferences, go to Menu - Preferences and Timings.

In the Timings tab, there is a section called MIB Timings.

This defines the default **MIB Fade** and **MIB Delay** times. This value is input as time. For more information about MIB fade and MIB delay, please read above.

The property **MIB Transition** defines which transition type will be applied to the fade of MIB. Read more about the different types of transitions in the **Cue Timing topic**.

The MIB Transition can only be defined for all MIB fades globally in the show file. It is not possible to define a different transition type for a single MIB fade per cue or cue part.

In the Cues tab, there is a section called **MIB Preferences**.

Here, it is possible to change the defaults that will be set to a new cue when it is stored.

- **MIB Mode**:
  The MIB Mode is used when storing a new cue that can execute MIB.
- **MIB Fade**:
  The MIB fade time is used when storing a new cue that can execute MIB. Default uses the time set in the timing tab.
- **MIB Delay**:
  The MIB delay time is used when storing a new cue that can execute MIB. Default uses the time set in the timing tab.
- **MIB MultiStep**:
  The MIB MultiStep settings are described above.

For more information about these settings, read above.

Sequence MIB Settings

There are sequence-wide MIB settings within the sequence settings. Read more about the settings in the **Sequence Settings topic**.

The purpose of the sequence MIB settings is to have the option to overwrite the cue-based MIB settings with the MIB settings.

When setting the MIB to any option except Enabled, the sequence sheet displays the corresponding setting in yellow in the MIB Mode column header. It also adds exclamation marks before and after the mode set in the cues, for instance, !Late!, to indicate that the set value is overwritten.

The MIB Mode setting selects the default MIB mode used when doing a MIB. The MIB Mode per cue or cue part must be set to Default.

For more information about the different MIB Mode types per sequence, please read the above.

MIB Color Indicators

Attribute values will be displayed with special colors within the fixture sheet and the sequence sheet in track mode to show when the MIB is performed.

The tracking sheet view shows when the MIB is performed in this example.

Two versions of the sequence sheet showing MIB data and the Track Sheet

The fixtures need to be ready for cue 5. In cue 5, the MIB Mode is set to "UponGo". This means that when the fixtures fade to 0% in cue 2, they are ready to MIB with the next cue trigger. When cue 3 is activated, they will perform the MIB for cue 5.

The default color indicators are a deep-sea green background and black text color. Read more about the MIB colors in the fixture sheet and other grandMA3 colors in the **Colors topics**.

1.30.12. Cue Timing

Each cue part has a lot of timing information.

The default times are used if nothing is defined on cue creation. The default can be changed. Read more about defaults in the **Store Options and Defaults topic**.

Any cue timings can be changed at any point using the command line or the GUI (**Sequence Sheet**).

There are a lot of elements that affect how a fixture changes values, and they can be divided into different groups:

- **General cue times**
- **Feature Type Timing**
- **Individual Attribute times**
- **Executor Time at point of cue execution**
- **Dynamic changed Rate**

This is also the priority list from lowest to highest priority.

There is also the **cue transition** that defines how the values change from one to another.

General Cue Times

There are six different general cue timings. Each has its own column in the Sequence Sheet when condensed timing is turned off - read about it in the **Look at Cues and Sequences topic** :

- **Cue In Fade**
  This is the fade time used for all intensity values changing from a lower value and going to a higher and any other attribute changing value. It starts when the cue is triggered and after the Cue In Delay has counted down.
- **Cue In Delay**
  This is the delay before the fade. This defines a countdown time between the cue is triggered and when the fade should begin. The In Delay affects the In Fade. The default value is 0, meaning that there is no delay.
- **Cue Out Fade**
  The out fade is used by intensity values fading from a higher value to a lower. It is executed after the Cue Out Delay. The default value is the in fade, meaning that it is the same as whatever the cue in fade is.
- **Cue Out Delay**
  This is the delay time for the Out Fade. It can be used to delay when intensity values should start to fade down in value. The default value is in delay, meaning it is the same as the Cue In Delay value.
- **Snap Delay**
  Some attributes are defined to Snap. This means that they do not fade from one value to another. They change values as fast as possible. This can make sense for attributes like gobos. This delay is used to delay when the snap is performed. It makes it possible to have the

fixture fade out before snapping to a new gobo. This delay affects all snap attributes stored in the cue. It can be overwritten by individual attribute timing.

- **Command Delay**
  This delays the execution of a command. This is the only place where this delay can be defined. There is no individual timing for this.

These values can be changed using this syntax:

Cue ["Cue_Name" or Cue_Number] CueInFade [New_Cue_Time]

Just use the relevant keyword: **CueInFade**, **CueOutFade**, **CueInDelay**, **CueOutDelay**, **SnapDelay**, or **CommandDelay**.

There are two more options for setting the cue fade and delay. They are called **CueFade** and **CueDelay**. They can be used to set the four cue fade and delay times mentioned above. They are used to set the in and out time using just one keyword.

The in and out time is separated by a slash. The time before the slash is the **In** time. The time after the slash is the **Out** time. It is not necessary to specify both in and out. The slash can be used to set just one of them. The desired time has to be on the correct side of the slash.

If only one time is set without any slash, then the in time gets the actual value, and the out time is linked to the in time - it is technically set to time "None". The result is that, for instance, the fade in and fade out will be the same.

For example, setting the in fade to 5 seconds and the out fade to 8 seconds in cue 3 could be done with the following keystrokes:

Cue 3 Time 5 / 8 Please

The command line would read:

If only one time is given, both in and out will use the time. For instance,

This makes cue 4 use 7 seconds to both fade in and out.

The CueFade and CueDelay commands can address only the in or out time by adding the slash and a number to the relevant side. For instance, setting the CueOutFade to 3 and the CueInDelay to 1 on the currently active cue:

Both CueFade and CueDelay can be addressed using the **Time key**. Read more about delays by following the links above to the keywords.

Feature Type Timings

Each feature type has its own fade and delay time, and the Sequence Sheet has columns for each.

These times are used by all fixtures changing the values of that feature type. Overwriting the general cue timing for that feature type.

	Important:
	Intensity values changing to a lower value are not affected by the "Dimmer Fade" time. They are controlled by the "Cue Out Fade".

The feature type fade and delay default is general cue timing. This means they are the same as the time for the Cue In Fade and Cue In Delay.

Individual Attribute Timing

Each individual attribute can have an individual fade and/or delay time. This is called individual time. These individual times are selected using the **programmer** and added to the cue when it is **stored**. They can also be edited in the sequence sheet while in **tracking mode**.

There are two columns in the sequence sheet called **Indiv Fade** and **Indiv Delay**. They display the highest of the individual times. The **Indiv Duration** column shows the complete individual time (delay + fade).

Executor Time

There is an **Executor Time** master fader. It can be seen and changed in the **Master Controls pop-up**. It can also be **assigned to physical controls or executors**.

If this function is turned On, the set time will be used instead of the stored fade and delay timing. The **Executor Time** faders position is registered when the cue is triggered. The fader can be moved afterward without affecting the running cue fade. Read more in the **Time Control topic**.

Sequences can be protected from this function by turning Off the **Use Executor Time** setting in **Sequence Setting**.

Rate Time

The **Rate** allows the timings to be dynamically adjusted while the cue fade runs. It does not change the stored times. It simply adjusts the time to be faster or slower.

Moving the fader up makes the fade faster, making the stored times appear smaller. Pulling the fader down makes it go slower, extending the stored times.

Times affected by the rate has an asterisk (*) in front of the time. If the rate is all the way down, the times will show ***Stopped**. If the rate is all the way up, the times will show ***0**.

An executor with the rate function shows the rate as a percentage in the executor label. If the rate is exactly 100%, then the label shows **1:1**. If the rate is at maximum, the label show **1:∞**. If the rate is at minimum, the label show **Stopped**.

The rate can be reset using the **Rate1 keyword**.

Cue Transition

The cue part property **Transition** allows for modifying the path values used during the fade from one cue to another. The transition can be set per cue part.

There are nine different transition forms available. They are visualized with these images.

- **Linear**:
  This is the default transition. There is no acceleration or deceleration of the values.
  

- **Slow**:
  This uses slow acceleration and fast deceleration.
  

- **Slow+**:
  This is an exaggerated version of **Slow**.
  

- **Fast**:
  This is a fast acceleration and slow deceleration.
  

- **Fast+**:
  This is an exaggerated version of **Fast**.
  

- **SCurve**:
  This uses slow acceleration and slow deceleration.
  

- **Swing-**:
  This uses fast acceleration and fast deceleration.
  

- **Swing**:
  This is a slight exaggeration of **Swing-**. The values change slows down in the middle of the transition.
  

- **Swing+**:
  This is an exaggeration of **Swing**. The values go backward in the middle before accelerating forward again.
  

1.30.13. Renumber Cues

Cues can be renumbered. This cannot be used to change the order of the cues.

Single cues can be renumbered but it can also be a range of cues.

Giving the same cue number to two or more cues is impossible - they must be unique.

Setting the cue number to a previous number already used is prevented by the system.

Entering a cue number used later in the sequence, all following cues will be renumbered until no collision with existing cue numbers occurs.

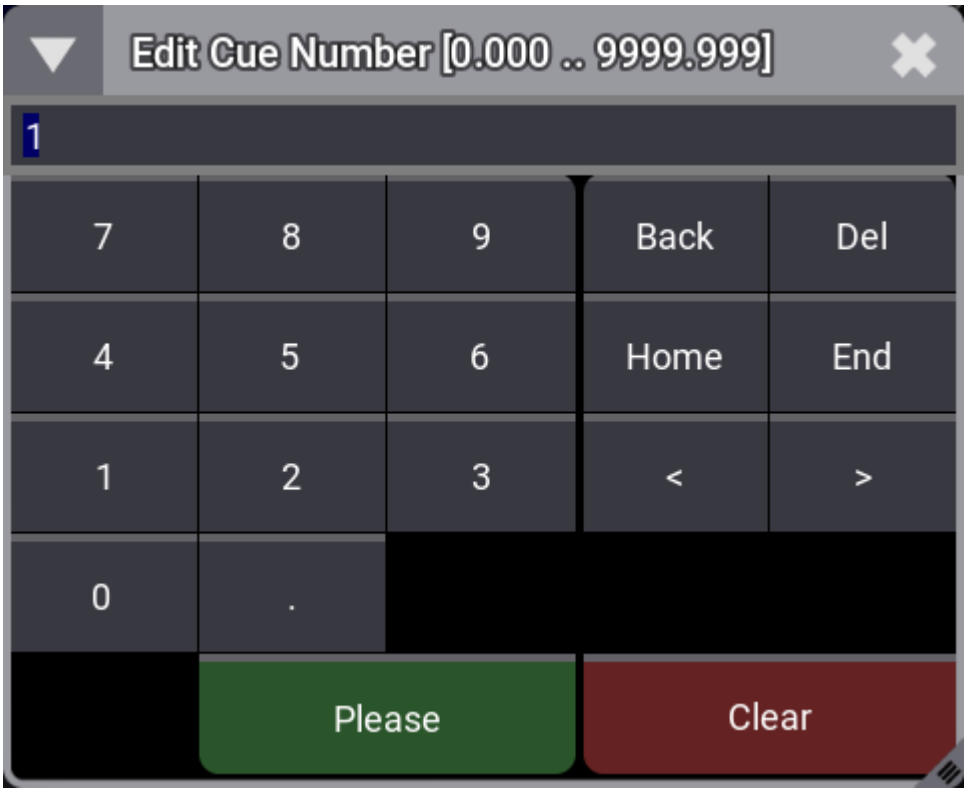
There is no command or keyword to renumber cues. It can be done in the **Sequence Sheet**.

Requirement:

Have a sequence with cues.

Procedure:

1. Open the sequence sheet window.
2. Select the cue numbers to be renumbered using the number column in the sheet.
3. Press Edit and tap inside the cue selection.
   This opens an Edit Cue Number pop-up.



Edit cue number pop-up.

4. Type the new cue number.

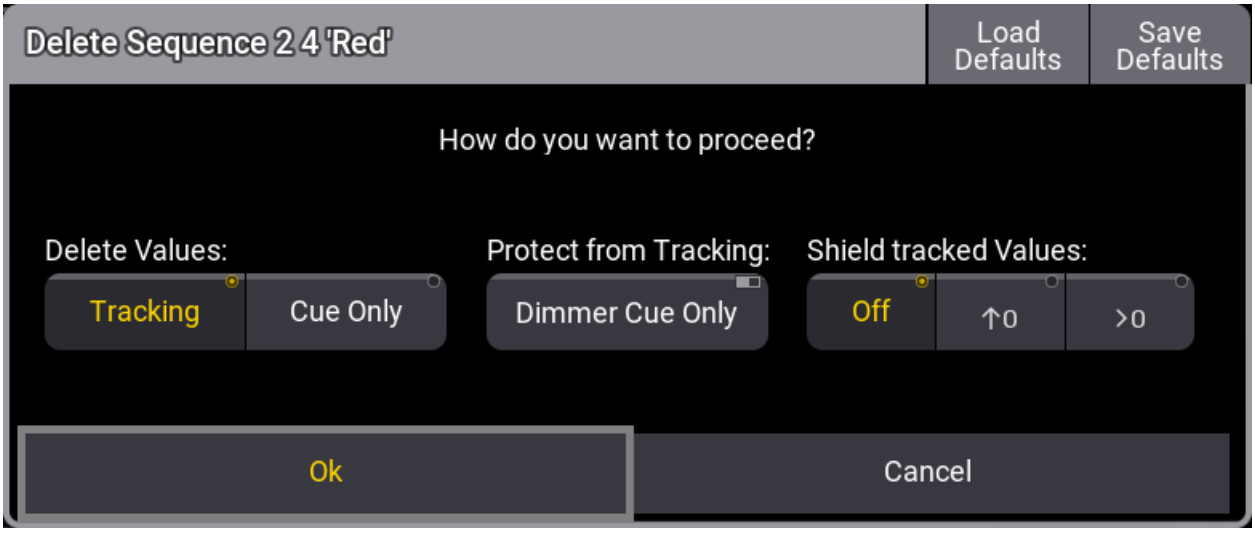
If several cues are to be renumbered, use the lasso selection when tapping the cue numbers.

1.30.14. Delete Cues

Cues can be deleted using the **Delete keyword**.

This can be accessed using the command line input or the **Delete key**.

Deleting a cue opens a pop-up asking for confirmation.



Delete Cue pop-up

Some settings influence how the deletion affects the following cues.

Delete Values has two options: Tracking and Cue Only. The option defines whether the deletion should be done using Tracking or Cue Only rules.

Protect from Tracking can be used to delete the dimmer attributes using Cue Only.

Shield tracked Values can be used to define how the Tracking Shield should respond to the deletion. Learn more in the **Tracking Shield topic**.

Tap Ok to delete the cue with the selection options and close the pop-up.

Tapping Cancel closes the pop-up and does not delete the cue.

Deleting the last cue in a sequence does not open the pop-up; it simply deletes the cue.

1.31. Executors

The executors are handles and controls to other objects.

They are often used to control sequences, but they can control other objects.

The idea is that an object (for instance, a sequence) is assigned to the executor. The executor can then control the object. Several executors can control the same object.

A sequence assigned to an executor is running cues from the sequence pool. In essence, the executor is manipulating or sending commands to the sequence in the pool.

Executors are physical keys (executor buttons), knobs (executor knobs), and faders (executor faders) on the grandMA3 hardware. They can also be represented as on-screen virtual executors - these can be viewed and operated in the **Playback Window** (see below).

Read more about the physical executor hardware in the **Executor Elements topic**.

There are 4 rows of executors. The bottom row is numbered from 101 up to 190. This row only has a single executor button. The row above is numbered from 201 up to 290. This row has an executor button and an executor fader. The next row is from 301 up to 390. This row has an executor button and a rotating executor knob. The top row is from 401 up to 490. This row also has an executor button and a rotating executor knob.

The Xkeys are also executors. The executor buttons labeled X1 | Clone to X8 | DMX are executor 291 to 298. The executor buttons labeled X9 to X16 | Exec are executors 191 to 198.

Executors are organized in wings. There are vertically 15 columns of executors on a wing. The columns are organized in sections of 5. There are 60 (15 columns x 4 rows) executors on a wing. There are 6 wings for a total of 360 (6 wings x 60 executors) executors plus the 16 from the Xkeys - the total is 376.

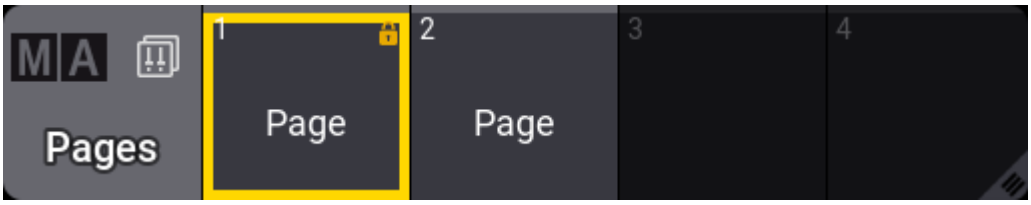
The physical device might have fewer executors than the software's amount. For instance, the grandMA3 compact console has 2 x 5 columns of executors.

The executors can also be used from the command line using the **Executor keyword**.

Executor Pages

The 376 executors are just on one page. It is possible to create up to 9 999 executor pages in each **Data Pool**. Each page has its own setup of executors. Executors that are active on one page are still active when the selected page is changed to another.

The executor pages can be seen and labeled in the **Page Pool** - it can be **created as a window**.



Page pool with page 1 selected.

The page pool is also available as a pop-up that can be opened by tapping the middle part of the page selector.

Tap the middle part of the page selector to open the page pool pop-up

There are no functional differences between the pool as a pop-up or window.

Pages are automatically created when the Page+ and Page- keys are used to change through the pages. The up and down arrows in the page selector can also be used to change the page, just like the physical keys.

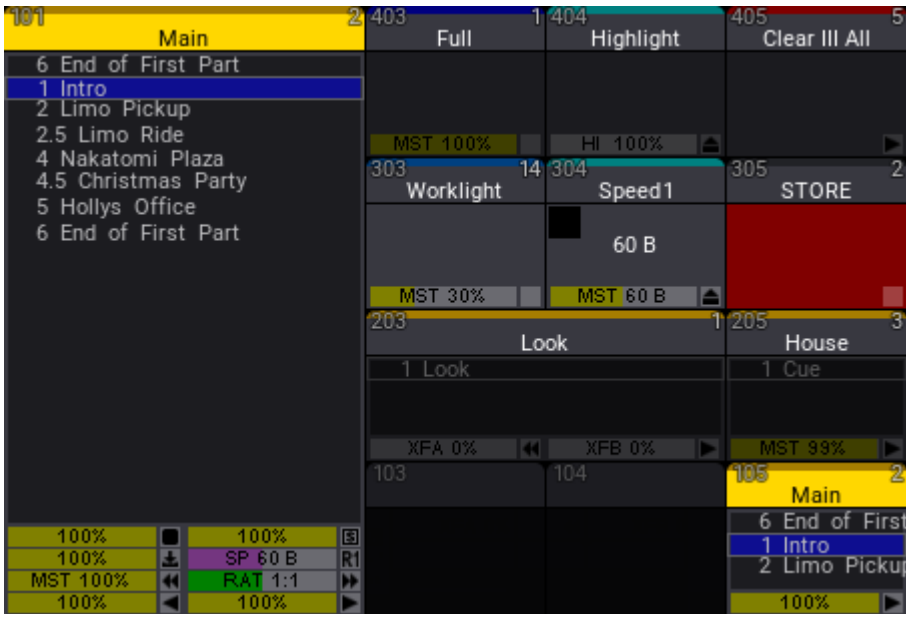
A page can also be created using the following syntax:

Store Page [Page_Number] (["Page_Name"])

The page name is optional in the syntax above. Read more in the **Page keyword** topic.

Executor Labels

The letterbox screens above the executors show the labels for the executors. The bottom of the left command screen shows the labels for the Xkeys.



Example of the playback bar for the first section of 5 x 4 executors

The executor labels above the executors are a part of the Playback Bar. Read more bout them in the **Playback bar topic**.

The labels display how the encoders are grouped, what object they control, and what handle function each executor has.

Read more about how to change all these things in the **Assign Object to an Executor topic**.

The labels take appearance and scribbles from the object assigned. For example, if a sequence has a special appearance, this appearance is also displayed in the playback bar.

Playback Window

The Playback window is a virtual representation of the physical executors.

Playback window showing all four rows and both labels and executor handles

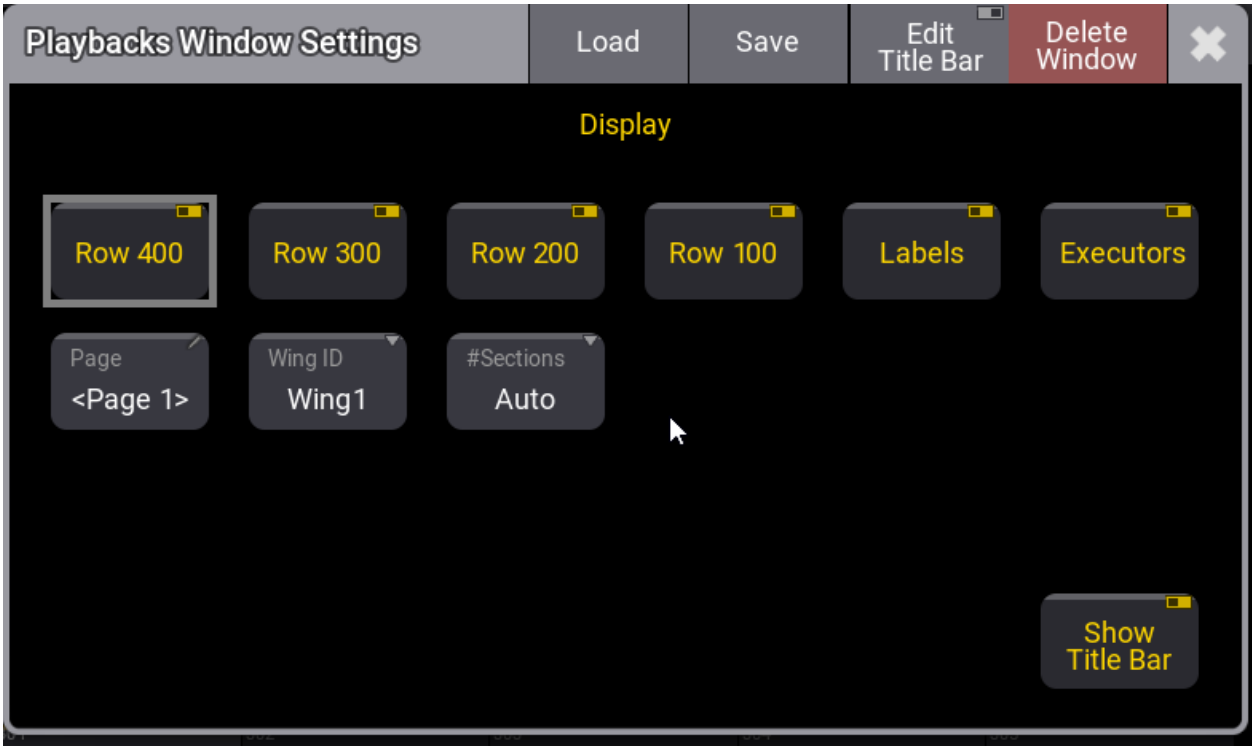
The window can automatically adjust the number of visible executors based on the window's width and the selected number of sections. This means that a window can minimum have 5 x 4 executors and a maximum of 15 x 4 (one wing). The window's minimum size to display a section of 15 executors is 10 squares wide on a screen.

The virtual executors can be operated on the screen.

Playback Window Settings

The settings allow for customization of the window.

The settings can be accessed by tapping the MA logo in the upper left corner.



Playback window settings

There are toggle buttons that can show or hide the four executor rows. There are also settings to show or hide the labels and the executor handles. This makes it very flexible. For instance, it could just show the labels for the 200 row of executors or just the executor handles for the 100 row executors.

The Page and WingID settings are also described in the **Window Settings** topic.

The Executors setting toggles the displayed on-screen copies of the relevant faders, knobs, and keys.

In additon, Labels shows or hides the executor or Xkeys labels.

The #Sections setting selects how many sections of five columns the window should display. The properties are:

- **Auto**:
  The window displays as many sections as possible based on the width of the window and a minimum width of each column.
- **1**:
  The window only shows the first section of executors - 5 columns.
- **2**:
  The window shows the first and second sections of executors - 10 columns.
- **3**:
  The window shows all three sections of executors in the wing - 15 columns.

The properties with a specified number of sections will always show the selected amount, even when the window width makes it hard to use.

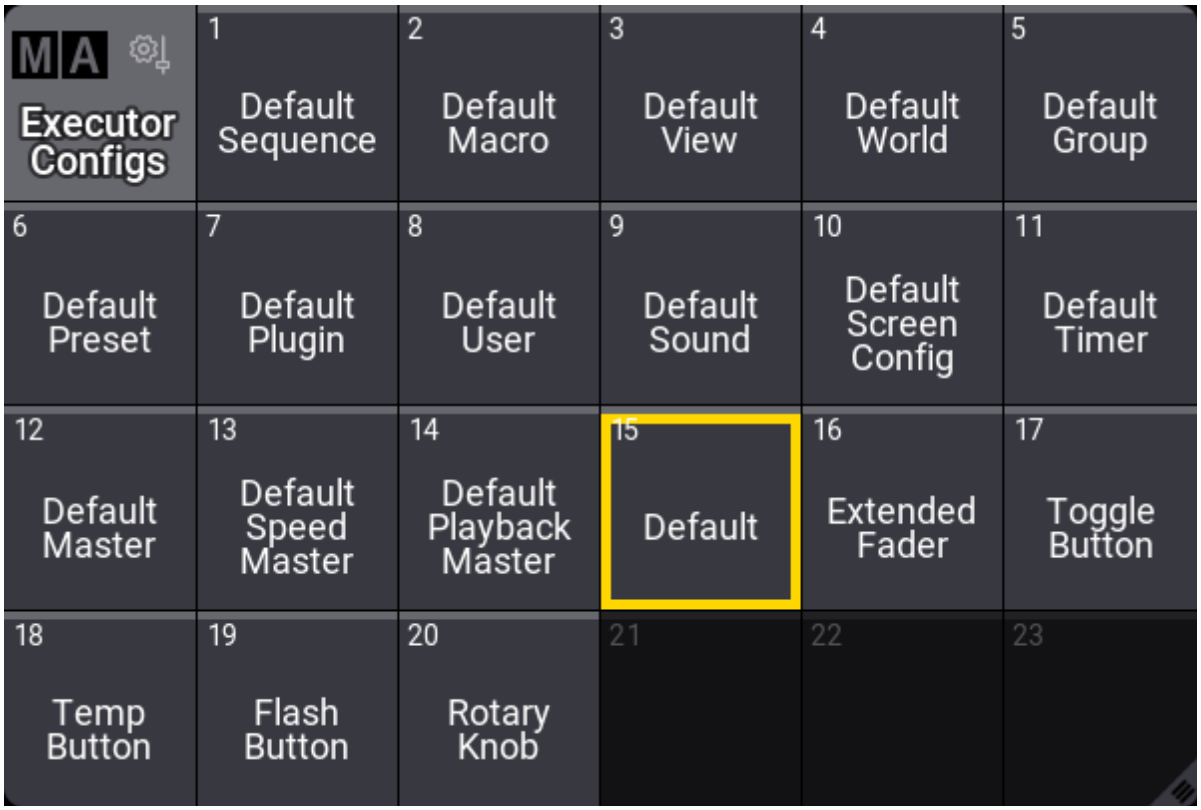
Subtopics

- **Executor Configurations**
- **Assign Object to an Executor**
- **Running Playbacks**
- **Special Executors**

1.31.1. Executor Configurations

Executor key and fader configurations can be saved and reused. Read the **Assign Object to an Executor topic** for information on changing the assignment.

The configurations are stored in an Executor Configuration pool. This can be created as any other window. Read the **Add Window topic** to learn how.



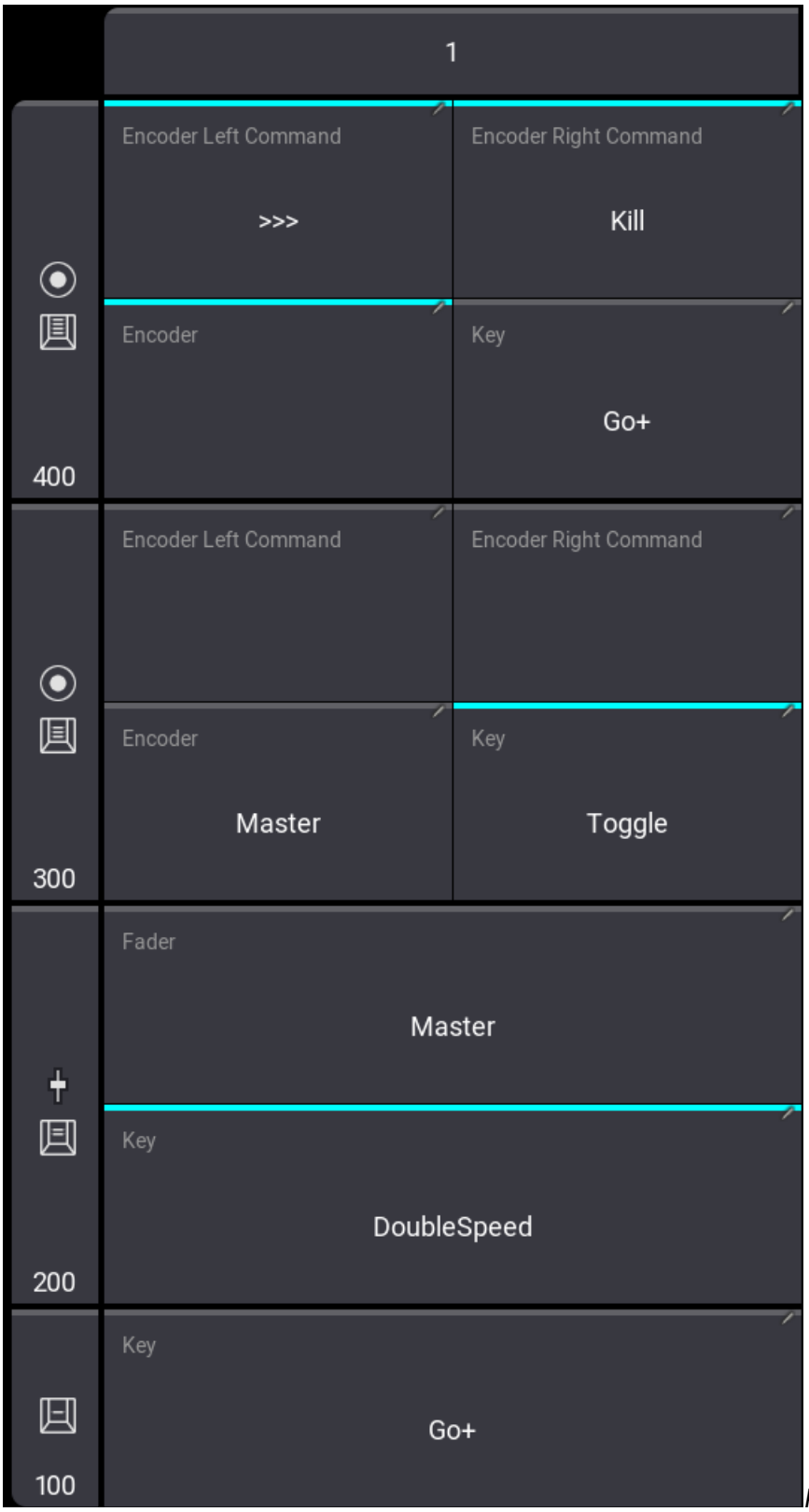
Executor Configuration pool with a selected configuration

Each object type that can be assigned to an executor has a default executor configuration used if nothing changes. This is described below.

The selected pool object is used as the default for sequences.

This configuration can be changed in the **Assign Menu.**

The Assign Menu can be opened by pressing Assign and then an executor button. The menu is used to set up executors. Learn more in the **Assign Object to an Executor topic**.



Key and Fader configurations

In the example above, there are several Keys and Encoders that have a different assignment than the original saved configuration. The different assignments are marked with a cyan color bar.

	Important:

	The encoder commands are mutually exclusive from the normal encoder function. In the example image above, in row 300 there is only an **Encoder** function and in row 400 only an **Encoder Left Command** and **Encoder Right Command.** Editing any command clears the encoder. Editing an encoder function, clears both commands.

In the title bar of the assign menu, there are three buttons: Executor Config., Load, and Save.

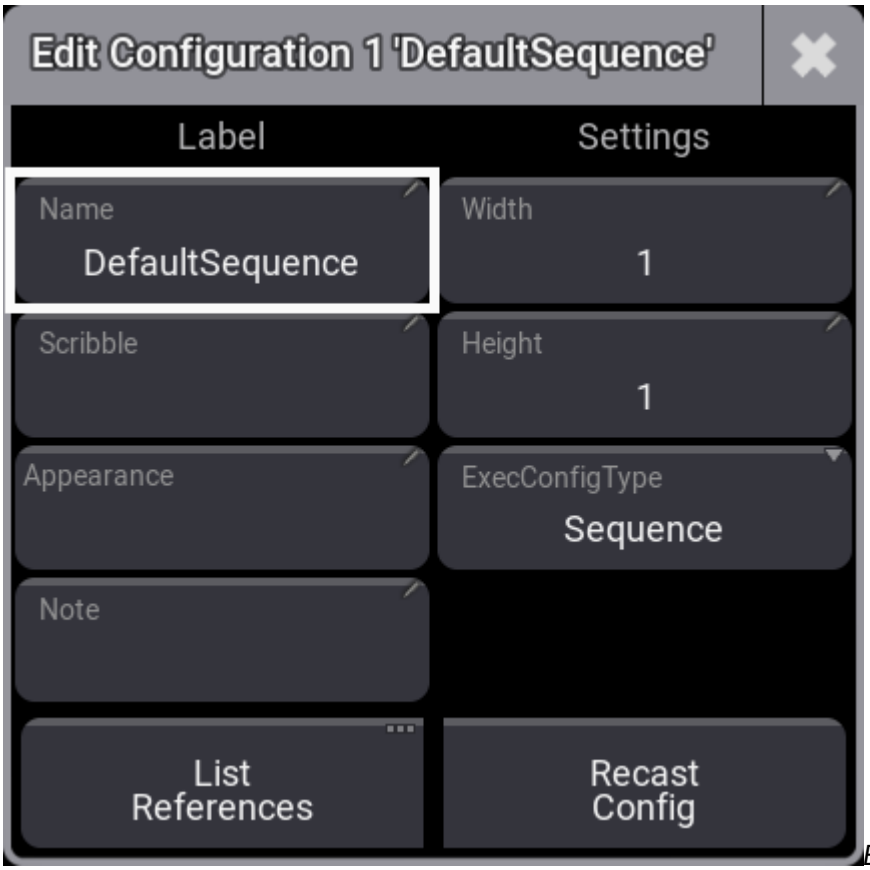
Title bar of the assign menu

Executor Config. shows the currently selected configuration, and tapping it opens a small select pop-up where any existing executor configurations can be selected or a new one can be created.

	Hint:
	Creating a new executor configuration this way will discard all changes that were made until then.

Load is used to load the selected configuration onto the keys and faders. Save can be used to save the currently assigned functions to the selected configuration.

Editing the configuration pool object opens an editor like this:



Edit configuration pop-up for the first pool object

Here it is possible to change the name of the executor configuration.

This is also where the executor configuration's width and height are set. It is not shown here, but the configuration also knows the starting row for the configuration. This means that a configuration of 1x1 can be different for each of the four rows.

The ExecConfigType defines what type of object the setting is relevant for. Tapping the button opens a small pop-up with all the possible objects.

The executor configuration can also have an appearance and scribble assigned. This is only visible in the pool.

A note can be added to the executor configuration.

Tapping List References opens an info pop-up showing the objects that reference the configuration and what other objects this configuration might depend on.

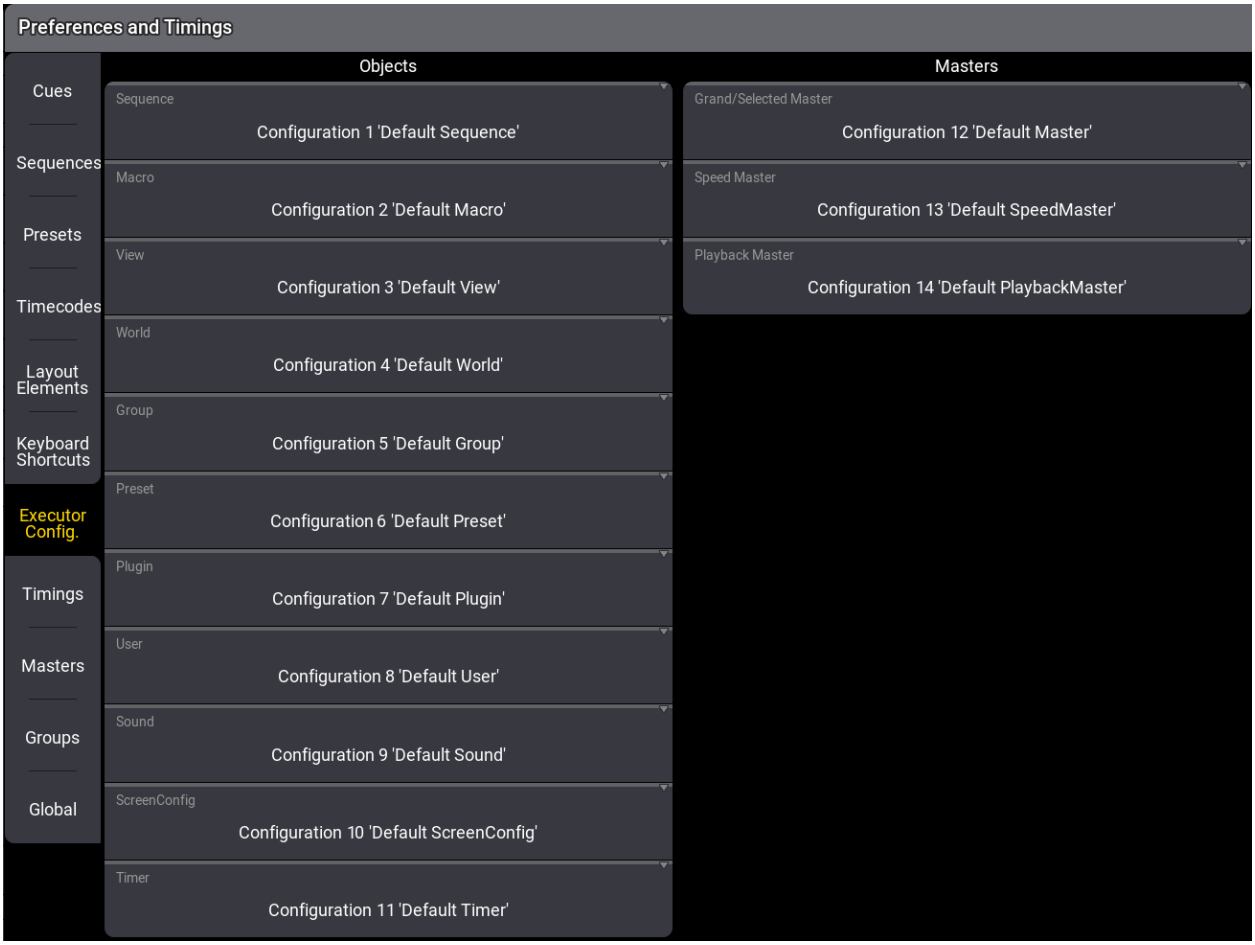
Recast Config is used when a configuration has been saved with changes, and these changes should also be applied to the other executors that use this configuration. Saving a change to a configuration does not automatically apply the changes to other executors.

Pressing Recast Config applies the changes to all the executors using the configuration. The recast can also be applied using the **Recast keyword**.

Executor Configuration Preferences

The default executor configuration for each object is stored in the Preference and Timing.

This is accessed by pressing Menu / and then Preference and Timings, and finally Executor Config.



Preferences and Timings - Executor Config menu

This menu has a setting for each object type that can be assigned to an executor. Tapping each of these settings opens a small select pop-up where any of the executor configurations can be selected as the default settings for the object type.

1.31.2. Assign Object to an Executor

Many objects can be assigned to an executor. The executor is a physical key, fader, or knob that controls the assigned object. The physical devices can also be represented as on-screen controllers.

	**Quick Steps:** 1. Press Assign. 2. Tap what should be assigned or press the relevant keys. 3. Press the executor where it should be. These are the simplest steps to assign something to an executor - read below for details.

Assigning Objects Using Keys and Pools

It is easy to assign something to an executor.

Press Assign followed by the desired object and then the executor to which it should be assigned.

Here are three variations on how it works. The examples use sequences, but it can be any of the allowed types:

Example 1

Using only the keys to assign sequence 3 at executor number 105 on the current page, type:

Assign Sequ 3 At MA + X16 | Exec 1 0 5 Please

Example 2

It is also possible to use a combination of keys and pools.

Having a pool visible on one of the screens makes it possible to combine key presses with pool selection.

1. Tap and swipe out of the sequence pool object that should be assigned.
2. Swipe to the Assign option and release the screen.
3. Press one of the keys associated with the desired executor.

Example 3

Pressing the keys puts keywords into the command line. This means it can also be typed as a command line input.

This command will assign sequence 4 to executor 301 on executor page 2. The page keyword needs to be used when addressing executors on specific pages. The page needs to exist before it can be addressed.

Assign Objects Using the Assign Menu

The **Assign** menu offers a visual approach to assigning something to the executors. Use the assign menu by selecting the executor first and then selecting the object.

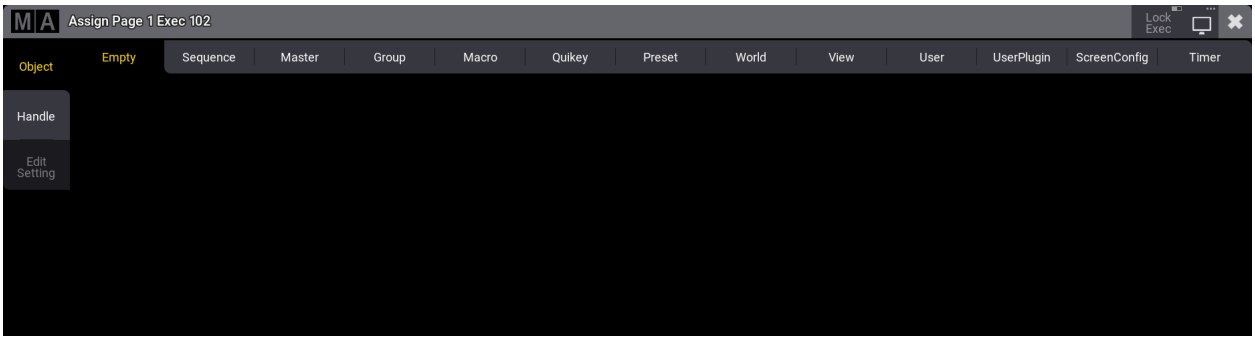
There are two main ways to open the Assign Menu.

- Press Assign followed by any of the keys associated with the desired executor.

-Or-

- Tap the executor label in the **Mini Executor Bar** (do not do a long press - it opens the editor instead).

This is the Assign menu:



Assign Menu - Object page

The title bar has a toggle button called Lock Exec. This can be used to lock the executor from changes. It does not lock the executor from playing back or performing other functions normally; the lock only prevents making changes to the executor.

This is the **Object** page of the Assign menu. This page is selected by tapping Object on the right side.

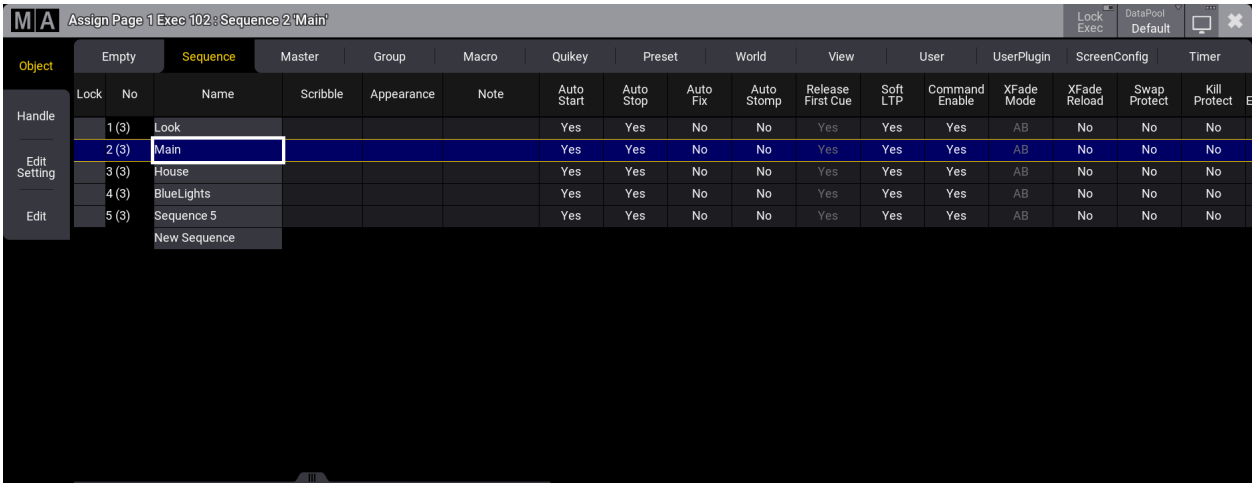
The top has several tabs. One for each object type that can be assigned to the executor and a special one used to select an **Empty** object. The other tabs open a selection list. The list will contain the possible objects of that type.

The different types are:

- **Sequence**
- **Master**
- **Group**
- **Macro**
- **Quickey**
- **Preset**
- **World**

- **View**
- **User**
- **UserPlugin**
- **ScreenConfig**
- **Timer**

Tap Sequence to open the list of possible sequences. It could look like this:



Assign menu in the sequence object tab

Each allowed type that can be assigned to an executor provides a list of the available objects. The DataPool in the title bar makes selecting an object from a different data pool easy.

Select the desired object by tapping it.

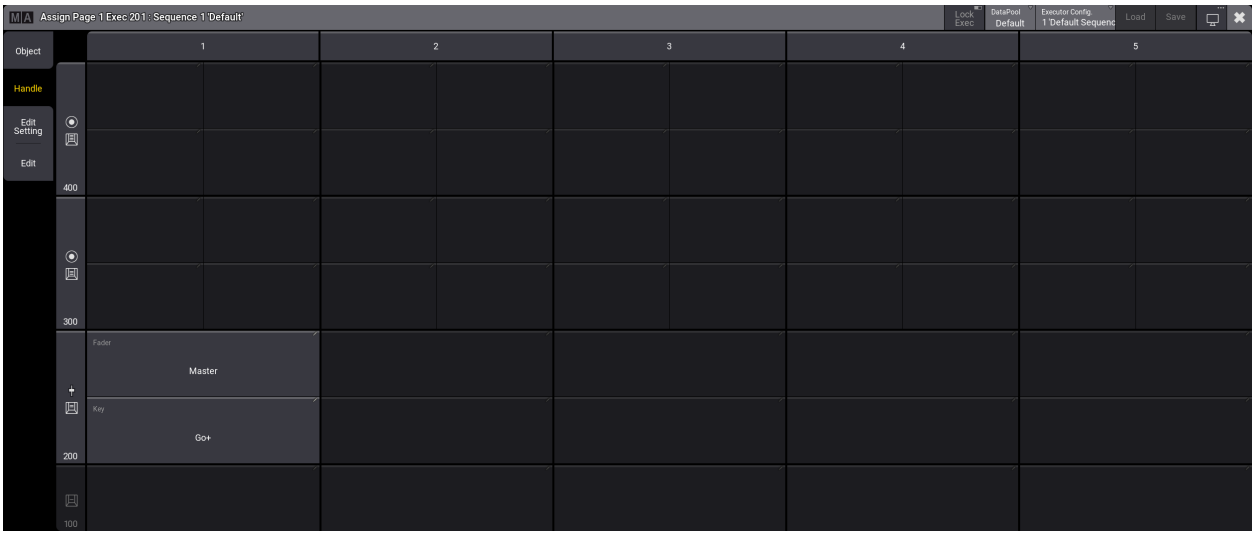
The default handle configuration is applied to the executor. This, including executor expansion, can be defined on the handle page.

Change Key Function and Executor Size

When the executor has something assigned, changing the functions assigned to the executor keys, faders, and knobs is possible.

If continuing to work in the Assign menu, tap Handle on the left side.

It could look like this:



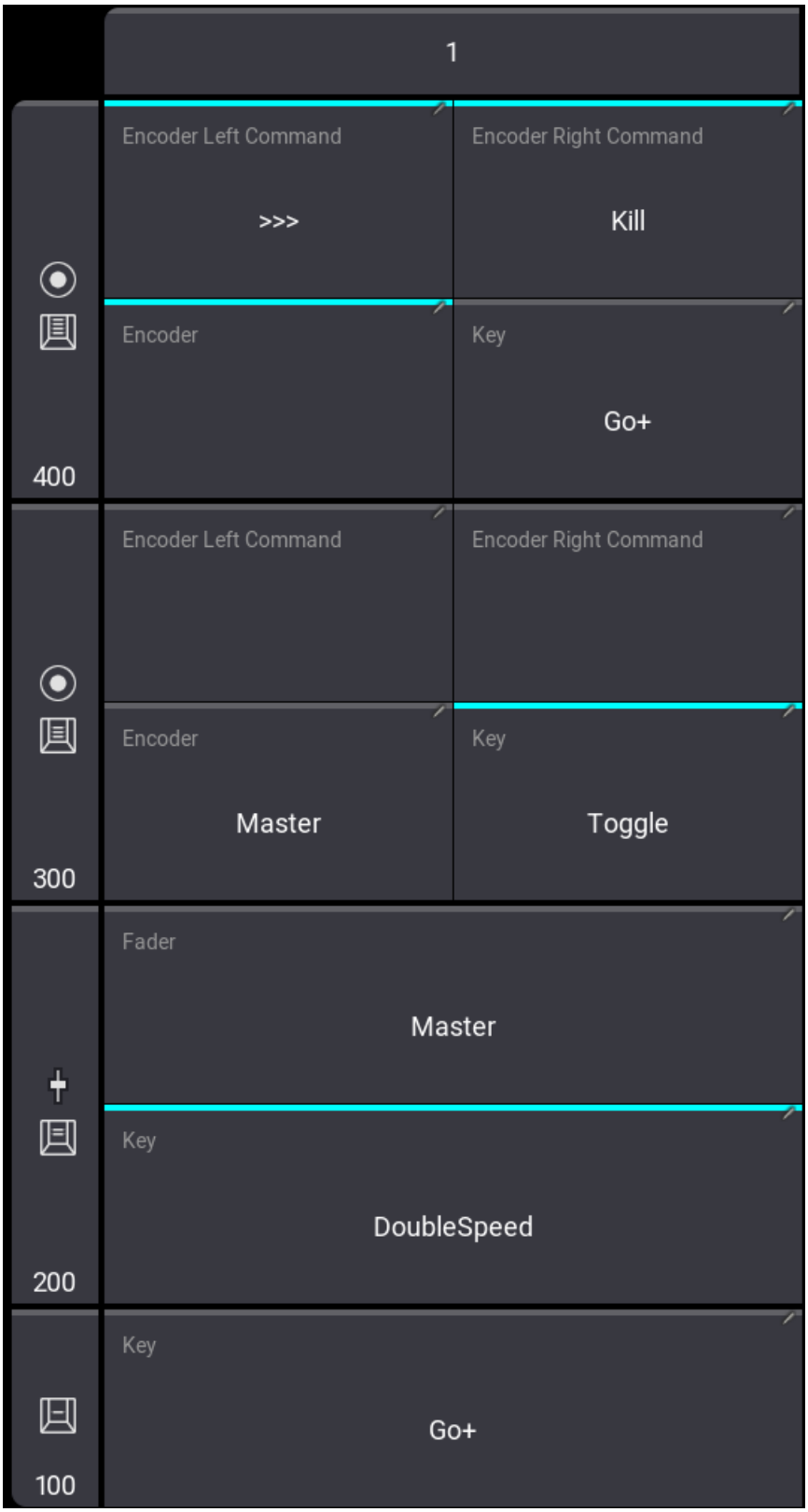
Assign menu on the Handle page

Executors can be expanded if the executors above and to the right are empty. They are grouped into sections of 5 in width, and this grouping cannot be crossed. For instance, executors 104 and 105 cannot be extended into executor 106.

They can also be extended upwards if there is available space. The example image above shows Executor 102 with space above and to the right. This executor can be extended to a width of 4 columns and a height covering all four executor rows.

It is expanded by tapping the column numbers on the top and the row numbers on the left. When it is expanded, then more buttons get available for the executor.

The 100-row only offers a single executor button. The 200-row has a button and a fader. Both the 300- and 400-rows have a button and an encoder knob. If an executor is extended to cover all four rows, then it could look like this:



Executor with all four rows in height

The keys, faders, and encoders can have different functions. The options might vary depending on the object assigned to the executor.

Tap any available button to get a small select pop-up with the available options.

The **Encoder Left Cmd** and **Encoder Right Cmd** open a pop-up that allows any command to be typed. There is even an option for different commands if the MA key is pressed while the encoder knob is turned.

	Hint:
	The encoder commands are mutually exclusive from the normal encoder function. In the example image above, in row 300 there is only an **Encoder** function and in row 400 only an **Encoder Left Command** and **Encoder Right Command.** Editing any command clears the encoder. Editing an encoder function, clears both commands.

The normal key options with their specific icons displayed in the **Playback window** are:

- **Empty**:
  The key has no function.
- **At**:
  Used with presets. Performs an **At command** on the assigned preset. Learn more about presets on executors in the **Use Presets topic**.
- **<<< (GoFastBackward)**:
  This jumps one cue back without cue timing.
- **>>> (GoFastForward)**:
  This jumps one cue forward without cue timing and will not trigger other cues.
- **Black**:
  This turns off the intensity of the executor as long as the button is pressed.
- **DoubleSpeed**:
  Pressing this doubles the resulting speed of the speed master.
- **Call**:
  Used with screen configurations. Performs a **Call command** on the assigned screen configuration.
- **Flash**:
  This turns the executor on and sets a virtual intensity master at 100, as long as you have the button pressed. When the button is released, it will return to the status before it was pressed.
- **Go+**:
  This executes a "Go". It uses fade and delay times.
- **Go-**:
  This fades backward using fade and delay times.
- **Goto**:
  This opens a pop-up where a cue can be selected. Tapping a cue in the pop-up immediately performs a **Goto command** on that cue. This means that the selected cue is now active.
- **HalfSpeed**:
  Pressing this halves the resulting speed of the speed master.
- **Kill**:
  This sequence playback action turns this sequence On and all other sequences Off. Sequences can be protected from the Kill action in the **Sequence Settings**.
- **LearnSpeed**:
  This sets the speed. By pressing it at least two times, it automatically adjusts the speed.
- **Load**:
  This opens a pop-up that lists the cues in the sequence. Tapping a cue performs a **Load command** on the cue, which will perform a go to the selected cue when the next Go command.

- **LogIn**:
  Used when a user is assigned. Performs the **LogIn command** for the assigned user.
- **Off**:
  This turns the executor Off.
- **On**:
  This turns the executor On.
- **Pause**:
  This holds an active fade and/or delay.
- **Rate1**:
  This resets rate fader.
- **Select**:
  This selects the executor.
- **SelectFixtures**:
  This selects the fixtures used in the object assigned to the executor.
- **Speed1**:
  This resets the speed fader to the default speed.
- **Swap:**
  Swap temporarily overrides the master level of executors to full and sets all other master levels to zero. Executors can be protected against the Swap in the **Sequence Settings**.
- **Time**:
  This toggles the time function for the executor. It overwrites the stored cue part times when it is On.
- **Temp**:
  This turns the executor on as long as it is pressed. The temp function uses the fade times from the cues and the level set by the intensity master.
- **Toggle**:
  This turns an active executor Off and an inactive executor On.
- **Top**:
  This fades to the first cue in an assigned sequence.

These main functions are keywords - they can be found in the **All keywords topics** with detailed descriptions of the keywords.

The functions can also be assigned using the keys or commands. Not all functions have a physical key that can be used; it can be done by writing the command using the keyboard.

Key Example

If the desired function has a physical key, assigning it to an executor key is easy. Press Assign, the function key you want, and the executor key where the function should be.

Example 1

For instance, to assign the **Off** function to the key associated with executor number 101, you need to press the following keys:

Assign Off – (the executor key in the lower-left corner on consoles)

Example 2

Assign the **Fix** function to an executor key

Assign MA + Pause [desired executor key]

Command Example

Functions can be assigned using the command line.

This is the general syntax:

Assign [Function] at [Location]

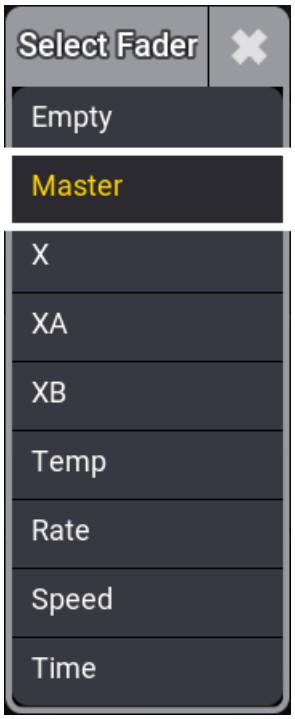
The location must be a specific physical key associated with an **Executor Page**.

To assign the Pause function to the top key associated with executor 5 on executor page 8, the following command could be typed:

Change Fader Function

It might be possible to change the function of the fader. This depends on what type of object an executor has. They are changed just as the executor keys - by tapping the onscreen representation in the assign menu (see above).

The **Select Fader** pop-up lists the possible options:



Select Fader pop-up

- **Master:**
  Controls the intensity. For more information, see **FaderMaster.**
- **X (CrossFade):**
  Crossfades between two cues. Current cue and next cue. The current cue will change when the fader reaches the other end position from where it started. For more information, see **FaderCrossFade Keyword.**
- **XA (CrossFadeA)**
  Is the first of a two-fader manual crossfade between two cues. Works along with XFadeB. The crossfade function can be changed in the **Sequence Settings**. For more information, see **FaderCrossFadeA Keyword.**
- **XB (CrossFadeB)**
  Is the second of a two-fader manual crossfade between two cues. Works along with XFadeA. The crossfade function can be changed in the **Sequence settings**. For more information, see **FaderCrossFadeB Keyword.**
- **Temp:**
  Crossfades the cue on when pulled up, and off when pulled down. For more information, see **FaderTemp Keyword.**
- **Rate:**
  Modifies the fade and delay time in a sequence by the value of the fader. If Speed from Rate is on, it is also valid for phaser speed stored in cues - see **Sequence Settings** topic. For more information, see **FaderRate Keyword.**
- **Speed:**
  Controls the phaser speed in a cue. For more information see **FaderSpeed Keyword.**
- **Time:**
  Sets the time for the executor time overwrite. For more information, see **FaderTime Keyword.**

The functions can also be assigned using the command line.

The syntax is the same as the key functions. Except for "empty" - see examples below.

Examples

Example 1

To assign the rate function to executor number 209 on the current page, you will need to type the following command:

Example 2

If you want to assign the "Empty" function to the executor fader, then it is necessary to use a different syntax. This example assigns empty to the left-most fader on page 1:

Save the Key and Fader Assignment

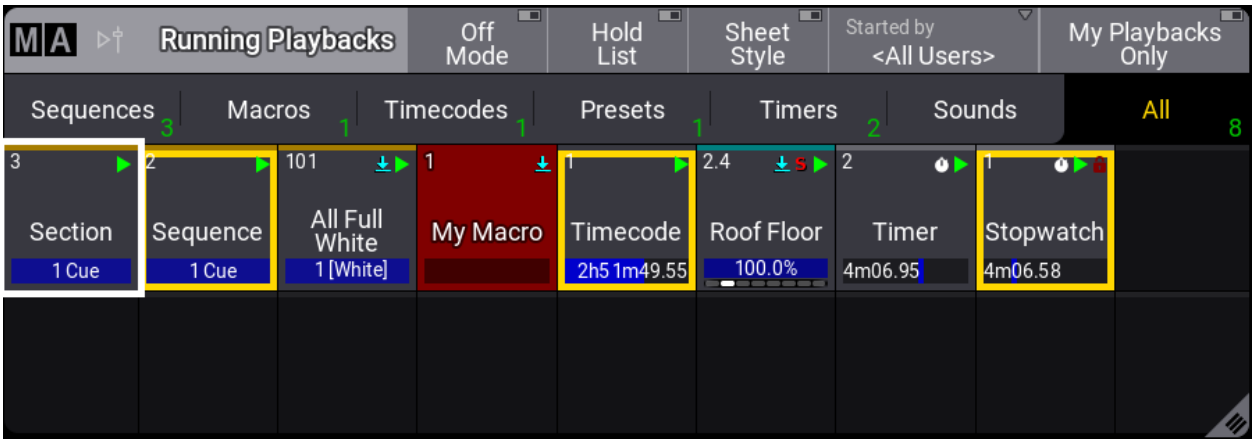
The current assignment of key and fader functions can be saved into a pool of different configurations. Read more in the **Executor Configurations topic**.

1.31.3. Running Playbacks

There are two ways to see the different running playbacks. This can be useful when trying to find out what is running and also to quickly turn off different playbacks.

Running Playback Window

It can be a window on a screen. This is called **Running Playbacks** and can be **created like any other window**. It is in the "More" tab.



Running playback window

Each running object is displayed as a pool object and can be interacted with like any other pool object. For instance, it is possible to use the **Off keyword** to turn off an object by pressing Off and then tapping the object in the window.

This window can show different types of objects: Sequences, Macros, Timecodes, Presets, Timers, SoundFiles, and Generators.

Toggle between them by tapping the Sequence, Macros, Timecodes, Presets, Timers, or SoundFiles tabs or select All to see all the different types.

The title bar has a button that can activate the "Off Mode". With this On, objects are immediately turned off when tapped in the window. Tap Off Mode to toggle between the On and Off states.

After Hold List is enabled, playbacks that are switched off will be grayed out.

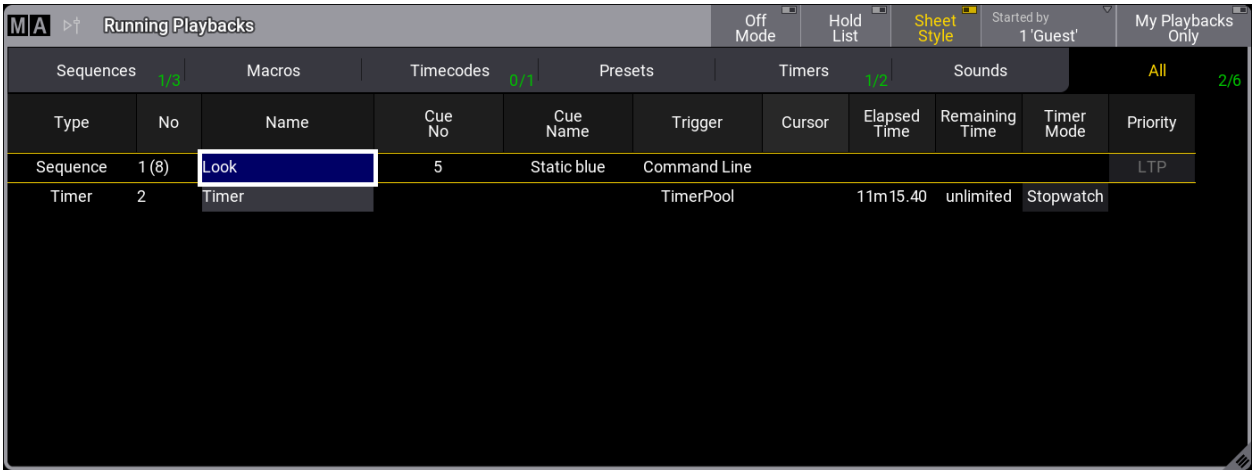
This stops the **Running Playbacks** window from constantly updating the list of running playbacks, especially when many sequences are flashing.

It is possible to filter the objects in the window to display only the playbacks started by a specific user by changing the Started by setting in the title bar or in the window settings. This can be overridden to only show playback triggered by the user currently logged in on the station, by tapping My Playbacks Only in the title bar. Disabling My Playbacks Only again resets Started by to its previous value.

The number in the lower right corner of each tab indicates how many playbacks of its object type are currently running. As soon as My Playbacks Only is active, or Started by is set to something else than **<All Users>** , or SelectedDataPool in the Window Settings is set to something else than **All DataPools,** two numbers will be displayed: **x** / **y**.

- **x**: Represents the number of running playbacks based on the made settings.
- **y**: Represents the number of all running playbacks.

The title bar also has a Sheet Style button. This turns the sheet style On or Off for this window. The default style is pool style, like the image above. Sheet style could look like this:



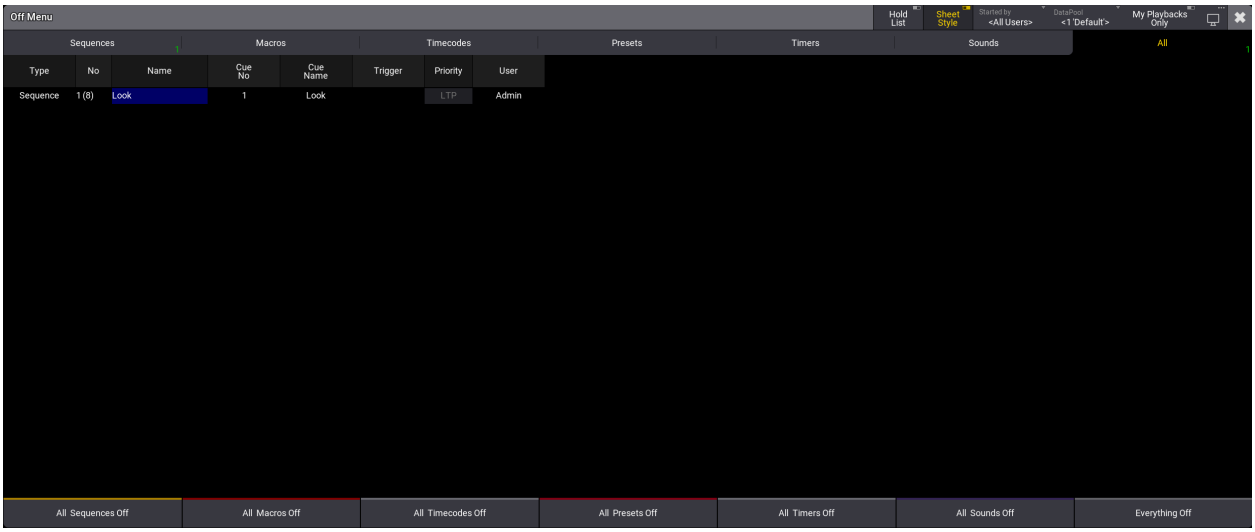
Running playbacks window in sheet style - one sequence and one timer started by Guest

This mode shows more details and displays all the extra information available on pool objects. For example, the **Trigger** column indicates which object started the playback.

Selecting a specific data pool and changing the font size are the only other options for the **Running Playbacks** window. All settings can be found in the **Running Playback Settings**, which can be opened by tapping the MA logo in the upper left corner of the title bar.

Off Menu

The other way to see running playbacks is to open a temporary version of the window. This is called the **Off Menu**. It is opened by pressing Off twice:



Off Menu in <All Users> mode

This is the same as the **Running Playbacks** window, with a few exceptions.

There are extra buttons at the bottom, and it has a permanent Off action, meaning that tapping any object instantly turns the object off.

The extra buttons at the bottom can be used to turn Off all objects of a specific type. The buttons change from **All** to **All My** if My Playbacks Only is On.

Tap Everything Off to immediately turn **all playbacks** off.

Tap (Screen Selector) in the upper right corner, to move the **Off menu** to a different screen.

The menu can be closed like any other pop-up by tapping the in the upper right corner.

1.31.4. Special Executors

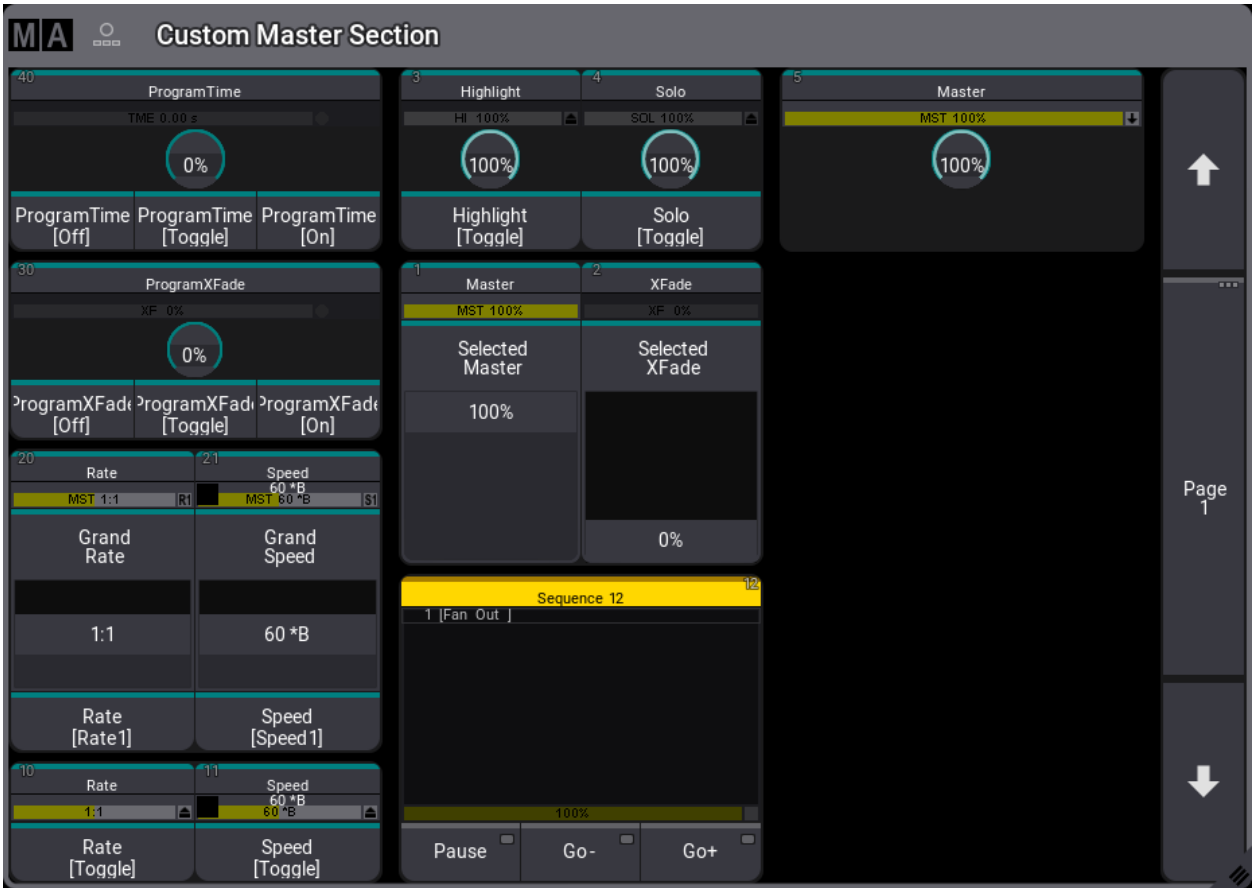
Special executors include the grand master knob as well as the keys, knobs, faders, encoders, and wheels found in the custom area and master area. By default, they are assigned to control a useful assortment of masters. As with other executors, changing the assignment of special executors allows control of many types of objects, including sequences, masters, plugins, groups, macros, worlds, and views.

The different areas containing special executors are always displayed to the right of the relevant letterbox screens corresponding to the hardware below the screens.

The **Custom Master Section** window displays the current assignment and status of the special executors in the two custom areas, the master area and the grand master knob. This window can also include on-screen versions of the special executor encoders, wheels, knobs, and keys; as well as the Go+, Go-, and Pause keys for the selected sequence.

Tap the icon in the control bar to open the **Custom Master Section** temporarily.

The **Custom Master Section** window is available under the More and All tabs in the **Add Window** pop-up.



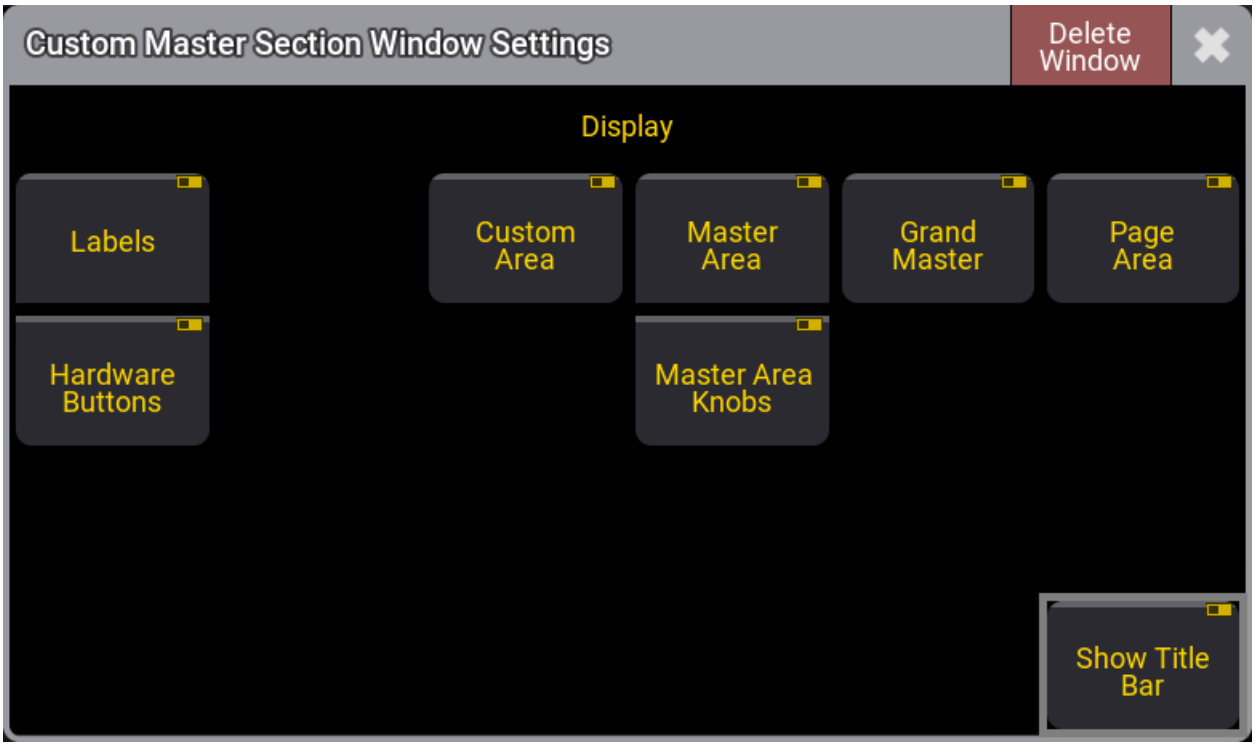
Custom/Master Section window

The Custom/Master Section window contains three main areas:

- Custom area encoders and wheels are displayed on the left side of the window when all sections are visible.
- Master area, including the Go+, Go-, and Pause buttons for the selected executor, is displayed in the center of the window when all areas are visible.
- Grand master area, displayed on the right side of the window when all sections are visible.

These three main areas can include labels and a visualization of the associated hardware. When both labels and hardware are visible, the label for each special master appears directly above the associated hardware. In addition, page navigation buttons can be shown along the window's right edge.

Tap MA in the upper left corner of the window to open the Window Settings pop-up.



Custom/Master Section Window Settings pop-up

Tap Labels to show or hide the labels for all visible sections. Tap Hardware Buttons to show or hide the hardware visualizations for all visible sections. Tap Custom Area, Master Area, Grand Master, Page Area, and Master Area Knobs to show or hide the corresponding area of special executors or page navigation controls. Tap Show Title Bar to show or hide the title bar of the window.

	Hint:
	The Hardware Buttons and Master Area settings must all be enabled in order for the **Default Playback** buttons to be visible.
	Hint:
	As elements are hidden, the remaining visible elements automatically rescale to fill the available space within the window.

Tap the label of the **Default Playback** area to open the **Edit Sequence** for the selected sequence.

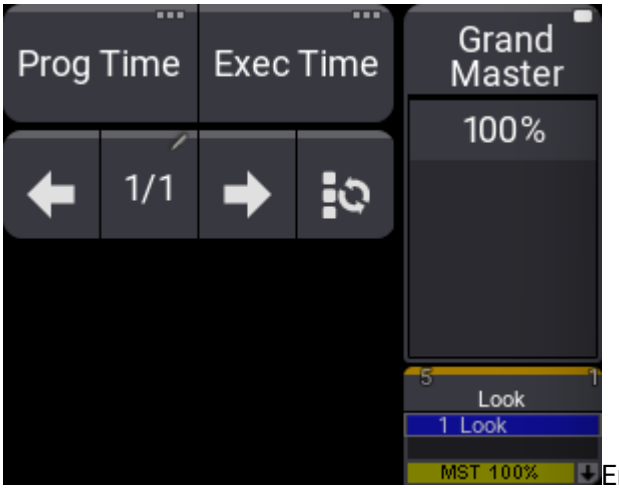
Tap the label of any special executor to open the **Assign Menu** for that special executor. The options available in the **Assign Menu** for a special executor are the same as the options for any other executor. For more information on the assignment process, see the **Assign Object to an Executor** topic.

Special Executor 5 and the Grand Master

By default, the grand master is assigned to special executor 5. Press Assign, then press the special executor 5 knob to open the **Assign Menu** for special executor 5. Use the **Assign Menu** to assign any desired object to special executor 5.

	Hint:
	Since the hardware associated with special executor 5 consists of only a knob with no additional key, it is possible to assign a normal key function to the press of the knob.

When any object other than the grand master is assigned to special executor 5, the area of the encoder bar that normally shows the grand master adjusts to show both the grand master and a label for the object assigned to special master 5.



Encoder bar showing both sequence 1, assigned to special master 5, and the grand master.

1.32. Masters

Masters are physical representations of various timing and level overrides that exist in the software.

Masters can be assigned to fader executors or any of the special executors in the **Master section**, **Custom section**, or the **Grand Master**.

They can all be assigned using the **Assign Menu**. Tap Object in the top-left corner, then Master at the top of the menu to reveal a tree structure separated into the five groups of Masters.

They can also be assigned and adjusted using the **Master keyword**.

For more information on assigning objects to executors, see the **Assign Object to an Executor topic**.

Quick access to all of the selected masters and grand masters is available in the master controls menu. For more information on this menu, see the **Master Controls topic**.

Masters have a limited selection of functions available as key assignments.

Please read about the available masters in the following subtopics.

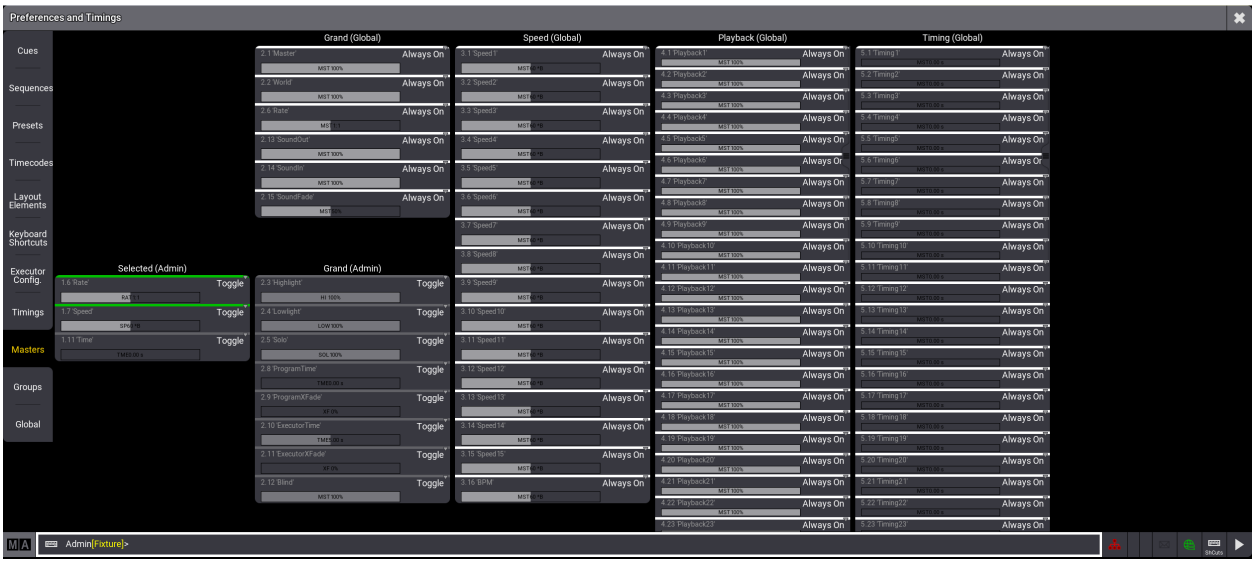
Master Modes

Masters have a mode that defines how they operate.

There are three available modes:

- **Toggle**:
  Switch the master On or Off. The fader value of the master can always be changed and affects the output when the master is On. The indicator colors are gray (Off) and green (On).
  Masters that are set to toggle can be executed with the button function Temp when they are off. They will then be temp'ed to their current master fader level. For more information about Temp, see **Temp Keyword**.
- **Always On**:
  The master is always On and cannot be switched Off. The fader value of the master can always be changed and affects the output. The indicator color for Always On is white.
- **Disabled**:
  The master is switched Off and its fader value internally uses its default value. For example, the grand master uses 100% and the program time uses 0s. The indicator color for Disabled is red.

The modes of all masters can be changed in Menu - Preferences and Timings - Masters.



Masters settings in Preference and Timings menu

This menu displays the masters separated by their type and their membership in the show file.

Masters can have a global membership which means that these masters are controlled by all users of the show file. Masters with a user membership are located within the user profile which means that every user profile has its own individual set of these masters.

Changing one of these masters does not affect the same master in a different user profile.

The section of masters that are controlled by the user profile displays the name of the user behind their type, for example, Selected (Admin) for the masters that control the selected sequence of the Admin user.

Masters of the global membership display (Global) behind their type, for example, Playback (Global).

The mode can be changed by tapping the current mode of the master. This toggles through the three modes.

This menu also shows the current value of each master.

Subtopics

- **Selected Masters**
- **Grand Masters**
- **Time Control**
- **Speed Masters**
- **Playback Masters**
- **Timing Masters**

1.32.1. Selected Masters

Selected masters give access to the individual masters of the selected sequence. They adjust the levels and timing of the selected sequence.

Selected masters can be **assigned** to executors and special executors.

Read more about the functions of the masters and assigning masters to executors in the **Assign Object to an Executor** topic, especially the **Change Fader Function** section.

The selected masters are:

- **Master 1.1 'Master'**
  Controls the intensity.
- **Master 1.2 'XFade'** - this is also called Crossfade.
  Crossfades between the current cue and the next cue.
- **Master 1.3 'XFadeA'** - this is also called CrossfadeA or XA.
  Is the first of a manual crossfade between two cues. Works along with XFadeB.
- **Master 1.4 'XFadeB'** - this is also called CrossfadeB or XB.
  Is the second of a manual crossfade between two cues. Works along with XFadeA.
- **Master 1.5 'Temp'**
  Crossfades the cue on when pulled up, and off when pulled down.
- **Master 1.6 'Rate'**
  Modifies the fade and delay time in a sequence by the value of the rate master. If Speed from Rate is on, it is also valid for phaser speed stored in cues - see **Sequence Settings topic**.
- **Master 1.7 'Speed'**
  Controls the phaser speed in a cue.
- **Master 1.8 'Highlight'**
- **Master 1.9 'Lowlight'**
- **Master 1.10 'Solo'**
- **Master 1.11 'Time'**
  Sets the time for the time overwrite. For more information on time, see the **Time Keyword**.

Hint:

The Highlight, Lowlight, and Solo masters are only functional for the grand masters. For more information, see **Grand Masters**.

1.32.2. Grand Masters

The Grand Masters are a selection of 14 different masters that can be assigned to executors and special executors.

They are:

- **Master 2.1 'Master'**
  This is the grand master. This has a dedicated virtual fader, but can also be assigned to any executor or special executor.
- **Master 2.2 'World'**
  Functions like a grand master but only for fixtures included in the selected world. For more information on worlds, see the **Worlds and Filters topic**.
- **Master 2.3 'Highlight'**
  Inhibits the effect of the highlight function on selected fixtures. It offers proportional control over dimmer as well as all other highlighted attributes. It also works as an inhibitor for the lowlight function. For more information on highlight, see the **Highlight keyword topic**.
- **Master 2.4 'Lowlight'**
  Inhibits the effect of the lowlight function on highlighted fixtures. It offers proportional control over all lowlighted attributes. For more information on lowlight, see the **Lowlight keyword topic**.
- **Master 2.5 'Solo'**
  Inhibits the effect of the solo function on selected fixtures. It offers proportional control over dimmer attributes. For more information on solo, see the **Solo keyword topic**.
- **Master 2.6 'Rate'**
  Modifies the fade and delay time in sequences. For more information on rate, see the **FaderRate Keyword topic**.
- **Master 2.8 'ProgramTime'**
  Modifies the fade time used by the programmer. For more information on program time, see the **Time Control topic**.
- **Master 2.9 'ProgramXFade'**
  Crossfades new programmer values to the output. It affects all programmer values changed after ProgramXFade is activated.
- **Master 2.10 'ExecutorTime'**
  Overrides stored cue timing. For more information on exec time, see the **Time Control topic**.
- **Master 2.11 'ExecutorXFade'**
  Crossfades one or multiple executors. It is like a crossfade for a single executor but affects all executors triggered after the ExecutorXFade is activated.
- **Master 2.12 'Blind'**
  Inhibits the effect of the blind function. It offers proportional control over dimmer as well as all other attributes. For more information on blind, see the **Blind keyword topic**.
- **Master 2.13 'SoundOut'**
  Volume of the console's audio output.
- **Master 2.14 'SoundIn'**
  Inhibits incoming audio signals. For more information about using sound input, see the **Sound Window** topic.

- **Master 2.15 'SoundFade'**
  Adjusts how quickly the console responds to changes in incoming audio signals. For more information about using sound input, see the **Sound Window** topic.

1.32.3. Time Control

When enabled, the Program Time and Executor Time masters can override playback timing within the programmer and executors, respectively, with a time range of 0 seconds to 10 seconds. They can be assigned to executors as well as special executors in the **Master area**, **Custom area**, or the **Grand Master**. This gives physical control of their activation and level. Quick access to these masters is available in the master controls menu. For more information on this menu, see the **Master Controls topic**.

Program Time

Program Time adds a fade time to changes of attribute values in the programmer. Set the master to the desired value and activate Program Time by pressing On or Toggle on an executor assigned to the program time master, or tap Prog Time in the master controls menu so there is a green marker. The set time will be used for all value changes in the programmer except for those made by turning an encoder or the dimmer wheel.

To deactivate program time, press Off or Toggle on an executor assigned to the program time master, or tap Prog Time again in the master controls menu. The green marker will disappear.

	**Hint:**
	Recalling presets that include individual fade time and delay time will continue to use their stored times. Presets with no individual fade time will respect the program time setting.

Executor Time

Executor Time overrides cue fade times during sequence playback, regardless of whether the sequence is running from an executor or directly from the sequence pool. Set the master to the desired value and activate Executor Time by pressing On or Toggle on an executor assigned to the executor time master, or tap Executor Time in the master controls menu so there is a green marker. Cues stored with only basic timing will change values using the specified executor time. Individual fade and delay and times will continue to playback as stored. Cues with follow or time triggers will continue to use the stored triggers.

To deactivate executor time, press Off or Toggle on an executor assigned to the executor time master, or tap Executor Time again in the master controls menu. The green marker will disappear.

If the Use Executor Time option in the sequence settings menu is enabled, the sequence will follow the executor time master. Disabling Use Executor Time allows all of the cues in the sequence playback using their stored timing. For more information on the sequence settings menu, see the **Sequence Settings** topic.

1.32.4. Speed Masters

15 different Speed Masters can be assigned to executors and special executors.

This group of masters also includes the BPM Master. This master adjusts the expected speed of the incoming audio. The console will adjust to the actual detected Beats Per Minute closest to this setting. For more information about using sound input, see the **Sound Window** topic.

Speed masters can be assigned to sequences. This makes it possible to sync multiple sequences to the same speed.

Read more about assigning speed masters to executors in the **Assign Object to an Executor** topic. Read about assigning a Speed Master to a sequence in the **Sequence Settings** topic.

The colored indicator bar above the speed master label blinks in accordance with the speed. This gives visual feedback about the current speed.

1.32.5. Playback Masters

50 different Playback Masters can be assigned to executors and special executors.

Playback Masters can be assigned as playback settings of sequences. The masters function as an inhibitive dimmer control for the assigned sequences. All sequences with the same playback master assigned can be simultaneously inhibited by that master.

Read more about assigning playback masters to executors in the **Assign Object to an Executor** topic.
Read about assigning a playback master to a sequence in the **Sequence Settings** topic.

1.32.6. Timing Masters

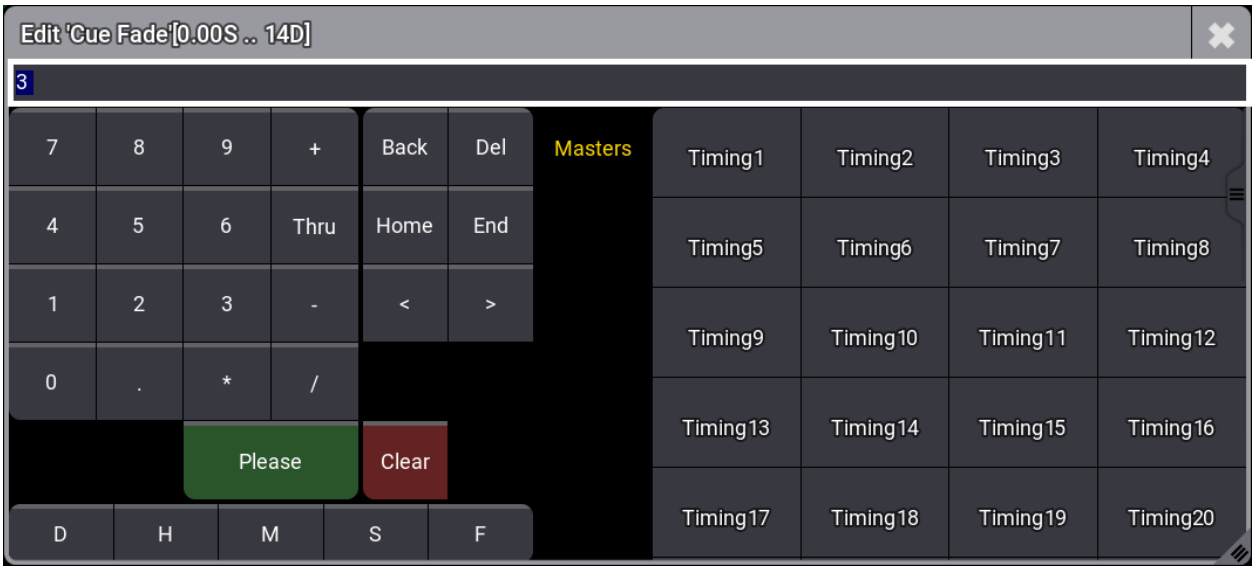
50 different Timing Masters can be set for playback timings instead of a numeric value. They can also be used for timings in combination with **playback commands**.

The range of values for timing masters is between 0 seconds and 10 seconds.

The following properties can be set to a timing master:

- **Timings in sequences**, except for Command Delay, Snap Delay and Trig Time
- Playback Timings in Preferences and Timings
  To show and edit these playback timings, press Menu – Preferences and Timings – Timings

To set a timing master, edit the value of one of these properties and choose a timing master in the **calculator**.



Calculator Cue Fade with timing masters

To set different timing masters for in and out timings when **Condensed Timing** is enabled in the sequence sheet, type manually into the Cue Fade or Cue Delay calculators. For example, Timing1/Timing2 (no spaces between name and number).

To use a timing master in combination with **playback commands**, type the timing master instead of a numeric value.

Example:

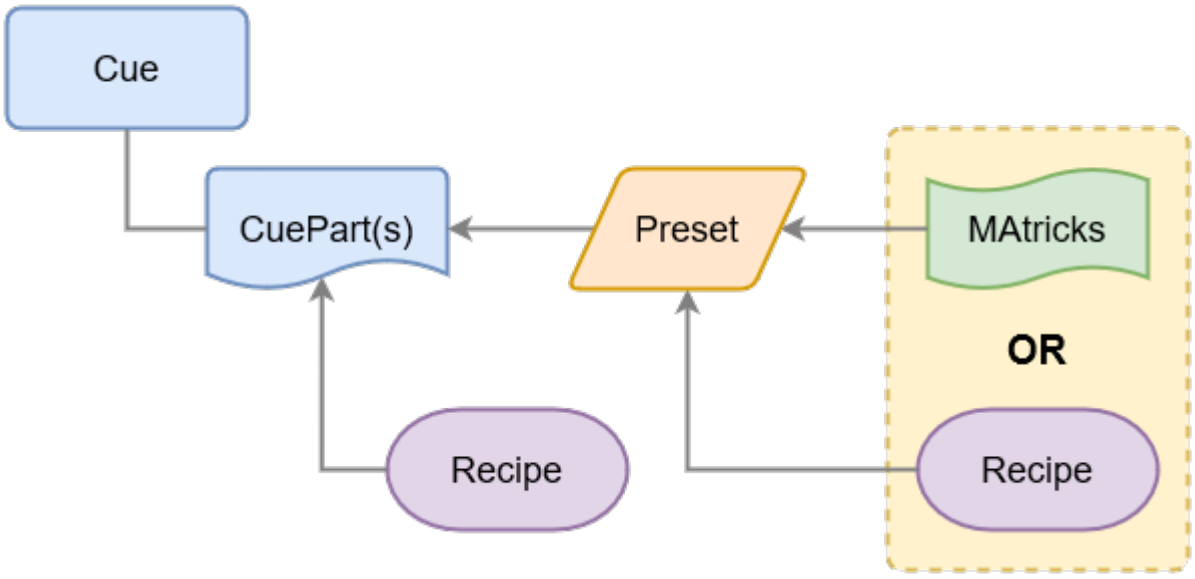
```
MA  User name[Fixture]>Goto Cue 5 Fade "Timing5"
```

Timing masters can be assigned to executors. Read more about assigning masters to executors in the **Assign Object to an Executor** topic.

1.33. Recipes

Recipes can be a very useful tool for touring shows or when the changes are expected.

Recipes can be stored in cue parts and presets.



Recipe data flowchart

There is a possibility to create some conflicting data using recipes and stored values in both cue parts and in presets.

Values can be stored in a cue part. Cue parts can also contain recipe data.

The cue part recipe needs to be cooked into the cue part. If this is cooked with the "Merge" option then existing values in the cue part are not overwritten by the recipe. If the "Overwrite" option is selected, then all existing values are overwritten by the recipe - values that are not affected by the recipe are removed from the overwritten cue.

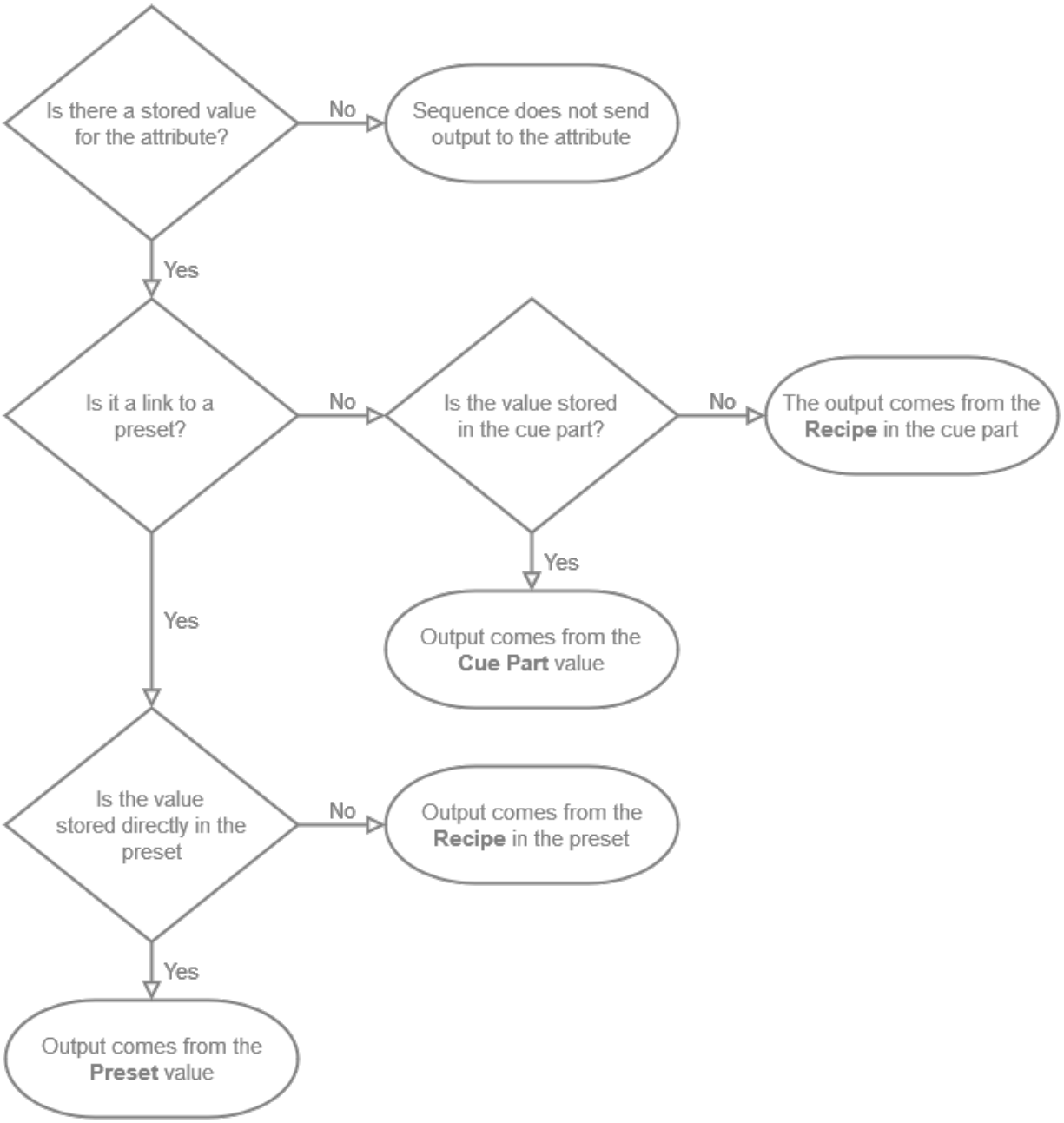
Preset recipes are automatically cooked into the preset, and a preset can be used in a cue part.

All this means that an attribute can potentially get a value from four different locations, but only one of these values can be output:

- The cue part
- The cue part recipe
- The preset
- The preset recipe

Values stored in the cue part have higher priority than the cue part recipe value. Values stored in the preset have a higher priority than the preset recipe values. Storing a preset to a cue part creates the preset reference to the preset so in that case the output is decided by the preset priority.

This flow chart can be used to determine where the output comes from in a sequence:



Output decision flowchart

A recipe contains one or multiple recipe lines describing what should happen based on a set of information. The recipe can "cook" values into the cue part, preset, or programmer. This cooked data is marked by a small pot icon and the cooked data can easily be removed again if needed.

A recipe line can contain information about a selection of fixtures, group, preset, MAtricks, individual fade, delay, speed, and phase values.

Values from recipes can be combined with conventionally stored values.

The flexibility in the recipe system allows for a variety of uses. The recipes could be used to:

- Create recipe presets for groups referencing other presets for a flexible fixture setup.
- Create template presets with ranged values that can be applied to a flexible selection of fixtures.
- Cues that contain a recipe on how different elements create the desired look.

If a recipe is present in a cue or preset, then there is a small pot icon. It can look different depending on the recipe status:

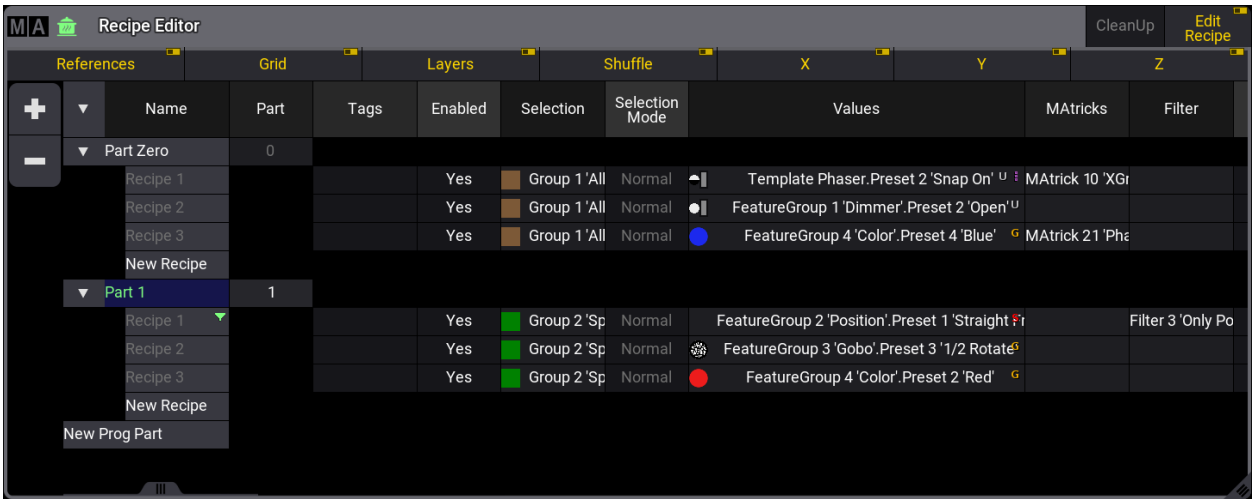
- Green pot: All recipe lines are valid.
- Red pot: One or more recipe lines cannot be cooked.
- Open pot: This is a Recipe Template.

The grandMA3 software offers multiple tools and methods to create and edit recipes. Read the **Cue Recipe** and **Preset Recipe** topics for details on how to create and edit recipes directly in cues and presets. Additionally recipes can be created with the programmer and the Recipe Editor window in Edit Recipe mode.

Recipe Editor Window

The Recipe Editor window is useful for working with **Programmer Parts** and offers a visualization of recipe lines. All recipe lines can be edited.

The Recipe Editor window can be found in the **Add Window** pop-up in Tools - Recipe Editor.



Recipe Editor window with Edit Recipe enabled and Part 1 selected

To enable the edit recipe mode, tap Edit Recipe in the title bar.

If a Group has multiple recipe lines with different presets for the same attribute, only the last entry will generate output. To delete all recipes that use the same selection with multiple presets of the same feature group, tap CleanUp in the title bar. Only the last object that generates the output is kept. For example, when Group 1 + Red + Yellow + Blue is selected and then CleanUp is tapped, the recipe lines for Group 1 + Red + Yellow will be deleted. If you tap an already selected preset again to deselect it, the corresponding recipe line will also be deleted. CleanUp is only active when the same fixtures are used

with multiple items of one feature group. Otherwise the button will be greyed out. With Edit Recipe disabled this button is not visible.

+ and - on the left side can add or delete Programmer Parts and Recipes.

Tap the **Name** column to select it.

It depends on the destination whether recipes of the different parts are stored or not:

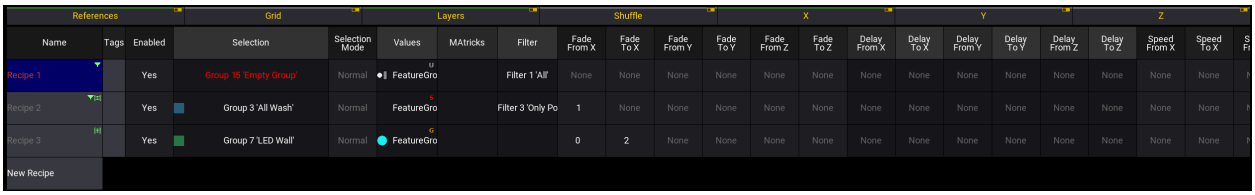
- If cue parts are the destination for storing recipes, recipe lines will be stored in the selected part, which is displayed in green.
- If cues are the destination for storing, all parts will be stored as respective cue parts.
- If presets are the destination for storing, the selected part which is displayed in green will be stored into the preset.

As soon as recipes are stored they will be cleared from the Recipe Editor.

Recipe Sheet

The recipe sheet with its multiple columns is part of recipes in cue parts, presets and the Recipe Editor window (programmer). Besides minor differences, the recipe sheet covers the same information in all three places.

This is an example of a recipe sheet:



Recipe sheet with three lines

A top bar with toggle buttons can show or hide different elements of the recipe.

- **Name:** The Name column can also be used to give the recipe line a name. If the name text is red, that indicates that the line cannot be cooked. For example, if the values are not valid for the selection or if crucial ingredients are missing. MAtricks values that do not come from a pool object, Worlds, or Filters, are indicated by small green icons (, , or) on the right in the Name cell.
- **Tags:** Displays assigned Tags. Tap and hold to edit the cell and assign or unassign tags.
- **Enabled:** If set to **Yes** recipe lines will be cooked. With **No**, the recipe line is marked red and will not be cooked.
- **References**:
  These are the columns for referenced data.
  - **Selection**:
    This is the selection of fixtures using this recipe line. If the selection is a group, then the number and the name of the group is displayed. If the group is empty, the text is red. If the selection is from the programmer, then it says **<Recipe>**.

- **Selection Mode**:
  This defines how subfixtures are handled when only main fixtures are part of the selection of the recipe:
  - **Normal**: Values are passed down to the subfixtures.
  - **Strict**: Values are strictly applied to only the fixtures in the selection.

Hint:

When migrating show files from v1.9.7.0. or earlier, the **Selection Mode** setting will be set to **Strict** in existing recipes.

- **Values**:
  This is the value reference used in the recipe line. If the value is a preset, then the number and the name of the preset and many of the different preset icons indicating Preset Mode, MAtricks, MultiStep, Layer information, and so on are shown. If the value is from programmer values, then it says **<Recipe>**. If the fixtures of the selected group are only partly used, the text in the values column is displayed in orange. This happens for example if not all fixtures of the selection in use can use the selected preset or a world is added to a recipe line. If it is red, it is not compatible to fixtures in the Selection column.
- **MAtricks**:
  This is a reference to an existing MAtricks pool object. Having a reference to an existing MAtricks adds referenced values in the Grid columns.
- **Filter**:
  This makes it possible to assign a filter or world to the recipe line.

- **Grid**:
  These columns are the same as known from **MAtricks**. There is a set of columns for X, Y, and Z axes in the grid. The columns can be filtered by activating one or several of the **X**, **Y**, and **Z** in the top bar. This section also has columns for invert. Some of these are linked to X, Y, and Z. Some are not linked to an axis but a common setting:
  - **X:**
    Edit the value **X** of the MAtricks Editor.
  - **Y:**
    Edit the value **Y** of the MAtricks Editor.
  - **Z:**
    Edit the value **Z** of the MAtricks Editor.
  - **Group**:
    The number of groups the selection is split into.
  - **Block**:
    The number of fixtures blocked together.
  - **Wings**:
    The number of wings the selection is split into.
  - **Width**:
    This changes the width of the selection in the **Selection Grid**.
  - **InvertStyle**:
    This defines if Invert is applied to Pan, Tilt, Pan and Tilt, or All attributes.
  - **PhaserTransform**:
    Transform can be set to Mirror. This mirrors values depending on the other grid settings, for example, Blocks, Groups, Wings. Learn more in the **Transform topic**.
  - **Invert, Inv, InvB, InvG, InvW**:
    These are Yes/No inverting settings for each of the grid axes.

- **Layers**:
  These are the timing layers. X, Y, or Z also needs to be active for any of the layer columns to be shown.
    - **Fade From / Fade To**:
      The two fade values allow spreading the fade time over a range.
    - **Delay From / Delay To**:
      The two delay values allow spreading the delay time over a range.
    - **Speed From / Speed To**:
      The two speed values allow spreading the speed values over a range.
    - **Phase From / Phase To**:
      The two phase values allow spreading the phase values over a range.
- **Shuffle**:
  These are the shuffle columns. X, Y, or Z also needs to be active for any of the layer columns to be shown.
    - **Shuffle**:
      This value can be set to shuffle the selection order.
    - **Shift**:
      This value can be set to shift the selection in the selection grid.
- **X**, **Y**, and **Z**:
  Each of the grid axes has columns. If the recipe only uses one or two axes, then the columns can be limited to only show the relevant columns.

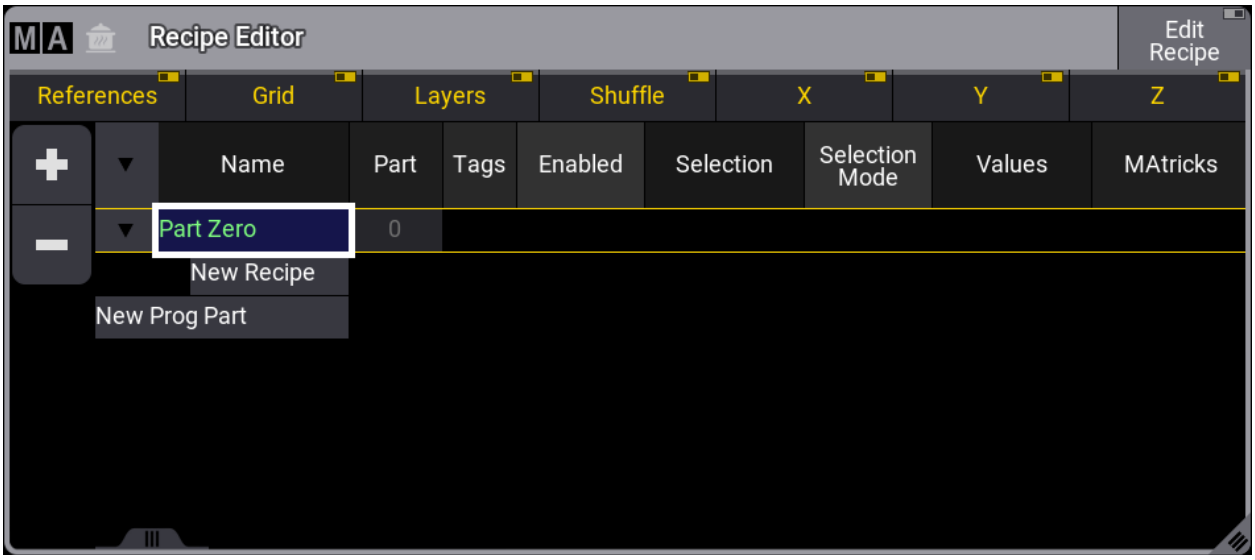
	**Hint:**
	The MAtricks and Shuffle settings are described in detail in the **MAtricks and Shuffle section**.

Recipes do not need information in all columns, but they can have information in all. They often only have information in a few columns.

Create Recipes with the Edit Recipe Mode

Besides handling recipes in cues parts and presets, recipes can also be created and edited within the programmer. This mode is based on a **programmer workflow** for quickly and efficiently creating recipes. All pool item selections are logged, saved in a recipe, and then can be easily stored into presets and cues.

The Recipe Editor window is a handy tool to visualize and edit recipes while in edit recipe mode. For more information on the Recipe Editor window, see **above**.



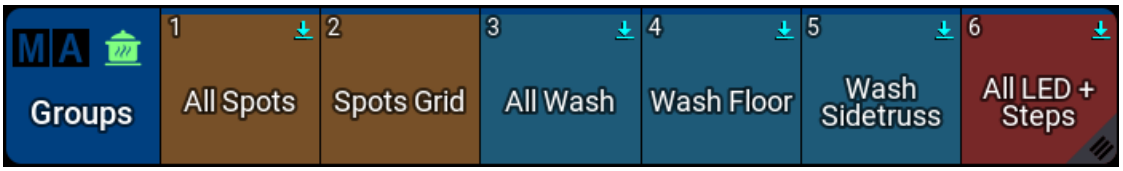
Recipe Editor window

There are multiple ways to enable the edit recipe mode:

- Tap Edit Recipe in the title bar of the recipe editor window.
- Enable Edit Recipes in the **At Overlay.**
- Use the **EditRecipe** keyword.

While the edit recipe mode is enabled, Edit and Esc flash alternately. To disable the edit recipe mode, press Esc or use one of the options described above to enable it. Once the edit recipe mode is disabled, all pools work as usual.

Each object type that can be used for a recipe will have its pool window marked with a green , when edit recipe mode is enabled:



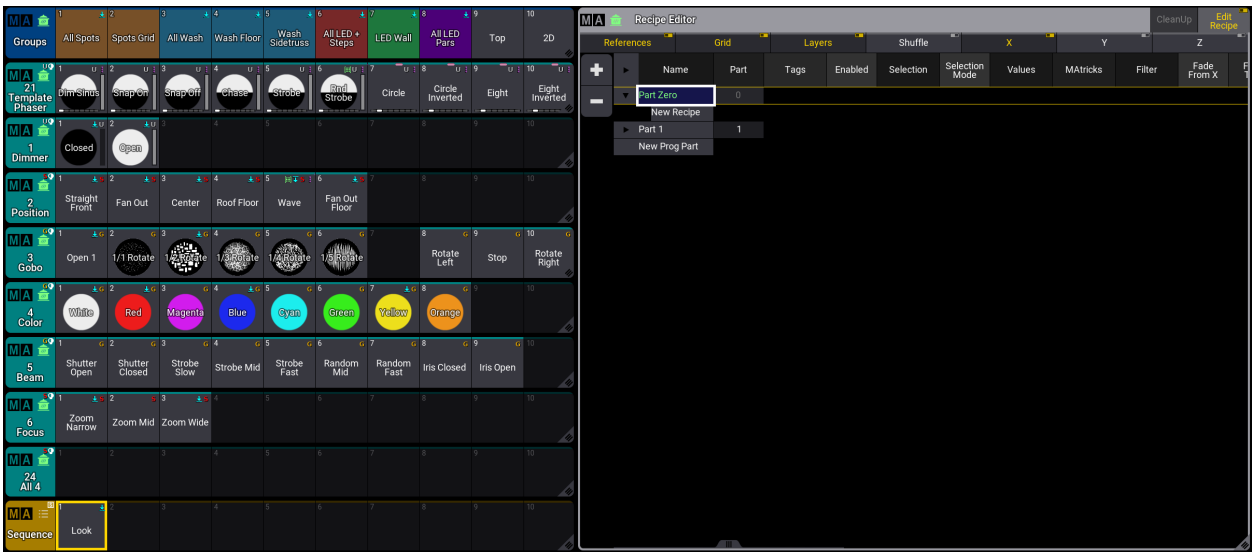
Groups pool with edit recipe mode enabled

The following windows use the recipe indicator:

- Groups
- Preset Pools
- MAtricks
- Worlds
- Filters
- Layout Viewer
- Fixture Sheet

	Hint:
	Make sure you set the Pool Action setting in the corresponding pools to **SelFix/At** or **At** so you can use them for recipes.

For creating recipes with the Edit Recipe mode, it is useful to have a groups pool, several preset pools, a sequence pool, and the recipe editor window visible:



Several pool windows and recipe editor window with edit recipe mode enabled

The following steps outline the general workflow for creating recipes. For a practical example, see **below**.

1. Select groups. The selected groups are indicated by a green frame around the pool objects. Multiple groups can be selected at the same time.
2. Select presets. Presets that are not compatible with fixtures of the selected group will be grayed out. All objects selected for recipes of the currently selected groups are indicated by a green frame in their pool windows. Tap on a selected object again to deselect it. You can add multiple values to a selection, a single recipe line is created for each selected value in the recipe editor window. You can define a different selection, for example, by tapping Group 2. A brown frame around pool objects indicates the values for previous selections, for example, Group 1. Tap Group 1 again and the previously defined values for this selection are highlighted in green again. A feature group indicator bar at the bottom of the group objects show the feature groups that are active in the recipe for for this very group.

	Hint:
	It is not possible to directly enter values into a recipe, they have to be stored in a preset first. If no preset is selected, the values are stored into the programmer.

3. Store the new recipe to a cue, cue part, or a preset. Storing a recipe clears the recipe editor and only groups will remain as selected objects.

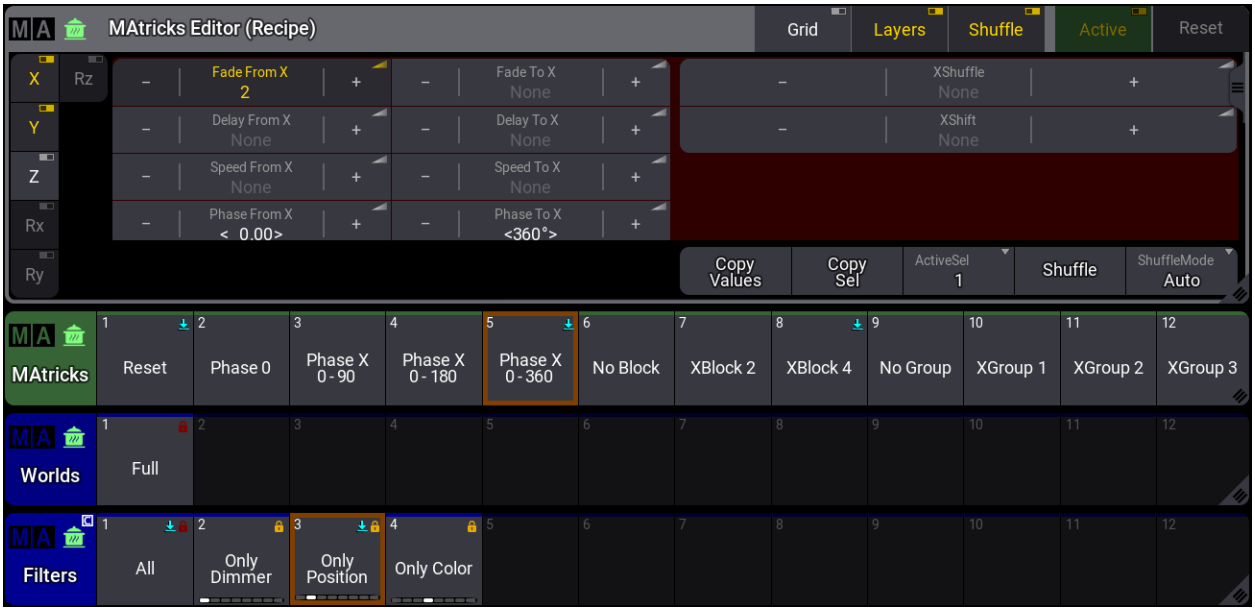
	Hint:
	The order of recording selections and values has an impact on how the recipe is handled. The last recipe line determines the output if several recipe lines with the same selection refer to the same attribute.
	Hint:
	Values that are entered for creating a recipe, do not overwrite or replace values that are directly entered into the programmer. The automatically

	cooked recipes are treated with low priority. To actively overwrite or merge programmer values, cook the programmer using the **Cook Keyword**.
	**Important:**
	Storing recipes into **Preset Pools** for specific feature groups, for example the Color preset pool, stores not only values for Colors, but all recipe lines of the selected programmer part.

Executing the **EditRecipe** keyword with a corresponding cue part or preset, enables the edit mode for the specific object in the recipe editor mode. Use the usual syntax of the **Off Keyword** or Off and tap a group or a preset to remove the corresponding recipe line. If a group or a preset has multiple lines linked, all corresponding lines will be removed.

Open the **MAtricks Editor** to define specific MAtricks values to the recipe lines. While using the MAtricks editor to edit a recipe, the editor is displayed with "(Recipe)" and the green pot icon in the title bar.

To add MAtricks, Worlds and Filters to the last created recipe, tap on the objects in the pool. Pool objects of MAtricks, Worlds, and Filters have a brown colored frame around them when they are used by a recipe:



MAtricks Editor with a selected MAtricks and Filter pool object

Example

This is a simple example of how to create recipe lines for a cue using the recipe editor window.

Requirements:

- Load the demo show file.
- Have preset pools, a groups pool, the recipe editor window, and the matrix editor visible. Also the sequence pool can be useful, but it is not a requirement.

Follow these steps to create two recipe lines and store it into a cue:

1. Enable Edit Recipe in the recipe editor window.

2. Select a group: **Group 1**
3. Select a dimmer preset: **At Preset 1.2**
4. Select a color preset: **At Preset 4.2**
   At this point, two recipe lines are created. Group 1 is the **Selection** for both lines. For **Values**, the first line has Preset 1.2, the second Preset 4.2.
5. Now select a different group: **Group 3**
   The prior selections of all pool objects is now deselected and displayed with a brown frame.
6. Select a dimmer and color preset again: **At Preset 1.2** and **At Preset 4.4**
   All selected presets are now connected to **Group 3**.
7. Select a position preset: **At Preset 2.3**
8. Select a template phaser preset: **At Preset 21.1**
9. Add some values for the MAtricks Editor: At Phase from X to 0, At Phase to X to 360°
   The MAtricks Editor values are added to the last created phaser preset recipe line. This is indicated by a small MAtricks icon in Name column of **Recipe 6** of the recipe editor window.
10. The last step is to store all recipe lines to a cue: **Store Sequence 2 Cue 1**

Cue 1 with all six recipe lines is created and all lines are removed from the recipe editor window.

To see the example, tap the video below:

1.34. Phasers

Phasers are the effects and chasers of grandMA3, creating dynamic output from a single preset or cue.

Phasers change the output for attributes using a set of information in two or more steps. A normal, static cue or preset only contains one step of information. Adding additional steps creates a phaser.

Example

A phaser has two steps with an absolute dimmer value of 100% in step 1 and an absolute dimmer value of 0% in step 2. With all other phaser layers at their default values, the resulting output is a smooth, repeating fade of the dimmer output from 100% to 0% and back to 100%.

Layers

Value Layers

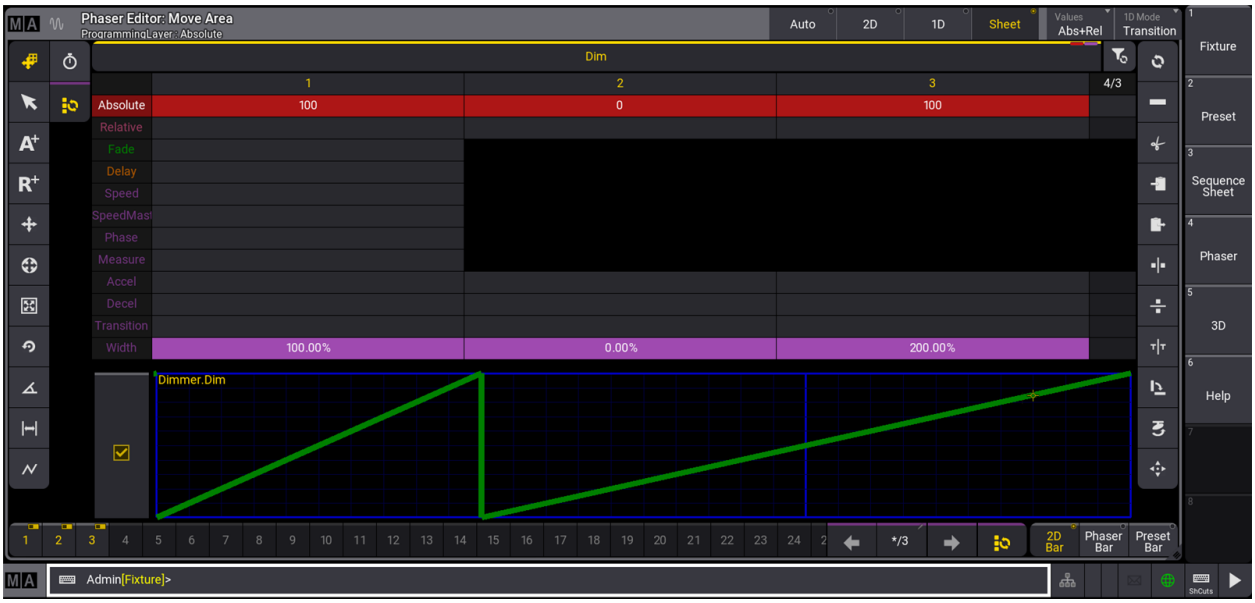
Each phaser step can include attribute value information. These values can be an absolute value (for instance, a dimmer value of 50%) or a relative value (for instance, a dimmer value of -20%). Steps can contain both absolute and relative values simultaneously.

	**Important:**
	The grandMA3 software handles storage, editing, and playback of absolute values and relative values separately. This separation allows the user to combine different absolute and relative values as desired.

Step Layers

Each phaser step also includes additional layers, which define the overall width of the step, the percentage of the step dedicated to transitioning to the new value, and the amount of acceleration and deceleration used when changing to the new value.

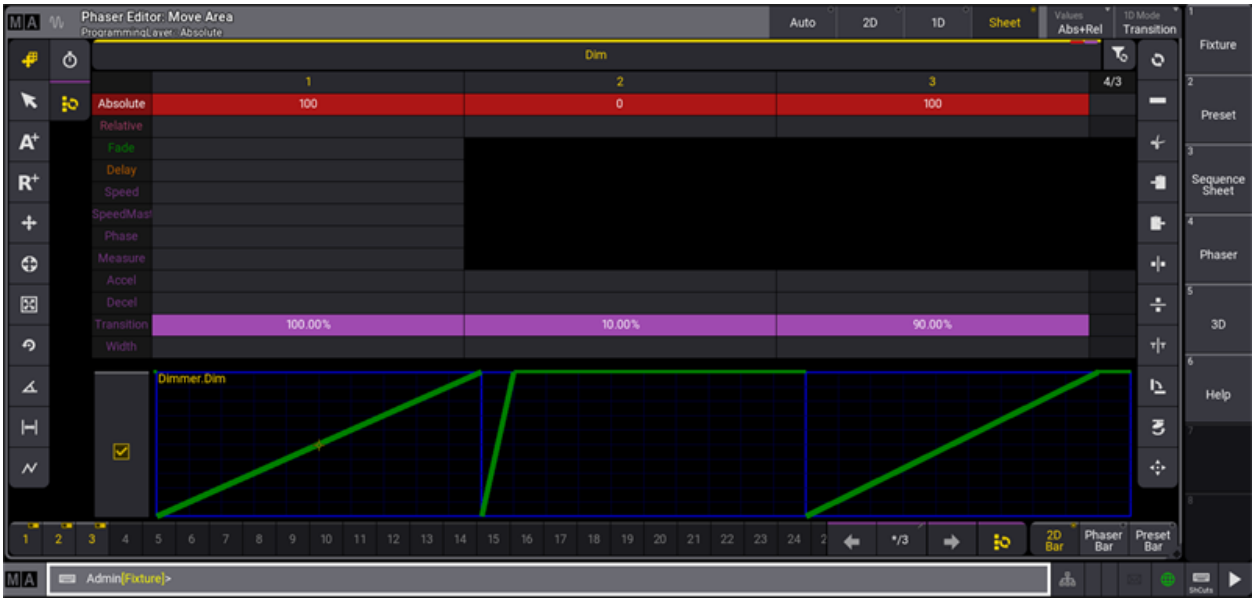
- **Width**



Phaser Editor showing steps with various widths.

The Width layer defines the total amount of time from the beginning of a step to the beginning of the next step. Width is shown as a percentage of one beat as defined by the Speed layer. The image above shows the first step with 100% width, so the change in value takes place over the time of one beat. The second step has a width of 0%, so the values change instantaneously and the next step begins immediately. The third step has a width of 200%, so the values change over the time of two beats.

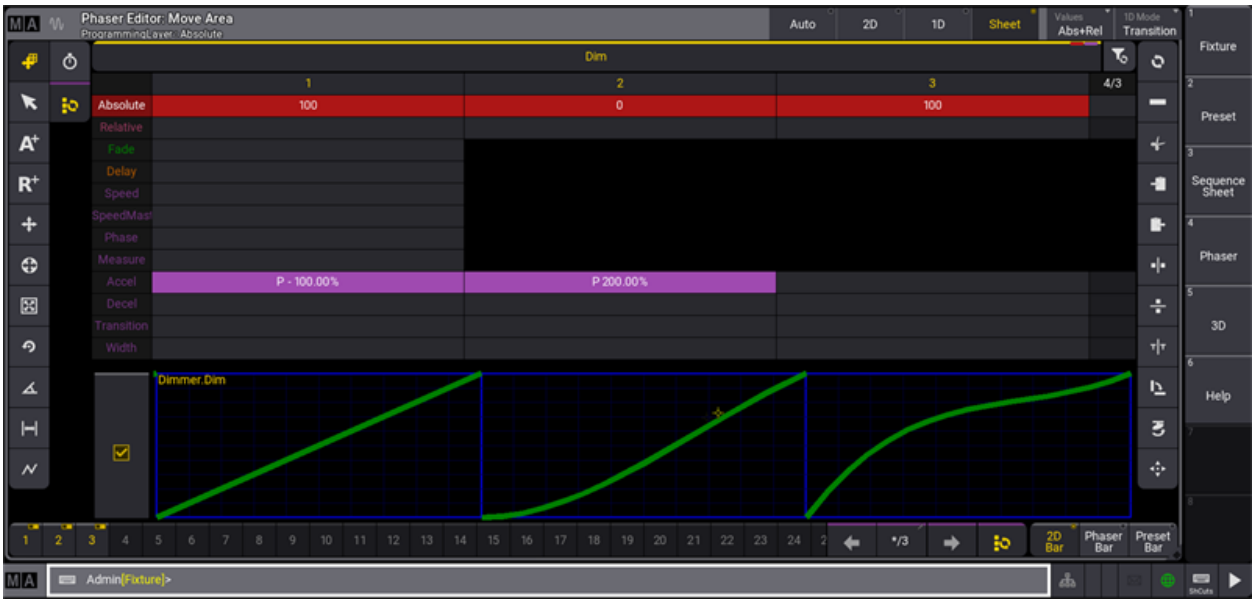
- **Transition**



Phaser Editor showing steps with various transitions.

The Transition layer defines how much of the step width is used to adjust values to their new levels. The transition of a step is shown as a percentage of the width of that step. The image above shows the first step with a transition of 100%, so the values change during the entire time of the step. The second step has a transition of 10%, so the values change during the first 10% of the width of the step, then remain at the new level until the end of the step. The third step has a transition of 90%, so the values change during the first 90% of the width of the step, then remain at the new level until the end of the step.

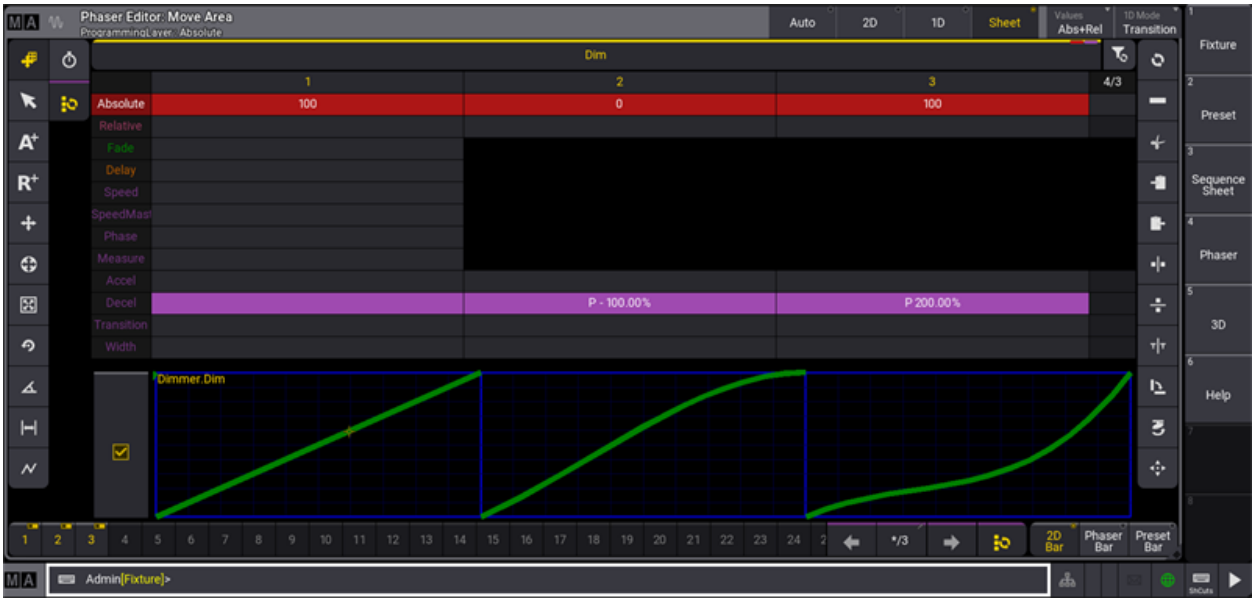
- **Accel**



Phaser Editor showing steps with various acceleration values.

The Accel layer defines how abruptly or gently the attributes begin to change from the value defined by the current step toward the value of the next step (acceleration). In the image above, the first step shows an acceleration of -100%, resulting in a smooth start to the transition away from the value at the end of the step. The second step shows an acceleration of 200%, resulting in an abrupt start to the transition away from the value at the end of the step.

- **Decel**



Phaser Editor showing steps with various deceleration values.

The Decel layer defines how abruptly or gently the attributes reach the value defined by the current step (deceleration). In the image above, the second step shows a deceleration of -100%, resulting in a smooth end to the transition into the value for the step. The third step shows a deceleration of 200%, resulting in an abrupt end to the transition into the value for the step.

	Hint:

	For linear transitions, keep Accel and Decel values at 0%. For smooth transitions, closely resembling sinus or cosinus waves, Accel and Decel values of -100%.

Phaser Layers

Each phaser additionally includes layers, which affect all steps of the phaser. Each attribute of each fixture can have its own speed, speed master, phase, and measure value in a phaser, but these values will be the same for all steps in a phaser.

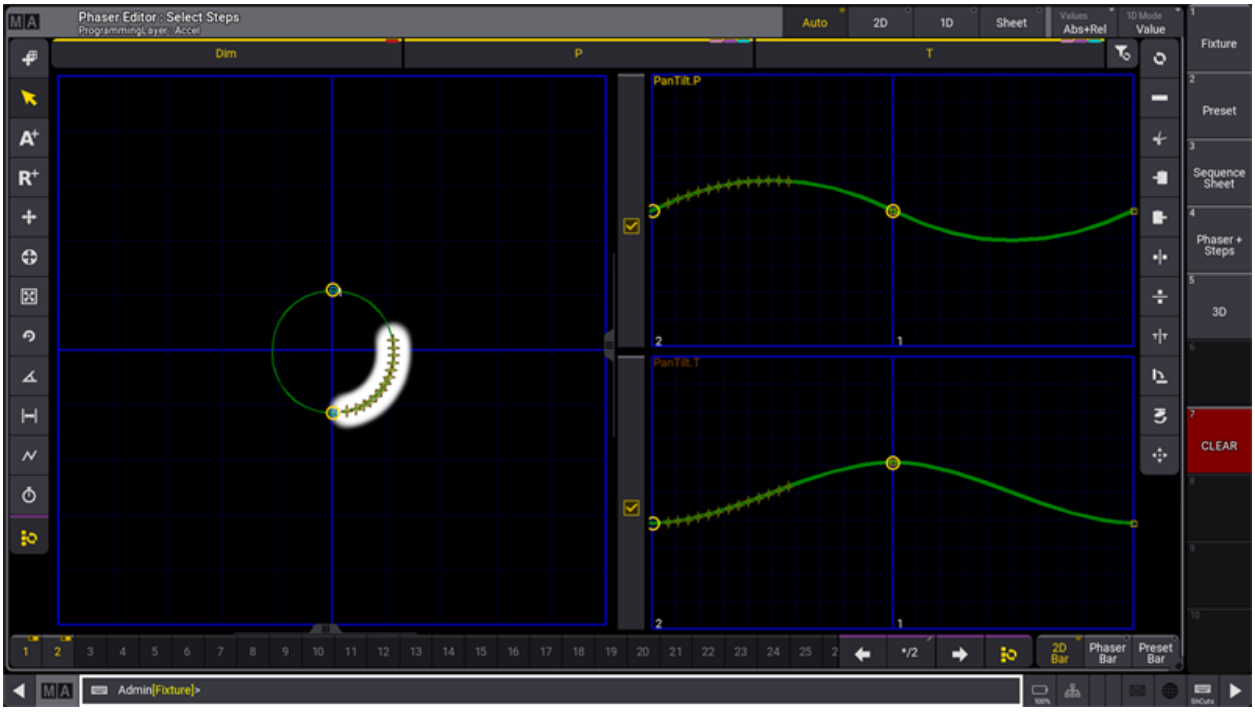
- **Speed**

Speed defines the overall rate at which the console plays back the steps of a phaser. The grandMA3 displays speed in units of either BPM (beats per minute), Hz (beats per second), or Seconds (seconds per beat).

- **SpeedMaster**

SpeedMaster defines per attribute to which master of the type speed the phaser adapts its speed to. Phasers speeds can be assigned to executors by using speed masters. For more information, see **SpeedMaster keyword**.

- **Phase**



Phaser editor showing a range of phase offsets.

Phase defines the timing offset of each fixture's attributes within a phaser. Phase is displayed in degrees (°). The image above shows a selection of fixtures in a circle phaser with phase offsets ranging from 0 to 90 degrees. The Phaser Editor displays this offset in both the 2D and 1D layouts.

- **Measure**

The optional Measure layer defines the number of beats in the repeating phaser loop. For example, a phaser with a measure of 4 beats and a speed of 120BPM will repeat every 2 seconds.

	Hint:
	When using the Measure layer, multiple combinations of step widths can produce the same result. In this case, it may be easiest to think of the width as a percentage of one beat in the measure.

All phaser layers, as well as a step selection tool, appear in the **Encoder Toolbar** any time attributes are mapped to the encoders. Phaser layers are indicated by a dark purple color. Phaser layers behave just like other attribute layers when activating, deactivating, storing, and clearing. For more information about attribute encoders and layers, see the **Operate Fixtures** topic. The Phaser Editor provides a graphical view and powerful manipulation of phaser layer information.

Steps

The grandMA3 offers multiple tools and methods for creating, selecting, and deleting phaser steps.

One intuitive method to quickly create steps based on presets involves the Step key. To create steps using this method, press and hold the Step key, then tap a preset for each desired step.

Example

To create a 3-step color phaser from white to red to blue:

1. Select the desired fixtures.
2. Press and hold the Step key.
3. Tap the white color preset.
4. Tap the red color preset.
5. Tap the blue color preset.
6. Release the Step key.

When using this method, the software will only create the next step when tapping a new preset, which contains any of the same attributes as the current step. This behavior allows each step to reference multiple presets as long as those presets apply to different attributes.

Example

To create a 2-step phaser, where each step contains both color and position presets:

1. Select the desired fixtures.
2. Press and hold the Step key.
3. Tap the desired color preset for step 1.
4. Tap the desired position preset for step 1. This preset is added to step 1.
5. Tap the desired color preset for step 2. Step 2 is automatically created.
6. Tap the desired position preset for step 2. This preset is added to step 2.
7. Release the Step key.

The Step key can also quickly change which preset is referenced by a step.

Example

Start with the 3-step color example above, but afterward, change the blue step to yellow:

1. Follow all directions in the 3-step color example above.
2. Press and release Step.
3. Press and release 3.
4. Tap the yellow color preset.

Delete specific steps from the programmer using standard syntax with the Delete keyword. For example, to delete step 3, type:

```
User name[Fixture]>Delete Step 3
```

Alternatively, delete specific steps from the programmer by selecting the steps in the Step Bar and deleting the **programmer**.

```
User name[Fixture]>Delete Programmer
```

An easy way to access this command (especially on grandMA3 onPC) is to tap Delete Steps in the **At Overlay**.

Stomp

The stomp function stops a phaser, running either in the programmer or from a playback, and resolves the stomped attributes to a single, static step. The console calls and activates the last static output value for each stomped attribute into step 1 of the programmer. All other steps of the stomped phaser are discarded.

With the programmer in step 1, calling a static preset onto any attributes currently running in a phaser automatically uses the stomp function to stop the phaser and output only the static look from the preset. Phasers follow this behavior whether the static preset is called directly in the programmer or as part of a recipe.

Sync

The Sync option ensures that phasers recall with predictable timing and phase offsets.

When calling phasers into the programmer, the Sync option is available in the **Encoder Bar**.

When recalling phasers stored in cues, the Sync option is available as a setting for each cue part in the Sequence Sheet. For more information about settings in cues and cue parts, see the **Look at Cues and Sequences** topic.

DelayToPhase

The DelayToPhase setting links the phase calculation of a phaser to any active individual delay time for the attribute.

When enabled, the phase calculation for each attribute begins as the values start to change at the end of the delay time. Because this delay timing may be different for different attributes or fixtures within a single cue, the resulting output may appear to include different phase offsets than originally stored within the phaser.

When disabled, the phase calculation for all attributes starting a phaser in the same cue starts when the cue is triggered, regardless of any individual delay times. This allows for more predictable, synchronized phaser playback, even with different individual delay times.

The DelayToPhase setting applies to all phasers in the show and appears in the Timing tab of the Preferences and Timing menu.

Subtopics

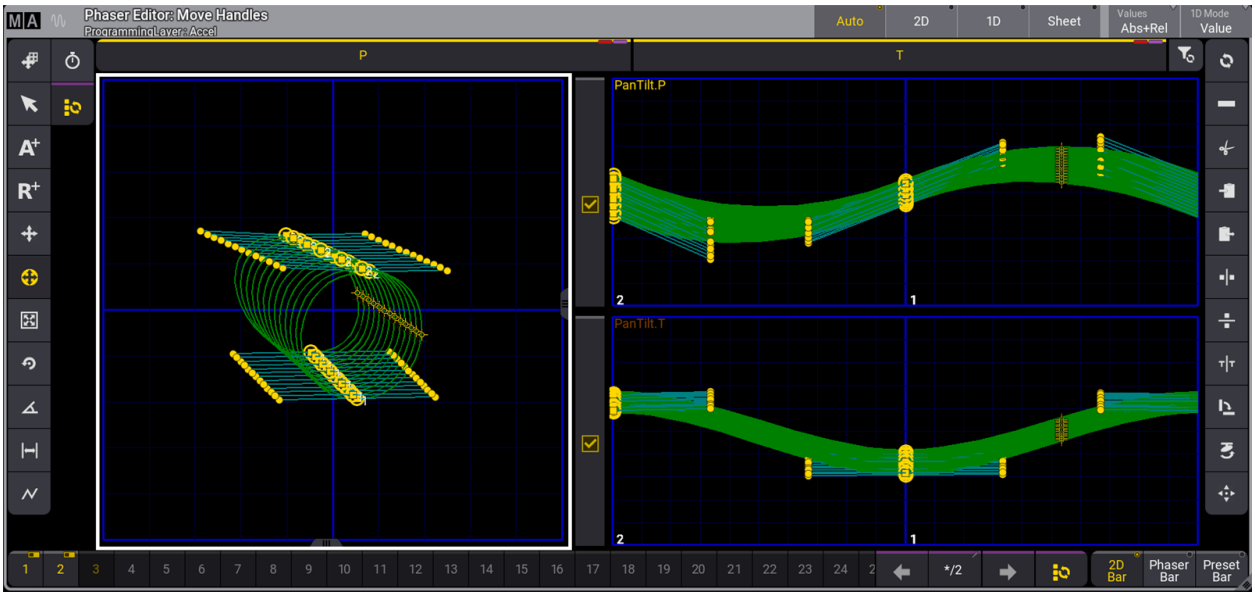
- **Phaser Editor**
- **Create a Sinus Dimmer Phaser**
- **Create a Circle Phaser**
- **Create a Circle Phaser Around a Position Preset**
- **Create Color Rainbow Phaser**

1.34.1. Phaser Editor

The **Phaser Editor** is a diverse tool for manipulating phasers.

It offers multiple means of visualizing running phasers as well as tools for dynamically creating and editing phasers. It can also be used as a simple trackpad to adjust the position attributes of fixtures.

Open a storable **Phaser Editor** window using the **Add Window pop-up**. Alternatively, open a temporary version of the **Phaser Editor** by tapping the Phaser button in the **Encoder Bar**.



Create and manipulate phaser steps in the phaser editor

The image above shows the Phaser Editor in the Auto view mode. Choose between multiple available view modes using the radio buttons in the title bar or the drop-down button in the Window Settings pop-up. Read below for images and descriptions of the other available view modes.

The blue grid on the left represents the entire pan range (horizontally) and the entire tilt range (vertically). This is called the 2D layout.

The smaller blue grids on the right are 1D layouts. Each attribute with more than one step in the phaser is displayed in its own 1D layout. The bold, vertical, blue lines in the 1D layouts represent beats, based on the speed of the phaser. The example above shows pan and tilt attributes, each with two steps.

When fixtures are selected, they are visualized by small yellow cross-hairs. The beams can be represented in the 2D layout by toggling the Show Beams button in the Window Settings pop-up. When Show Beams is enabled, the intensity and color of each fixture are visualized as a larger circle behind the corresponding yellow cross-hair.

Adding points in the 2D layout adds steps to the programmer. The first point moves the fixtures to the specified position. Each subsequent point adds another step to the programmer.

When a step contains both an Absolute point (shown as a yellow or cyan-filled square) and a Relative point (shown as a hollow, yellow or cyan square), a thin red line connects the relative point and the related absolute point. The cyan-colored points show that the value is from a preset. Yellow-colored points are hard values from the programmer.

Selected points have a yellow circle around them.

A green line describes the path of the fixtures.

A row of buttons immediately below the title bar includes one button for each attribute included in the active phaser. These buttons function as a direct link to the **At Filter**. Phaser adjustments apply to any attributes with a yellow bar and yellow text. Attributes with a grey bar and grey text will ignore adjustments. Tap the button immediately to the right of the last attribute button in this bar to enable or disable all attributes in the bar.

	Hint:
	If any or all attributes in the bar are disabled, the first tap of the button to the right of the attributes enables all displayed attributes. If all displayed attributes are already enabled, tapping this button disables them all.

The tool buttons on the left side are mouse or touch tools, and the tool buttons on the right are operational functions and shortcuts - read more below. The right side tool buttons might change depending on the selected tool on the left side.

Step Bar

The Step Bar appears across the bottom of the Phaser Editor. The Step Bar is a quick step selection tool. This element is optional in the Phaser Editor window, and it is enabled by default. Tap the Step Bar button in the Window Settings pop-up to show or hide the Step Bar. A storable window version of the Step Bar is also available in the **More** tab in the **Add Windows pop-up**.

Step Bar window with step one selected

Each step has a small square, which displays a status of empty, deselected, or selected.

Empty steps are steps that do not have any values. They have a dark background color.

Deselected steps have values in them, but they are not selected and are not affected by step-specific adjustments. Deselected steps have a light gray background color and an On/Off toggle icon in the upper-right corner.

Selected steps have values in them and are affected by step-specific adjustments. Selected steps have yellow text and a yellow On/Off toggle icon in the upper-right corner.

Tapping any of the steps toggles the selected status.

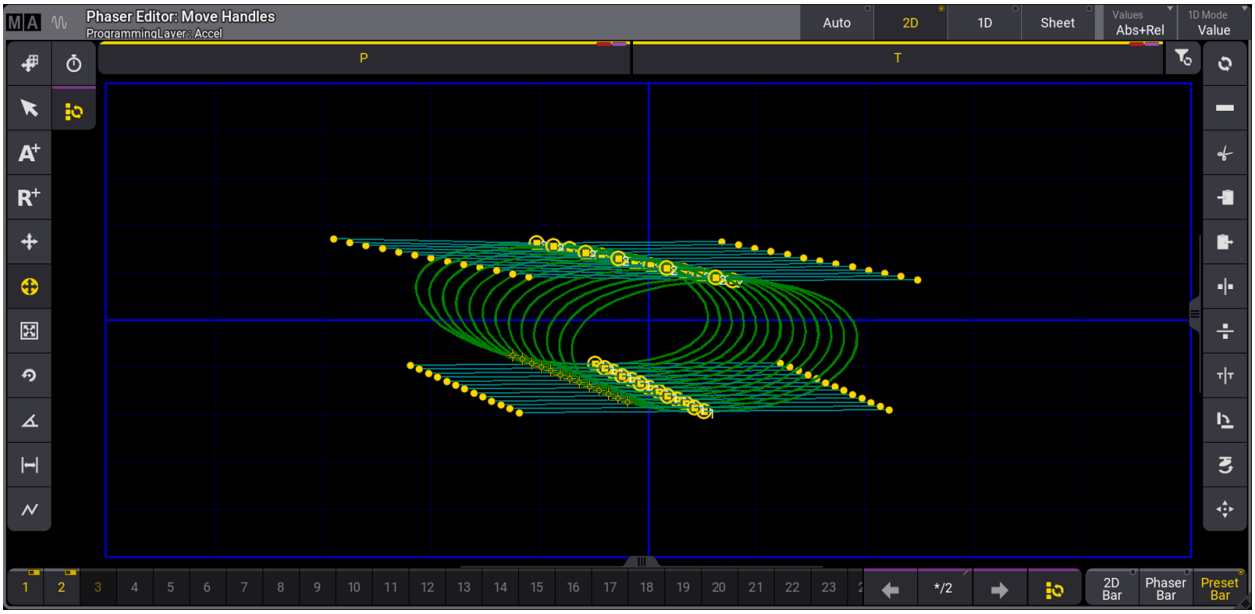
Additional controls appear on the right side of the Step Bar. These controls include left and right arrows used to change between single steps. An informative field appears between the arrows, showing which step is currently selected and the number of steps that contain values in the active phaser. If multiple steps are selected, the first selected step is shown with two dots after it (for example, 1../3). If all steps are selected, the first number is replaced with an asterisk (for example, */3).

Tap the button to the right of the right arrow in the Step Bar to select all steps. Tapping this selects all not-empty steps if only one step is currently selected. If more than one step is already selected, then only step 1 is selected. Tapping the button executes this command: **Step Toggle Executor**.

The Step Bar in the Phaser Editor window includes three radio buttons. Tap one of these buttons to call up the desired encoder toolbar. The Preset Bar opens the standard attribute **Encoder Toolbar** with access to all layers and all attributes. See below for Images and descriptions of the 2D Bar and Phaser Bar.

	Restriction:
	The storable window version of the Step Bar does not include the encoder toolbar radio buttons.

2D View Mode



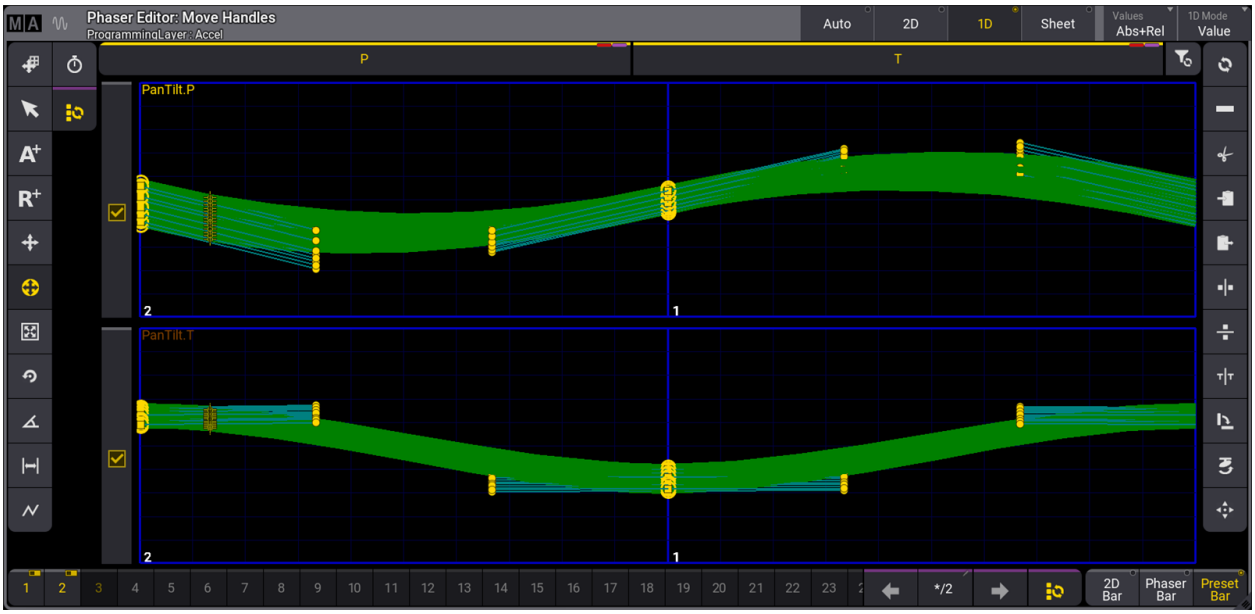
Phaser Editor in 2D View Mode

The 2D view mode removes other layout elements and expands the 2D layout to cover the main area of the window. This view mode is helpful when working with position phasers.

The Values button in the title bar allows or prevents the display and editing of absolute points or relative points. Tap to cycle through the available options or tap and swipe to open a list of available options. Options include:

- Absolute - Displays and allows editing of absolute points and their handles.
- Relative - Displays and allows editing of relative points and their handles.
- Abs+Rel - Displays and allows editing of absolute and relative points, as well as their handles.

1D View Mode



Phaser Editor in 1D View Mode showing Values

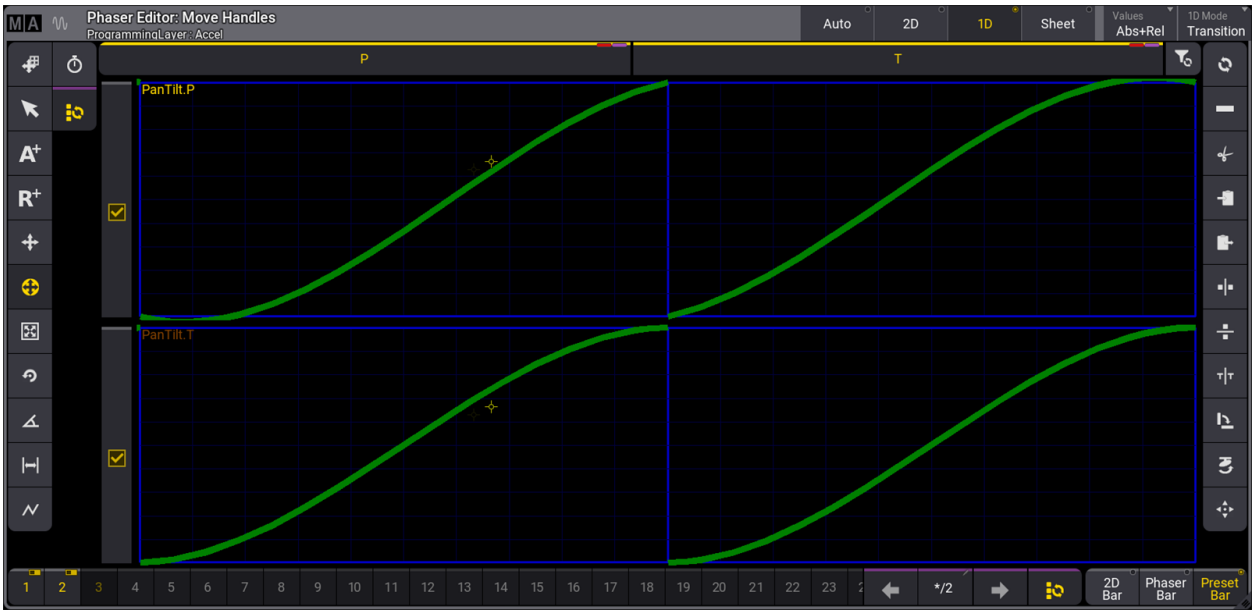
The 1D view mode removes other layout elements and expands the 1D layouts to cover the main area of the window. This view mode displays a layout for every attribute included in the active phaser.

To change the height of the attribute lines, tap the MA logo in the upper-left corner of the window, then tap LineHeight in the **Window Settings** menu, enter the desired height, and press Please.

The checkboxes to the left of the attribute lines represent the status of those attributes in the **At Filter**. These checkboxes relate directly to the attribute buttons across the top of the Phaser Editor. A yellow checkmark indicates that adjustments are allowed on the corresponding attribute. An empty, grey box indicates that adjustments are not allowed.

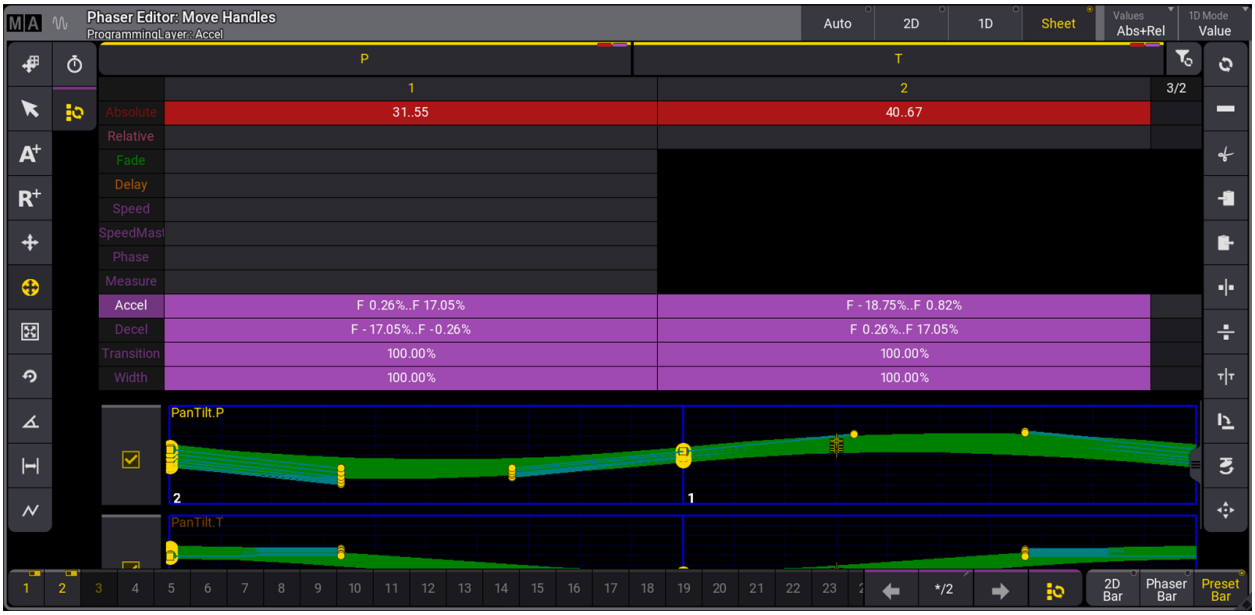
Tap the 1D Mode button at the right end of the title bar to toggle between the two available 1D drawing modes. These modes change the vertical scale of the 1D layouts.

- Value - In this mode, the vertical scale of each 1D layout equals the full range of values available for the displayed attribute.
- Transition - In this mode, the bottom of the vertical axis represents the beginning value for each step, and the top represents the ending value for the step.



Phaser Editor in 1D View Mode showing Transitions

Sheet View Mode



Phaser Editor displayed using Sheet View Mode

The sheet view mode displays phaser steps using a configurable spreadsheet format similar to the fixture sheet. Beneath all of the relevant spreadsheet data, this mode also includes 1D layouts for each attribute included in the phaser.

The default configuration of the sheet view mode arranges phaser steps as columns and layers as rows. Layers that affect all steps of the phaser appear only under step 1. Each cell in the sheet displays an overview of all values for the layer in that step. In cases where the cell includes multiple values or presets, the sheet displays the lowest and highest values in the cell separated by two dots.

Tools

The tool buttons on the left side are:


1
2
3
4
5
6
7
8
9
10
11
12

- **1 - Move Area**:
  Moves the entire blue pan/tilt square to show relative values that can be outside the position range.
- **2 - Select**:
  Selects single steps or uses lasso selection for multiple steps. The selection can be filtered to **Absolute** and **Relative** steps by toggling the respective buttons in the title bar.
- **3 - Add Absolute**:
  Adds a step using absolute value. The encoder bar follows the absolute layer. A filled rectangle represents absolute values. The **Single Step Mode** is automatically activated as long as you hold a finger or the mouse cursor in the editor while adding an absolute point.
- **4 - Add Relative**:
  Adds a step using a relative value. The encoder bar follows the relative layer. An empty rectangle represents relative values. The **Single Step Mode** is automatically activated as long as you hold a finger or the mouse cursor in the editor while adding a relative point.
- **5 - Move Point**:
  Moves selected steps. It is also possible to use the Align function. The **Single Step Mode** is automatically activated as long as you hold a finger or the mouse cursor in the editor while moving a single point.
- **6 - Move Handles (The yellow highlight shows this is the current tool.)**:
  Move handles of selected steps. It is also possible to use the **Align** function. Each point/step has two handles to influence acceleration and deceleration at the same time. Each handle has a cyan line connected to the point it controls. Move both handles of a point by touching the point and dragging in the desired direction. A handle can be moved individually by touching and moving the handle (small yellow circle).
- **7 - Change Size**:
  Enlarges/reduces all selected steps evenly from the center point of the selection. It is also possible to use the **Align** function.
  Pressing MA while changing the size enlarges/reduces all selected steps from the center point of the selection on the horizontal or vertical axis.
- **8 - Change Rotation**:
  Rotates the selected steps around the center of the selection. It is also possible to use the **Align** function.
  Pressing MA while changing the rotation, rotates the selected steps around the center of the selection in 5-degree increments.
- **9 - Change Phase**:
  Opens an overlay with yellow dots symbolize the value of the phase for each fixture. Here the phase value can be shifted independently from the selected steps for the selected fixtures. The **At** filter can be used to move phases for individual attributes. It is also possible to use the **Align** and the **Transition** mode functions. If several attributes are displayed along with different phases, the selected feature groups are displayed in a brighter color. If the fixtures are allocated to grid positions, the values of the phases can be vertically or diagonally aligned. Selecting this also changes the functions on the right side to quick selections for alignment. This changes the tool buttons on the right side to give quick access to standard phase values and a Reset and Invert button.
- **10 - Change Width:**
  The width can be changed with this tool. It changes the functions on the right side to shortcuts of different percentages. If all steps are selected and the Speed shortcut is tapped, then the movement speed is equalized between the steps. Default is that each step has the same width with no regard to the distance the fixtures are moving in the step. Equalizing the step width

makes the fixture move at the same speed by adjusting the width. There is a Reset shortcut that resets the width. Default is a width of 100%. This means that each step has the same width with no regard to the travel distance of the fixtures in each step.

- **11 - Select Form:**
  Selecting a form overwrites the values of the transition layer, accel layer, and decel layer to create the desired effect. Available forms are listed in the toolbar on the right side and include Rectangle, Sawtooth, Sine, and Circle.
- **12 - Change Speed:**
  Tap to display various speed tools in the right side toolbar. Available speed tools include functions which multiply or divide the current speed of the phaser by a given amount. Special speed options include:
  - **Loop:** Tap Loop to multiply the current phaser speed by the number of steps currently in the phaser.
  - **Fixture:** Tap Fixture to divide the current phaser speed by the number of fixtures in the current selection.
- **13 - Select all Steps**:
  This is not a tool like the others. Tapping this toggles between all steps selected or the selection of single steps. It keeps the previously selected tool active.

	**Hint:**
	Some tools don't require the user to precisely touch points in the 2D and 1D layouts to adjust them. Instead, make sure the desired steps are enabled in the **Step Bar**, and use the blue grid areas in the **Phaser Editor** as trackpads to make the desired adjustments.

The standard operational functions on the right side are:

- **1 - Reset**:
  Reset spline of the selected steps.
- **2 - Delete**
  Deletes the selected steps.
- **3 - Cut Programmer**:
  Removes all programmer values in the currently selected fixtures and moves them to the clipboard.
- **4 - Copy Programmer**:
  Copies all programmer values in the currently selected fixtures and moves them to the clipboard.
- **5 - Paste Programmer**:
  Inserts all values from the clipboard to the currently selected fixtures in the programmer.

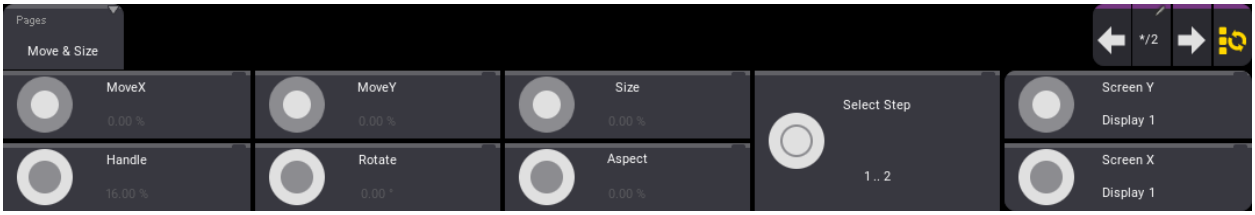
- **6 - Mirror X**:
  Mirrors all selected points on the x-axis.
- **7 - Mirror Y**:
  Mirrors all selected points on the y-axis.
- **8 - Mirror Time**:
  Swaps the order of the steps. Select all steps first. This is the same as reversing the direction of the movement.
- **9 - Swap XY**:
  Swaps the X and Y axes of the selected steps.
- **10 - Flip**:
  Flips the position of the selected fixtures.
- **11 - Reset Zoom**:
  Resets the area with the blue square.

Phaser Encoder Toolbar

Depending upon the type of encoder toolbar selected with the radio buttons in the bottom-right corner of the **Phaser Editor**, the encoder toolbar changes to display helpful tools, which are specific to the manipulation of Phaser data.

2D Bar

To enable the 2D Bar, tap 2D Bar in the bottom-right corner of the Phaser Editor.



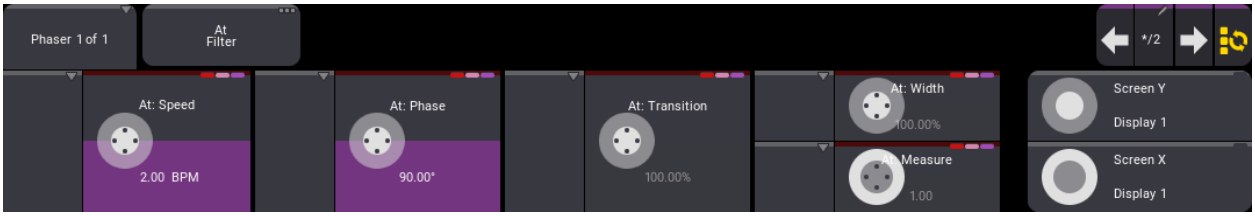
2D Phaser Encoder Bar

This encoder toolbar includes the following encoder controls:

- **MoveX:** Turn the inner encoder to adjust the X position in the 2D Phaser grid of all steps currently enabled in the **Step Bar**.
- **MoveY:** Turn the inner encoder to adjust the Y position in the 2D Phaser grid of all steps currently enabled in the **Step Bar**.
- **Size:** Turn the inner encoder to adjust size of all steps currently enabled in the **Step Bar**. This allows encoder access to the **Change Size** tool mentioned above.
- **Handle:** Turn the outer encoder to adjust the handle length of all steps currently enabled in the **Step Bar**. This allows encoder access to the **Move Handles** tool mentioned above.
- **Rotate:** Turn the outer encoder to rotate all steps currently enabled in the **Step Bar**. This allows encoder access to the **Change Rotation** tool mentioned above.
- **Aspect:** Turn the outer encoder to adjust the aspect ratio of all steps currently enabled in the **Step Bar**. This tool simultaneously adjusts positions of points as well as length and direction of handles in order to create a wider or taller version of the current 2D form.
- **Select Step:** Turn the inner or outer encoder to select individual steps. Press the inner encoder or the outer encoder key to toggle between selecting all steps and a single step.

Phaser Bar

To enable the Phaser Bar, tap Phaser Bar in the bottom-right corner of the Phaser Editor.

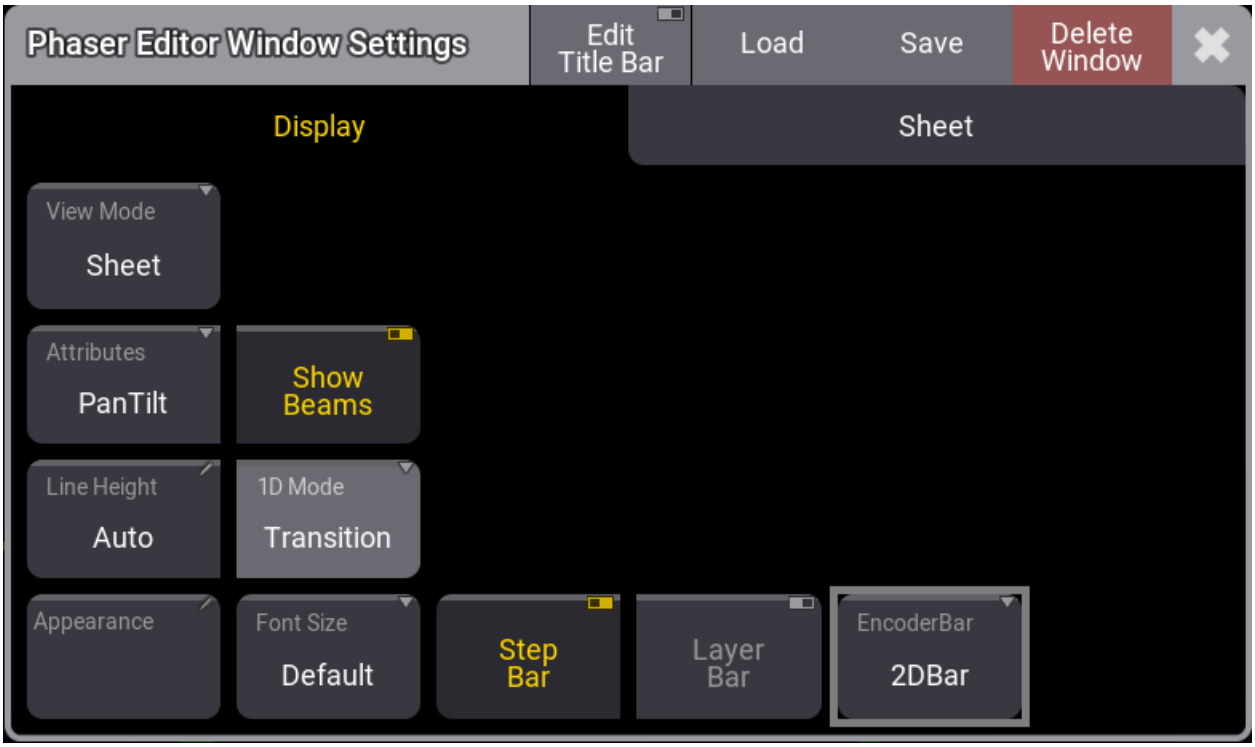


Phaser Bar

This encoder toolbar allows for adjustment of **Speed**, **Phase**, **Transition**, **Width**, and **Measure**, based upon the current fixture and step selection, as well as the current status of the **At Filter**.

Window Settings

The Window Settings pop-up includes some **Common Window Settings** as well as some that are specific to the Phaser Editor.

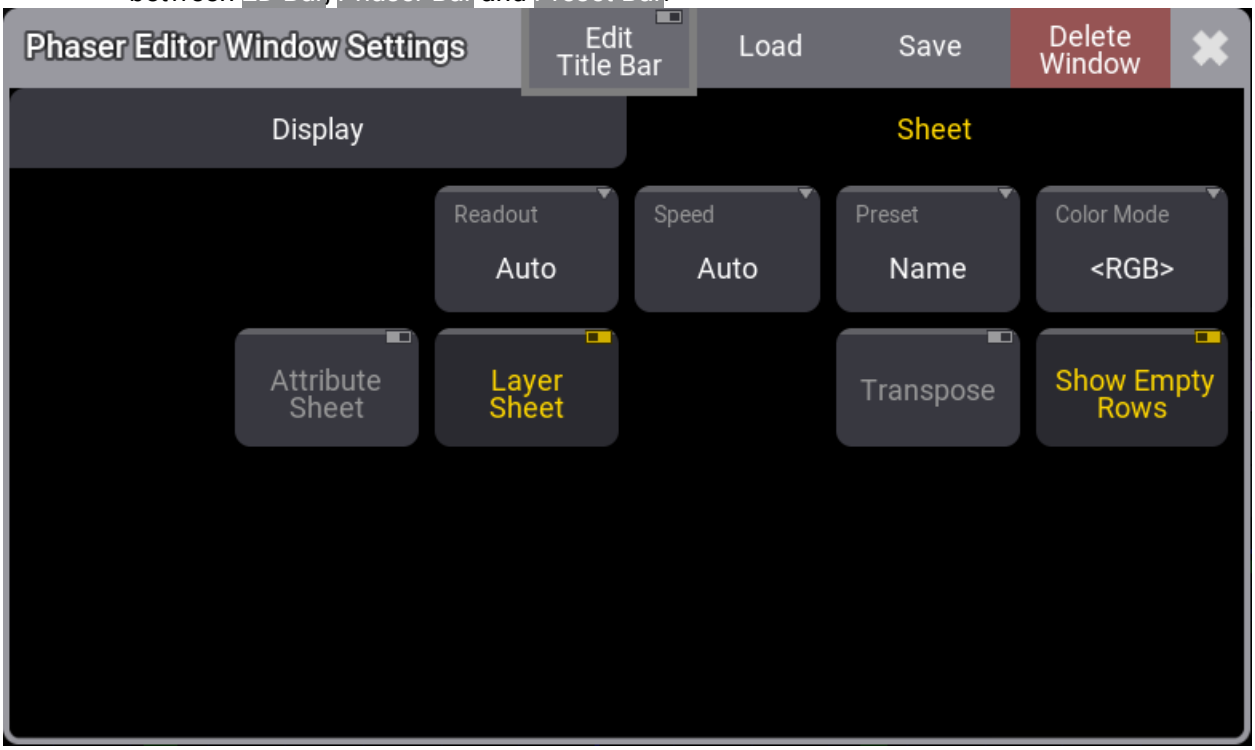


Display tab of the Phaser Window Settings Pop-up

- View Mode - The view mode defines how the different data and information are displayed in the view. The view mode can be changed in the settings or the title bar of the window.
    - **Auto:**
      Displays the 2D and 1D layouts.
    - **2D:**
      Displays the 2D layout.
    - **1D:**
      Displays a 1D layout for every attribute included in the active phaser.

- **Sheet:**
  Displays phaser steps in a spreadsheet format and the 1D layouts for the phaser attributes.

- Line Height - Defines the height of the 1D layouts. Valid options range from 50 to 500. **Auto** adjusts the lines to the size of the Phaser Editor within this range.
- Appearance - Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.
- Font Size - This selects the font size in the window. It is a swipe button that opens a list of sizes from 10 to 32. There is also a **Default** property. The default is the same as size 18.
- Show Beams - Tap to show or hide the beams in the 2D layout.
- 1D Mode - Tap to toggle between the two 1D drawing modes. These modes change the vertical scale of the 1D layouts.
  - **Value:**
    The vertical scale of each 1D layout equals the full range of values available for the displayed attribute.
  - **Transition:**
    The bottom of the vertical axis represents the beginning value for each step, and the top represents the ending value for the step.
- Attributes - Defines which position attributes are used when creating a position phaser. Tap to toggle between PanTilt, XY, XZ, and YZ.
- Step Bar - Tap to show or hide the Step Bar.
- Layer Bar - Tap to show or hide the Layer Bar.
- EncoderBar - Defines which encoder bar is displayed in addition to the step bar. Tap to toggle between 2D Bar, Phaser Bar and Preset Bar.



Sheet Tab of the Phaser Window Settings Pop-up

- Readout - This selects the value readout for fixture attributes. It is a swipe button that opens a list of readout types with the following options:
  - **Auto**:
    This makes the sheet follow the selected readout in the **Encoder Bar**.

    - **Natural**:
      Each attribute has a defined Natural readout. This is defined in the **Attribute Definition**. Selecting this option will show the different readouts defined for the attributes.
    - **Percent**:
      This is a range from 0 to 100.
    - **PercentFine**:
      This is a range from 0.00 to 100.00.
    - **Physical**:
      This uses the physical range defined in the fixture type definition.
    - **Decimal8**:
      This is a decimal range from 0 to 255.
    - **Decimal16**:
      This is a decimal range from 0 to 65 535.
    - **Decimal24**:
      This is a decimal range from 0 to 16 777 215.
    - **Hex8**:
      This is a hexadecimal range from 00 to FF.
    - **Hex16**:
      This is a hexadecimal range from 0000 to FFFF.
    - **Hex24**:
      This is a hexadecimal range from 000000 to FFFFFF.
- Speed - It sets how the speed value is displayed. It has the following options: Auto (following the User Profile setting), Hertz, BPM (Beats Per Minute), and Seconds.
- Preset - This defines how the preset information is displayed in the sheets. There are six properties which are different combinations of these three elements:
    - **ID**:
      Shows the ID number of the preset.
    - **Name**:
      Shows the name of the preset.
    - **Value**:
      Shows the values stored in the preset.
- Color Mode - This switches the color readout between Auto (following the User Profile setting), RGB and CMY. The default value is to follow the setting in the **User Profile**. The user profile setting is shown between "<>".
- Attribute Sheet - Tap to show or hide individual rows for each attribute included in the phaser. If Layer Sheet is enabled, each layer will include all of these individual layers.
- Layer Sheet - Tap either show all layers simultaneously or hide most layers, showing only the layer that is currently accessible on the attribute encoders.
- Transpose - This On/Off button flips the columns and rows in windows.
- Show Empty Rows - Tap to show or hide empty rows.

1.34.2. Create a Sinus Dimmer Phaser

This topic details a couple of examples of how to create a sinus dimmer phaser using keys and buttons as well as the phaser editor and encoder bar. There are many ways to reach the same goal in grandMA3. Many of the steps in the examples below are interchangeable while producing identical results. These examples use a combination of keys and the calculator interface. If you are working with the onPC software, it might be easier to type commands into the command line rather than use the on-screen keys.

It is recommended to read the **Phasers topic** and the **Phaser Editor topic** before this topic.

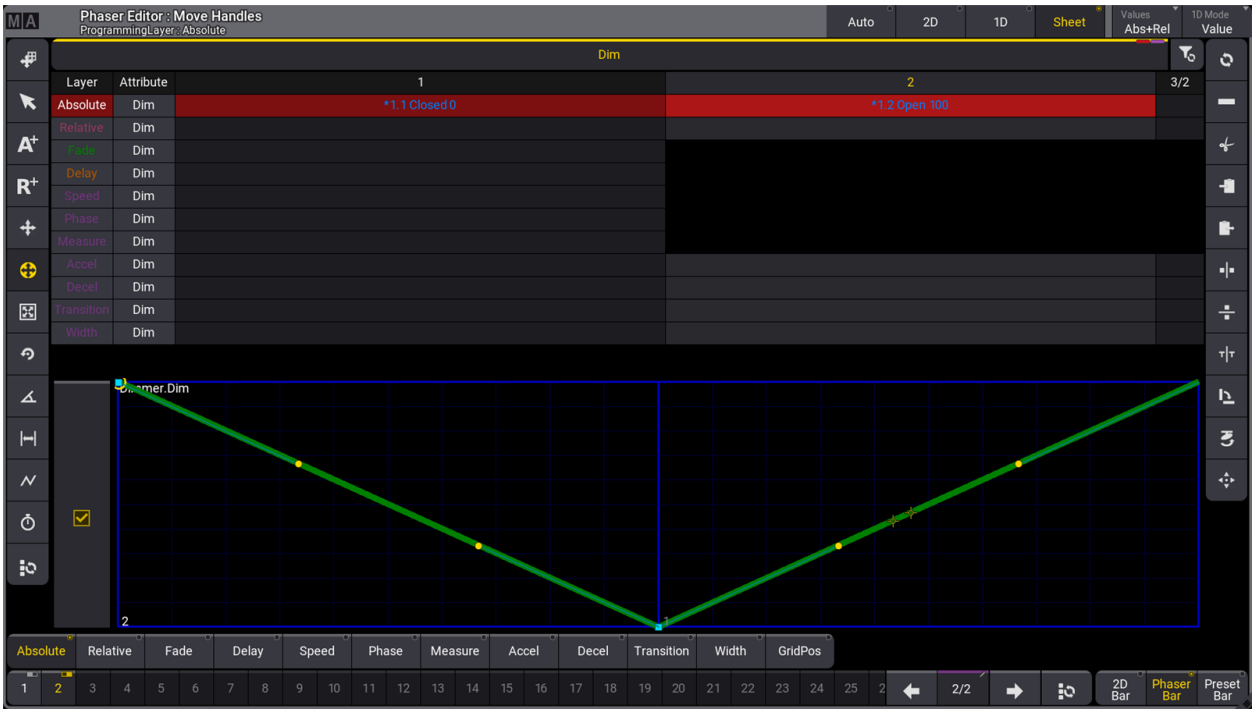
Requirements:

- Have a show with some lights patched.
- An open **Phaser Editor** window with the Step Bar enabled is recommended, though it is also possible to use the temporary Phaser Editor.
- Arranging the fixtures in the 3D window can be useful, but it is not a requirement.
- As it is highly recommended to use presets while programming, an open Dimmer preset pool is recommended with presets for dimmer values of 100% and 0%.

Quickly Create Steps and Apply Adjustments with the Phaser Editor

1. Select the desired fixtures. For example: Fixture Thru Please
2. Choose a dimmer preset for step one: Press and hold Step and tap the preset with the 0% dimmer value.
3. Choose a dimmer preset for step two: While still holding Step, tap the preset with the 100% dimmer value.

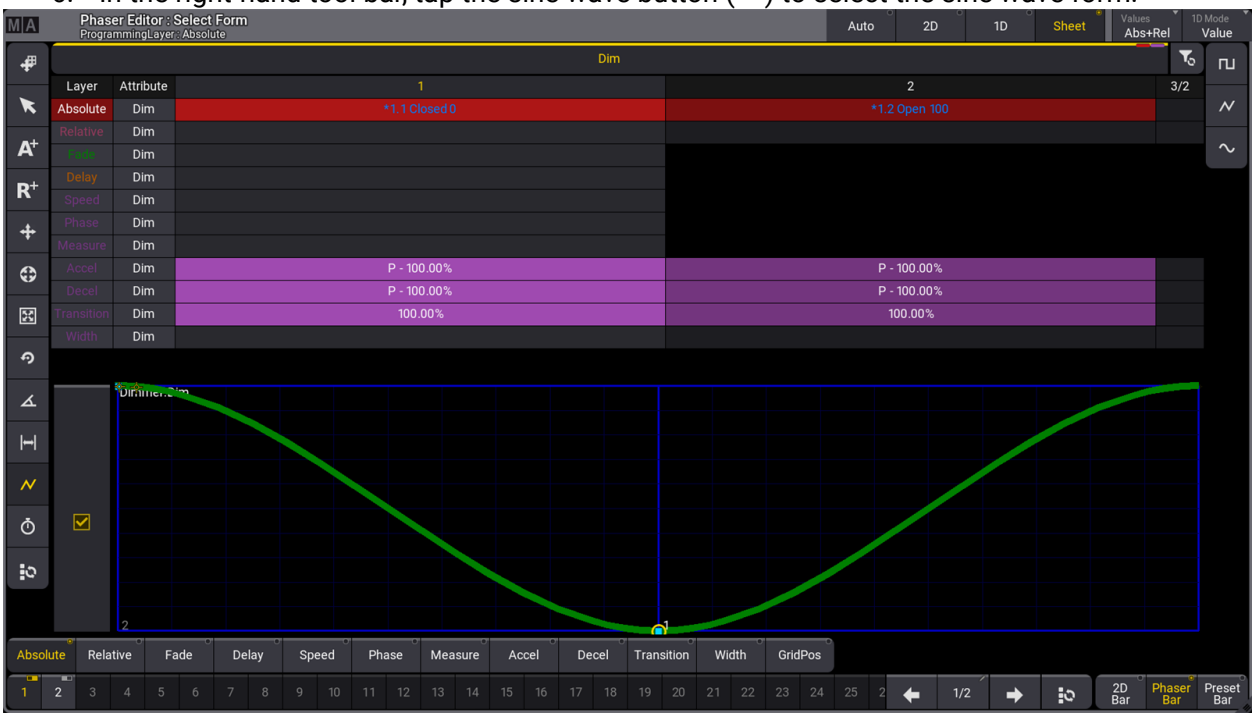
This is the base for the phaser. There are now two steps with a dimmer preset of 0% in the first step and a preset of 100% in the second step. All of the fixtures are changing together from step one to two with a linear fade (not yet using a sine curve). The Sheet view mode of the **Phaser Editor** shows:



Linear Dimmer Phaser with No Phase Distribution

Adjust the curve of both steps using tools in the Phaser Editor:

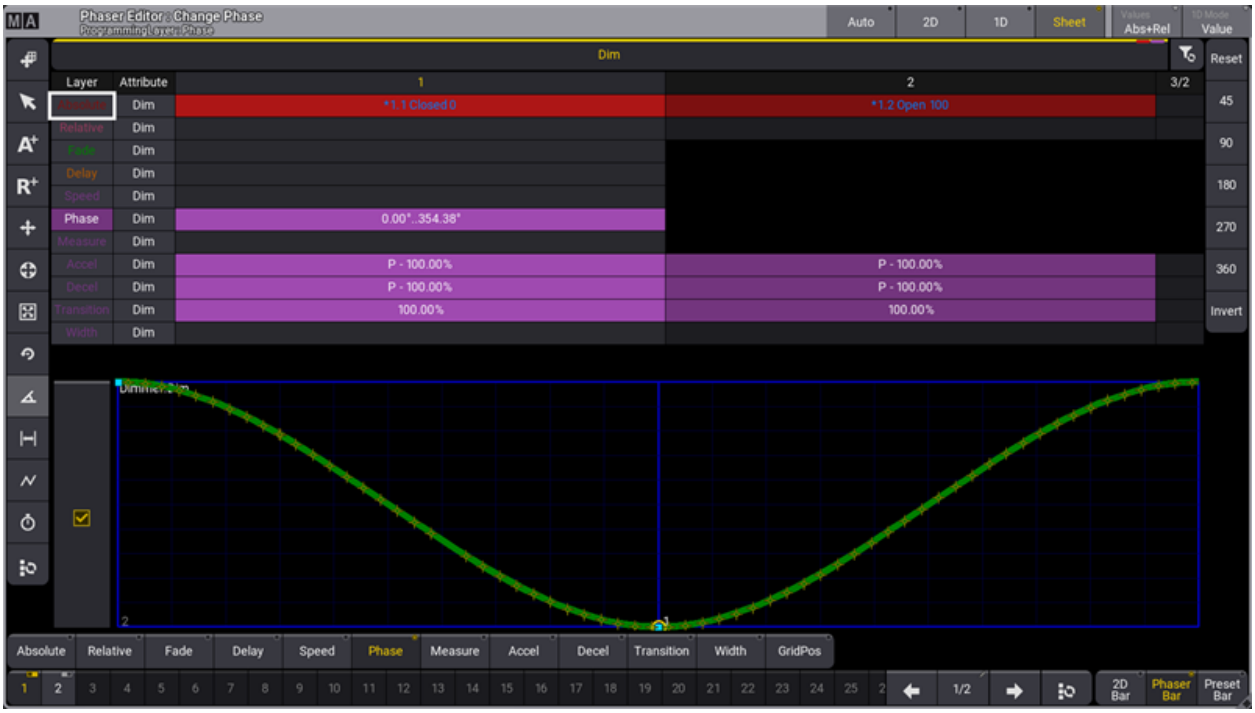
4. Select all steps by tapping the button at the right end of the Step Bar or at the bottom of the left-hand tool bar.
5. Tap to select the Select Form tool ( ) in the left-hand tool bar.
6. In the right-hand tool bar, tap the sine wave button ( ) to select the sine wave form.



Sinus Dimmer Phaser with No Phase Distribution

Adjust the phase of all fixtures using tools in the Phaser Editor:

7. Tap to select the Edit Phase () tool in the left-hand tool bar.
8. In the right-hand tool bar, tap the 360 button to evenly distribute the selected fixtures over the wave form.



Sinus Dimmer Phaser with Full Phase Distribution

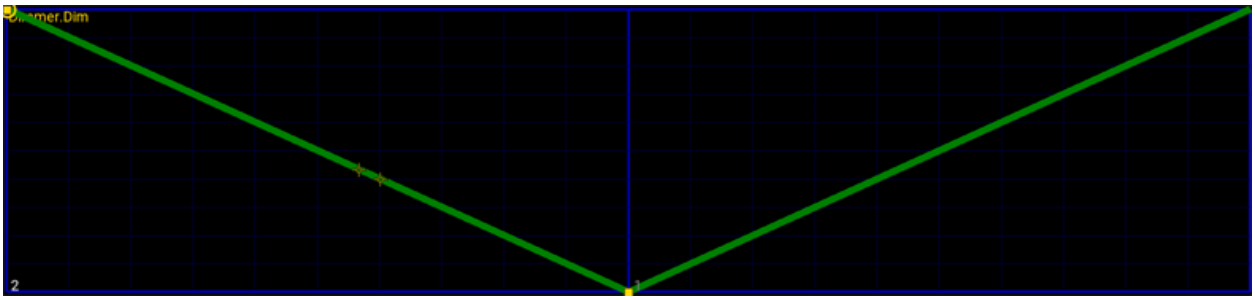
Store the phaser into a preset or a cue.

Alternate Methods to Create a Sinus Dimmer Phaser

Follow these steps to create a sinus dimmer phaser:

1. Select the desired fixtures. For example: Fixture Thru Please.
2. Set them at a dimmer value of 0: At 0 Please.
3. Create step 2 and select it: MA + Next (Next Step).
4. Set the fixtures at full: Full.

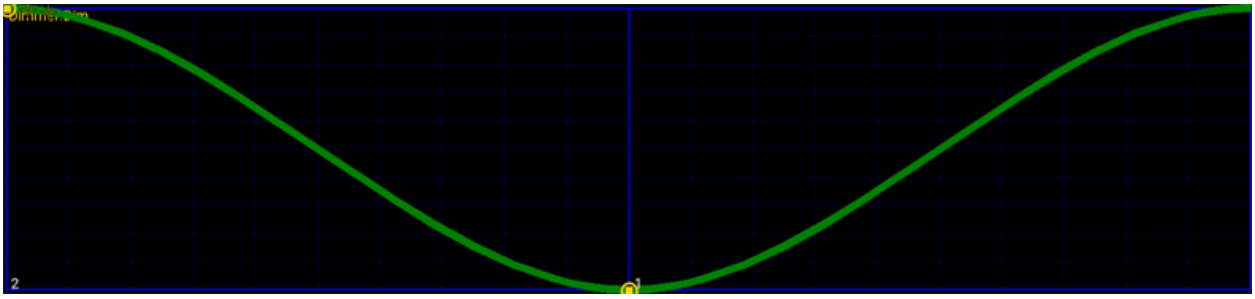
This is the base for the phaser. There are now two steps with a dimmer value of 0 in the first step and a value of 100 in the second step. All the fixtures are changing together from step one to two in a linear direction and do not fade using a sine curve. It looks like this in the **Phaser Editor**:



The curve is corrected by adjusting the acceleration and deceleration in both steps.

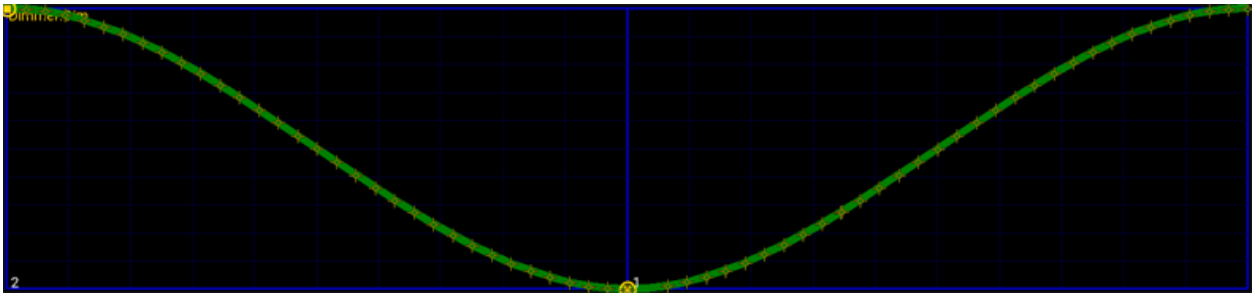
5. Select both steps: MA + Set (Step Toggle).
6. The steps need to have an accelerate value of -100. This is done using the **Encoder Toolbar**.
    a. Tap Dimmer to make sure it is selected.
    b. Tap Accel to select the acceleration layer.
    c. Tap the left attribute encoder to open the calculator.
    d. Tap +/- 1 0 0 Please. This gives a known and precise value. If **Handles** are used then it might not be as precise as typing a value.
7. Do the same for the Decel layer.

Now the attribute accelerates out of and decelerates into each step. The curve now looks like this:



8. The final thing to do is to spread out the fixtures using the phase value. This is also done from the encoder toolbar.
    a. Tap Phase to select the phase layer.
    b. Tap the left attribute encoder to open the calculator.
    c. Tap 0 Thru 3 6 0 Please.

Now the fixtures are spread out equally over the entire phase of the phaser loop.



This is a sinus phaser that can be stored in a cue or preset.

It is created using a specific selection of fixtures so remember to select the desired **Preset Mode** when **storing a preset**.

1.34.3. Create a Circle Phaser

This topic presents an example of one method for creating a simple circle phaser. It uses absolute position values and will always move the lights in the specified circle. See the **Create Circle Phaser Around Position** topic for an example that covers creating a relative phaser circle, which can be used with different absolute base positions.

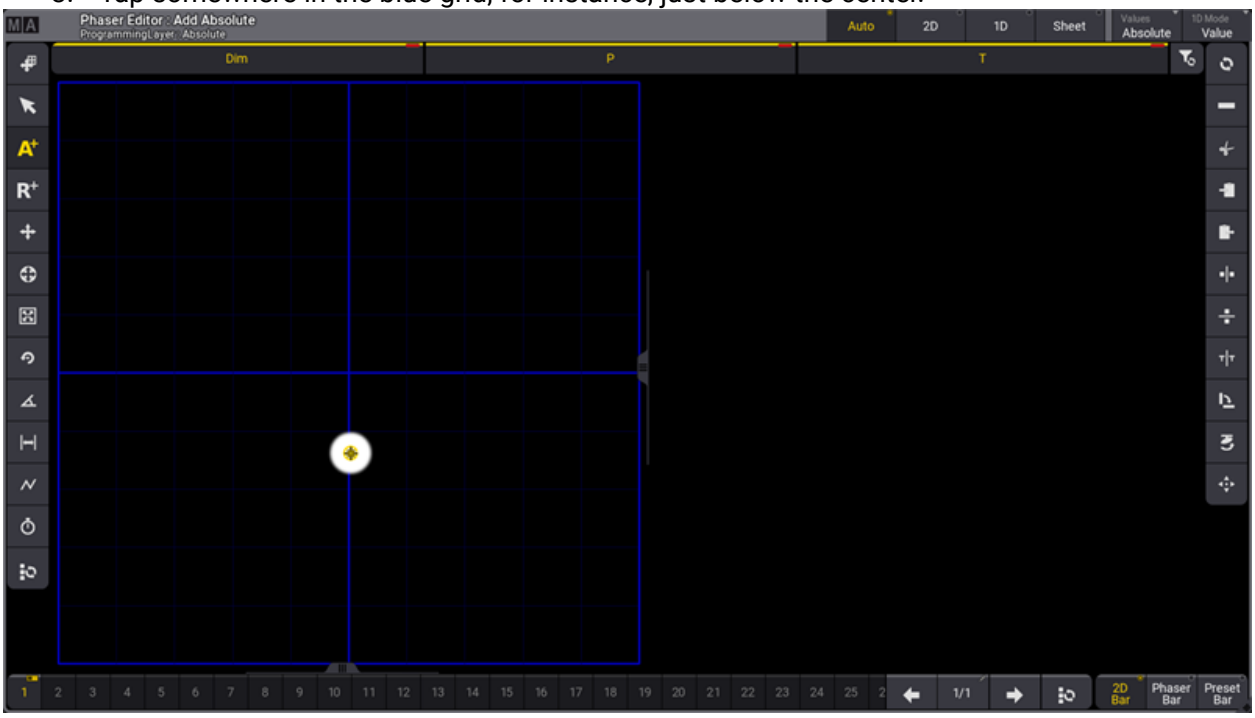
Requirements:

- Have a show with some moving lights patched - for instance, the demo show.
- An open **Phaser Editor** window is needed.
- Arranging the fixtures in the **3D window** can be useful, but it is not a requirement.

	Hint:
	For phasers that use position values, it is recommended to use a slower speed than the default of 60 BPM, such as 20 BPM. Otherwise, the fixtures may not be able to move properly.

Follow these steps to create a circle:

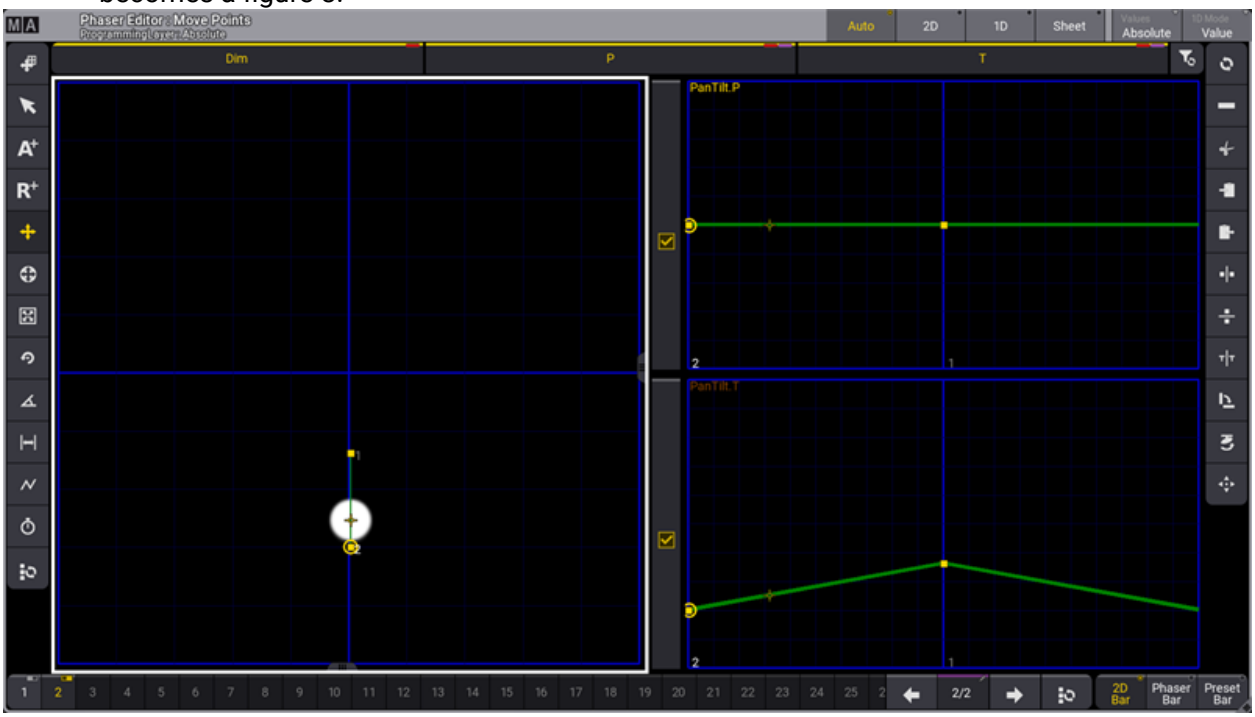
1. Select the desired fixtures and bring their dimmers to full.
   The Phaser Editor shows a white beam in the center of the blue 2D grid.
2. Tap the A+ button on the left of the editor.
3. Tap somewhere in the blue grid; for instance, just below the center.



Phaser Editor with fixtures turned on and one absolute position

This creates the first absolute point. The selected fixtures move to this static position. This is an absolute position value in step one.

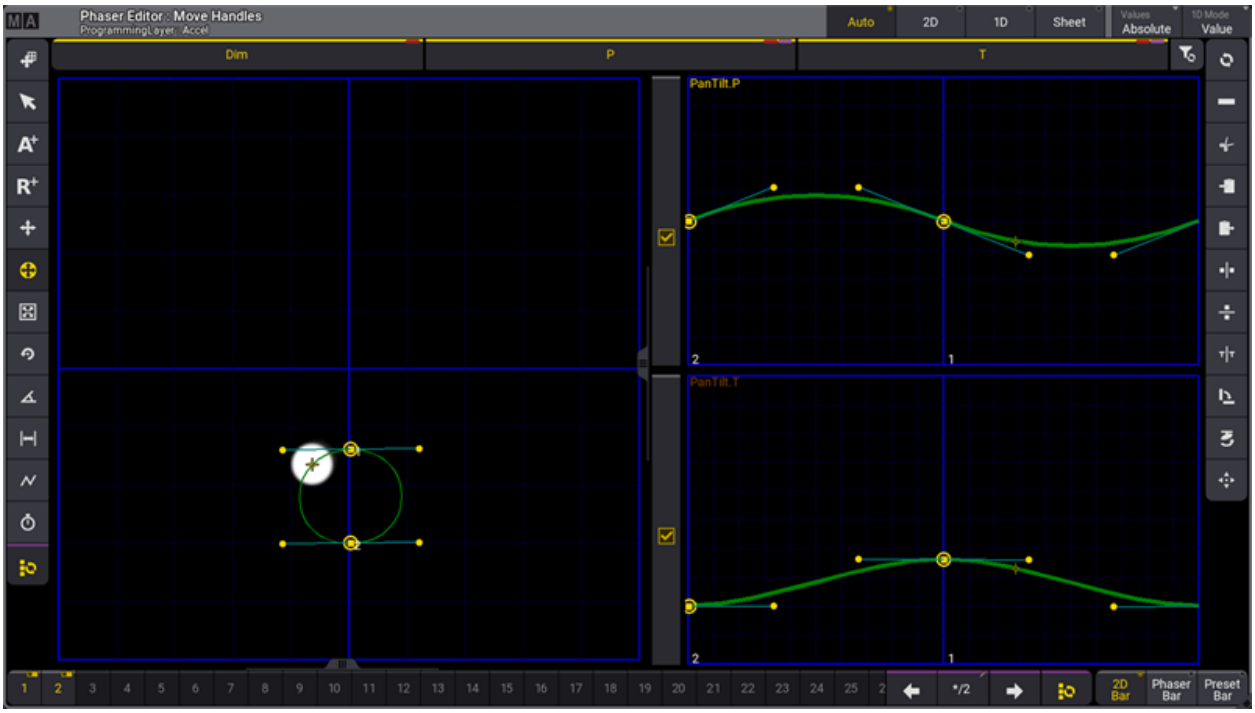
4. Tap somewhere else in the blue grid to create a second absolute position. Please tap 90 degrees vertically to the first point and do not cross the horizontal blue line - otherwise, the circle becomes a figure 8.



Two absolute positions - one in each step

Now, the fixtures move between the two absolute points (two steps).

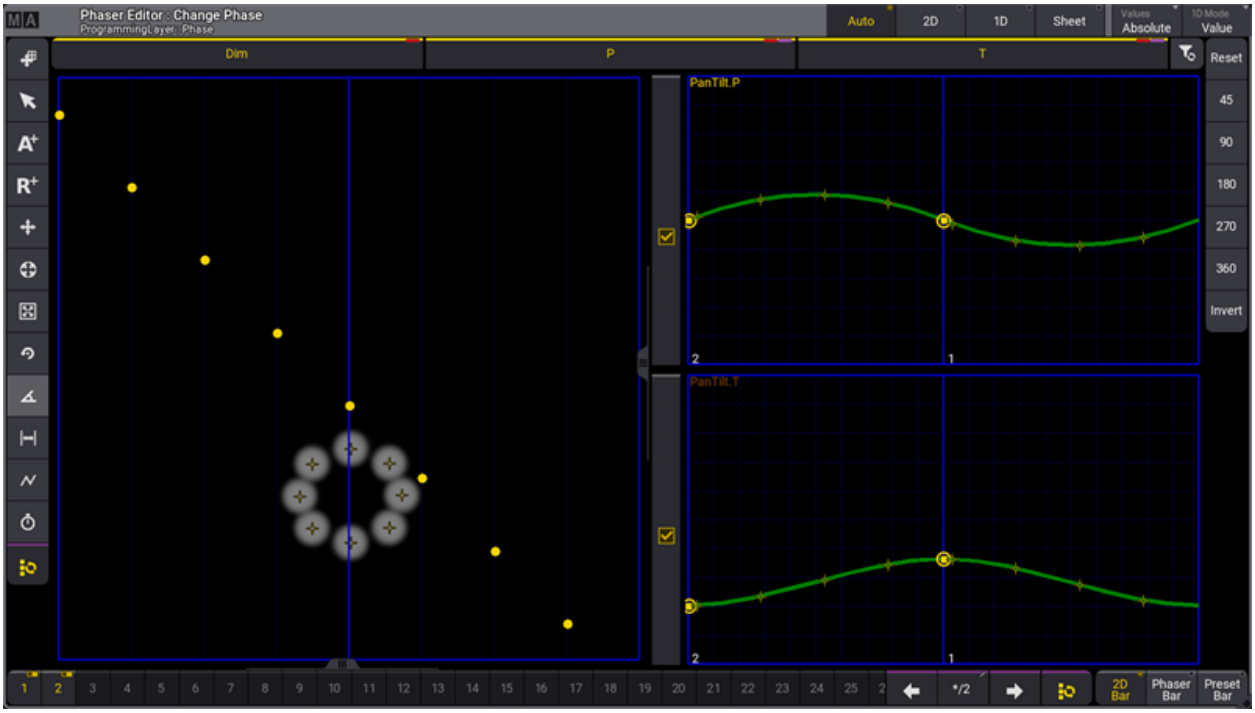
5. Tap the **Select All Steps** button ( ).
   This selects both steps.
6. Tap the **Move Handle** button ( ).
7. Tap one of the two points and drag horizontally.
8. Release the screen when there is a nice circular movement.



The fixtures now move in a circle

9. Tap the **Edit phase** button () on the left menu.
10. Tap 360 in the menu on the right
    This distributes the fixtures evenly along with the form.

The finished result could look like this:



Finished circle phaser

The **Change Phase** part of the phaser editor represents each fixture as a yellow dot in a grid where the horizontal axis is the phase value.

The circle phaser is now finished and it can be stored into a cue or a preset.

	Important:
	Following the steps above exactly results in an active dimmer value in the first step of the phaser. If you prefer that your circle phaser only contain position values, deactivate the dimmer attribute before storing.

1.34.4. Create a Circle Phaser Around a Position Preset

This topic presents an example of one method for creating a simple relative circle phaser.

	Hint:
	Combining relative phasers with separate absolute presets provides greater flexibility for creating dynamic looks.

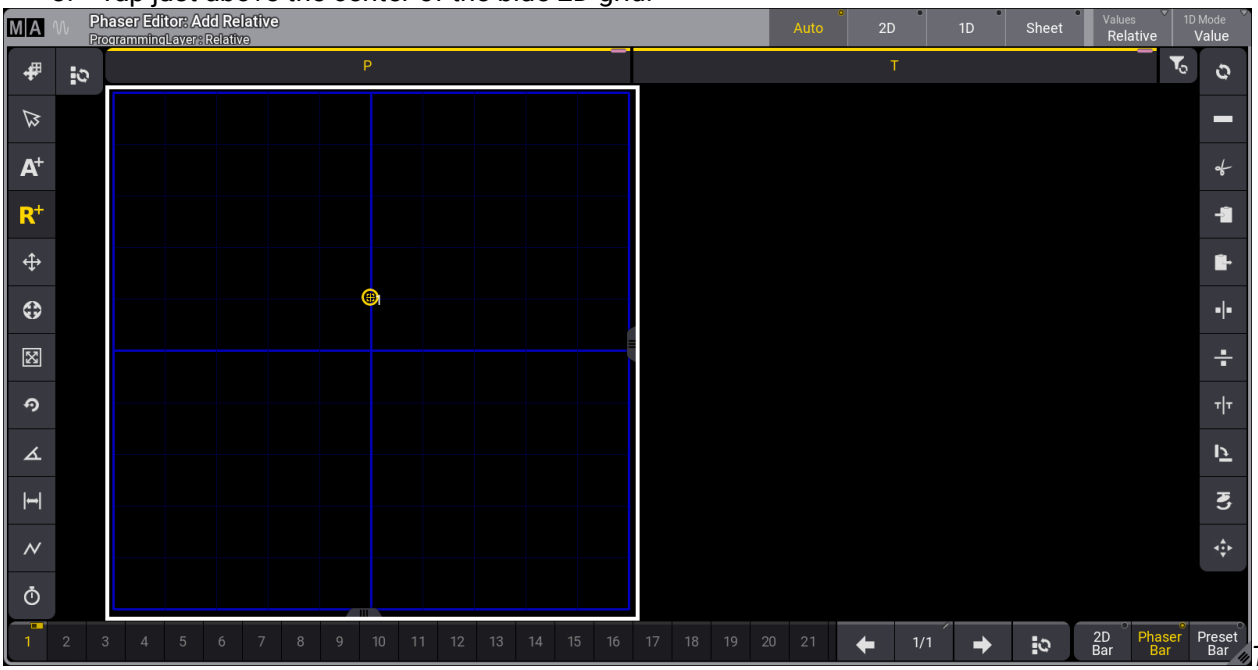
Requirements:

- Have a show with some moving lights patched. The following examples reference presets included in the demo show.
- An open **Phaser Editor** window is needed.
- Arranging the fixtures in the **3D window** can be useful, but it is not a requirement.

This example shows one possible workflow for creating a relative phaser. Alternate workflows can also produce the same results.

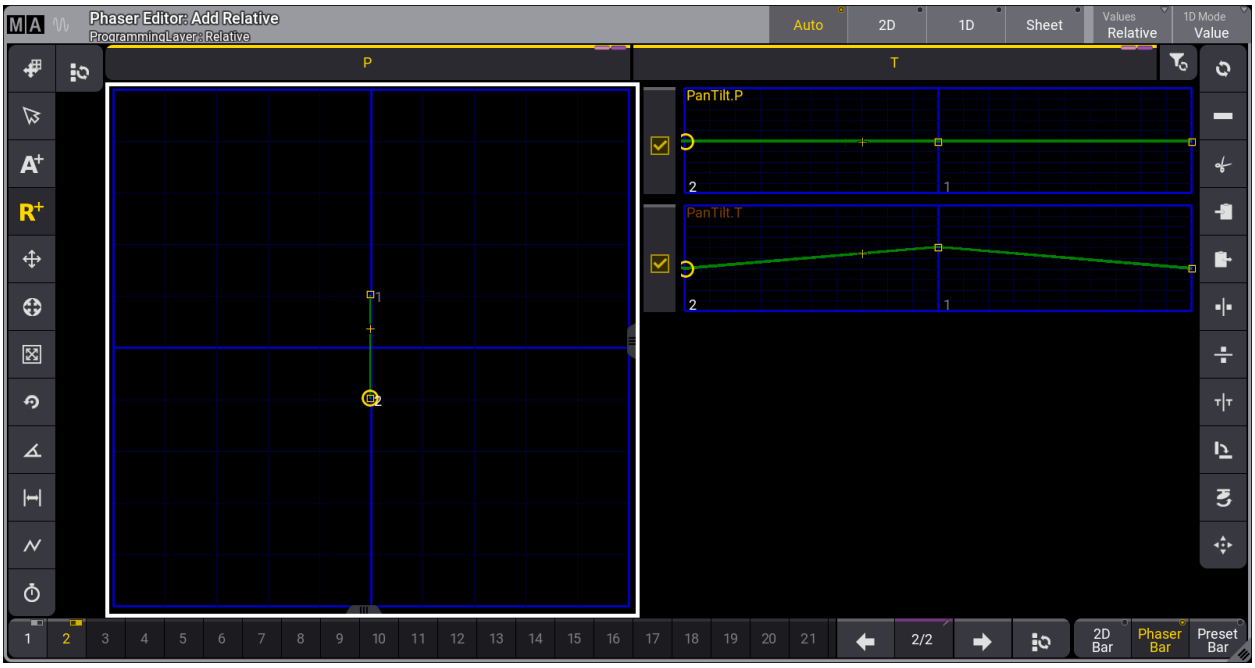
Follow these steps to create a relative circle preset:

1. Select all of the desired fixtures.
2. Tap R+ on the left side menu in the **Phaser Editor**.
3. Tap just above the center of the blue 2D grid.



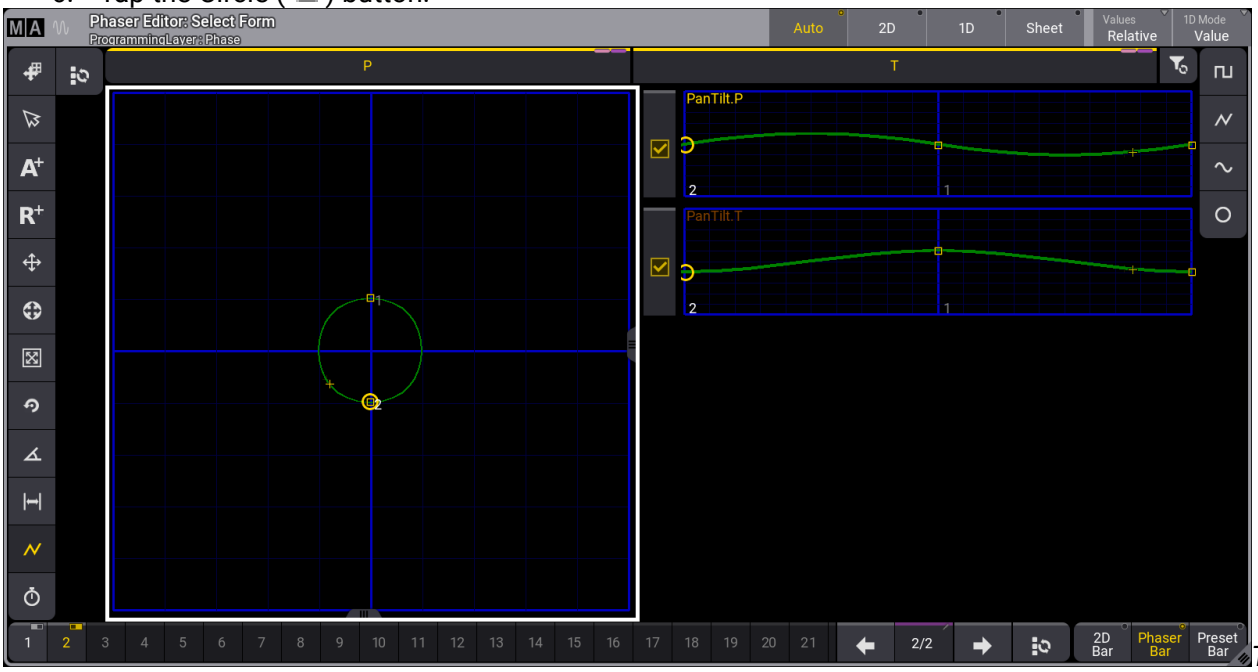
The first point in a relative circle

4. Tap just below the center of the blue 2D grid.
   This creates two relative position points.



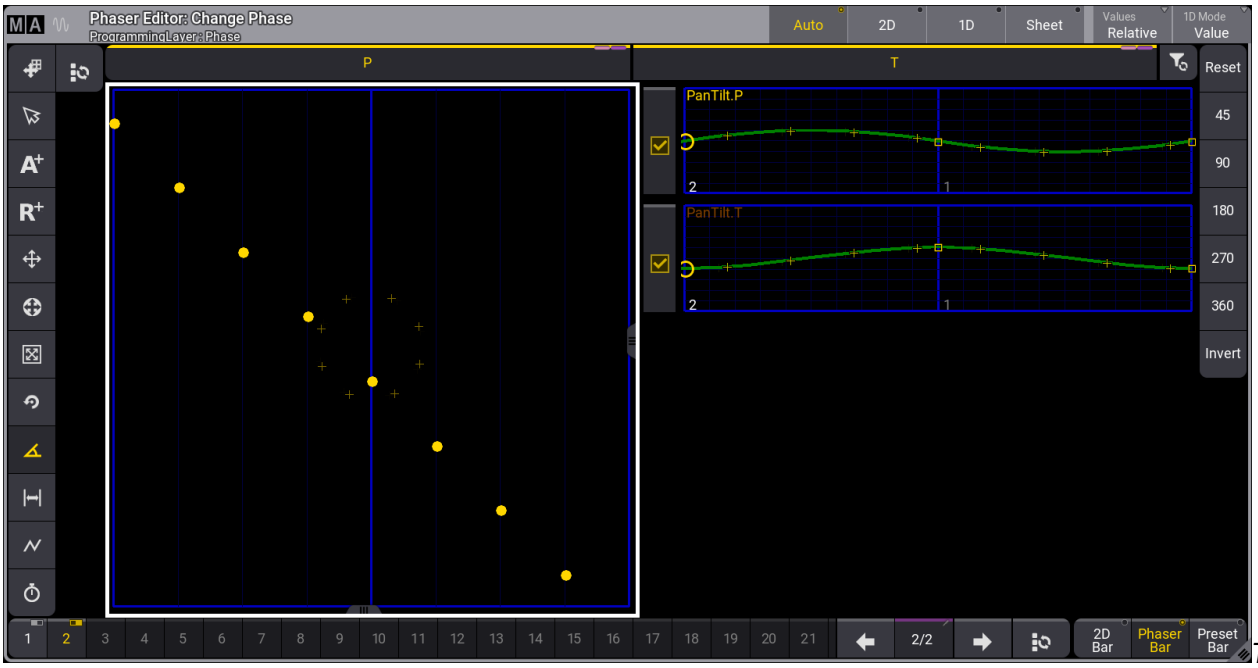
Two steps /points are the basis for the circle

5. Tap the Select Form ( ) button.
6. Tap the Circle ( ) button.



The relative circle with all selected fixtures moving simultaneously.

7. Tap the Change Phase ( ) button on the left menu.
8. Tap 360 in the menu on the right.



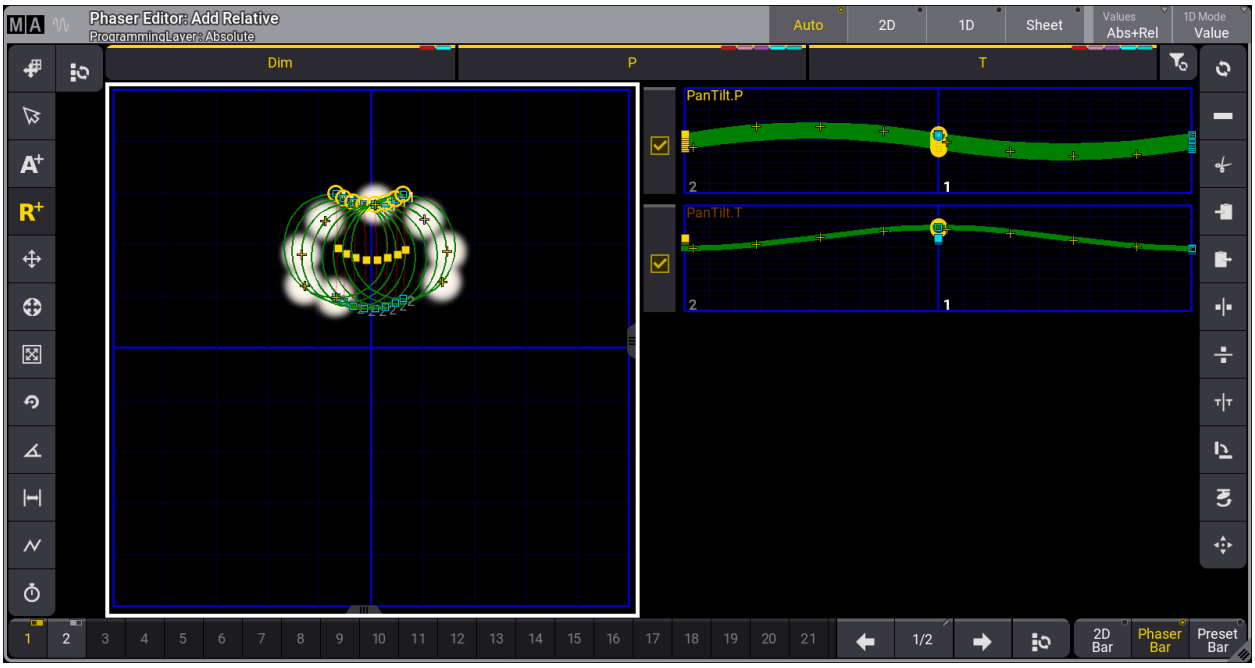
The relative circle with selected fixtures aligned evenly around the form.

9. Store this as a preset.

Follow these steps to create a cue combining this relative circle phaser with a static absolute position:

1. Select the desired fixtures.
2. Tap dimmer preset 2 "Open" to bring the fixtures to full.
3. Tap position preset 3 "Center" to move the lights to the downstage-center position.
4. Tap the new preset with the relative circle.

The resulting output shows the fixtures moving in a circle around the downstage-center position.



Fixtures moving in a circle around the downstage-center position.

Continue editing as desired or store this as a new preset or store as a cue.

1.34.5. Create Color Rainbow Phaser

This topic presents an example of one possible workflow for creating a rainbow color phaser. This example creates the rainbow chase using three steps with a primary color in each step. An additional example at the end shows an easy way to change the color used in one of the steps.

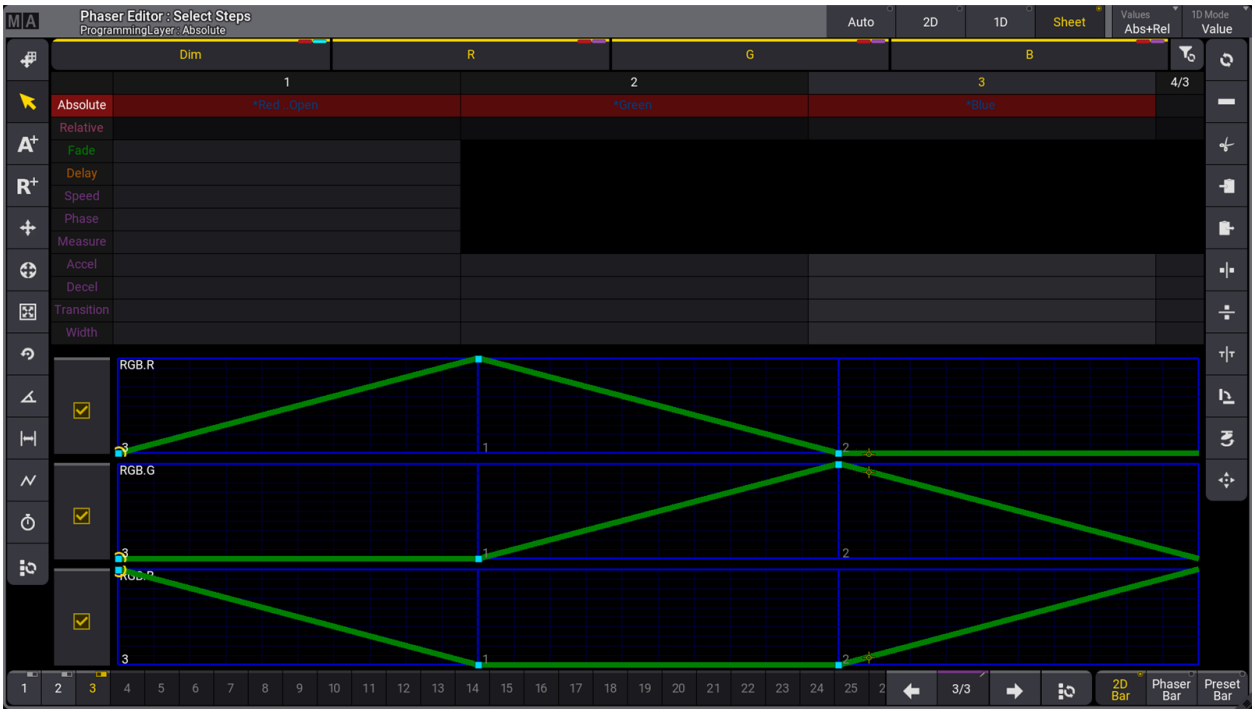
Requirements:

- Have a show with some fixtures that have a color mixing system (not color wheels with static colors). The steps below reference fixtures and presets included in the demo show.
- Have three color presets with Red, Green, and Blue colors stored using the color mixing system. The additional example also uses a White preset. These presets exist in the demo show file.
- An open **Phaser Editor** window is needed.
- An open Color preset pool for quickly calling the desired presets.
- Arranging the fixtures in the **3D window** can be useful, but it is not a requirement.

Follow these steps needed to create the phaser.

1. Select the desired fixtures. The following screenshots use fixtures 1 thru 8 in the demo show.
2. Bring the dimmers to full so the beams are visible.
3. Press and hold the X5 | Step key, and tap the red color preset.
4. While still holding the X5 | Step key, create the second step by tapping the green color preset.
5. While still holding the X5 | Step key, create the third step by tapping the blue preset.

The phaser editor shows the three steps and all referenced presets.



Phaser Editor with a three-step, three-color phaser

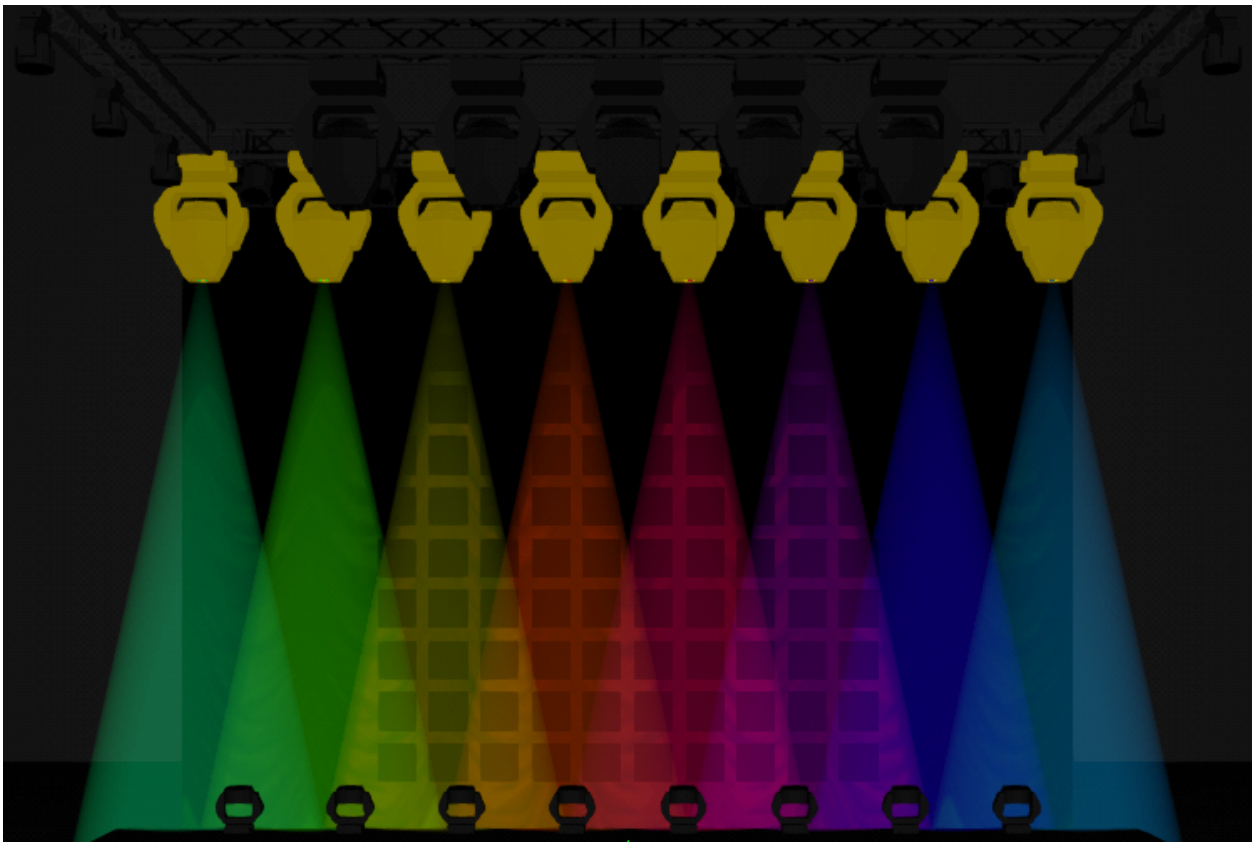
All the fixtures now change color together through the rainbow colors. If this is the desired result then store this in a cue or preset.

	Important:
	Following the steps above exactly results in an active dimmer value in the first step of the phaser. If you prefer that your rainbow phaser only contain color values, deactivate the dimmer attribute before storing.

To spread the rainbow across the selected fixtures, adjust the Phase:

1. Tap the Change Phase ( ) button on the left menu in the **Phaser Editor**.
2. Tap 360 on the right menu in the **Phase Editor** to spread the fixtures evenly.

The 3D window shows the result:

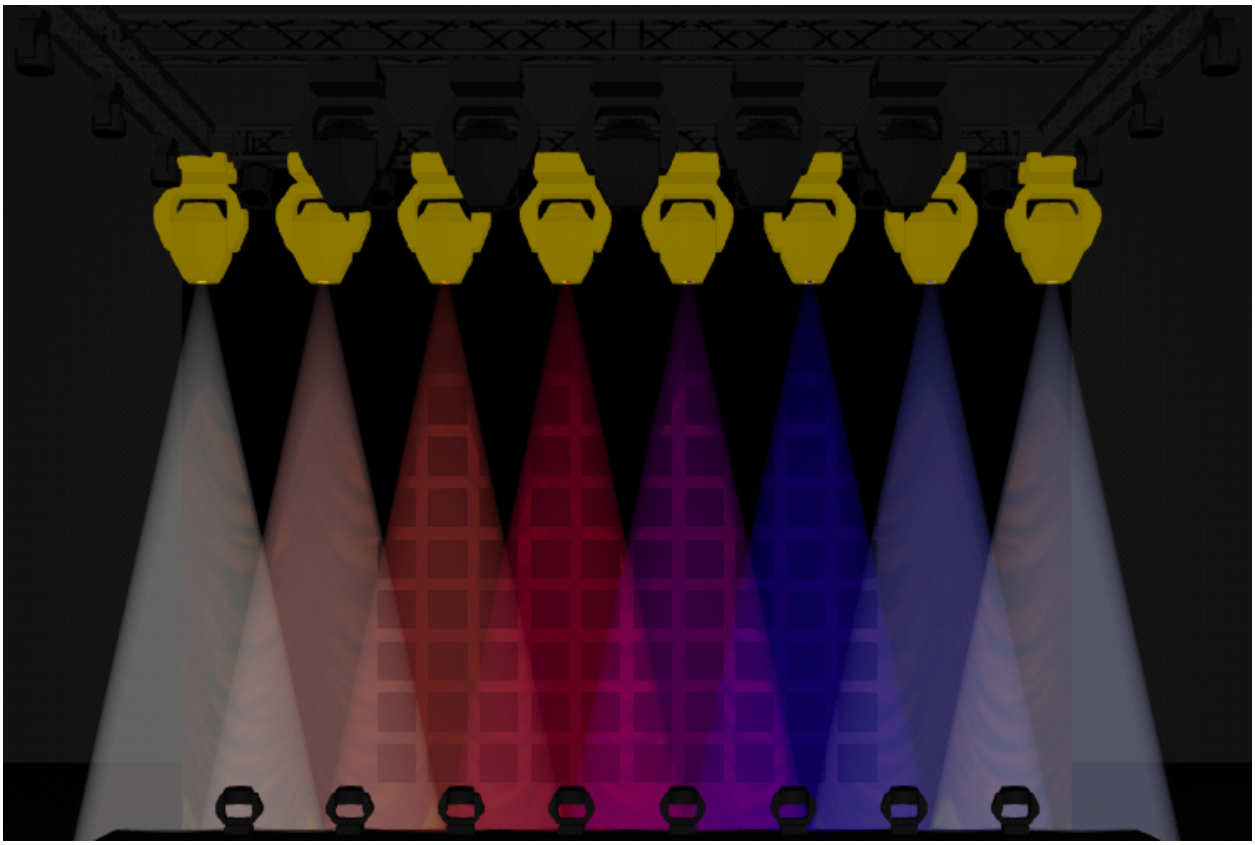


Rainbow chase across a range of fixtures

This can be stored as a preset or in a cue.

Quick Workflow for Changing the Preset Referenced in a Step

In the case that a different combination of colors is desired, the following workflow allows for such adjustments to be made quickly. With the rainbow from the above example still active in the programmer, use the following command to change the preset used in step two from green to white, so that the fixtures transition from red to white, then to blue: Press X5 | Step then 2 and tap the white preset.



Red, white, and blue phaser

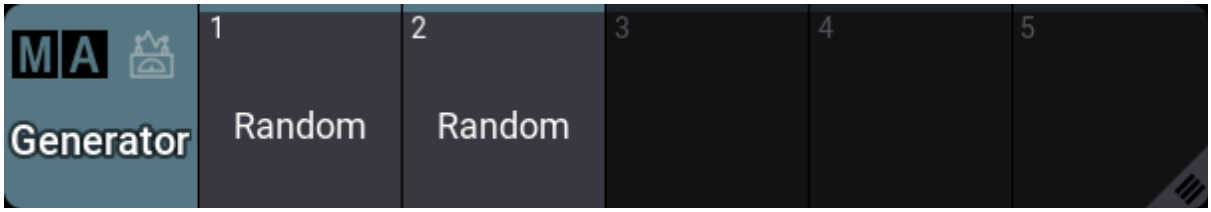
The resulting phaser now references the red, white, and blue presets.

1.35. Generator - Random

Generators are objects that allow a dynamic absolute value generation for special purposes, like randomization of absolute values on attributes.

Random

The generator type **Random** allows the randomization of absolute values of attributes in different ways. Randoms are organized in the **Generator** pool. The **Generator** pool is part of the data pools. To address the generator pool objects, use the **Generator keyword**.



Generator pool

The key to randoms are the different **variance** values. Without the variance, it would be a very simple phaser, but the variance allows the different values to vary between each "loop" of the random.

Randoms are called into the programmer and affect the defined attributes for the selected fixtures. This programmer information can be stored in presets or cues.

Randoms can only affect the values on the absolute layer.

The random generators have a lot of different settings that affect how the attribute output is randomized. This is a short description of all the different settings:

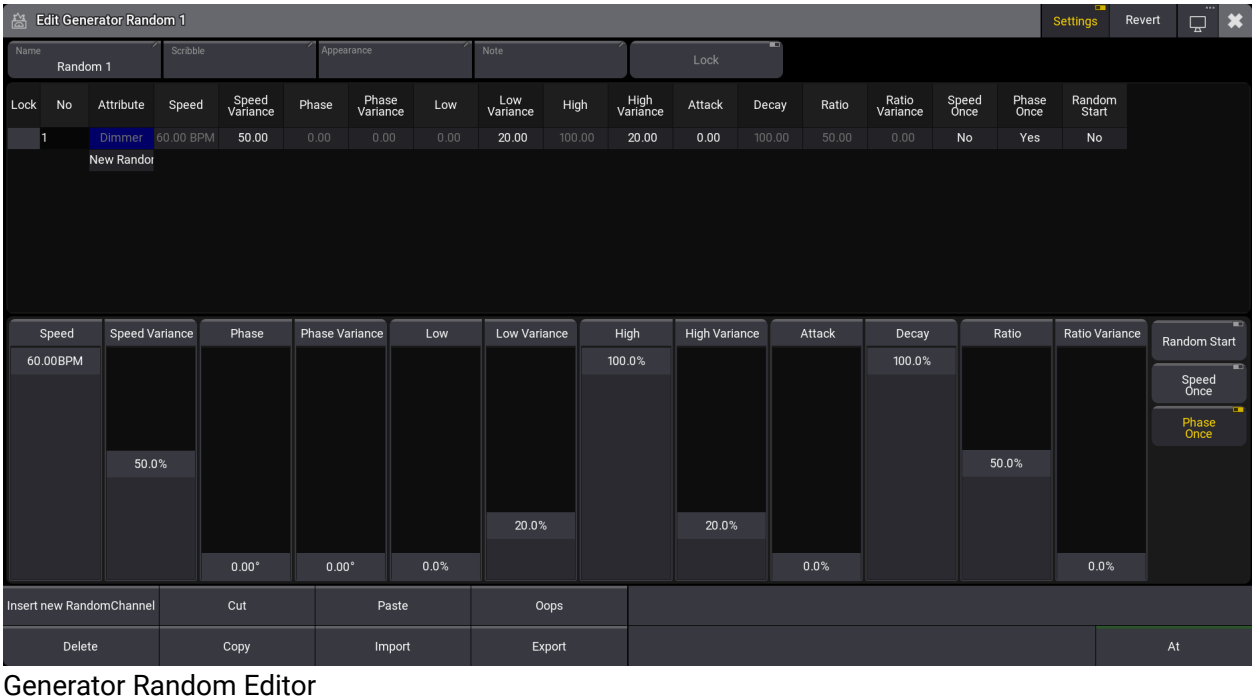
- **Attribute**:
  This is the attribute affected by the random generator.
- **Speed**:
  Defines the speed at which the attribute values are randomized.
- **Speed Variance**:
  Defines how big the **Speed** value may differ through the selection.
- **Phase**:
  Sets the phase of the random between 0° and 360°.
- **Phase Variance**:
  Defines the size of variance of the **Phase** value. If the selected fixtures use the random feature and this value is not 0%, their behavior will change.
- **Low**:
  Defines the lowest value the random can reach.
- **Low Variance**:
  Defines the degree of variation of the **Low** value. This adds some variation through the selected fixtures for the Low value. This variance acts only on the positive side of the Low value.

- **High**:
  Defines the highest value the random can reach.
- **High Variance**:
  Defines the size of value variance of the **High** value. This adds some variation through the selected fixtures for the High value. This variance acts only on the negative side of the High value.
- **Attack**:
  Defines the size of the linear transition towards the **High** value.
- **Decay**:
  Defines the size of the linear transition towards the **Low** value.
- **Ratio**:
  Defines the ratio of how long fixtures will use the **Low** or **High** value.
- **Ratio Variance**:
  Defines the amount of variance in the **Ratio** value.
- **Speed Once**:
  When **Speed Once** is set to No, speed changes are applied immediately to the running random. When it is set to Yes, the random must be called again to see the change.
- **Phase Once**:
  When **Phase Once** is set to No, changes to the phase parameters are applied immediately. By default, this setting is set to Yes.
- **Random Start**:
  When **Random Start** is set to No, the random will always start the same every time the Random gets started. This might be useful for environments that need a predictable start. When Random Start is set to Yes, the Random will start differently every time it will be started.

These settings are applied to a **Random Channel**. A Random can have multiple Random Channels.

Create a Random Generator

Randoms can be created by editing an empty pool object. This opens the **Generator Random Editor**.



Generator Random Editor

The top part of the editor contains a sheet area where the Random Channels are rows, and the different settings are columns.

The bottom part has on-screen faders that can be used to edit the selected rows.

The values can be edited on the sheet and fader parts.

There are buttons at the bottom for normal actions like Cut, Copy, Paste, etc. This includes an At button, which can be used to apply the random values to the current selection.

There is a special encoder bar when the editor has focus.

Generator Random Encoderbar

This encoder bar also allows editing the generator values.

The editor's title bar has a button called Revert. Tapping this discards all current changes to the valid values when the Generator editor was opened.

Using a Random

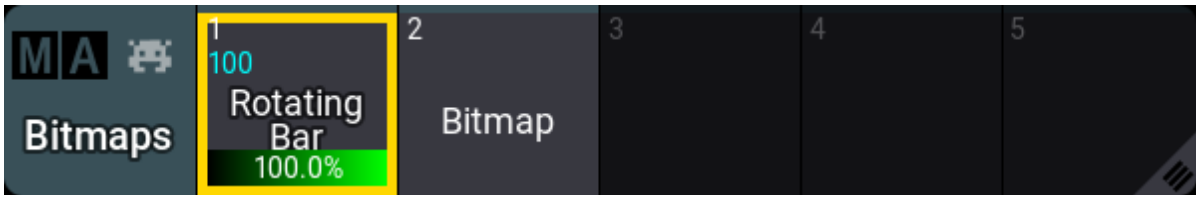
Random Generators are not dependent on specific fixtures. They affect attributes. Having a fixture selection and tapping a Random pool object applies the Random objects values to the attributes of the fixtures on the absolute layer. These are programmer values that can be stored or used as live playback.

The **Generator keyword** can also be used to apply Random Generators.

1.36. Bitmap

	Known Limitation:
	The system can overload and crash if a high-resolution (8K) video is previewed in the Layout Viewer. The video source should not exceed a resolution of 1920 x 1080 pixels.

The Bitmaps allow the use of media files (images, gobos, symbols, or videos) for mapping them to a selection of fixtures. Bitmaps are organized in the **Bitmaps** pool. The Bitmap pool is part of the data pools. To address the Bitmap pool objects, use the **Bitmap keyword**.



Example of the Bitmaps pool

	Hint:
	For testing purposes, three simple simple video clips are provided. These can be imported into the Video pool.
	Hint:
	If NDI input is used in a setup with several consoles and processing processing units, the NDI stream must be available at all calculating stations.

Bitmap Canvas

The canvas is an essential concept for Bitmaps.

To be able to output a Bitmap to fixtures, it is necessary to select fixtures and arrange them in the **Selection Grid**. The Bitmap applies the dynamic value changes coming from the media file to a canvas. This canvas is mapped to the selection grid.

The Selection Grid window shows the selected fixtures and their arrangement. It is a very useful programming tool. Learn more about it in the **Selection Grid topic**.

A purple rectangle in the selection grid represents the canvas. It is visible as long as one attribute of the selection uses a Bitmap as a value. The canvas should be in the same area as the arranged fixtures for the fixture to be affected by the Bitmap.

The size and aspect of the canvas can be changed using different settings in the Bitmap Configuration.

The selected fixtures should be arranged in the selection grid in a manner that makes sense for the desired output. The canvas can be adjusted automatically or by different built-in functions.

The Bitmap Editor is described below. The editor has some buttons related to the canvas and the fixture arrangement in the canvas area.

Tapping At applies the Bitmap to the current selection.

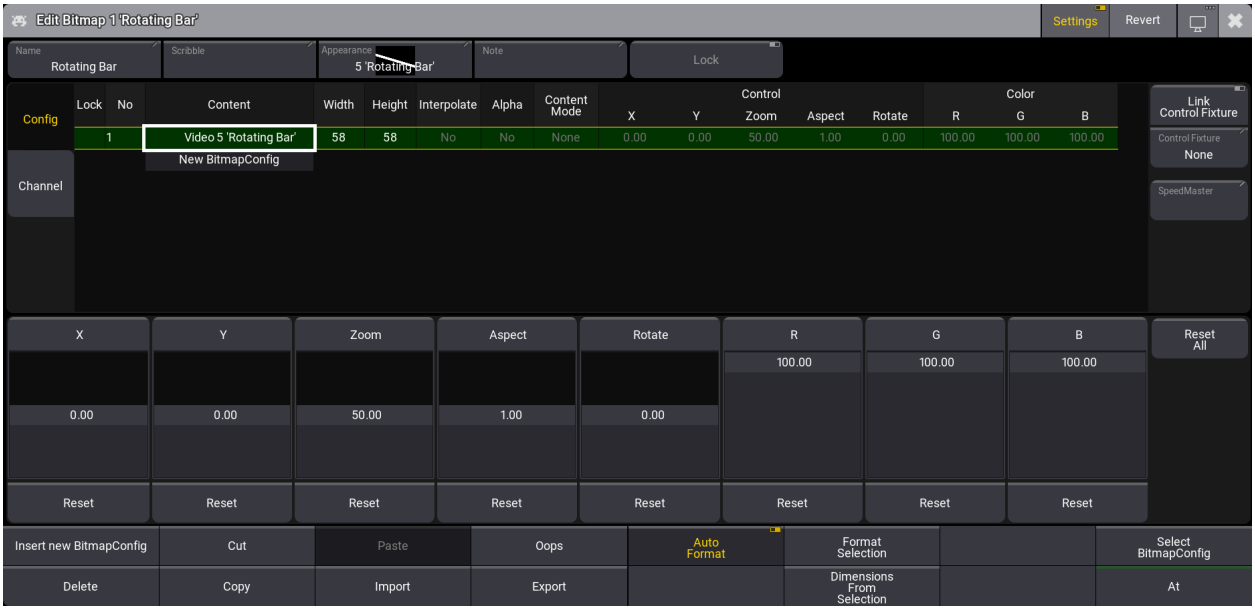
Tapping Dimensions From Selection changes the width and height of the selected Bitmap Configuration (see below) to match the current selection's size in the selection grid.

Tapping Format Selection scales the current selection in the grid so it fits as close as possible to the size of the canvas. The ratio of the selection in the selection grid is taken into account when scaling the selection.

Toggling On Auto Format automatically performs a "Format Selection" when the Bitmap is applied to a new selection.

Bitmap Editor

The best way to create a new bitmap is by editing an empty pool object. This opens the **Bitmap Editor**:



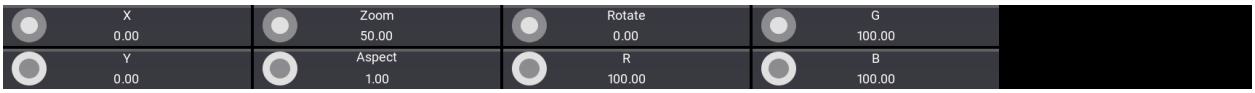
The Bitmap editor can be used to edit the Bitmap.

The top area of the editor gives access to the Bitmap Configurations or the Bitmap Channels. Tap the relevant tab on the left side to switch between the two. On the right side of this area, there are some buttons to add connections to a Bitmap Control Fixture (read more about this below) and a Speed Master.

Below this area are on-screen faders that give access to change the **Control** and **Color** attributes of the selected Bitmap Configuration. Each on-screen fader has a Reset button below. Tapping this will reset the associated fader value to the default value. A Reset All button resets all the attributes represented by the on-screen faders.

At the bottom of the editor are the usual editor buttons and some special buttons for the Bitmap editor. The special buttons are described in the following text.

The encoder toolbar changes to show the same attributes as the on-screen fader while the editor has focus.



This is the encoder toolbar shown when the Bitmap editor has focus.

The title bar has a Revert button. Tapping this discards all current changes to the valid values when the Bitmap editor was opened. There is also a Settings button which toggles the visibility of the bitmap settings.

Bitmap Configuration

The bitmap configuration is a sheet with columns containing settings and rows representing different media files. The bitmap can contain several rows with bitmap configurations but only plays the selected bitmap configuration. The selected row is marked with a green background color. A row can be selected by tapping it in the list and then tapping Select Bitmap Config in the bottom area.

The Bitmap Configuration has the following settings organized in columns:

- **Content**:
  This is the media file played back by the Bitmap Configuration.
- **Width**:
  This is the width of the canvas to which the content is mapped. The default value is 64 pixels or squares in the selection grid.
- **Height**:
  This is the height of the canvas to which the content is mapped. The default value is 64 pixels or squares in the selection grid.
- **Interpolate**:
  Smoothens the transition from fixture to fixture when playing back a media file.
- **Alpha**:
  The used media file's alpha channel is considered and made transparent when enabled.
- **Content Mode**:
  This setting has three different options:
  - **None** (default value):
    The media file uses the part of the canvas based on the media file size, the canvas size, and the zoom. The result can be that the media file only takes up a small part of the canvas.
  - **Clip**:
    The media file scales to use the whole canvas size.
  - **Wrap**:
    The media file is displayed several times across the canvas.
- **Control - X**:
  Moves the canvas on the X-axis in the selection grid.
- **Control - Y**:
  Moves the canvas on the Y-axis in the selection grid.
- **Control - Zoom**:
  Zooms the canvas in the selection grid and keeps the aspect ratio.
- **Control - Aspect**:
  Changes the aspect ratio of the canvas in the Y direction in the selection grid.
- **Control - Rotate**:
  Rotates the canvas in the selection grid.

- **Color - R**:
  Changes the red color of the played-back media files.
- **Color - G**:
  Changes the green color of the played-back media files.
- **Color - B**:
  Changes the blue color of the played-back media files.

Adding a new bitmap configuration will set the values for Control, Width, and Height to the values of the currently selected bitmap configuration.

Bitmap Channel

The bitmap channels define which fixture attributes respond to the media file content.

The bitmap channel sheet has different fixture attributes in rows and the channel settings in columns.

These are the settings:

- **Attribute**:
  The fixture attribute defines what the bitmap should affect.
- **Source**:
  The source defines which part of the media file should be analyzed to create the values for the fixture attribute.
- **Value - Low**:
  Defines the low value of the fixture attribute to which the source should be mapped.
- **Value - High**:
  Defines the high value of the fixture attribute to which the source should be mapped.

Attributes can be added to change what the Bitmap effects. For instance, Tilt can be added, and a pixel's intensity in a video can be used to dynamically change the tilt between the low and high values.

Bitmap Control Fixture

One or several special Bitmap Control fixtures can be added to the patch.

The Bitmap Control fixture has attributes to change the Color values and some of the Control values in the Bitmap Configuration. It can also be used to change the selected Bitmap Configuration.

The right side of the Bitmap editor has a Control Fixture button. Tapping this opens a small select pop-up where one of the patched Bitmap Control fixtures can be selected. Selecting a fixture here automatically toggles On the Link Control Fixture. When this is On, then the selected Bitmap Control fixture can be used to control the Bitmap Configuration values. This fixture has programmer values that can be stored in cues or presets.

A suggested workflow could be the following. It assumes normal patched fixtures and an existing Bitmap.

1. Patch at least one fixture using the MA Lighting Bitmap Control fixture type. Give it at least a FID number.
2. Select the fixtures which should use the Bitmap and arrange them in the Selection Grid.

3. Tap the desired Bitmap in the pool or use the keyword to apply the Bitmap to the selected fixtures.
4. Select the Bitmap Control fixture.
5. Apply the desired Bitmap to the Bitmap Control fixture by tapping the Bitmap generator object in the pool or by using the At Bitmap syntax. This will set the Bitmap Generator as the value on the Object attribute in the Gobo feature group of the Bitmap Control fixture. To apply a different Bitmap Configuration, scroll through the Config attribute.

The attributes X, Y, R, G, B, and Zoom of the Bitmap Control fixture modify the corresponding settings of the Bitmap generator.

Speed Master

The speed of the Bitmap is relevant when the content is a video. As a default, the Bitmap will play the video using the speed of the video. A Speed Master can be assigned to the Bitmap by tapping SpeedMaster on the right side. This Speed Master will then adjust the playback speed of the Bitmap.

Using Bitmaps

Bitmaps must be applied to a selection of fixtures. This can be done by tapping a Bitmap with a selection of fixtures. It can also be done using the following syntax:

At Bitmap ["Bitmap_Name" or Bitmap_Number]

This applies the Bitmap to the relevant attributes, based on the Bitmap Channels, in the programmer.

These programmer values can be stored in presets or cues.

The Bitmap Fixture connected to a Bitmap can also be stored in Presets and Cues.

1.37. XYZ

Fixtures can be programmed using XYZ values instead of PanTilt values.

This offers some advantages over standard PanTilt programming. For instance, when fixtures move from one position to another, they do it in a straight line from position A to B when using XYZ values. When using PanTilt, the fixtures fade from DMX value A to B, this means that two fixtures often do not follow the same path.

XYZ values can, just like PanTilt, be relative and absolute.

PanTilt values can be mixed with XYZ values in a sequence.

All combinations of PanTilt and XYZ, and absolute and relative, are supported.

When XYZ position parameters are added, a Distance attribute is also added. Distance is connected to Pan and Tilt. Pan and Tilt values can, in combination with a Distance (from the fixture) and the fixture's known position, be calculated into XYZ position information. And vice versa.

	**Important:**
	When mixing absolute PanTilt with relative XYZ, the value of the Distance attribute in the PanTilt feature is important. This value defines where the relative XYZ values are added.

Calling XYZ presets with absolute values or hard values in the absolute layer knocks out absolute PanTilt values. The same is true with relative values. Absolute PanTilt values can be combined with relative XYZ values and vice versa.

When knocking in attributes of one position type (PanTilt or XYZ) while the other one has active values in the programmer or is played back, the values will be converted to the knocked-in position type and converted to match the same position on stage.

If several playbacks with values for both positioning systems run, Pan, Tilt, and Distance will be knocked in.

If the position feature group is knocked in while a transition from one system to the other is happening, Pan, Tilt, and Distance will be knocked in.

On FeatureGroup "Position"."XYZ" knocks in all XYZ attributes, while **On EncoderPage "Position".2** will knock in only the attributes that are currently on this encoder page (in most cases, this will be X, Y, Z, and Flip).

The **OffWhenOverriden** setting of sequences turns off playbacks with relative PanTilt when relative XYZ is played back, and vice versa.

XYZ Attribute Range

The absolute XYZ attributes range from 0 to 100 % of the space they use.

The relative XYZ attributes range from -100 to 100% of the space they use.

The default space for the absolute values is the default stage space volume. Read about stage spaces in the **Stages topic**.

The physical readout of the XYZ attributes displays the values in meters of the stage (when MArker is set to 0) or of the size of the target space of the selected MArker fixture.

Turning the encoders of the XYZ attributes by one click in the encoder resolution Coarse changes the value in physical readout and natural readout by 1 meter. The other encoder resolutions and encoder factors are also based on this size.

When a MArker fixture is set up on the relative layer for the MArker attribute, the relative XYZ values are relative to the movement space of the MArker fixture.

When a fixture does not have XYZ values active in the programmer or played back, activating the MArker attribute will set the values for XYZ in the programmer to 0.

When the value of the MArker attribute on the relative layer is 0, the absolute MArker value will be used.

Important:
XYZ needs to be activated for fixtures that want to use MArker fixtures. See the **MArker Fixture topic** for more information about this special fixture.

Subtopics

- **Activating XYZ for Fixture Types**
- **MArker Fixture**

1.37.1. Activating XYZ for Fixture Types

XYZ needs to be activated for a fixture type to get access to the XYZ attributes.

	**Important:**
	Turning On XYZ adds extra virtual parameters for each fixture using the fixture type mode. These extra virtual parameters do not count against the parameter limit, but they are shown in the **System Info window**.

XYZ is turned on in the Fixture Type Editor.

1. Press Menu.
2. Tap Patch.
3. Tap Fixture Types in the menu on the left.
4. Select the fixture type that needs XYZ.
5. Tap Edit in the menu at the bottom.
6. Edit the XYZ cell for the desired DMX Mode so it says "Yes".
7. Close the editor and the patch and save the changes.

Now, XYZ can be used for the selected fixture type mode.

XYZ is activated for the selected fixture type mode. So make sure it is activated for the mode that is used.

1.37.2. MArker Fixture

A fixture type called **MArker** from the manufacturer MA Lighting is part of the fixture library for the MA source.

A MArker is a virtual fixture, that allows operating (for instance, moving around or rotating) objects such as fixtures, actors, or stage props in the 3D stage environment. Learn more about this in the **3D window**.

It can also function as a target for XYZ-enabled fixtures. Learn how to activate XYZ in the **Activating XYZ for Fixture Types topic**.

MArker fixtures have a **Target Space** in the Stage. The **Space** defines the volume around the MArker. This space is also the boundary for X, Y, and Z attribute values for the fixtures pointing to the MArker. For instance, if the target space is 2 meters wide on the X-axis, then a fixture pointing to this maker can move inside this 2-meter wide space volume. Setting the X attribute to 0 will move the fixture to the minimum X position in the MArkers target space. The default size for a target space is 200 meters on the X and Z axes (very much bigger than the default stage space).

Moving MArker fixture also has a Movement Space. This defines the space volume in which the MArker fixture can be moved. The default values for this space are also 200 meters in X, Y, and Z.

Read more about the spaces in the **Stage topic**.

MArker as an Object Mover

For each object or group of objects that shall be moved separately, a MArker fixture needs to be added to the patch in the **Moving** mode.

As soon as a MArker fixture is patched, other patched fixtures can be added as children of the MArker. After applying these changes to the patch, the MArker fixture can be operated like any other fixture in the show.

The MArker fixture provides these attributes:

- X, Y, and Z are used to move the MArker and all its children together along the corresponding axes.
- Rot X, Rot Y, and Rot Z rotate the MArker and all its children together around the corresponding axes.

These attributes are located within the features XYZ and Rotation within the feature group Position in the encoder bar.

Select the MArker fixture and turn the encoders for the described attributes in order to see the MArker and its children moving around in the 3D window. The children of a MArker are always moved around relative to the position set up of the MArker itself.

This means that if the fixtures are already at a height of 5 meters and then attached to a MArker that is at a height of 0 meters, then the MArker needs to move below zero to move the fixtures below 5 meters.

	Important:
	A MArker fixture needs to have a DMX address patched in order to be able to see its changes within the 3D window.

MArker as a Target

The MArker fixtures can be a target for other fixtures that are XYZ enabled. Read about activation in the **Activate XYZ for Fixture Types topic**.

The MArker fixtures need to have the MArker IDType in the patch and a CID number.

The MArker fixtures can be of the Moving or Still mode. Still markers can be positioned using the patch or any other method described in the **Position Fixtures in the 3D Space** topic. Still MArkers are meant to be used for positions in the 3D space that do not move around. Moving MArkers are meant to be used with positions that can move in the 3D space.

The fixtures with XYZ enabled have a MArker attribute. Giving this attribute a number makes the fixture point to the MArker fixture with the matching number.

Moving the MArker, makes the light fixtures pointing at the MArker move the beams to match the MArker fixtures' position.

If the light fixtures pointing to MArker fixture, are be moved using a different MArker fixture while pointing to a MArker, then the light fixture will try to keep the light beam pointing to the MArker.

Moving the MArkers by a Tracking System

The input of PSN trackers can be linked to moving MArkers. This allows moving around the MArker and its children via an external tracking system.

To do so, set up the number of markers needed, and add the desired children to them within the grandMA3 software.

Then set up the PSN system and configure the **PSN input** within the grandMA3 software. Within the PSN menu, the columns called IDType and ID allows entering the MArker IDType and ID of the MArker fixture that shall be linked to the input of each tracker.

The column DMX Priority defines at which level of the grandMA3 playback priorities the PSN data shall be processed. This allows overwriting the input of the PSN system by using a sequence with a higher priority if needed.

	Important:
	The DMX universes the marker fixtures are patched to which are receiving data from PSN trackers need to be set to the **merge mode Prio**.

1.38. Tags

Tags allow you to organize, link and cross-reference objects throughout the software. They are also a great tool for busking shows. All objects that have the same tag can be triggered or selected together.

Tags are organized in the Tags pool. To open this pool, see the Pools tab in the **Add Window** pop-up.

Tags Pool
The default action for objects in the tags pool is **ListReference**.

Assign Tags

Tags can be assigned to other objects, like for example sequences or groups. It is possible to assign a tag to another tag.

Multiple tags can be assigned to the same object.

There are multiple ways to assign tags to objects:

- **Tag Pool Objects**

1. Press Edit and tap on an object in the Tags pool. The tag editor opens.
2. Tap Add New Tag Reference. A pop-up opens.
3. Select an object.
4. Tap Assign.

- **Pool Objects**

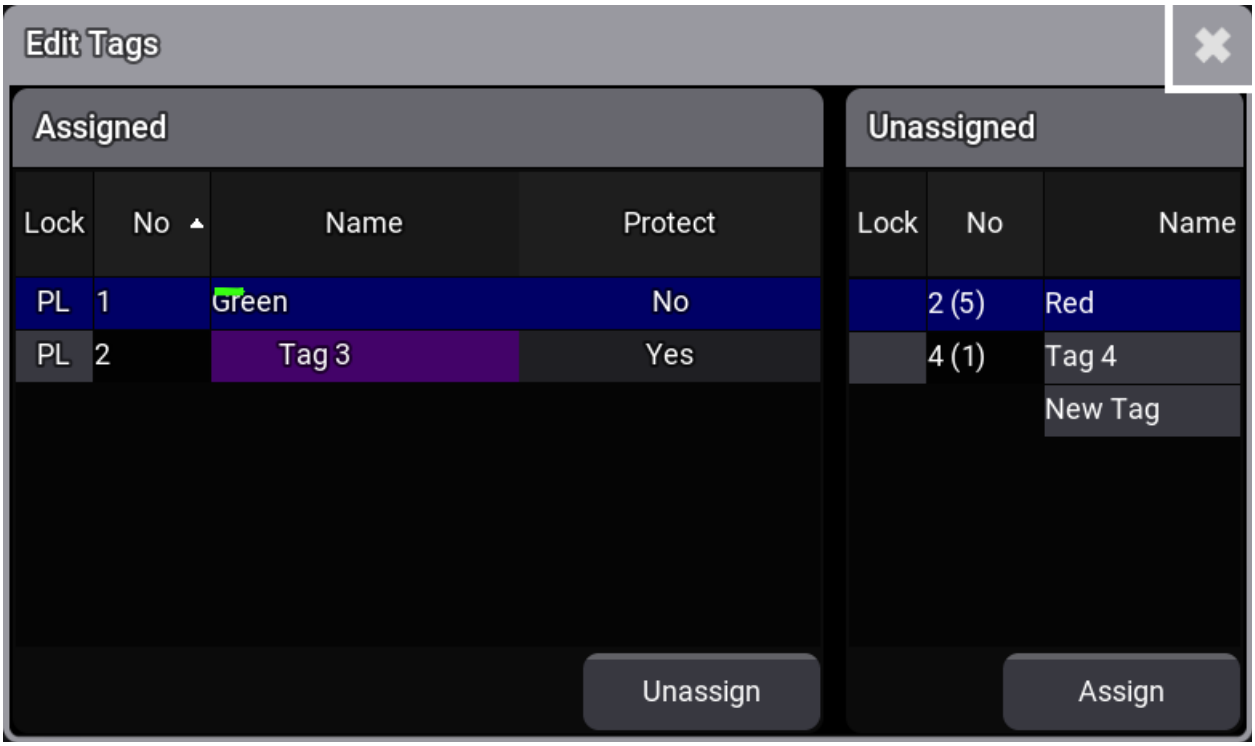
1. Open the **swipey commands** on the pool object and select Edit or Edit Setting. A pop-up opens. For some pool objects, for example sequences or macros, additionally tap Settings in the title bar of the pop-up.
2. Tap Tags. The Edit Tags pop-up opens.
3. Select a tag and tap Assign.

- **Other Objects**
  This applies for example to recipes, cues, or macro lines.

1. Edit the cell in the Tags column. The Edit Tags pop-up opens.
2. Select a tag and tap Assign.

- **Command Line**
  For more information on assigning tags using the command line, see **Tag Keyword**.

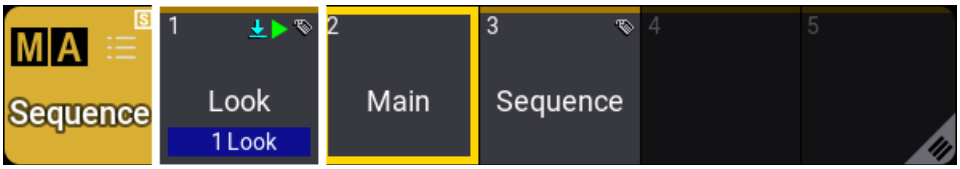
Already assigned tags can also be unassigned using the options listed above.



Edit Tags pop-up

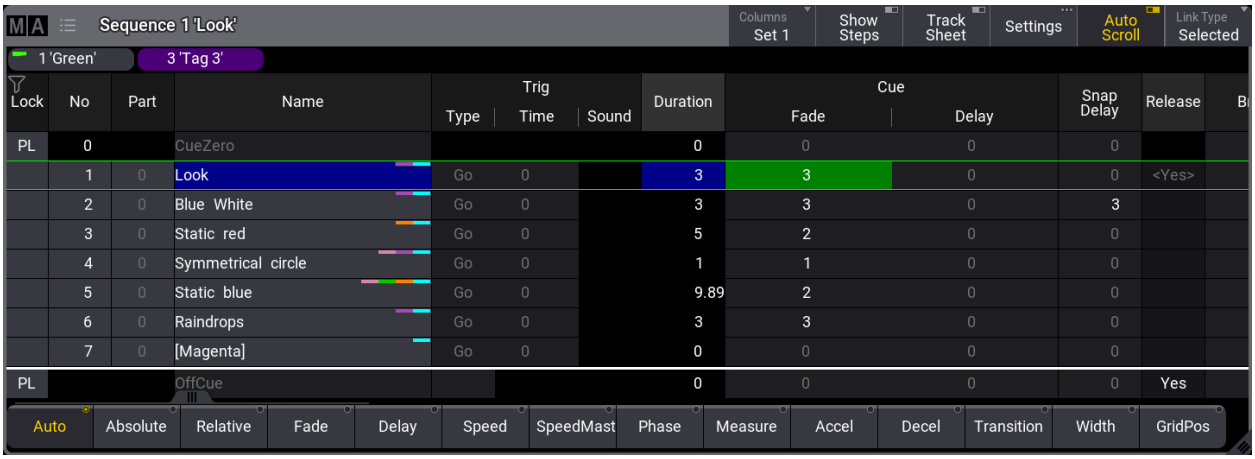
The left side of the Edit Tags pop-up shows tags that are assigned to the corresponding pool object. The right side shows tags that are not assigned.
In the pop-up, tags can also be locked and unlocked and protected against Kill Instant and Kill Delayed (see **below**).

When a tag is assigned to a pool object, is displayed on the pool object. The names and numbers of assigned tags are displayed on Tags in the pool object settings.



Sequence pool with tags assigned to sequences 1 and 3

In the sequence sheet and editors like the sequence editor or the macro editor, the assigned tags are displayed in the area between the title bar and the grid. They can be edited and perform pool actions like the objects in the Tags pool. The background color of the appearance assigned to the tag defines the background color of the tag displayed in the editor.



Sequence sheet with two assigned tags

Edit Tags

To edit a tag, press Edit and tap on an object in the tags pool.

The Tag Editor shows the objects the tag is assigned to in the grid offering information about:

- **Datapool:** Shows the corresponding Data Pool.
- **Class:** Shows the object type of the assigned reference.
- **No:** Shows the number of the corresponding pool object.
- **Name:** Shows the name of the pool object.
- **Protect:** The assigned reference is protected against Kill Instant and Kill Delayed. The default is **No**.

Tag Editor

Multiple objects can be assigned to a tag at the same time using Add New Tag Reference in the tag editor. In the Add Tag References pop-up, multiple objects can be selected and assigned consecutively. To do so, select an object line and tap Assign.

If you enable Settings in the title bar of tag editor, Name, Scribble, Appearance, Tags, Note, and Tag Type can be set and Forward Commands can be toggled on or off.

The following Tag Types can be selected:

- **Kill Instant:** Other playbacks using the same tag will start their OffCue immediately when starting the sequence.
- **Kill Delayed:** The sequence that was started will complete its fade in first and then the other playbacks using the same tags will start their OffCue.

For tag types to function, the tag and tag type need to be set before triggering the sequence. Otherwise, the tag type will work as soon as the corresponding sequences have been triggered once.

When a playback is started by a tag, the Trigger column in the **Off Menu** and **Running Playbacks** window reports the tag.

	Hint:
	Kill Protect does not protect a sequence from being disabled by Kill Instant or Kill Delayed executed from a tag. For more information on Kill Protect see **Sequence Settings** and **Kill Keyword**.
	Hint:
	For temporary playback actions (Flash, Temp, Swap, and Black) Kill Instand and Kill Delayed do not switch off other playbacks permanently.

If Forward Commands is toggled on, **playback commands** can be executed for all references of a tag. It is on by default.
This applies to >>>, <<<, Go+, Go-, Goto, Halfspeed, Load, On, Off, Pause, Rate1, Speed1, Toggle, and Top.

Examples

Requirements:

- Have a show with some lights patched, for example the Simple_Show.
- An open 3D Viewer is recommended to see the effects of the tags more clearly.

Create three individual sequences 1, 2, and 3 for fixtures 1, 2, and 3 with the same cue:

- Dimmer 100%
- Fade 2 seconds

Set up tag 1:

1. Edit Tag 1:
   a. Press Edit.
   b. Tap on the first pool object in the Tags pool.
2. Add tag references:
   a. Tap Add New Tag

Editor Tag 1

References.
b. Select sequences 1 thru 3.
c. Tap Assign.
3. Set Tag Type to **Kill Instant**.

Result: If you trigger one of the sequences with Tag 1 assigned, the other sequences are switched off immediately.

Edit Tag 1 and change the Tag Type to **Kill Delayed**.

Result: If you trigger one of the sequences with Tag 1 assigned, the other sequences are switched off after the triggered sequence has completed its fade in.

See the effects of the tag types Kill Instant and Kill Delayed in this video:

Edit Tag 1 and set Protect to **Yes** for Sequence 2.

Result:

- When triggered, Sequence 1 switches off Sequence 3 but not sequence 2.
- Sequence 2 switches off Sequence 1 and Sequence 3.

See Protect being applied for a sequence in the following video:

1.39. Macros

Macros

What are Macros

Macros are commands stored in a pool object.

The commands can be simple, very complex, and everything in between.

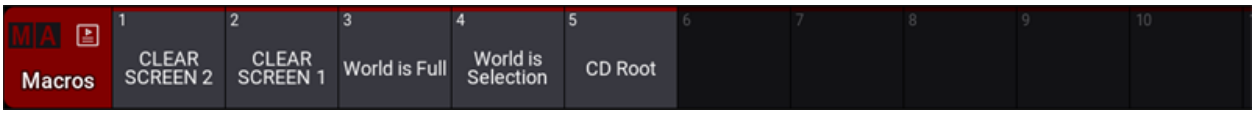
Macros can make programming faster and more convenient. They allow you to perform trivial or complex operations with a button push.

Macro Pool

Macros are stored in the **Macro Pool** but can be assigned to physical keys and view buttons. The macro pool is shared between all users in the show file. For example, User A stores a macro on pool object number 10, it will also be available for User B as macro number 10.

A Macro Pool can be created like any other window. **See Pools Window topic**. It is under the **Data Pools** tab.

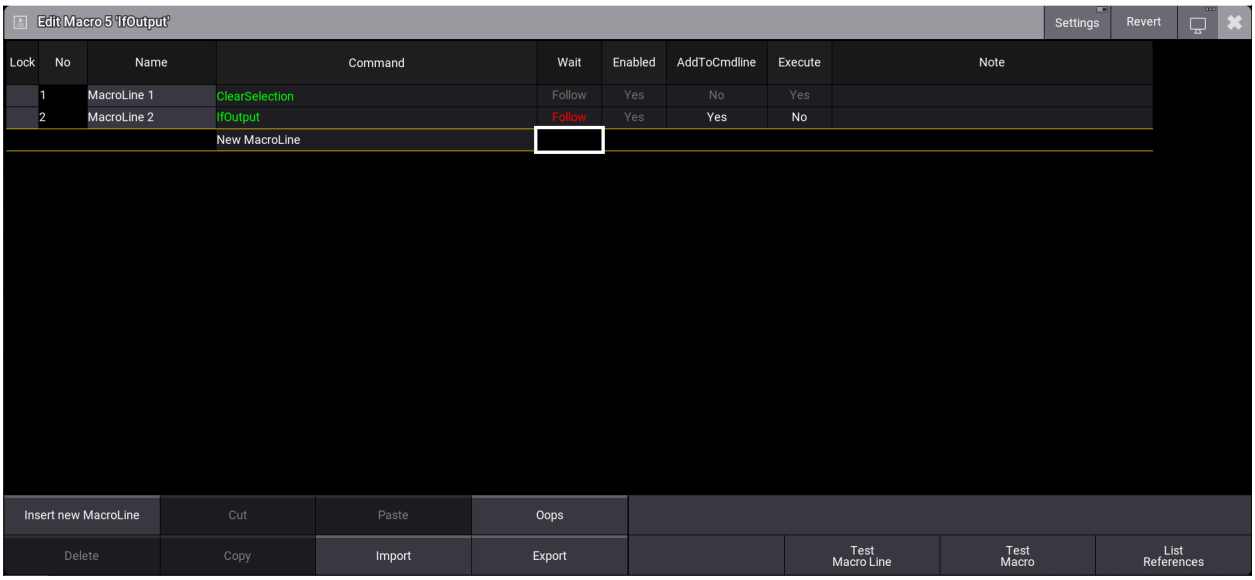
The Macro Pool could look like this:



Macro pool

Elements in a Macro

A macro consists of one or more rows.



Macro Editor

Each row has information about a **Command** the row executes.

Each row has several cells that specify how the row is handled:

The **Wait** cell adds a wait time before moving to the next row in the macro. This wait time is added after the command in the row is executed. It is relevant if there is more than one row in a macro. The first row will execute, wait the specified time, and then execute the next row. There are two special wait commands. They are not a time value but a special value:

- **Follow**: This is the same as having a wait time of 0. As soon as this row has executed the command, the next row will be executed.
- **Go**: This special value pauses the macro after this row until it receives a new Go+ command.

The **Enabled** field indicates if the row is enabled. This is a **Yes** or **No** field. If it is **Yes**, it is executed when the macro row is triggered.

The **AddToCmdline** function makes it possible to append the command from the macro line to existing content in the command line. This is a **Yes** or **No** field. If set to **Yes,** it is added to the command line.

The **Execute** field defines whether the macro row is automatically executed. If set to **Yes**, the row is executed (An automatic Please is executed at the end of the row). If set to **No**, the command will be placed on the command line, ready for user interaction.

Importing Macros from the Library

A factory library of macros can be imported using the import command.

For more information on how to import macros, read the **Import Macros** topic.

Read more about importing in the **Import Keyword topic**.

Subtopics

- **Command Editor**

- **Create Macros**
- **Import Macros**
- **Edit Macros**
- **Assign Macros to Keys and Buttons**
- **Variables**
- **Example Macros**

1.39.1. Command Editor

Command Input

When editing a command, for example, in a macro, a cue, or an agenda event, the **Edit Command** pop-up opens:

The entered command is previewed to see if it will be interpreted by the software as intended.

Preview Variables

Enable Preview Variables to display the content of a specific variable. The content can only be displayed if the variable exists:

For more information on Variables, see **Variables**.

Resolve Executor Assignments

Resolve Executor Assignments Disabled

In the following example, Sequence 1 'FirstSong' is assigned to executor 3.201. If Resolve Executor Assignments is disabled, pressing executor 3.201 when inputting a command enters Page 3.201 in the command:

Resolve Executor Assignments Enabled

In the following example, Sequence 1 'FirstSong' is assigned to executor 3.201. Pressing that executor when inputting a command when Resolve Executor Assignments is enabled, the handle of the object assigned to the pressed executor is entered in the command. The preview displays the object type, number, and name in yellow text.

Handles can also be used in the command line:

For more information, see **#[ ] Keyword**.

	Hint:
	Using handles instead of the object number or name eliminates the need to update macros and commands after moving an object to a different location or changing its name.

1.39.2. Create Macros

Create a macro using the GUI editor or the command line.

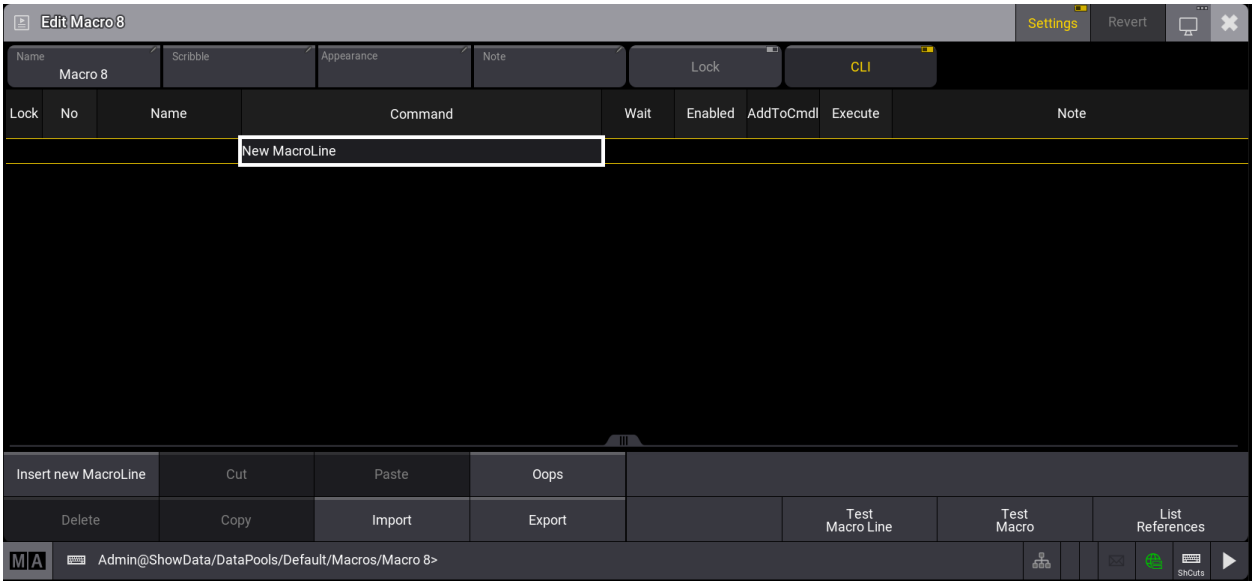
For a better understanding of macros, see the **Macros topic**.

Create a Macro

Requirement:

A visible macro pool. Open the Add window dialog, tap Data Pools, then tap Macros.

1. Pressing Edit, then tapping an empty macro pool object creates a new macro and opens the editor:



2. Tap Insert new MacroLine. The command editor opens:

3. Write a command into the editor. See **Examples** for more information.
4. Repeat steps 2 and 3 to add several lines.

There are several buttons in the editor. Here is a short explanation of the specific buttons:

- Settings: Enable Settings in the title bar to display the Name, Scribble, Appearance, Note, Lock, and CLI buttons.
- Test Macro Line: Tests only the selected line.
- Test Macro: Tests the macro.
- List References: For more information, see **Info Window**.
- Tapping Import will open a pop-up containing predefined macros. See **Import Macros**.

When CLI is disabled (it is enabled by default), CLI is displayed in red text on the pool object, and the macro is executed without affecting the command line.

Macro 3 with CLI deactivated

To edit macro 3 when CLI is disabled, use the Swipey Commands or type:

```
User name[Fixture]>Edit Macro 3
```

To execute the macro, close the editor and tap the macro in the Macro pool.

Create a Macro by Using the Command Line

Macros can be added using the command line. It is relevant when the GUI is unavailable, for instance, when the software is accessed via the command line only, using the terminal window.

Use the **List** command during the creation process to see the result in the command line feedback.

Requirement:

- The **command line feedback** needs to be visible.

Here is the creation process:

1. Navigate to the macro part of the software:

```
User name[Fixture]>ChangeDestination Macro
```

2. List the existing macros to see empty macro objects:

```
User name@ShowData/DataPools/Default/Macros>List
```

3. Locate an available number (one that is not listed).

4. Store a macro with the number:

```
User name@ShowData/DataPools/Default/Macros>Store [Available_Number]
```

5. Navigate into the new macro:

MA	User name@ShowData/DataPools/Default/Macros>ChangeDestination [Available_Number]

6. Insert a line into the macro:

MA	User name@ShowData/DataPools/Default/Macros/Macro #>Insert

5. Add the command to the text field line number using the **Set** command:

MA	User name@ShowData/DataPools/Default/Macros/Macro #>Set [Line_Number] Property "Command" [Command Inside Quotations"]

6. The wait time can be set in the same manner as the text:

MA	User name@ShowData/DataPools/Default/Macros/Macro #>Set [Line_Number] Property "Wait" ["Wait Time"]

	Hint:
	The wait time can be entered as a number in seconds without quotation marks or as text with quotation marks. The two special times called **Follow** and **Go** can be entered without quotation marks, but the software will add the marks. They are case-sensitive. Typing "go" gives an error, but typing "Go" works.

7. Add more macro lines by repeating steps 6 through 8, but remember to use the correct line number with the Set command.
8. Return to the root of the command line:

MA	User name@ShowData/DataPools/Default/Macros/Macro #>ChangeDestination Root

A new macro is created.

Label a Macro

Macro pool objects are labeled like any pool object. Read more in the **Label Pool Objects topic**.

	Hint:
	A unique macro name makes it possible to run the macro using their names.

1.39.3. Import Macros

Importing a Macro from the Macro Library

Requirement:

- Have a Macro Pool visible on one of the screens. For more information, see **Create Macros**.

To import a macro by using the macro pool:

1. Edit an empty pool object. This automatically creates a new macro and opens the Editor.
2. Tap Import at the bottom of the editor. This opens the macro library.
3. Use the search function or scroll through the list to find a macro, select a macro, and then tap Import.

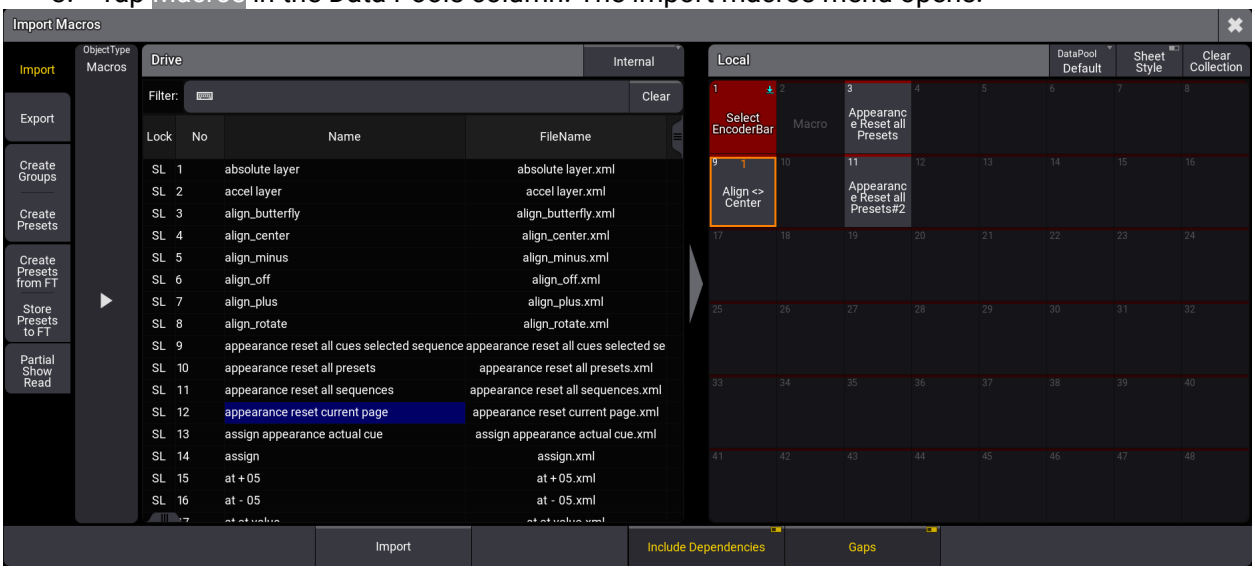
The macro is imported.

Importing Macros by Using the Import/Export Menu

For more information about the Import/Export menu, see the **Import / Export menu**.

Predefined macros can be imported from the internal library as well as custom macros, for example via USB.

1. Press Menu.
2. Tap Show Creator and then tap Import. The Import menu opens.
3. Tap Macros in the Data Pools column. The import macros menu opens:



Import menu

4. To select a macro from the library, tap a macro from the list on the left side of the window.
5. Tap an empty pool object (on the right side of the window) where you want to import the macro.
6. Tap Import at the bottom of the window.
7. The macro is imported.

	**Hint:**
	If no pool object is selected, the macro will be imported at the next available empty pool object.

1.39.4. Edit Macros

There are two ways to edit macros: The GUI and the command line.

Edit a Macro by Using the GUI

Requirement:

- An open Macro Pool on one of the screens.

To edit a macro by using the GUI:

1. Tap, hold, and swipe out of the macro in the macro pool that is to be edited.
2. Swipe to the Edit swipey command and release the screen.
   The editor opens.
3. Edit the fields that are to be changed.
4. Close the editor when the macro is correct.

Read the **Create Macros** topic to learn about adding lines and labeling a macro.

Macro lines can be deleted by selecting the line and then tapping the Delete button in the editor.

Edit a Macro by Using the Command Line

Requirement:

- The **command line feedback** is very nice to have visible.

Remember that the **List** command displays the content at the current location and can be used anytime.

This is the editing process:

1. Navigate to the macro that is to be edited:

```
MA  User name[Fixture]>ChangeDestination Macro ["Macro_Name" or Macro_Number]
```

2. Edit the fields using the **Set** command. This simply overwrites the current content in the field.
3. When done editing, return to the root location:

```
MA  User name@ShowData/DataPools/Default/Macros/Macro #>ChangeDestination Root
```

Read the **Create Macros** topic to learn about adding lines and labeling a macro.

Delete a Macro Row Using the Command Line

To delete a macro row by using the command line:

1. Navigate to the macro where the row is to be deleted:

MA	User name[Fixture]>ChangeDestination Macro ["Macro_Name" or Macro_Number]

2. Use the **Delete** command followed by the row number:

MA	User name@ShowData/DataPools/Default/Macros/Macro #>Delete [Macro_Row_Number_List]

	Hint:
	To delete multiple rows, type a list of numbers.

3. When done deleting rows, return to the root location:

MA	User name@ShowData/DataPools/Default/Macros/Macro #>ChangeDestination Root

The macro rows are deleted.

1.39.5. Assign Macros to Keys and Buttons

The macros can be assigned to executors or view buttons for easy access.

The macro is still in the macro pool. The executor or view button simply runs the macro in the pool. It is, therefore, a requirement to have the macro in the pool before assigning it anywhere.

There are different ways to assign macros to keys or buttons:

Assign Macros by Using the Keys

Requirements: A visible macro pool and a stored macro.

This is maybe the fastest way to assign the macro when using a console:

1. Press Assign.
2. Press MA while pressing X14 | Macro.
3. Enter the macro number using the numeric keys.
4. Press the button or tap the key where the macro is to be assigned.

Alternatively, if the macro pool is visible:

1. Press Assign.
2. Tap the macro to be assigned in the macro pool.
3. Press the button or tap the key where the macro is to be assigned.

Assign Macros by Using the Swipey Commands

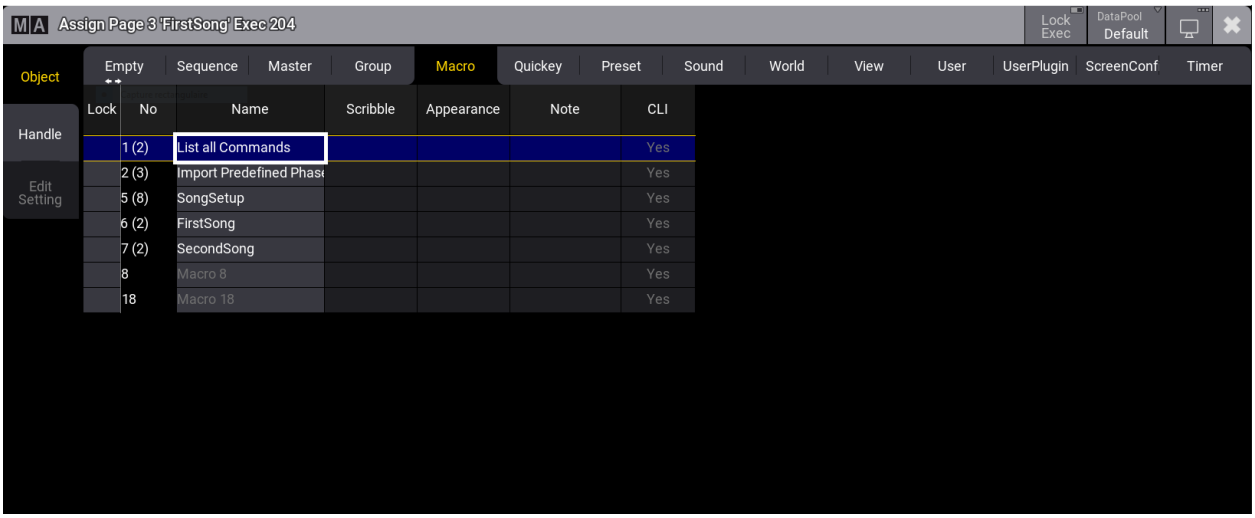
Requirement: Have the macro pool visible on the screen.

1. Tap and hold the macro pool object.
2. Swipe out of the pool object. The swipey commands open.
3. Swipe to the Assign button and release the screen.
4. Press the button or tap the key where the macro is to be assigned.

The macro is assigned to a key.

Assign Macros by Using the Assign Menu for Executors

1. Press Assign and press the desired executor. The Assign Menu opens:



1. Tap Object on the left.
2. Tap Macro at the top of the menu.
3. Tap the desired macro in the list.
4. Close the Assign Menu.

The macro is assigned to an executor.

Assign to an Executor Using the Command Line

To assign a macro to an executor on a specific page, use this syntax:

```
User name[Fixture]>Assign Macro ["Macro_Name" or Macro_Number] At Page ["Page_Name" or Page_Number].[Executor_Number]
```

To assign a macro to the current page, use this syntax:

```
User name[Fixture]>Assign Macro ["Macro_Name" or Macro_Number] At Executor [Executor_Number]
```

1.39.6. Variables

Variables can be used in macros or command line entries.

Variables have a user-defined name that the variable content replaces when the command is executed.

A variable can contain three different types of data:

- **Integer**: A signed whole number.
- **Double**: A signed fixed-point number with six decimals.
- **Text**: Any text string.

Signed numbers can be negative numbers, for instance, -7.

Variables are typeless. This means that an existing variable can be changed to contain a different type.

The name of a variable can be composed of any character. Variable names are case-sensitive.

	**Hint:**
	It is not a requirement, but avoiding spaces in variables is a good idea. Instead, consider using camelCase or PascalCase for variable names.

There are two different variable scopes.

Variables can be scoped for the user, or they can be global variables.

User variables can only be seen and used by the user profile that creates them, while global variables can be seen and used by all users in the session.

SetGlobalVariable and SetUserVariable

Creating a variable is the same whether it is a user or global variable, but we use two different keywords to set one or the other: **SetGlobalVariable** or **SetUserVariable**.

This is the syntax:

SetGlobalVariable ["Variable_Name"] ["Content text with or without spaces"]

SetGlobalVariable ["Variable_Name"] [Integer or Double]

SetUserVariable ["Variable_Name"] ["Content text with or without spaces"]

SetUserVariable ["Variable_Name"] [Integer or Double]

	**Important:**
	The quotation marks are very important if the variable name contains

	spaces. If the name does not contain spaces, then they can be omitted. The same is true for text strings as the variable content. Variable content must be in quotation marks to ensure the software stores a text string.

Examples

```
User name[Fixture]>SetUserVariable MyFavoriteNumber 9
User name[Fixture]>SetUserVariable MyFavoriteText "9"
```

These two commands create two variables that might seem to have the same content, but the first is a number, and the second is a text. So, it is not because of the names but the quotation marks in the second example.

Single and Double Quotation Marks

It might be needed to store a variable with a text string that includes something in quotation marks. This presents an issue with how the software interprets the input and how it is interpreted when the variable is called during runtime.

The variable system supports single and double quote pairs.

This means that text strings can contain quoted text as long as the other quotation marks are used.

Examples

Working with a specifically named group ("Spots Grid") in different contexts is necessary. A variable with "Group" and the name can be stored as a variable:

```
User name[Fixture]>SetUserVariable MySpecialGroup "Group 'Spots Grid' "
```

Notice the single quotes around the group name and the double quotes around the variable text string. These two pairs can be reversed for the same result.

Now, the variable can be used to address the group:

```
User name[Fixture]>$MySpecialGroup At 80
```

The $ is placed before the command so we can recall the variable.

This is the software's reply:

```
OK: Group "Spots Grid" At 80
```

GetGlobalVariable and GetUserVariable

The variables' content and type can be listed using the **GetGlobalVariable** or **GetUserVariable** keywords.

These keywords need to know what variable to show. The syntax is:

GetGlobalVariable ["Variable_Name"]

GetUserVariable ["Variable_Name"]

These commands show information about the variable name, content type, and content in the **Command Line Feedback window**.

They only show the variables in their scope. This means that the global version only shows global variables, and the user version only shows the user variables.

This assumes that the variable names are known.

The **asterisk (*)** wildcard can show all variables or a filtered list of variables for which a part of the name is known.

Examples

The current content of the variable called "MySpecialGroup" can be shown using this command:

This is the feedback in the command line window:

```
"MySpecialGroup": Type = Text, Value = "Group 'Spots Grid' "
OK: GetUserVariable  "MySpecialGroup"
```

The following example shows a list of all the global variables where the name contains the word "Our":

The "O" is replaced with the asterisk to ensure the variables beginning with "Our" are also shown. The asterisk wildcard at the beginning defines that there must be additional letters in front of the letters we specify. In reality, the example will also show variables beginning with "Hour" or, in fact, anything with "ur" somewhere in the name. To specify variables starting with "Our," use the following example:

This example shows all the user variables:

Use Variables

Variables are used where the content of the variable is needed.

The following variable was used in a previous example:

The variable's text string ("Group 'Spots Grid' ") replaces the **$MySpecialGroup**. This works because the variable contains a text string with commands.

	Important:
	Using quotation marks when using variables is essential and makes a

	difference.

If the variable is used with quotation marks:

```
User name[Fixture]>$"MySpecialGroup" At 80
```

This will fail because the content of the variable is explicitly called as a string, and the resulting feedback is:

```
Illegal object: Fixture  "Group 'Spots Grid'"  At  80
```

The software tries to use the variable content as a text string and does not interpret it as a command.

This might be useful in other cases. When the variable name contains a text string that should be used as text, it must be in quotation marks. See the example below.

Example

This example uses a variable that contains the name of a group:

```
User name[Fixture]>SetUserVariable MyGroupName "Spots Grid"
```

Now, this variable can be used where the text would be used otherwise:

```
User name[Fixture]>Group $"MyGroupName" At 80
```

It will fail If the quotation marks are omitted:

```
User name[Fixture]>Group $MyGroupName At 80
Illegal object: Group  "Spots"  Grid At  80
```

The text string is interpreted as a command.

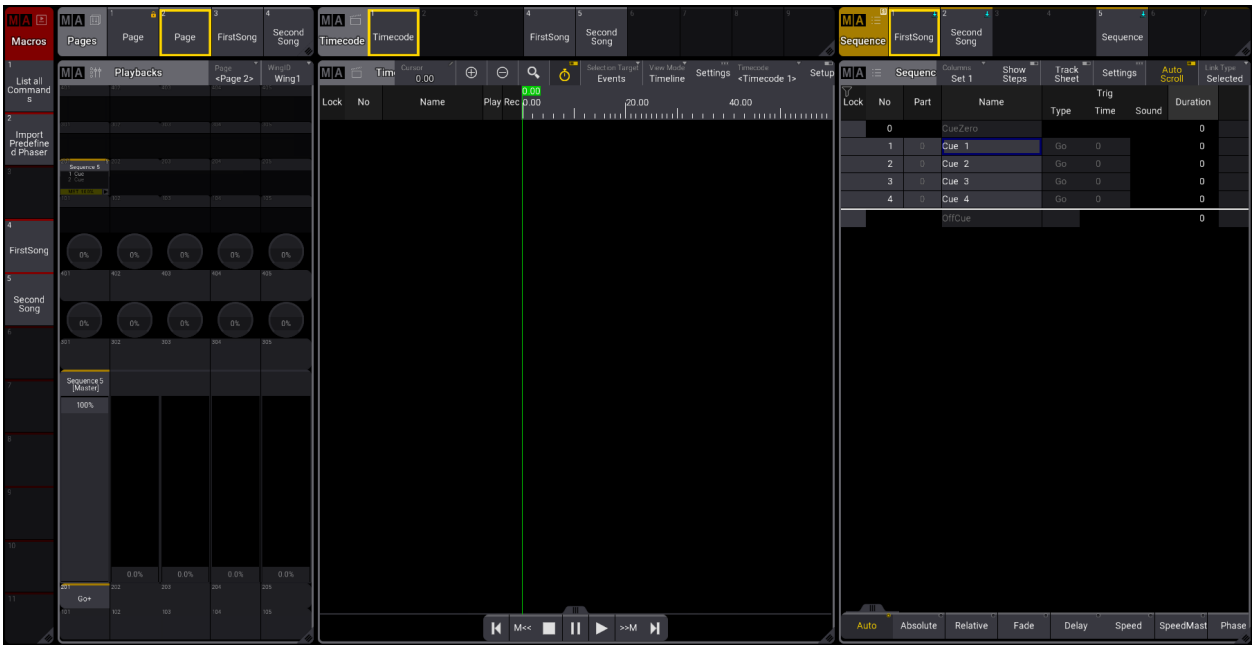
Example

Let's assume you are running a musical show with multiple songs that will run off timecode.
The following macro example will create a variable that will be used to set the desk ready for each song.

Opened windows requirement:

- Macro Pool
- Pages Pool
- Timecode Pool
- Sequence Pool
- Playbacks
- Timecode Viewer
- Sequence Sheet

It could look like this:



For this example, you will need to:

- Store two sequences, one labeled FirstSong and a second labeled SecondSong. Both sequences should have cue one as the first cue. For more information, see **Cues and Sequences**.
- Store two timecode shows, one labeled FirstSong, where the track's target is the sequence labeled FirstSong, and a second timecode show labeled SecondSong, where the track's target is the sequence labeled SecondSong. For more information, see **Timecode - View Mode**.
- Assign the sequence labeled FirstSong to page 3 executor 201 and the sequence labeled SecondSong to page 4 executor 201.
- Store a macro labeled FirstSong.
- Store a macro labeled SongSetup.

Here's the workflow:

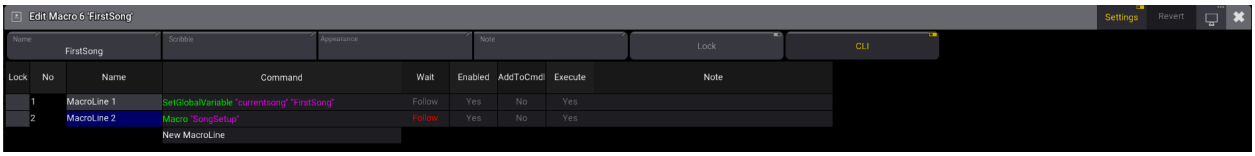
1. Edit the macro labeled FirstSong.
2. Insert two macro lines.
   - **Macro Line1**: First, we need to create a new variable. Edit the command on macro line 1 and type:

This is the feedback in the command history window when the command is executed:

OK:SetGlobalVariable "curentsong" "FirstSong"

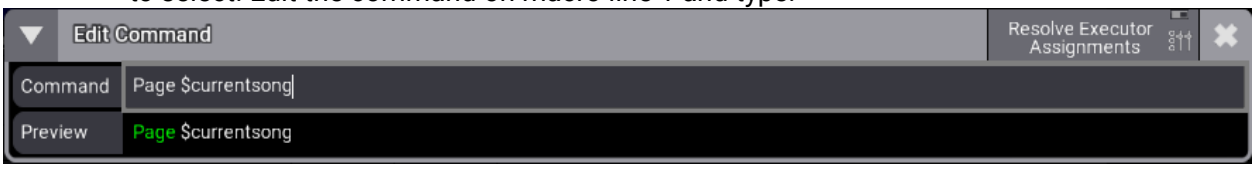
- **Macro Line 2**: Here, we are going to call the macro that we have labeled SongSetup. Edit the command on macro line 2 and type:

This is what the macro looks like:



Next, we build a macro that will be used throughout the show to set the desk ready for each song.

1. Edit the macro labeled SongSetup.
2. Insert eight macro lines.
   - **Macro Line 1**: Using the new variable "currentsong", we will instruct the software on what page to select. Edit the command on macro line 1 and type:



This is the feedback in the command history window when the command is executed:

OK: Page "FirstSong"

You can see here that the variable "current song" is replaced with the variable content "FirstSong".

   - **Macro Line 2**: Now we must instruct the software which executor to select. Edit the command on macro line 2 and type:

This is the feedback in the command history window when the command is executed:

OK: Select Page "FirstSong" Executor 201

   - **Macro Line 3**: Stop any running timecode is a good idea. To do so, edit the command on macro line 3 and type:

This is the feedback in the command history window when the command is executed:

OK: Off Timecode 1 Thru

- **Macro Line 4**: Here, we will rewind all timecodes to time 00h00m00.0. Edit the command on macro line 4 and type:

This is the feedback in the command history window when the command is executed:

OK: <<< Timecode 1 Thru

Macro Line 5: To select the timecode show, edit the command on macro line 5 and type:

This is the feedback in the command history window when the command is executed:

OK: Select Timecode "FirstSong"

Macro Line 6: Now we instruct the software what timecode show to execute. Edit the command on macro line 6 and type:

This is the feedback in the command history window when the command is executed:

OK: Go+ Timecode "FirstSong"

- **Macro Line 7**: Here, we instruct the software to turn off all running executors from page 2 through page 100 except the one running the sequence labeled FirstSong. It's usually a good idea to keep page 1 for any sequences that need to be manipulated manually outside of the running timecode sequences; this is why the command also excludes sequences running on page 1. Edit the command on macro line 7 and type:

This is the feedback in the command line window:

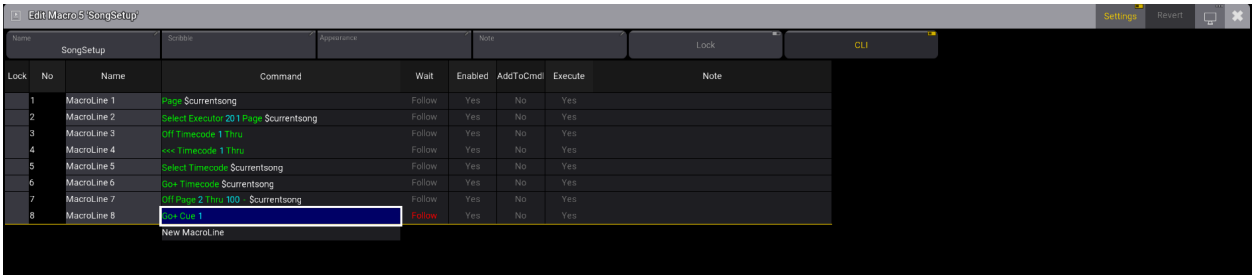
OK: Off Page 2 Thru 100 - "FirstSong"

- **Macro Line 8**: Finally, even if the timecode should trigger cue 1, it is safe to add the following command. Edit the command on macro line 8 and type:

This is the feedback in the command line window:

OK: Go+ Cue 1

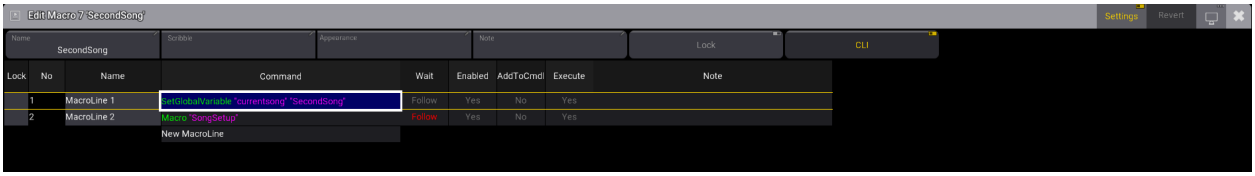
This is what the macro looks like:



This is where it gets interesting. To create a macro for the next song:

1. Copy the macro you have created labeled FirstSong to an empty macro pool object.
2. Edit that new macro and label it SecondSong.
3. Edit the command on **Macro Line 1** by replacing the variable content "FirstSong" with "SecondSong".

This is what the macro SecondSong looks like:



The only thing that changed is the variable content on **Macro Line 1**: SetGlobalVariable "currentsong" "**SecondSong**". So, each time the variable is recalled, it will refer to that new content. This method can be repeated for all songs in the show.

DeleteGlobalVariable and DeleteUserVariable

Variables can be deleted using the **DeleteGlobalVariable** and **DeleteUserVariable** keywords.

This is the syntax:

DeleteGlobalVariable ["Variable_Name"]

DeleteUserVariable ["Variable_Name"]

	Important:

	Deleting a variable cannot be oopsed. There is not much else to say about deleting variables. When deleted, they do not exist anymore.

Concatenate Variables

Variables can be concatenated by storing them into a new variable.

The desired result of the concatenation is to create a compound text string containing the concatenated variables. The output is always a text string. See below for the mathematical addition of number variables.

This is the syntax:

SetGlobalVariable [MyConcatResult] $[VariableName]$[VariableName]

SetUserVariable [MyConcatResult] $[VariableName]$[VariableName]

There must be no space between the variables that need to be concatenated or between any extra text that needs to be part of the concatenation.

Text can be added to the variable in quotation marks with no space between the end and start of quotation marks. However, there can be spaces inside the quotation marks. See the example below.

Examples

These examples use three variables:

Variable Name	Content Type	Content
MyInteger	Integer	9
MyKeyword	Text	Group
MyGroupName	Text	Perfect Macro

The goal is to concatenate these into a command string that can be used to store a group.

It can be done with one command but is split up to look at different functions here.

First, the keyword is combined with the integer into a new variable:

```
User name[Fixture]>SetUserVariable MyCommandString $"MyKeyword"" "$MyInteger
```

This command uses many quotation marks. Setting the variable names in quotation marks ensures the variable content is interpreted as text. This also means that the integer variable is converted to a text string.

Furthermore, there is a " " between the two variables to add a space between them.

Show results using this command:

```
User name[Fixture]>GetUserVariable MyCommandString
```

Next, add the macro name to the string:

```
User name[Fixture]>SetUserVariable MyCommandString $"MyCommandString"" ""$"MyGroupName""""
```

Again, it can be hard to see what is happening. The two names of the variable are in quotation marks. Between the two are a space and a single quotation mark in double quotation marks. After the last variable name, there is another single quotation mark inside the double quotation marks.

Here you can spot them better: SetUserVariable MyCommandString $"MyCommandString"" ' "$"MyGroupName"" ' "

The command looks like this without the double quotations: **SetUserVariable MyCommandString $MyCommandString '$MyGroupName'**

The existing content of MyCommandString is added to MyCommandString, plus the content of MyGroupName in single quotations.

The result is that MyCommandString is **Group 9 'Perfect Macro'** as a text string.

Now, some fixtures can be selected, and this command can be executed:

```
User name[Fixture]>Store $MyCommandString
```

Then, a group with the name is stored at group 9. Quotation marks must not be used in the command above.

Variable Addition

Concatenating variables with integer and/or double types performs a mathematical addition instead of an actual concatenation. Here, quotation marks are not used.

The result needs to be stored in a variable.

Examples

This example uses three variables:

Variable Name	Content Type	Content
MyInteger	Integer	9
MySignedInteger	Integer	-2
MyDouble	Double	6.500000

The first two variables can be added to each other with this command:

```
User name[Fixture]>SetUserVariable MyResult $MyInteger$MySignedInteger
```

The result can be seen by looking at the command line feedback. It shows that the variable is stored with a value of 7.

The double can be added with an integer. The result will be a double to avoid data loss.

```
User name[Fixture]>SetUserVariableMyResult $MyInteger$MyDouble
```

This stores the value of 15.500000 in MyResult.

Type Conversion

Type conversion has been happening in the examples above.

An integer or double can be converted to text by putting the variable name in quotation marks. The command will convert then when it is executed.

Setting variables without quotation marks can convert text to an integer.

An integer can be converted to a double by adding it to a double with the value of **0.000000**.

A double cannot be converted to an integer.

Examples

These examples use the following variables:

Variable Name:	Content Type:	Content:
MyInteger	Integer	9
MyEmptyDouble	Double	0.000000
MyText	Text	Hello

This example converts the integer to text:

Converting this back to an integer can be done with the following command:

Converting an integer to a double can be done using this command:

1.39.7. Example Macros

This topic has some example macros. They are meant as inspiration for other macros. In addition, they show some of the possibilities with macros.

Calling Macros by Macros

This example shows how macros triggered by other macros can have different results.

1. Create two Macros as shown below:

Macro 1

Lock	No	Name	Command	Wait	Enabled	AddToCmdl	Execute
	1	MacroLine 1	Macro 2; Echo "Macro 1 Line 1 - Trigger Macro 2"	Follow	Yes	No	Yes
	2	MacroLine 2	Echo "Macro 1 Line 2"	Follow	Yes	No	Yes
	3	MacroLine 3	Echo "Macro 1 Line 3"	Follow	Yes	No	Yes
	4	MacroLine 4	Echo "Macro 1 Line 4"	Follow	Yes	No	Yes
	5	MacroLine 5	Echo "Macro 1 Line 5 - Last"	Follow	Yes	No	Yes

Macro 2

Lock	No	Name	Command	Wait	Enabled	AddToCmdl	Execute
	1	MacroLine 1	Echo "Macro 2 Line 1"	Follow	Yes	No	Yes
	2	MacroLine 2	Echo "Macro 2 Line 2"	Follow	Yes	No	Yes
	3	MacroLine 3	Echo "Macro 2 Line 3"	Follow	Yes	No	Yes
	4	MacroLine 4	Echo "Macro 2 Line 4"	Follow	Yes	No	Yes
	5	MacroLine 5	Echo "Macro 2 Line 5 - Last"	Follow	Yes	No	Yes

2. Tap Macro 1. As a result, all lines except line 5 of macro 2 are executed.
3. Disable lines 3 and 4 of Macro 1. As a result, Lines 1, 2, and 5 of Macro 1 are executed and Lines 1 and 2 of Macro 2 are executed.

Macros triggered by other macros in this way will also stop when the first macro stops. In order to overcome this, **Call Macro X** command should be used.

Change User

This macro opens the login window where you can enter a new username and password.

Lock	No	Name	Command	Wait	Enabled	AddToCmdl	Execute
	1	MacroLine 1	LogIn	Follow	Yes	No	Yes

World Is Selection

This predefined macro will create a temporary world for the selected fixtures.

Lock	No	Name	Command	Wait	Enable	AddToCmdl	Execute
	1	MacroLine 1	Delete World 999 /NoConfirmation	Follow	Yes	No	Yes
	2	MacroLine 2	Store World 999	Follow	Yes	No	Yes
	3	MacroLine 3	World 999	Follow	Yes	No	Yes

Block Sequence

This macro will prompt you for a sequence ID and a cue number to block.

Lock	No	Name	Command	Wait	Enable	AddToCmdl	Execute
	1	MacroLine 1	Block Sequence (Sequence #) Cue (Cue #)	Follow	Yes	No	Yes

Hint:

If no cue number is entered, the entire sequence will be blocked

1.40. Agenda

The agenda allows you to schedule objects, such as sequences, macros, or plugins, to be executed by the console based on the calendar. Commands can also be defined, such as shutting down the system at a certain time.

You can also schedule events that repeat every minute, every day, every week, every month, and/or every year, for example, the Saint-Patrick's Day cue every year.

To learn about the Agenda keyword, see **Agenda**

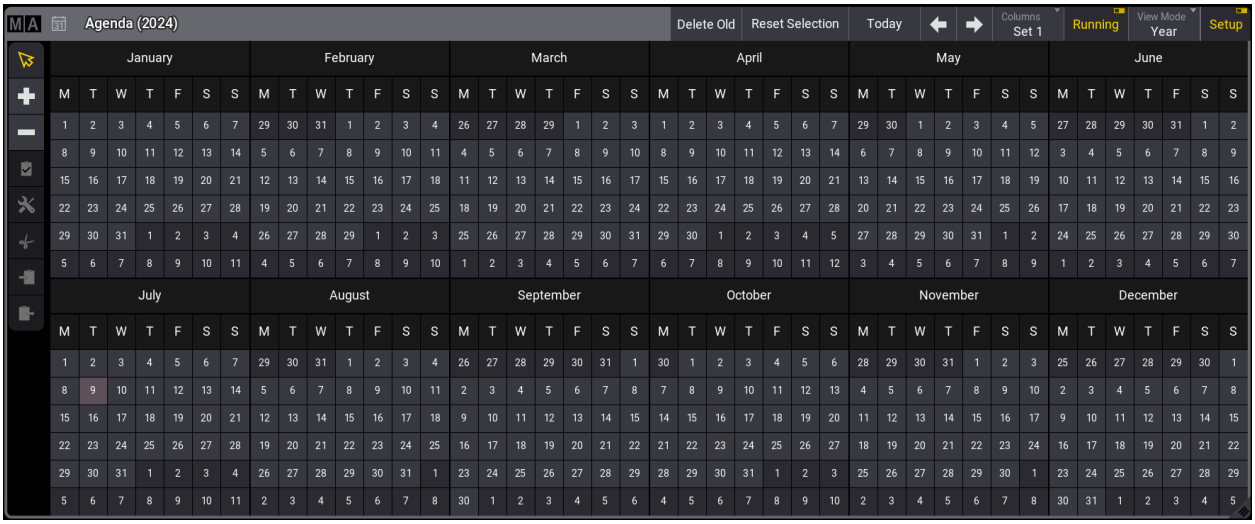
Subtopics

- **View Modes**
- **Create an Agenda Entry**
- **Edit an Agenda Entry**
- **Agenda Toolbar**

1.40.1. View Modes

The agenda window can display the data using five different view modes.

Open an agenda window, see **Add Windows**. From the add window dialog, tap Tools, and then tap Agenda Viewer.



Agenda window

To change the view, tap View Mode in the title bar.

- **Sheet**: Displays all agenda entries in a spreadsheet format.

	Hint:
	To sort the agenda entries in the sheet view by name, right-click or tap and hold a column header.

- **Year**: Displays a calendar view of the year. The current date has a light gray background. Days with at least one enabled event are indicated by a green background. Days that have all events disabled are indicated by a bright red background.
- **Month**: Displays all agenda entries of the selected month.
- **Week**: Displays all agenda entries of the selected week.
- **Day**: This mode is divided into two sections. It displays a smaller monthly overview on the left and entries for the selected day on the right.

	Hint:
	- Disabled events are displayed with red font color. - Repeated events are displayed with on the right side. This applies to the month, week, and day view modes.

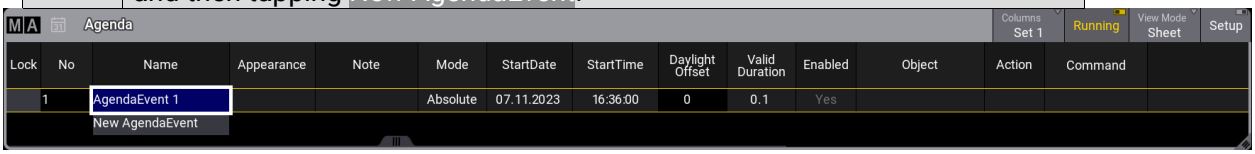
1.40.2. Create an Agenda Entry

Create an Agenda Entry Using the Sheet View Mode

1. Tap and hold View Mode in the title bar, then slide your finger into the list and select Sheet.
2. To create a new agenda event, right-click or tap and hold New AgendaEvent.

A new entry is created with the system date and time.

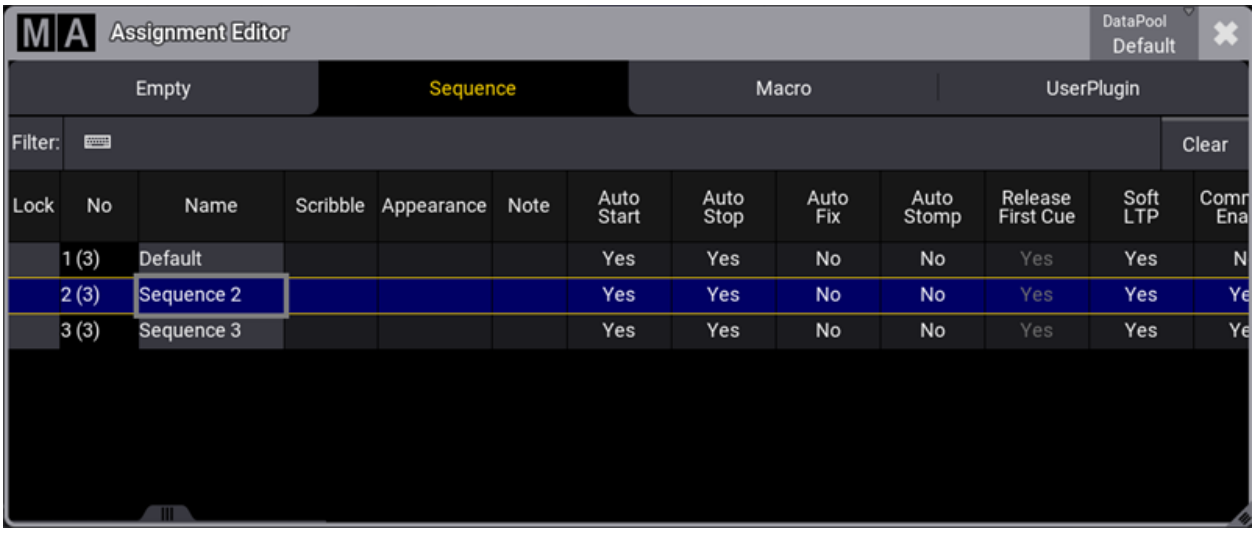
Hint:
You can also create an entry from the Sheet View mode by pressing Edit and then tapping New AgendaEvent.



Agenda Viewer

The following is a list of the properties to be defined by the user:

- **Name**: Enter the name of the event that will be displayed in all layout modes.
- **Appearance**: Assign an appearance to define how the entry looks in the calendar views.
- **Note**: For more information, see **Note**.
- **Mode**: When set to Absolute, the agenda uses the entered start date and time. Twilight times like Dawn, Sunrise, Sunset, and Dusk can also be used. See **Date and time** to learn how to configure the grandMA3 software to calculate the correct twilight time for your location.
- **StartDate**: Tap and hold a cell to open the **Edit StartDate** pop-up, then edit the day, month, and year. Use Today in the title bar to quickly enter today's date.
- **StartTime**: Tap and hold a cell to open the **Edit StartTime** pop-up, then edit the hours, minutes, and seconds. Use Now in the title bar to quickly enter the current time.
- **Daylight Offset**: When a twilight time is set in the mode column, it defines an offset for an event. For example, if you want an event to start 15 minutes before dawn. See **Date and time** to learn about twilight times.
- **Valid Duration**: Enter a duration value to allow the backcasting of events if the console is not switched on or the agenda is disabled during a scheduled entry.
- **Enabled**: To enable or disable an agenda entry. Disabled entries are displayed with a red color font.
- **Object**: Tap and hold a cell to open the Assignment Editor pop-up, and select a **Plugin**, **Macro**, or **Sequence** to be executed.

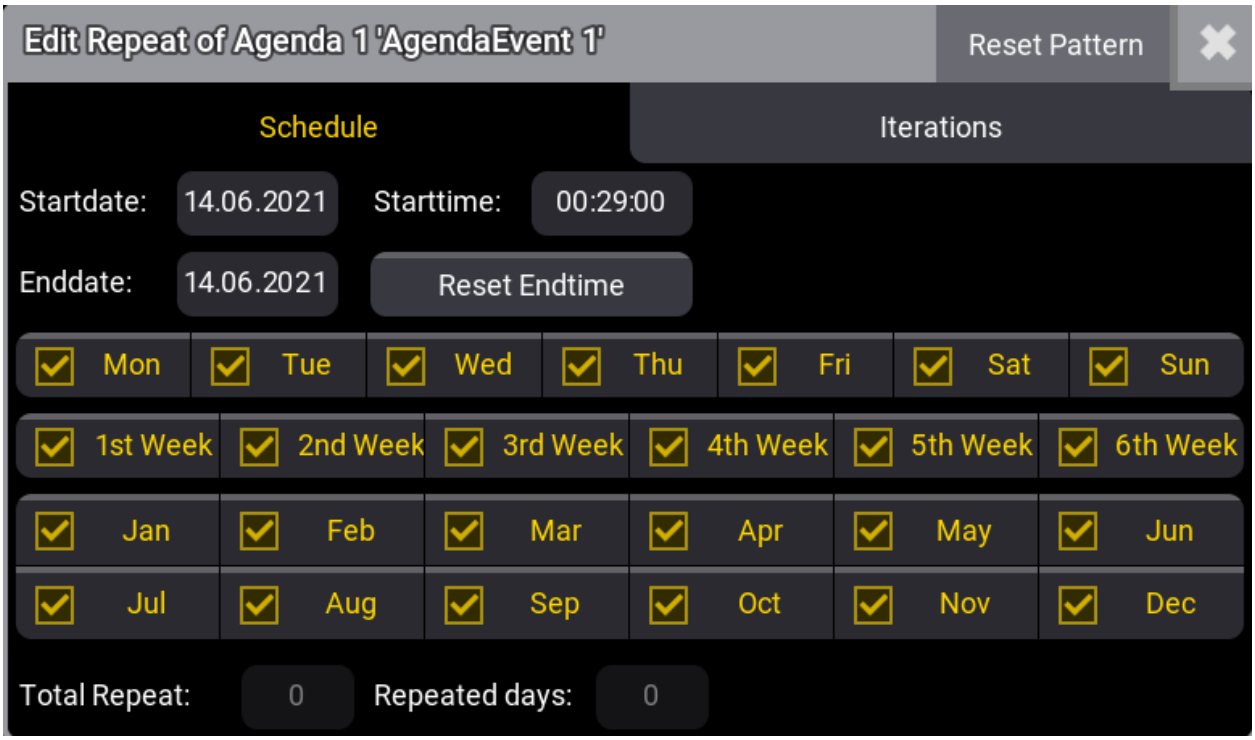


Assignment Editor

- **Action**: Tap and hold a cell, then select an action to use when the defined object is executed.
- **Command**: Define a command here to be executed instead of defining an object to be triggered. E.g., Go+ Executor 101.
- **Repeat**: This field defines whether an entry is repeated and, if so, how often. Tap and hold a cell to open the **Edit Repeat of Agenda** pop-up.

There are two main settings to define the repetition of an agenda entry, Schedule and Iterations.

Schedule Settings



The start date and time are linked to the agenda event and, therefore, are identical. To get repetitions, at least a different end date or end time must be set.

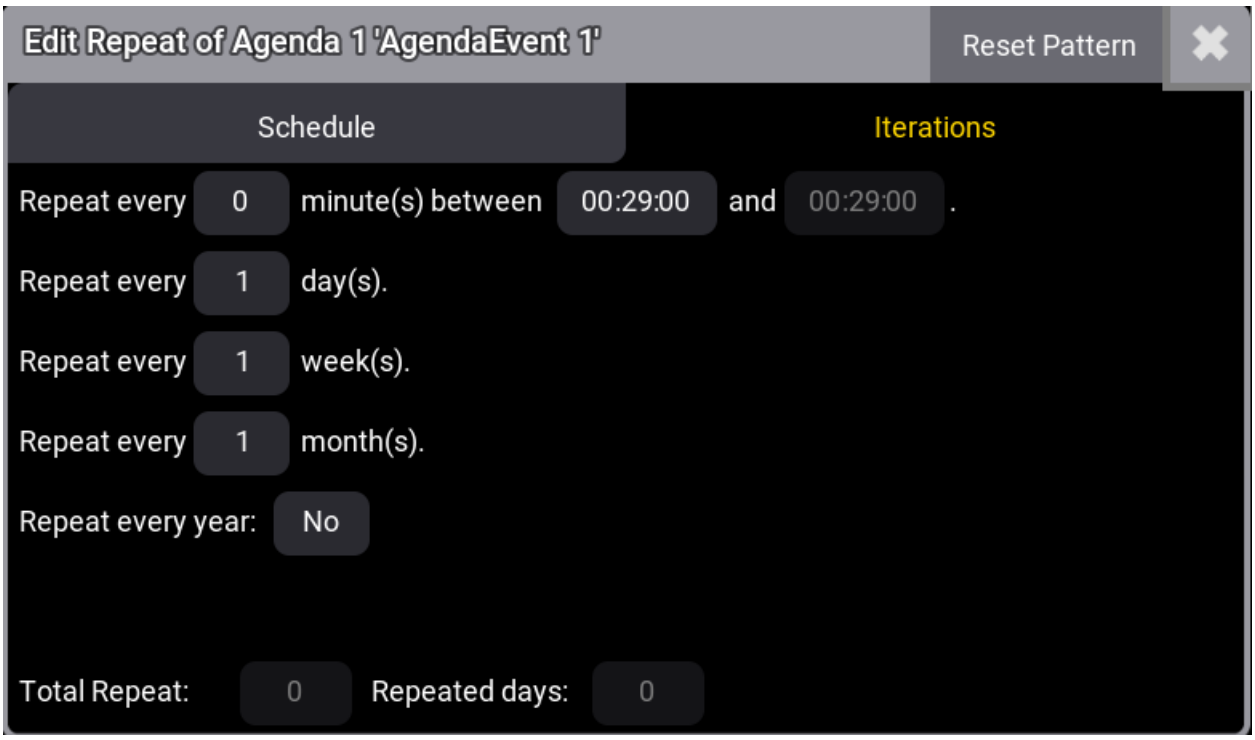
- Tap Reset Pattern in the title bar to reset the repeat settings you made previously.

- Tap Reset Endtime to reset the end date back to the agenda event date.

Tap a day, week, or month cell to enable or disable it. This will define at which days, weeks, or months the event will be repeated.

	Hint:
	The week settings are the weeks within a full month and not the weeks from the start date onwards.

Iterations Settings



Iterations settings allow defining repeat per minute, days, weeks, and months.

These settings are counted from the start date onward.
To repeat an event every year again, set **Repeat every year** to yes.

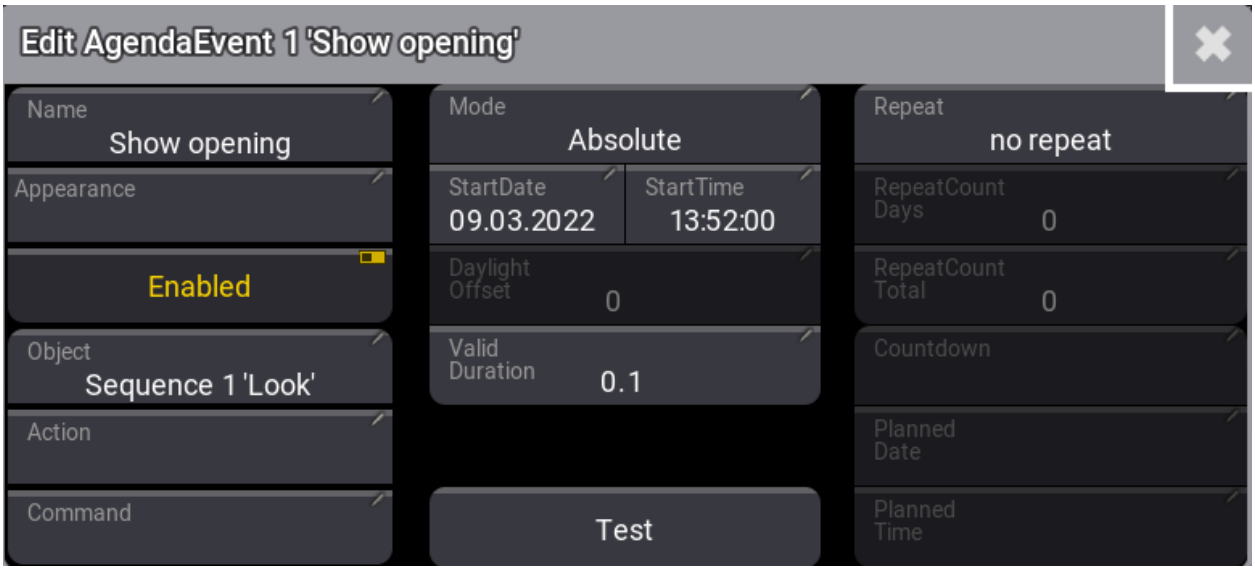
The following describes the last five columns of the agenda window in the title bar when ViewMode is set to **Sheet**. (These columns can not be edited.)

- **Countdown**: Displays the remaining days or time until the next launch of an event.
- **Planned Date**: Displays the next date when the event will be launched.
- **Planned Time**: Displays the next time when the event will be launched,
- **Repeat Count Days** and **Repeat Count Total**: These two columns are similar to the edit repeat pop-up. This is a quick way to verify the repeat pattern.

Tap Delete Old in the title bar to erase all preceding events, including the valid duration.

1.40.3. Edit an Agenda Entry

To edit an agenda entry from the **Month**, **Week**, and **Day** layout mode, press Edit, then tap the agenda entry you wish to edit. You can also tap and hold or right-click an agenda entry. The **Edit Agenda Event** pop-up opens.



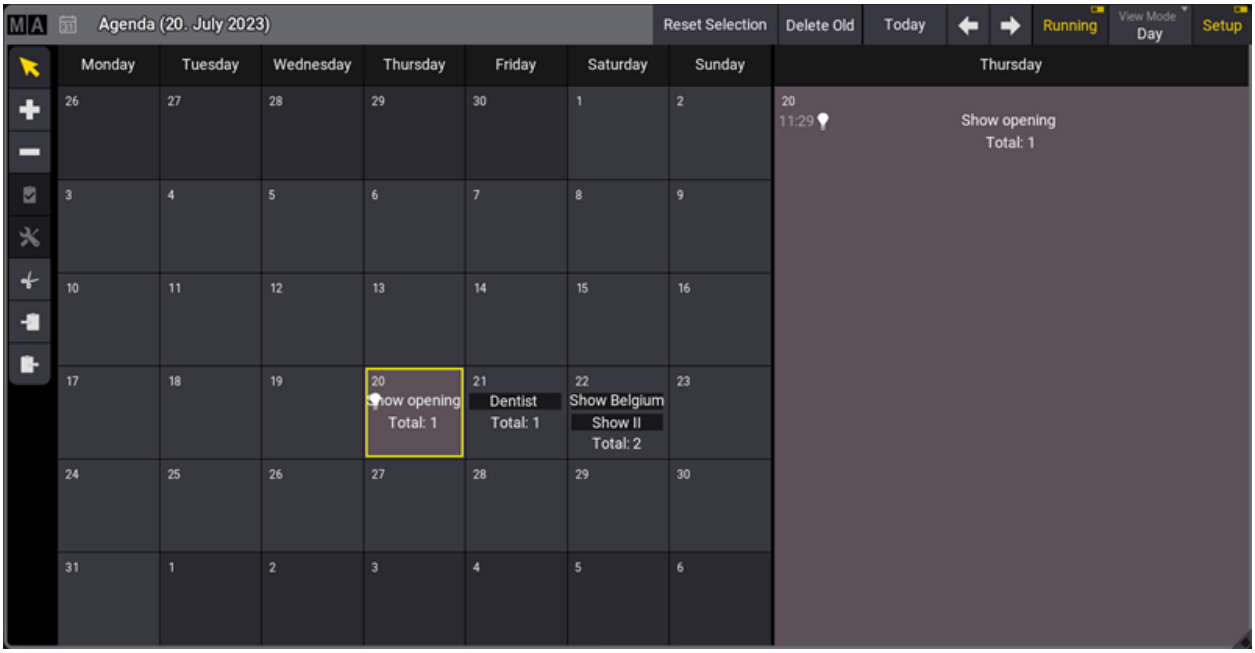
To learn about the properties that can be defined here, see **Create an agenda entry**.

	Hint:
	Agenda events can also be edited using the toolbar. See **Agenda toolbar**.
	Hint:
	Events can be deleted from any view mode by pressing Delete and then tapping the event you wish to delete.

1.40.4. Agenda Toolbar

Create or edit an agenda entry using the toolbar within the 5 available view modes; **Sheet**, **Year**, **Month**, **Week**, and **Day**. For more information, see **Agenda Modes**.

To enable the toolbar, tap Setup in the title bar.



Agenda window with view mode day

To select an event, tap in the toolbar, then tap the event you wish to select.

To create an event, tap in the toolbar, then tap a day in the agenda.

To delete an event, tap in the toolbar, then tap the event you wish to delete.

Select an event, then tap in the toolbar to execute it immediately. This will verify if the set object or command will be executed correctly.

Select the event you wish to edit, then tap in the toolbar. This opens the Edit Agenda pop-up. See **Edit an agenda entry** for more information.

To cut an agenda entry, tap in the toolbar, then tap the entry you wish to cut. Notice that (the paste tool) is selected in the toolbar and awaits the user to tap a day where to paste the cut entry.

To copy an agenda entry, tap in the toolbar, then tap the day you wish to copy. Notice that (the paste tool) is selected in the toolbar and awaits the user to tap a day where to paste the copied entry.

Use and in the title bar to change the day, week, month, or year.

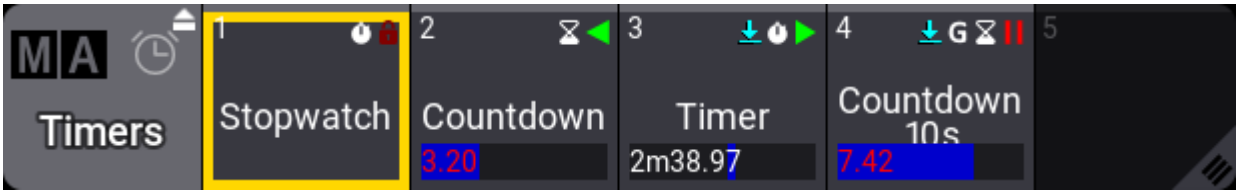
	Hint:
	The day, week, month, or year is displayed in brackets in the title bar according to the selected View Mode.

Tapping Today in the title bar will set the agenda to today's date.

1.41. Timers

The Timer pool can be used to manage stopwatches and countdowns.

The pool can be created like any other pool window using the **Add Window pop-up**.



Timers pool with different timers.

The Timers pool has a default action indicated by an icon in the title field's upper right corner. The factory default action is **Toggle** (). The default action is applied when a pool object is tapped without a valid keyword in the command line. Learn more about Pool Actions in the **Window Settings topic**.

There are two different types of timers: Stopwatch and Countdown. Each of these two types has their own topics. Follow the links below or in the menu on the left.

Each object in the Timers pool can be one or the other.

Changing the type from one to the other will stop and reset the timer if it is active.

The first pool object is a stopwatch that is locked from being edited, although it can be used.

Timers can be controlled using the pool objects or the **Timer keyword**.

Active timers can be seen in the pool with the time and a progress bar at the bottom of the pool object. Active timers can also be seen in the **Running Playbacks window**.

Running timers have a green play icon pointing right for stopwatches () and left for countdowns () in the upper right corner of the pool object. A paused timer has the pause icon () in the corner. Stopped timers do not have a playback icon.

Timers can be assigned to executors. Learn more in the **Assign Object to an Executor topic**.

The running state of the timer is not saved with the showfile. The time on the timer will be saved, and the timer will be on the saved time in a paused state when the showfile is reloaded.

Timers have a read-only property called "ElapsedTime". This can be used to read the current timer value.

All the different timer properties can be seen using the following commands in the command line and watching the Command Line History:

```
User name[Fixture]>List Timer
```

Playback Actions

A timer can be started with a **Go+** command or toggled On if it is Off or paused. It will start to count up or down in real-time.

A timer can be paused in its running. This keeps the current time on the timer. Restarting the timer with a Go+ command can either restart the timer or let it continue. This can be set in the setting for the timer.

The **Top** command will reset the time. If the timer is running, it will continue to run. If the timer is paused, the time will be reset, and the timer will be stopped.

Flash and **Temp** commands can be used to trigger timers. The timer will run while the function is active and pause when the command is stopped. For instance, a stopwatch can run while a flash executor button is pressed.

Stopping a timer resets the time to zero for stopwatches and to the set time value for countdowns.

Besides running timers manually, they can also be automatically triggered with specific sequence actions. See more below in the settings.

Using the Clock to See the Time

The **Clock Viewer** can be used to show and control the timers. The Clock Source should be set to **Timer**. Then, the Timer can be used to select the desired timer object.

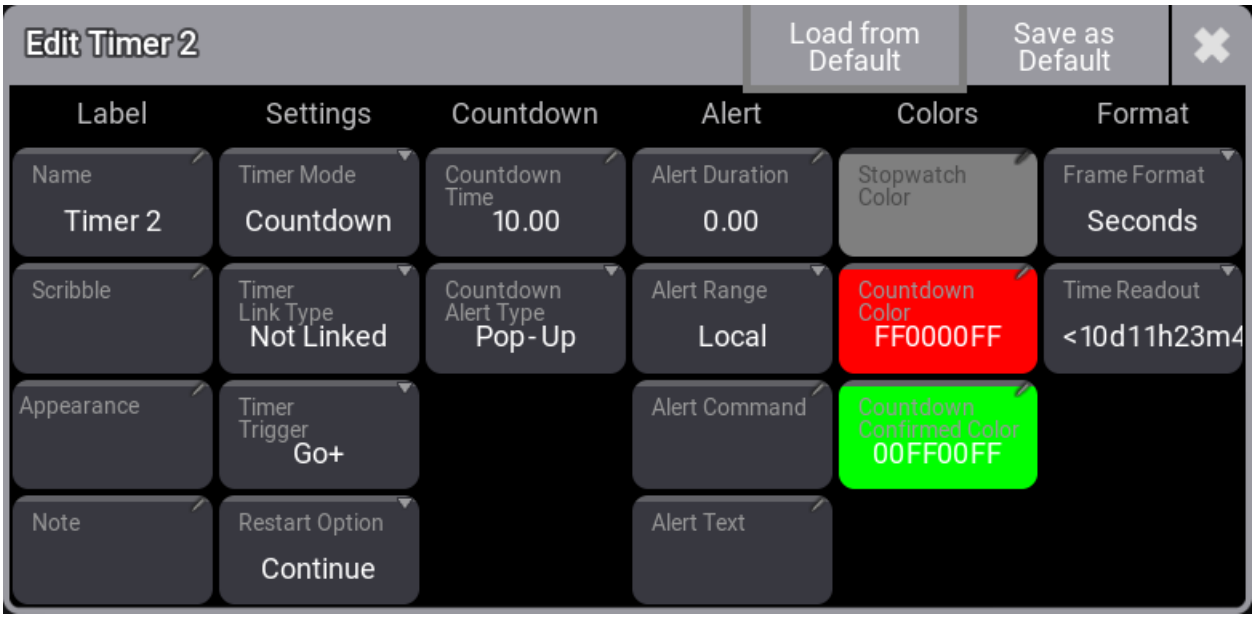
Two clocks showing a stopwatch and a countdown

The controls in the title bar adjust to match the selected timer.

Create and Edit a New Timer

A new timer can be created by editing an empty pool object.

Editing a timer pool object opens the editor:



Timer editor

There are some common settings for the two types of timers.

Label

- **Name**:
  This is the name of the timer object.
- **Scribble**:
  This is used to assign a scribble to the pool object.
- **Appearance**:
  This is used to assign an appearance to the pool object.
- **Note**:
  This is used to add a note to the pool object.

Settings

- **Timer Mode**:
  This is used to select one of the two modes: **Stopwatch** or **Countdown**.
- **Timer Link Type**:
  Tapping this opens a select pop-up. It contains a list of all the sequences in the show, plus three special options. The options are:
  - **Not Linked**:
    Nothing happens when starting a sequence playback.
  - **<Selected Sequence>**:
    Triggering the selected sequence will trigger the timer.
  - **Link Last**:
    Triggering any sequence will trigger the timer.
  - **[Sequence]**:
    Triggering the specific sequence will trigger the timer.
- **Timer Trigger**:
  This setting defines what action triggers a timer.
- **Restart Option**:
  This has two possible options:
  - **Continue**:
    When a stopped or paused timer is triggered, it will continue to run the time.

- **Reset**:
  When triggered, a stopped or paused timer will be reset and start from its default time.

Colors

The timer has a color defined. This color is used in the clock viewer and in the bar at the bottom of the pool objects.

The default color for stopwatches is white text color. Countdowns have numbers in red color with a countdown bar at the bottom of the timer and green color for a confirmed countdown.

These colors can be changed for each timer.

Format

There are two settings in the format column:

- **Frame Format**
- **Time Readout**

These are available for each timer object. They are described in the **User Settings topic**.

Subtopics

- **Stopwatch**
- **Countdown**

1.41.1. Stopwatch

Stopwatches are the default timer. Learn more in the **Timers topic**.

They count time up from 0 and do not have a practical upper limit.

The stopwatch timer has a stopwatch icon () in the upper right corner of the pool object.

A stopwatch will run for several days if it is not stopped or paused.

Stopwatches do not have any special settings.

1.41.2. Countdown

Stopwatches are the default mode for timers. The mode can be changed to the Countdown mode. Learn more in the **Timers topic**.

Countdowns count time down from a set value; when zero is reached, a trigger can fire an alert pop-up and a command.

The countdown timer has an hourglass icon () in the upper right corner of the pool object.

Besides the "ElapsedTime" property mentioned in the Timers topic (link above), the Countdown also has a "RemainingTime" read-only property.

Countdown Settings

The countdowns have more settings besides the common settings described in the Timer topic.

Countdown

- **Countdown Time**:
  Sets the duration time of the countdown. The maximum possible countdown time is limited to 14 days. The calculator sets the duration time in seconds, minutes, hours, and days.
- **Countdown Alert Type**:
  The alert type defines what should happen when the countdown reaches zero. There are four options:
  - **None**:
    Nothing happens when the timer runs out, except it stops.
  - **Pop-Up**:
    A pop-up appears with the text set in the **Alert Text** setting (see below). The pop-up appears for as long as the **Alert Duration** is set (see below).
  - **Command**:
    The timer executes the command set in the **Alert Command** setting (see below).
  - **Command & Pop-Up**:
    The pop-up appears, and the command is triggered.

Alert

- **Alert Duration**:
  Defines how long the pop-up should be displayed when the Countdown Timer reaches zero. An alert duration of zero seconds means the pop-up stays until confirmed. The countdown alert type needs to include "Pop-Up" for this setting to be active.
- **Alert Range**:
  The countdown alert range defines where the pop-up appears.
  - **Local**:
    The alert pop-up appears on the stations with users logged in using the same user profile as the one who started the countdown.
  - **All Stations**:
    The alert pop-up appears on all stations in the session. Countdowns with this setting have a small G icon () in the upper right corner of the pool object to indicate that it is a global countdown.

The pop-up has a confirmation button to close the alert (before the duration closes it). Confirm only confirms for the station where it is tapped. Confirm All Stations confirm the alert on all stations.

- **Alert Command**:
  When the countdown reaches zero, the command is executed on the GlobalMaster/IdleMaster station or the Standalone station. The **Countdown Alert Type** needs to include "Command" for this setting to be active.
- **Alert Text**:
  The defined text of this setting is displayed in the alert pop-up when the countdown reaches zero. The countdown alert type needs to include "Pop-Up" for this setting to be active.

1.42. Timecode Show

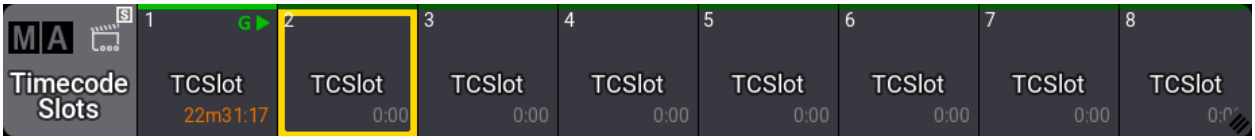
Timecode shows can execute events and move faders to a timed recording or at specific points in time based on a running time counter. This running time counter can be an internal timecode from the session master or an external timecode source.

The timecode show is organized in **Tracks**, and they exist inside a **Track Group**.

There can be multiple timecode shows that all live in the **Timecode** pool.

The timecode pool with a selected show and a running show. One of the timecode shows is the selected timecode show, which is indicated by a yellow frame.

There is another connected pool called **Timecode Slots**. These can be used as the source for the running time counter. There are eight timecode slots. This is a fixed number of slots.



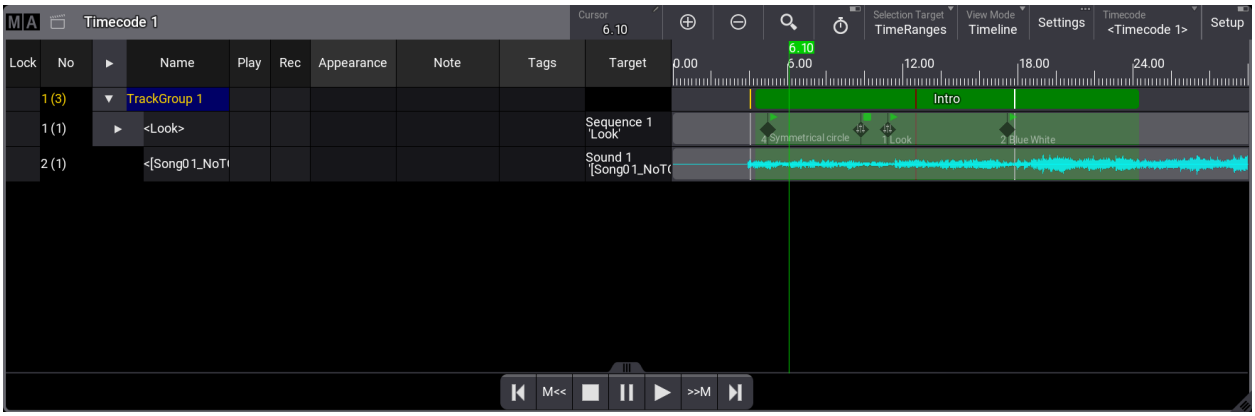
The Timecode Slots pools with a selected slot and a running time.

A slot can be connected to an SMPTE audio timecode, MIDI timecode, and Art-Net timecode input to allow an external source to control the running time. The slots can also generate a running time that can be used to run a timecode show. The **Clock Viewer** can show the time of a timecode slot. Learn more in the **What are Timecode Slots topic**.

The Clock Viewer shows the time. It can be the system time, the time of a timezone, the timer of a timer, or timecode slot.
Learn more in the **Clock viewer topic**.

icked.

The content of a timecode show can be seen in the **Timecode Viewer**.



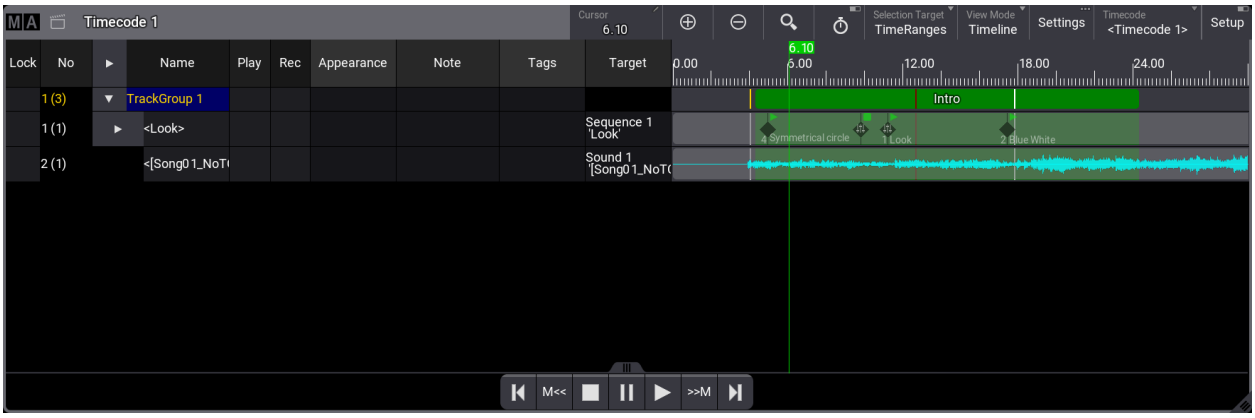
The Timecode Viewer window shows the content of the timecode show. The viewer can also be in a setup mode where the events can be added, edited, and deleted. The viewer can be a window in a view or a temporary pop-up.

Subtopics

- **Timecode Viewer**
- **Track Groups**
- **Markers**
- **Tracks**
- **Time Ranges**
- **Events**
- **Timecode Slots**
- **Timecode Settings**
- **Create a Timecode Show**
- **External Connections**

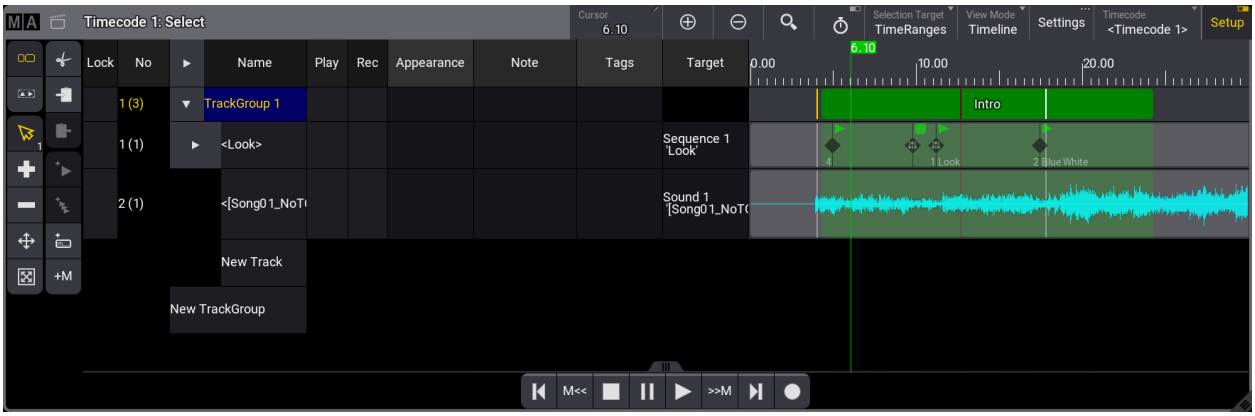
1.42.1. Timecode Viewer

The timecode viewer can be created as a window. It gives access to control the timecode show and can be in a setup mode where the timecode show can be edited.



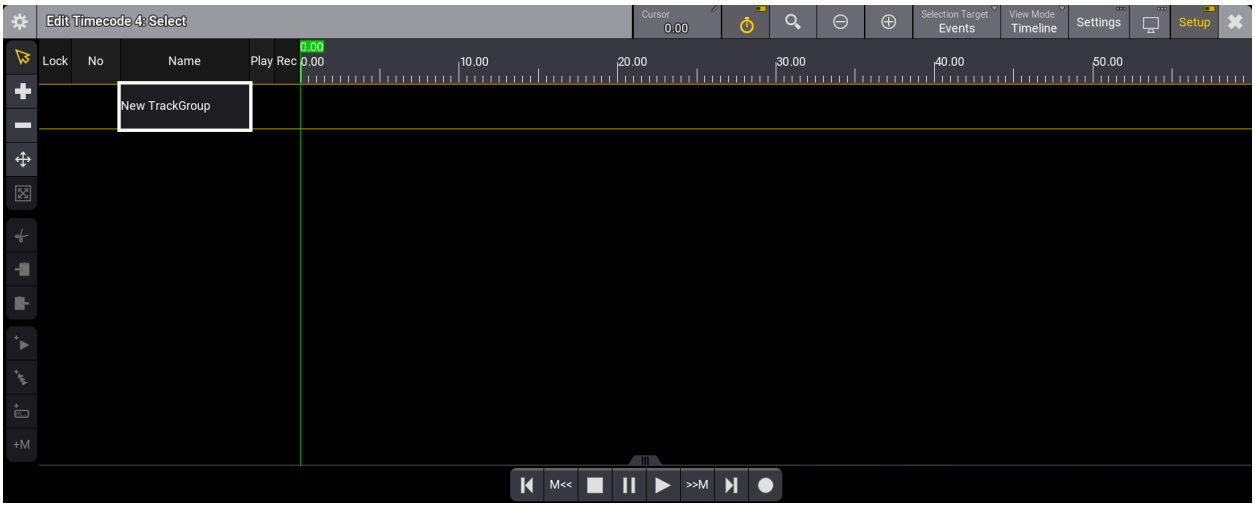
Timecode viewer window in normal mode.
In the normal mode, it can be used to see the timecode elements. The title bar gives access to some settings (this can be adjusted). A Playback toolbar can be visible at the bottom, giving quick access to playback functions.

The setup mode adds more elements to the viewer, turning it into an editor.



Timecode viewer in Setup mode.

The timecode viewer can also be opened as a pop-up editor by editing a timecode pool object.



Example of the timecode pop-up editor. The pop-up editor functions like the timecode viewer with Setup active. There are minor differences. A couple of tools are missing in the toolbar on the left, but the basic functionality is the same.

Title Bar

The default title bar gives access to the window settings by tapping the MA logo on the left side or the gear icon (⚙) in the pop-up editor. After the logo and the timecode icon, the title bar shows the timecode number and name (if the name is different than the default "Timecode"), and in setup mode, it also shows the selected tool. The buttons on the right can be edited, and they are described in the window settings **below**.

Left Side Toolbar

The left side toolbar is visible when the viewer is in setup mode. The toolbar is split into four different groups.

Select Target:

This group of tools defines which elements are being selected. These tools are only available in the timecode viewer with Setup active.

- **Select Time Range** (▢▢):
  Selecting this allows for the time ranges in the track groups to be selected.
- **Select Events** (▲▶):
  Selecting this allows the timecode events in the tracks to be selected.

Tools:

This is a group of tools interacting with the tracks and track groups.

- **Select Tool** (↖):
  This tool is used to select elements in the tracks and track groups. When there is a selection, there is a small number in the lower right corner of this button to indicate how many elements are selected.

- **Add Tool** ():
  This tool manually adds relevant elements based on the selected target and the tapped element.
- **Minus Tool** ():
  This tool manually removes tapped elements based on the tapped element and the selected target.
- **Move Tool** ():
  This tool can be used to move selected elements.
- **Resize Tool** ():
  This tool is used to change the length of a time range. The start location stays, but the end can be moved using this tool.

Selection:

This group has tools to cut, copy, and paste the selected elements. This group is only visible when the Select Tool is active (see above).

- **Cut Selection** ():
  This can be used to cut the selected elements.
- **Copy Selection** ():
  This can be used to copy the selected elements.
- **Paste Selection** ():
  This can be used to paste the cut or copied elements.

Add Menu:

This menu is used to add events and markers to the tracks

- **Add Event At Time Cursor** ():
  Tapping this adds an event on the selected track at the current time cursor location
- **Add Multiple Events At Time Cursor** ():
  Tapping this opens a small pop-up that allows to add multiple events with a specified time interval on the selected track beginning from the location of the time cursor.
- **Add Marker At Time Cursor** ():
  Tapping this will add a marker and open a pop-up where a name or text for the marker can be added. The marker is added to the currently selected track group.
- **Quickly Add Marker At Time Cursor** (+M) :
  Tapping this will add a marker to the currently selected track group without opening the name pop-up.

Playback Toolbar

The playback toolbar can be visible at the bottom of the viewer. The Playback Toolbar setting in the window setting defines if the toolbar is visible or hidden. Read about the settings **below**.

The toolbar looks like this when setup mode is active :

Playback toolbar in setup mode. This is a short explanation of the (up to) eight buttons:

- **Jump to start of timeline**( ):
  Tapping this jumps to the start of the timeline.
- **Jump to previous marker**(M<<):
  Tapping this jumps to the previous marker based on the current cursor location.
- **Stop**( ):
  This stops the timecode show.
- **Pause** ( ):
  This pauses the playback of the timecode show.
- **Play**( ):
  This activates play mode for the timecode show, which allows it to listen to the time signal and then play back the recorded events.
- **Jump to next marker**(>>M):
  Tapping this jumps to the next marker based on the current cursor location.
- **Jump to end of timeline**( ):
  Tapping this jumps to the end of the timeline.
- **Record**( ):
  Activates the record mode and allows the automatic record of executor events to the timecode show.

View Mode

The Timecode Viewer has two primary view modes (Text and Timeline) and a third (Both) that is a combination of the two.

The different modes do not change the left side of the window. This always shows the track groups and tracks.

The left side has the following columns:

- **Lock**:
  This can be used to lock a track or a track group.
- **No**:
  This is the number of the track group or track.
- **Name**:
  This is the name of the track group, track, and sub-track.
- **Play**:
  This can be used to disable playback of the track or track group. When the cell is empty, the track is played. Editing the cell adds a colored icon indicating that the track is not playing back.
  
- **Rec**:
  This can be used to prevent recording new events to the track or track group. When the cell is empty, new events can be recorded or overwritten. Editing the cell adds a colored icon indicating that the track is protected from changes when the record mode is active.
  
- **Appearance**:
  An appearance can be set for the track or track group. Learn more in the **Appearance** topic.

- **Note**:
  A note can be added to a track or track group. Learn more in the **Notes** topic.
- **Tags**:
  Tags can be added to a track or track group. Editing this cell opens the **Edit Tags** pop-up. Learn more in the **Tags** topic.
- **Target**:
  The track needs a link to a target object, which is often a sequence, sound, preset, group, or master. Editing this cell opens the **Assignment Editor** pop-up.

View Mode - Text

The text mode shows the events, tracks, and markers in a spreadsheet-type mode on the right side of the window.

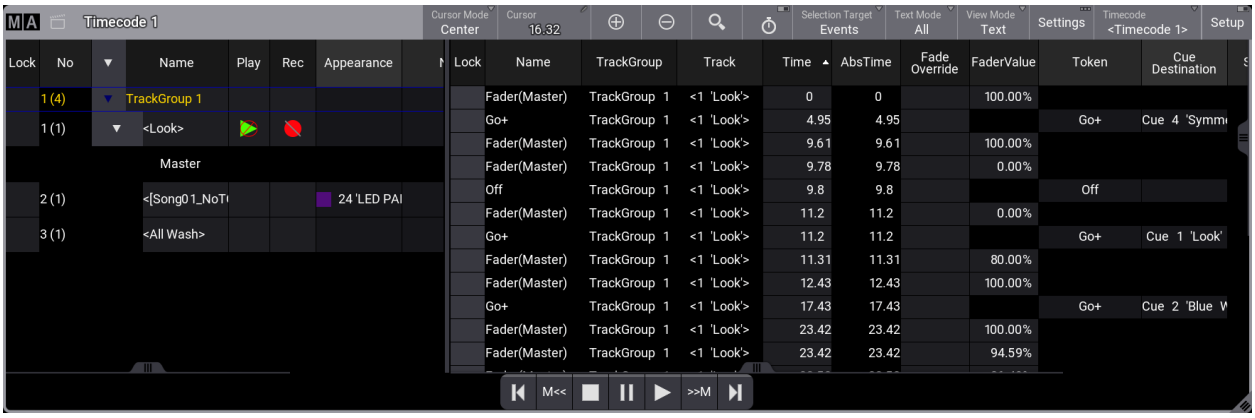
A **Text Mode** setting defines what information is shown. See more about the settings **below**.

There are four different modes. They can show the markers, the events, and the time ranges.

Be aware that the setting called **Selection Target** greatly influences what the Text Mode shows in the viewer. The selection target can be **Events** or **TimeRanges**. Learn more about the settings **below**.

- **All**:
  With the **Selection Target** being **Events**, the viewer shows all events in all tracks.
  With the **Selection Target** being **TimeRanges**, the viewer shows all markers and time ranges in all tracks.
- **Tracks**:
  With the **Selection Target** being **Events**, the viewer shows all events in the selected tracks.
  With the **Selection Target** being **TimeRanges**, the viewer shows the markers and time ranges for the selected tracks.
- **Selected**:
  With the **Selection Target** being **Events**, the viewer shows the selected events.
  With the **Selection Target** being **TimeRanges**, the viewer shows the selected markers and time ranges.
- **Markers**:
  This mode shows all the markers for the selected track group.

As a default, **View Mode**, **Text Mode**, and **Selection Target** are buttons in the title bar.



Example of the Timecode Viewer in Text mode showing the events.

The possible columns in text mode are:

- **Lock**:
  This can be used to lock the row.
- **Name**:
  This is the name of the element. The name of Event elements can not be edited.
- **TrackGroup**:
  This shows the track group for the element.
- **Track**:
  This shows the track.
- **Appearance** (Time Range):
  This is the appearance of the time range and markers.
- **Note** (Time Range):
  Notes can be added to the time ranges and markers.
- **Tags** (Time Range):
  Tags can be added or edited for the time range and markers.
- **Start** (Time Range):
  This can be used to edit the start time for the time range.
- **Duration** (Time Range):
  This can be used to edit the duration of the time range.
- **Speed Factor** (Time Range):
  This can be used to change the speed factor for the time range. Changing this will scale the time. The default value is 1.
- **Time** (Event):
  Reports the time of events inside their **Time Range**. The time can be edited.
- **AbsTime** (Event):
  Reports the absolute time of events inside the timecode show. The absolute time shows the time the event will trigger. The absolute time is a combination of the Time (above) and the start time for the time range. The start time for a time range functions as an offset.
- **Fade Override** (Event):
  This can be used to define a fade time for the event that overrides the cue fade time when the timecode show is played back.
- **FaderValue** (Event):
  When recording a track using a fader, the fader movement is recorded in steps between 0% and 100%. This column displays the fader's value at each step. Tap and hold a cell in this column to open the calculator, then set a new value.
- **Token** (Event):
  This column displays the action executed when the event is played back. Tap and hold a cell in this column to open the select token pop-up, then select a token.
- **Cue Destination** (Event):
  This is used to specify the cue related to the token (above). For instance, if the Token is **Go+**, and there is no cue destination, the next possible cue will be triggered. If a cue destination is defined, then the **Go+** will trigger the specified cue.
- **Status** (Event):
  This is used to see and edit the status of tokens that can have a status. For instance, if the token is **Flash**, an On or Off status can be used to define what the Flash should do.
- **Execute Command** (Event):
  Defines whether cue commands are executed when an event is played back. Tap and hold a cell in this column and set it to **Yes** or **No**. The default is **Yes**.
  If the cell shows "- - -", this indicates that the user cannot override the command execution of cue commands within the timecode show. Cue commands are recorded and later changes of

the commands in cues are not part of the timecode show. This is only valid when the **Playback and Record** setting is set to **All Events**.

View Mode - Timeline

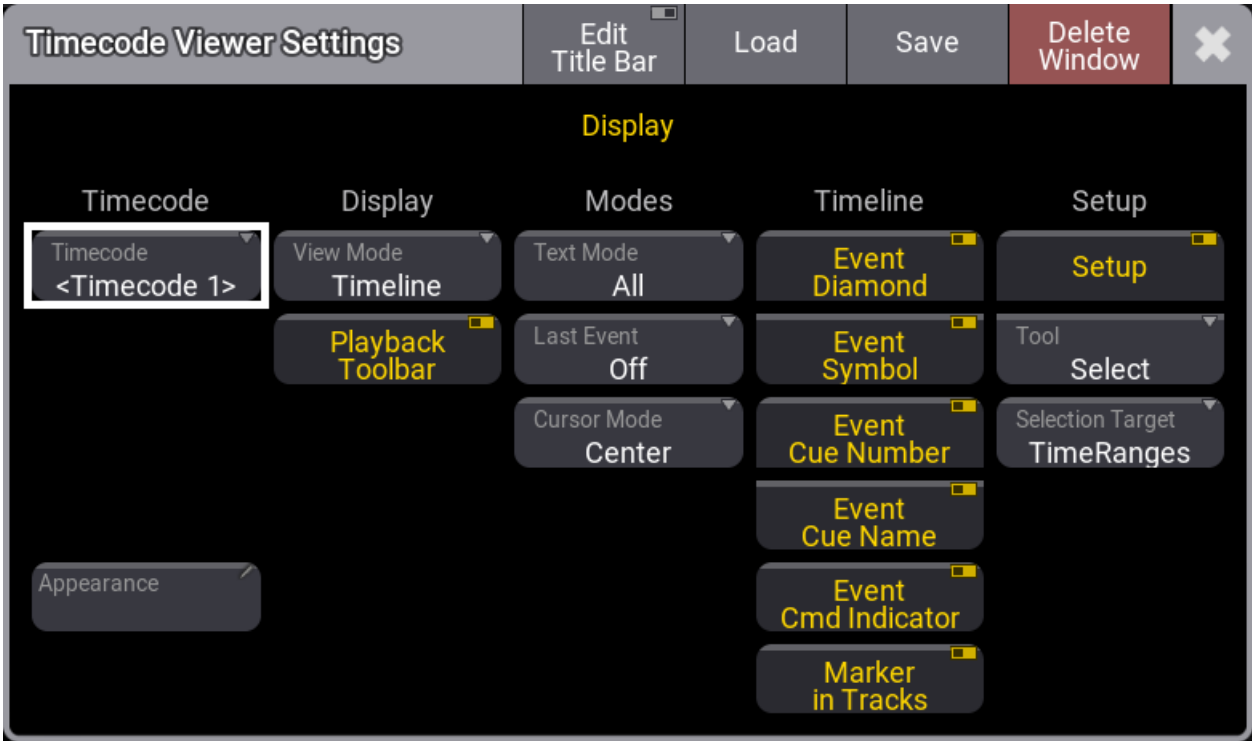
This displays the timecode show as a timeline where the tracks are rows, and events, markers, and time ranges can be shown in the timeline.

View Mode - Both

This shows both the timeline and the text mode next to each other.

Window Settings

The window settings can be accessed by tapping the MA logo in the upper right corner of the viewer or the gear icon (⚙) in the pop-up editor.



The Timecode Viewer Settings in normal mode.

Some settings are hidden and can be seen when the Edit Title Bar mode is on. Most of the hidden settings are button functions meant for the title bar.

Timecode Viewer Settings in edit mode. The settings are split into different sections.

Timecode

- **Timecode**:
  The timecode setting selects which timecode show is displayed in the viewer. The options include selecting a specific timecode show, but "<Link Selected>" can also be used to have the viewer always display the selected timecode show. Because the viewer can have different **View Modes** (**read above**), it could make sense to have multiple viewers visible linked to the selected timecode show, each displaying the timecode show in different modes.
- **Settings**:
  This toggles if the title bar has a button opening the Timecode show settings.
- **Timecode Cursor**:
  The title bar can have a Cursor input field. The field shows the current cursor location. Tapping the field (in the title bar) opens the calculator, which allows a new cursor position to be entered. If the time source is a timecode slot, then the slots frames per second are displayed in a small square. For instance, if the timecode slot is set to 30 fps then this icon is displayed:
  30
- **Timecode Offset**:
  A timecode show can have an offset from the received timecode source. This offset can be displayed in the title bar in an **Offset TC Slot** input field. Tapping the input field opens a calculator where the offset can be set. The offset is a part of the timecode shows settings.
- **Appearance**:
  Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.

Display

- **View Mode**:
  Tapping this toggles between the three different view modes. Tap and hold to open a small

select pop-up with the same options. It changes the right side of the viewer. Read more about the different view modes **above**.

- **Text**:
  The right side of the view displays events, markers, timeranges in a text grid with rows and columns.
- **Timeline**:
  The right side of the view displays the timeline of the timecode show. The tracks are displayed as rows and events, markers and time ranges are displayed along these tracks based on time
- **Both**:
  Both Text and Timeline are displayed on the right side.

- **Playback Toolbar**:
  The playback toolbar can be visible at the bottom of the viewer. Read about the playback toolbar **above**.
- **Reset View()**:
  This toggles if the Reset View icon is visible in the title bar. Tapping this button resets the zoom level in the timeline view mode.
- **Zoom In()**:
  This toggles if the Zoom In icon is visible in the title bar. Tapping this button zooms in on the timeline.
- **Zoom Out()**:
  This toggles if the Zoom Out icon is visible in the title bar. Tapping this button zooms out on the timeline.

Modes

- **Text Mode**:
  This mode defines what information is shown in the viewer when the **View Mode** is Text or Both. Read about the settings **above**.
- **Last Event**:
  Defines if the last played-back event in the timecode show is selected. It has the following options:
  - **Off**:
    The last played-back event will not be selected.
  - **Track**:
    The last event played back in the selected track will be selected.
  - **All**:
    Select the last played-back event, no matter which track.
- **Cursor Mode**:
  When the Follow Time Curcor () is enabled in the title bar, the cursor stays visible when the timecode runs. The following options are available:
  - **Page**:
    When an external time signal is coming in or generated by the console, the cursor (green timeline) moves until it reaches the end of the visible part of the timeline. Then, a new section of the timeline is displayed. The cursor moves from the beginning to the end of that new section, repeating until the timecode is stopped.
  - **Center**:
    When an external time signal is coming in or generated by the console, the cursor (green timeline) moves until it reaches the middle of the visible area of the timeline. The timeline starts moving to keep the cursor centered on the display. This is the default setting.

- **Follow Time Cursor(⏱)**:
  This toggles if the icon is visible or hidden in the Title bar.

Timeline

This can be used to toggle if different elements are shown or hidden in the timecode timeline. See the **Event topic** and the **Marker topic** to learn about the different elements.

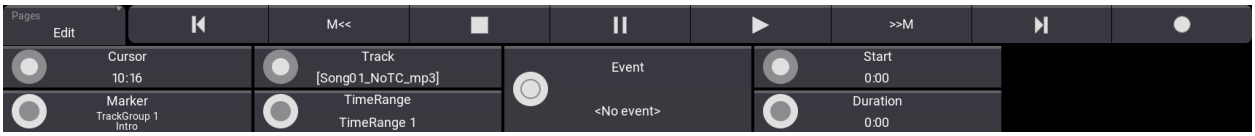
- **Event Diamond**:
  Toggle on to display event diamonds on the timeline.
- **Event Symbol**:
  Toggle on to display the green event icon next to the events on the timeline.
- **Event Cue Number**:
  Toggle on to display the cue number next to the events on the timeline.
- **Event Cue Name**:
  Toggle on to display the cue name next to the events on the timeline.
- **Event Cmd Indicator**:
  Toggle on to display the command indicator on or next to the events on the timeline.
- **Marker in Tracks**:
  Toggle on to display markers on all tracks of a track group. The markers will only be shown in the track group section when toggled off.

Setup

- **Setup**:
  This toggles the viewer's setup mode. When in setup mode, it is possible to edit the timecode show.
- **Tool**:
  This swipe button opens a list with the same tools found in the toolbar on the left side - read about them **above**. It can be used to select the desired tool. Besides the tools described above, there is an extra option called **Operate**. This does not have a tool function, it can be selected to avoid having one of the other tools selected.
- **Selection Target**:
  Toggle to put the focus on Events or Time Ranges. Read more **above**.

Timecode Encoder Bar

When the timecode viewer has Setup active and has focus, or when the timecode pop-up editor is open, the encoder bar changes to the Timecode Encoder Bar.



The Timecode encoderbar. The top row has the same buttons as the **Playback Toolbar**. Read more **above**.

The encoders has the following functions:

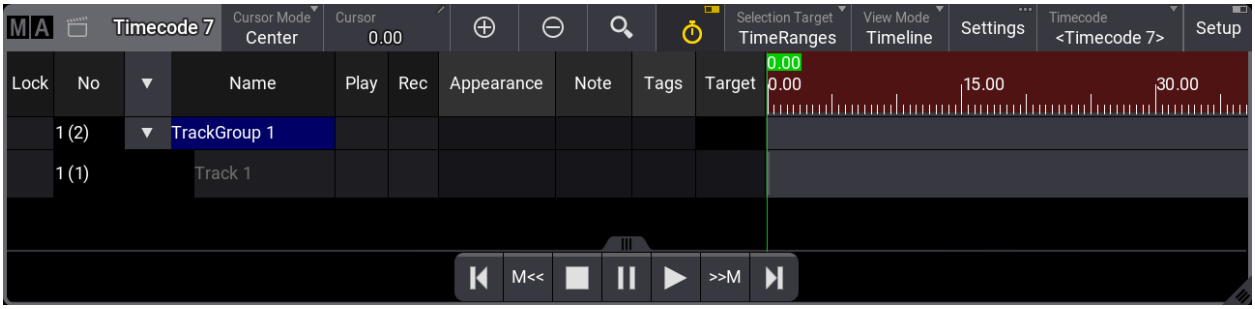
- **Cursor**:
  Rotate the encoder to change the cursor location. Pressing while turning moves the cursor faster. Pressing the encoder shortly opens the calculator to input a specific cursor location.

- **Marker**:
  Rotate the encoder to jump between markers in the selected track group. Press the encoder button (next to the encoder) to open a small **Select Marker pop-up**.
- **Track**:
  Rotate the encoder to select a Track or Track Group. Press the encoder shortly to open a small **Select Track pop-up**.
- **TimeRange**:
  Rotate to select different time ranges on the selected track or track group.
- **Event**:
  Rotate any of the two encoders to select the different events on the selected track or track group. Press the inner encoder shortly to open the small **Select Token pop-up**, where the token can be changed or the calculator to specify the fader position (if the selected object is a fader event). Press the encoder button (next to the encoder) to open a small **Select Event pop-up**.
- **Start**:
  When an event is selected, turning the encoder moves the event on the timeline.
  When a time range or marker is selected, turning the encoder moves the starting point for the element.
- **Duration**:
  Rotate the encoder to adjust the duration of the marker or time range.

1.42.2. Track Groups

Timecode shows are arranged in a hierarchical structure where **Track Groups** are the children of the timecode show. Track groups can contain one or more **Tracks**.

Markers can be assigned to the track group. **Time Ranges** are assigned to tracks. Read the following topics to learn more about these elements.



A timecode show with a single Track Group.

To select a track group, press Select and tap the desired track group in the viewer. The selected track group has a yellow text label.

Track groups can be expanded and collapsed by tapping the arrow on the left-hand side of the track group name.

Track groups can also be selected via the command line. For example, select track group 2 of timecode show 1:

Tracks can be added to track groups by tapping and holding New Track Group when setup mode is active. Learn about setup mode in the **Timecode Viewer topic**.

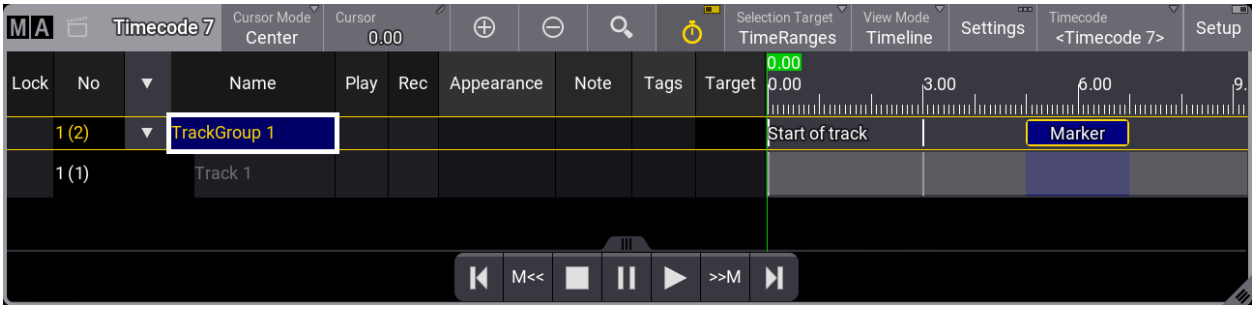
Deleting a track can be done in setup mode. Press Delete and then the desired track group.

1.42.3. Markers

Markers can be used to mark specific places in a **Track Group**. This could be used to mark when a specific part of a song or event is happening.

A vertical line indicates the position of a basic marker through all tracks in the track group.

Markers can be labeled. The name is shown in the track group. An appearance is displayed vertically through all tracks. The marker might need a duration for the appearance to be visible.



Timecode show with markers. The example above shows three different markers. The first one (at time 0) has a name. The second one is a simple marker with a label. The third one has a range, name, and appearance.

The two M<< and >>M buttons in the playback toolbar jump to the previous and next marker in the timeline.

Notes and Tags on markers are not indicated in the timeline view. They can be seen in text mode. Learn about the different modes in the **Timecode Viewer topic**.

Add Markers to the Track Group

To add a marker, the Setup mode must be active. Then, follow these steps:

1. Make sure the **Selection Target** is set to **TimeRanges**. This can be done by tapping in the toolbar on the left. If the Selection Target is visible in the title bar, it can be tapped to change the target.
2. Select the track group where the marker should be added.
3. Position the cursor at the correct location in the timeline.
4. Tap in the toolbar. The Edit Name pop-up opens.
5. Enter a name for the new marker and press Please or tap Enter.
6. The new marker is added.

Another option is the quick marker tool.

1. Make sure the **Selection Target** is set to **TimeRanges**. This can be done by tapping in the toolbar on the left. If the Selection Target is visible in the title bar, it can be tapped to change the target.

2. Select the track group where the marker should be added.
3. A marker is added every time the +M is tapped.

This can be useful during recording as it does not prompt for a name during creation.

1.42.4. Tracks

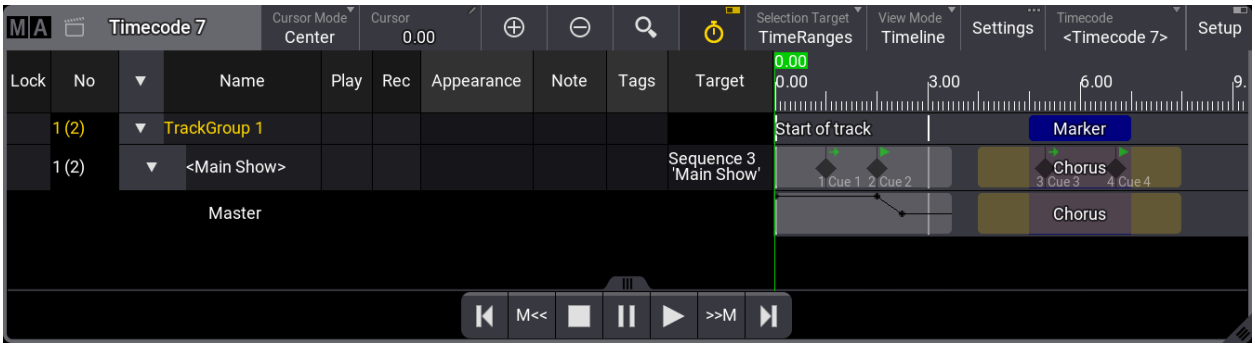
A **Track Group** can contain multiple **Tracks**. The track contains one or multiple **Time Ranges**. The time range can contain **Events**. The Track can have one or multiple **Subtracks**. Subtracks contain information about fader position and movement.

A track should have a **Target**. The target can be one of the following objects:

- Sequence
- Sound
- Timecode
- Timecode Slot (TCSlot)
- Preset
- Group
- Master

Subtracks

Subtracks contain information about fader positions and transitions.



An example timecode show with a Master subtrack.

Subtracks can be added by starting a recording and then moving a fader or rotating a knob. The timecode show will record a series of events containing a fader value. These points will recreate the fader movement. The events can be edited to only contain the relevant events.

Be aware that if **Auto Start** and **Auto Stop** are activated, then moving the master fader from and to a zero value might also record cue events.

Subtracks appear as rows nested in the tracks. The row does not have any settings that can be changed and appears with a black background. The name column indicates what fader type it affects. Only the events in the subtrack can be edited.

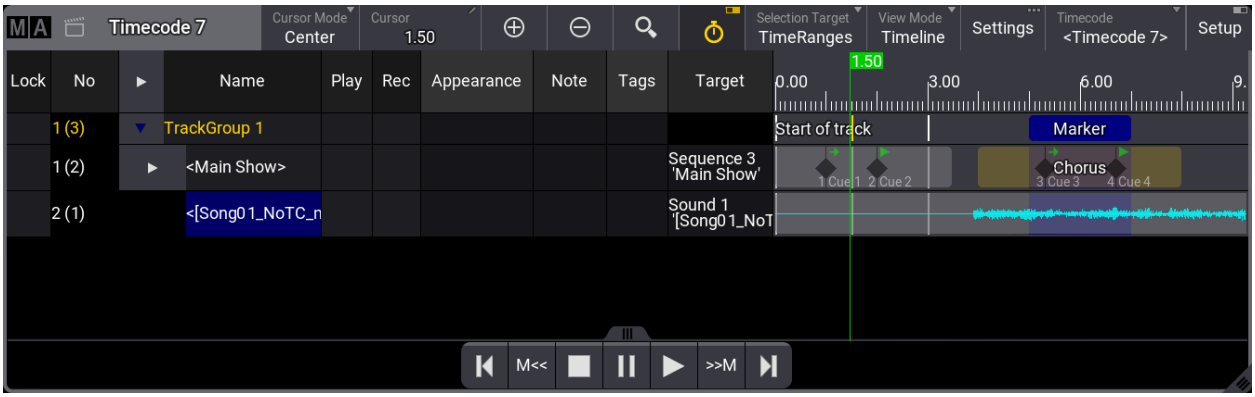
Subtracks can be deleted by pressing Delete and then tapping the subtrack.

Sound Track

Sound pool objects can be assigned to tracks and will display the waveform in the timecode show. For more information, see **Sounds Pool**.

The left from the right of the audio track is divided by a horizontal line across the waveform.

The sound assigned to a track is played back when playing the timecode show. For more information, see **Local Settings** and **Connect Sound Out**.



A timecode show with a sound track.

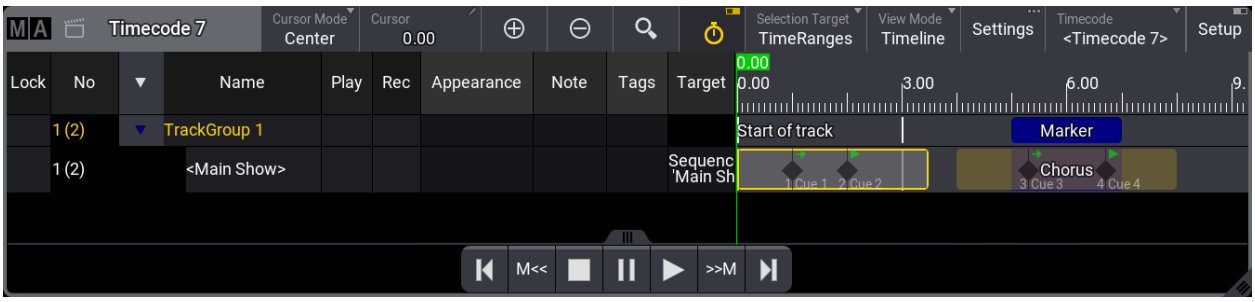
Events cannot be added to the sound track.

A warning icon (⚠) will be displayed on the left side of the track if the sound file is corrupted.

1.42.5. Time Ranges

Time Ranges can be used to mark a specific area in a **Track**. This could be used to indicate a specific grouping of **Events** in the track.

When a track is created, a Time Range is automatically created. This range automatically has the duration of the timecode show.



An example image with multiple time ranges in the timecode show.

A track can have multiple time ranges, and they can overlap. Events are children of the time range.

This is useful if there is a group of events that repeat. Then, the events can be stored in their own time range, and this range can be copied to multiple locations. The events' **Time** value is relative to the beginning of the time range. Moving the time range changes an event's calculated (Event Time + Time Range Start) **Absolute Time**, but not the **Time**.

A time range can be added to a track by tapping + followed by tapping the track when the setting **Selection Target** is **TimeRanges**. Tapping and dragging in the track will add the time range and the duration based on the range dragged.

The move tool () can be used to move the start time for the time range.

The resize tool () can be used to adjust the duration for the time range.

Time ranges can be copied or cut and then pasted into tracks using the tools in the toolbar on the left.

Deleting the time range by pressing Delete or tapping - followed by the time range also deletes the events in the time range.

Time ranges can have a label. The timeline view shows the name and appearance on the track and subtracks. Notes and tags can only be seen in the text view. Learn about the different views in the **Timecode Viewer topic**.

1.42.6. Events

The tracks can have Events positioned on the timeline. Events can have a **Token** assigned. Tokens are the actions an event triggers.

The assigned token is executed when the cursor (green timeline) reaches an event.

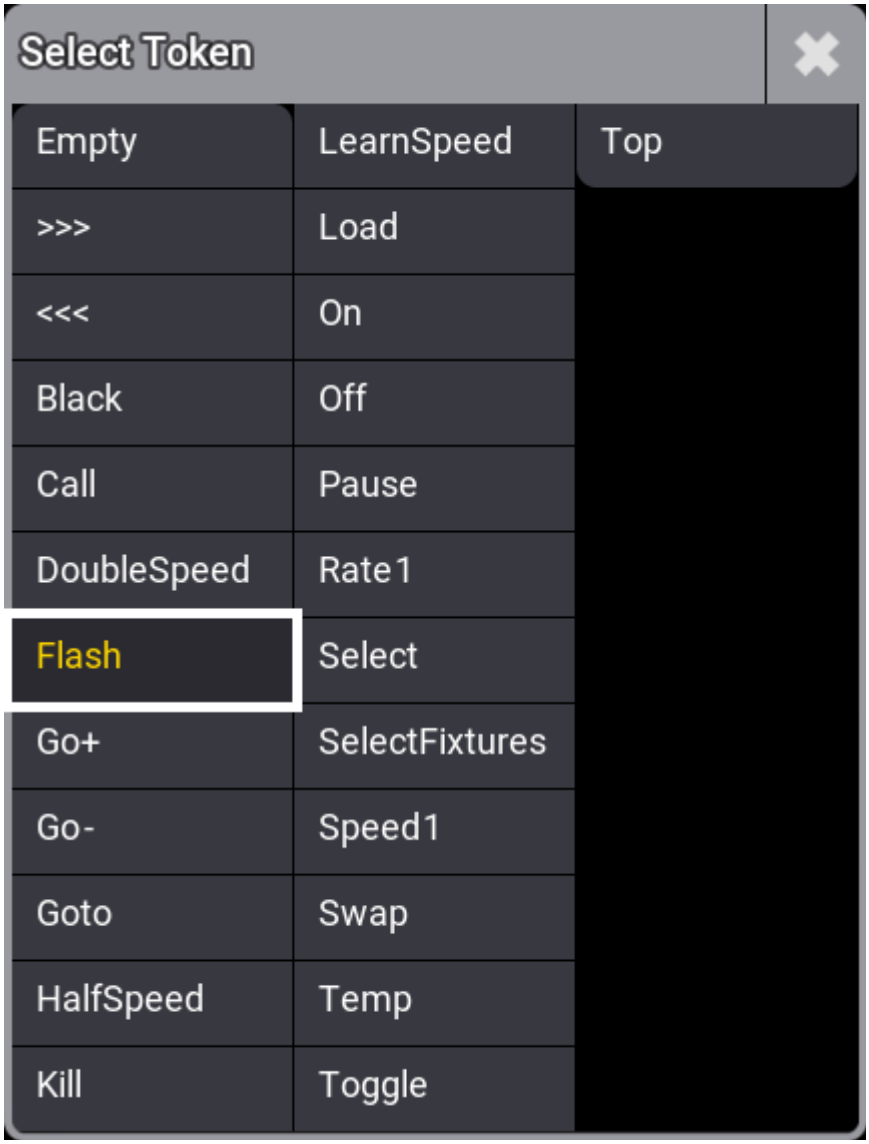
A diamond marks the event in the timeline.

An event diamond in the timeline.

The diamond is black when it is not selected and yellow when selected. There are some icons and text around the diamond. The visibility of these and the actual diamond can be changed in the window settings. Learn more about the settings in the **Timecode Viewer** topic.

The green icon above the diamond indicates the token action. Each action has its own icon. The token can be edited in the text mode or by pressing the event encoder in the encoder toolbar when in setup mode.

This is the list of available tokens for a cue:



Select Token pop-up. Some tokens have an extra status setting. For instance, **Flash**. This token can be On or Off to indicate the status of the token. The different status' changes the icon in the timeline.

Events automatically recorded will display an icon next to the event diamond or in the center of it:

- This event was recorded as a timed cue, for example, triggered by the follow or time triggers.
- Command icon:
  When displayed on the right side of the event diamond, a command was executed by the timecode event.
  When displayed in the center of the event diamond, a command was executed in the triggered cue.
- This event was triggered by moving a fader, and the assigned sequence's **Auto Start** or **Auto Stop** started or stopped the sequence.
- When a playback is switched off via Off When Overriden, for more information, see **Sequence Settings**.

-  The event was created by executing a macro. For more information, see **Create Macros**.
-  This icon is displayed when two or more events are too close to one another to be displayed. It disappears when zooming in.
- This icon is displayed when a toggle event is recorded. It will be accompanied by this icon  if the toggle starts the sequence or by this icon ■ if the toggle stops the sequence.
-  The event was created via a DMX remote.
-  This event was created via MIDI remote.
-  This event was created via a DC remote.

For more information about remotes, see **Remote In and Out**.

For events with disabled commands, the command icon will be displayed in red if you set **Execute Command** to **No** in the timecode show or by switching off Command Enable in the sequence.

See the **Create a Timecode Show** to learn about adding events.

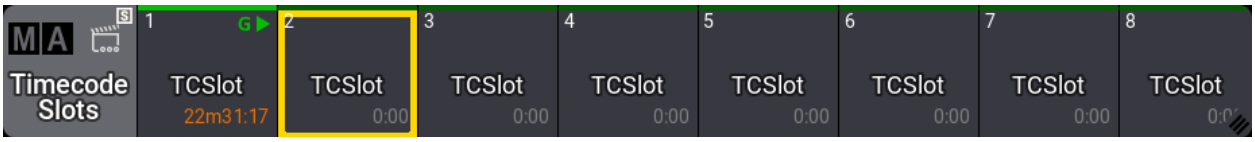
1.42.7. Timecode Slots

	Hint:
	The timecode slots can be edited but not added, copied, deleted, or moved.

A timecode slot is an integrated interface that interprets a timecode signal in hours (h), minutes (m), seconds (s), and frames (f).

The grandMA3 can receive up to eight different external timecode signals at the same time.

Timecode slots are located in the Timecode Slots pool.



The Timecode Slots pool. Each pool object represents a timecode slot. The slot can listen to external time sources and start counting when a time signal is received. Each slot can also generate a time signal.

	Important:
	Timecode slot settings are not part of the show file and will not be transmitted to another station within the show file.

Generator

The system can generate a time signal. Each slot is capable of generating a signal.

The generated signal can be used in the session or transmitted to external devices.

The generator can be controlled by sending commands to a timecode slot following this syntax:

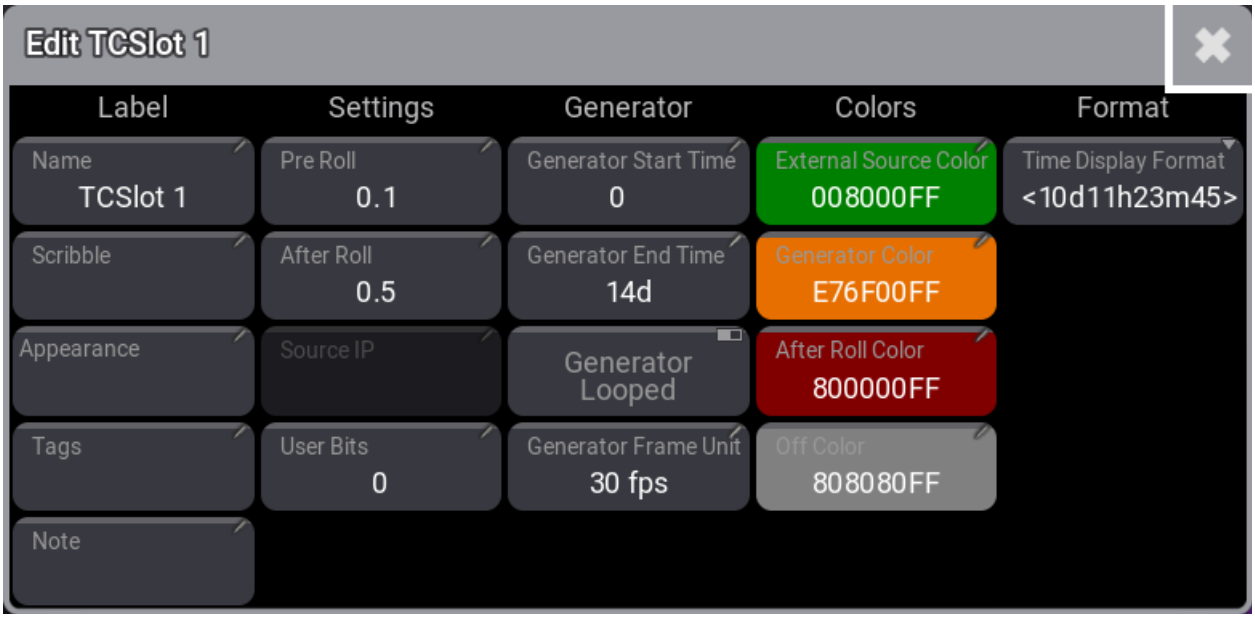
[Function] TimecodeSlot ["TimecodeSlot_Name" or TimecodeSlot_Number]

The generator supports the following functions:

- **Select**:
  Select the timecode slot pool object.
- **On**:
  Starts the generator at the current time.
- **Go+**:
  Starts the generator at the current time.
- **Pause**:
  It turns off the generator and does not reset the time.
- **Off**:
  Turns off the generator and resets the time.

Edit a Timecode Slot

Edit a timecode slot pool object using any of the edit methods. This opens the editor:



Example of the Timecode Slot Editor. The settings are separated into different sections.

Label

This is a short description of the settings in the **Label** section.

- **Name**:
  This is name of the timecode slot.
- **Scribble**:
  A scribble can be added to the timecode slot. Learn more in the **Scribble topic**.
- **Appearance**:
  An appearance can be added to the timecode slot. Learn more in the **Appearance topic**.
- **Tags**:
  Tags can be added to the timecode slot. Learn more in the **Tags topic**.
- **Note**:
  A note can be added to the timecode slot. Learn more in the **Notes topic**.

Settings

This is a short description of the settings in the **Settings** section.

- **Pre Roll**:
  States how long an external signal must be received before the timecode slot uses it. This is only relevant for external time sources.
- **After Roll**:
  If the external time signal stops, the timecode runs internally in the After Roll. For example, if 10 seconds were set, the time runs for another 10 seconds, even though the external signal ceased. This can be useful if the source is a little unstable to prevent the slot from starting and stopping unintentionally.
- **Source IP**:
  Displays the IP address of the grandMA3 device, which receives the running timecode signal.
- **User Bits**:
  Besides the 32 Bit for 8-digit timecode time, timecode executes 32 User Bits (8-digit) per frame.

Use User Bits to mark a timecode signal. For example, use User Bit 1 for light and User Bit 2 for pyro. This is not available when using an internal timecode source.

	Important:
	Do not change the user bits unless you are told to do so by the timecode supplier. If the user bits are changed and the timecode supplier does not know, the timecode signal will not be received anymore.

Generator

This is a short description of the settings in the **Generator** section.

- **Generator Start Time**:
  Sets the start time for the internal timecode generator.
- **Generator End Time**:
  Sets the end time for the internal timecode generator.
- **Generator Looped**:
  Toggle this On to repeat the internal timecode range indefinitely. The time range is only generated once if this is Off.
- **Generator Frame Unit**:
  Sets the frame unit for the generated signal.

Colors

This is a short description of the settings in the **Colors** section. The color is applied to the time displayed in the Clock and in the pool object. Tapping any of the settings opens the Color Editor pop-up, where a color can be selected.

- **External Source Color**:
  Defines the color when the time source is external.
- **Generator Color**:
  Defines the color when the time source is the generator.
- **After Roll Color**:
  Sets the color to indicate that the external signal was interrupted and that the timecode slot is in After Roll.
- **Off Color**:
  Sets the color of the clock when no signal is coming in.

Format

This is a short description of the settings in the **Format** section.

- **Time Display Format**:
  This can be used to define how the time should be displayed. The following options exist:
  - **Default**:
    This follows the user profile setting called **Time Readout**. Learn more in the **User Settings topic**.
  - **10d11h23m45**:
    The time is separated into days, hours, minutes, and seconds using letters as separators.

- **251h23m45**:
  This is separated into hours, minutes, and seconds using letters as separators. The hour number can become more than 24 if time is more than a day.
- **10.11:23:45**:
  The time is separated into days, hours, minutes, and seconds using a dot and colons as separators.
- **251:23:45**:
  This is separated into hours, minutes, and seconds using colons as separators. The hour number can become more than 24 if time is more than a day.

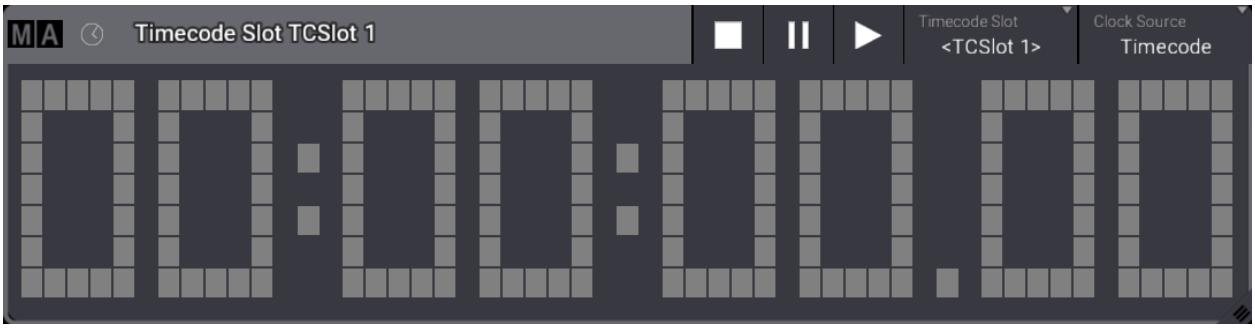
Clock

To use the clock to display the timecode slot, open the **clock window**.

The Clock Viewer shows the time. It can be the system time, the time of a timezone, the timer of a timer, or timecode slot.
Learn more in the **Clock viewer topic**.

To display the timecode clock, tap and hold Clock Source in the title bar to open the drop-down menu, then select Timecode.

Tap and hold Timecode Slot in the title bar to open the drop-down menu, then select the desired timecode slot. When set to **<Selected>** the selected timecode slot from the Timecode Slot Pool is displayed.



Clock window with Clock Source set to Timecode

The buttons in the title bar of the clock viewer can be used to control the timecode slot generator (Off, Pause, and Go+). These actions do not add Command Line History feedback. Learn about the generator **above**.

For general information on the clock, see **System – Clock**.

1.42.8. Create a Timecode Show

There are two basic ways to create a timecode show. The timecoded events can be recorded or added manually.

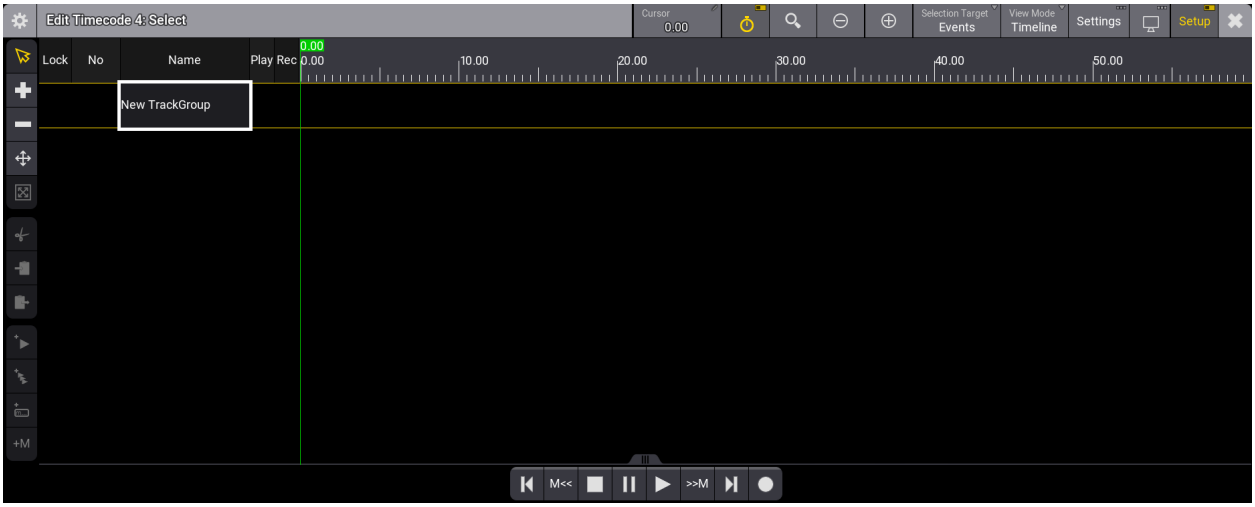
The two workflows can be combined.

A timecode show needs to be created in the Timecode pool. A new show can be added by performing a store or edit action on an empty pool object.

This creates an empty timecode show.

This show is created using the default settings. There are many settings connected to timecode shows and timecode elements like the viewer and the slots. Learn about the settings in the **Timecode Settings** topic.

If the edit action is used on the pool object, a pop-up editor appears, which functions like the **Timecode Viewer** with **Setup** active.



The timecode pop-up editor displays an empty timecode show with Setup active.
The pop-up editor is missing some tools in the toolbar on the left, which are available in the Timecode Viewer with Setup active. So, it might be better to use the **Timecode Viewer**.

The Timecode Viewer is described in the **Timecode Viewer topic**. This includes detailed descriptions of the different tollbars in the viewer.

Time Source

One of the most important elements is the source of the time. Each timecode show has a **setting** called **TC Slot**. This can be "Internal" or one of the Timecode Slots. Learn more about the timecode slots in the **What are Timecode Slots** topic.

There are a lot of different settings related to the timecode shows. Learn more about them in the **Timecode Settings Topic**.

If it is internal, it can be run by a Go command to the timecode show. This will put the show in play mode and run the time. The show's playback actions have an easy-access **Playback Toolbar** at the bottom of the Timecode Viewer window. This includes a play button(▶).

This toolbar allows control of the playback status of the timecode show. When the setup mode is active, the extra record button is available. Learn more in the **Timecode Viewer** topic.

If the time source is one of the timecode slots, then the timecode show must be in playback mode, and then the timecode slot can generate the time, or an external source can make the time run in the timecode slot.

Initially, a new timecode show is set to use the internal time source. If the timecode show should use an external source, then this source must be set up, and the timecode show must be set to listen to the relevant timecode slot. Learn how to set up external sources in the **External Connections** topic.

Record a Timecode Show

The timecode show can be recorded using the **Record keyword**. It also has a quick access button (●) on the **Playback Toolbar** at the bottom of the Timecode Viewer when Setup is active.

The Record keyword is used to start the recording mode of the timecode show. Learn more in the **Record Keyword topic**.

This toolbar allows control of the playback status of the timecode show. When the setup mode is active, the extra record button is available. Learn more in the **Timecode Viewer** topic.

When the record mode is active and the selected time signal is running, then the executor actions are automatically added to the show in relevant tracks. This includes the fader movements. The different events are added to tracks. Each track represents a target. A target object can be sequences, presets, sounds, group masters, masters, and other timecode shows and slots.

This requires that objects are assigned to executors and that these are valid timecode show target objects. Learn about the valid object in the **Tracks** topic.

The recording can be stopped using the stop button (■) or the Off command on the timecode show pool object.

The recorded events can be edited after the recording.

As a default, the tracks are added to the same **Track Group** and the same **Time Range**.

Track Groups are a way to organize the tracks. Learn more in the **Track Groups topic**.

Time Ranges are ranges of the time in the timeline. Time ranges are a part of the tracks. Each track must have one time range. Learn more in the **Time Ranges** topic.

If the **Timecode Settings** have the **Record Remote Events** active and the **Playback and Record** setting is **All Events**, then events triggering cues or other valid actions are automatically added to the timecode show. These settings also record events triggered by the cues in the timecode show. For instance, follow cues or cue commands.

The timecode settings contain all settings for the timecode show object. Learn more in the **Timecode Settings** topic.

Manually Add Events

Events can manually be added to tracks. The track must exist, and these can be manually created. The tracks must exist inside a Track Group. The track group must be created in an empty show, and then the track can be added.

	Restriction:
	Subtracks cannot be manually created.

Add a Track Group

Follow these steps to manually create a Track Group:

1. Make sure to have a **Timecode Viewer** visible and activate **Setup**.
2. Ensure the viewer displays the correct **Timecode Show** by either having the viewer display the selected timecode show and select the correct show or by selecting the correct timecode show in the timecode settings.
3. Tap and hold New Track Group.

Now, a new track group has been added. It can be unfolded by tapping the right-pointing white triangle ( ). This reveals that it already has one track with a time range.

Add a Track to a Track Group

Follow these steps to add a track to the track group:

1. Make sure to have a **Timecode Viewer** visible and activate **Setup**.
2. Ensure the viewer displays the correct **Timecode Show** by either having the viewer display the selected timecode show and select the correct show or by selecting the correct timecode show in the timecode settings.
3. Unfold the desired **Track Group**.
4. Tap and hold New Track.

A new track has been added to the track group.

5. Now, edit the **Target** cell for the track. This opens the Assignment Editor pop-up.
6. Select the desired object type by tapping one of the tabs at the top of the editor.
7. Select the desired target object in the list.

The track is now ready to get some events.

Points 5 to 7 can also be used to edit the desired target. Be aware this will break existing events in the track.

Add a Single Event to a Track

Single events can be added to the selected track. There are two ways to add events. The beginning is the same for both ways:

The first involves the green cursor:

1. Make sure to have a **Timecode Viewer** visible and activate **Setup**.
2. Ensure the viewer displays the correct **Timecode Show** by either having the viewer display the selected timecode show and select the correct show or by selecting the correct timecode show in the timecode settings.
3. Unfold the desired **Track Group**.
4. Select the desired **Track**.
5. Make sure the **Selection Target** is **Events** by tapping Selection Target in the title bar or by opening the settings and changing the setting in there . If the Timecode Viewer with setup active is used, then the  icon can be tapped to select the events as the selection target.

The first method involves the green cursor:

6. Move the cursor to the desired time. The cursor can be moved by tapping and holding it while moving it in the timeline or by using the **Timecode Encoder Toolbar**.
7. Tap  in the toolbar on the left. This adds the default event to the track.

These two steps can be repeated until all the events exist.

The second method involves tapping the track:

6. Activate the Add tool () in the toolbar on the left.
7. Tap the track where the event should be.

Repeat point 7 until all desired events exist.

	**Important:**
	Manually created cue events only trigger follow cues, timed cues, or cue commands if the **Playback and Record** setting is **Manual Events**. Learn more about this in the **Timecode Settings** topic.

Add Multiple Events to a Track

Multiple events can be added to the selected track:

1. Make sure to have a **Timecode Viewer** visible and activate **Setup**.
2. Ensure the viewer displays the correct **Timecode Show** by either having the viewer display the selected timecode show and select the correct show or by selecting the correct timecode show in the timecode settings.
3. Unfold the desired **Track Group**.
4. Select the desired **Track**.

5. Make sure the **Selection Target** is **Events** by tapping Selection Target in the title bar or by opening the settings and changing the setting in there . If the Timecode Viewer with setup active is used, then the icon can be tapped to select the events as the selection target.
6. Move the cursor to the desired time. The cursor can be moved by tapping and holding it while moving it in the timeline or by using the **Timecode Encoder Toolbar**.
7. Tap in the toolbar on the left. This opens the **Add multiple events** pop-up.

Add Multiple Events pop-up.

8. Type the number of desired events.
9. Change the time interval between the events.
10. Tap Add.

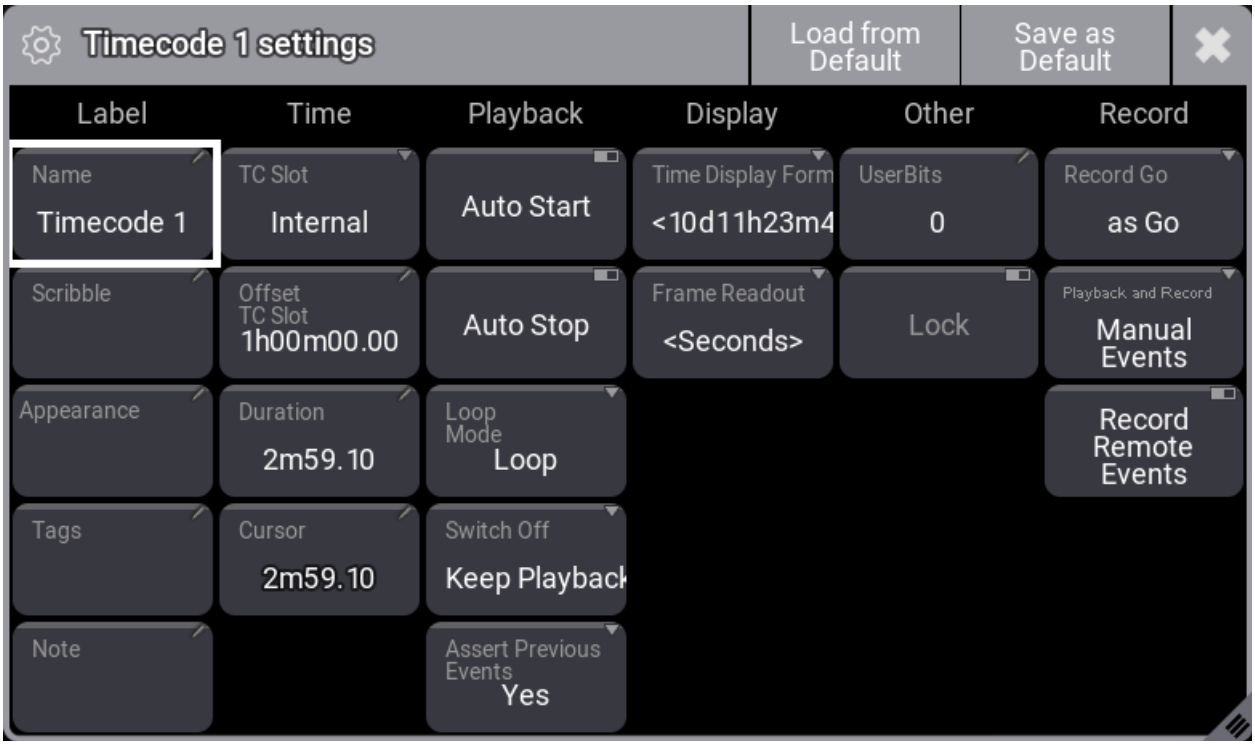
This adds the number of default events to the timeline using the interval from the cursor's location.

	Important:
	Manually created cue events only trigger follow cues, timed cues, or cue commands if the **Playback and Record** setting is **Manual Events**. Learn more about this in the **Timecode Settings** topic.

1.42.9. Timecode Settings

Timecode Settings

To open the Timecode Settings, tap Settings on the title bar of the **Timecode Viewer** window or press Edit twice (**EditSetting keyword**), followed by the Timecode pool object.



Timecode settings pop-up

The settings are split into different sections.

Label

This is a short description of the settings in the **Label** section.

- **Name**:
  This is the name of the timecode show.
- **Scribble**:
  A scribble can be added to the timecode show. Learn more in the **Scribble topic**.
- **Appearance**:
  An appearance can be added to the timecode show. Learn more in the **Appearance topic**.
- **Tags**:
  Tags can be added to the timecode show. Learn more in the **Tags Topic**.
- **Notes**:
  A note can be added to the timecode show. Learn more in the **Notes topic**.

Time

This is a short description of the settings in the **Time** section.

- **TC Slot**:
  This defines the time source for the timecode show. It has the following options:
  - **<Internal>**:
    The timecode show uses internal timing and is not connected to a timecode slot.
  - **<Selected>**:
    This links the timecode show to the selected timecode slot.
  - **TCSlot 1** to **TCSlot 8**:
    This sets a specific timecode slot as the source for this timecode show.
- **Offet TC Slot**:
  Tap to open the calculator, then set the offset for the timecode slot.
  When a timecode slot other than internal is set for a timecode show, Offset TC Slot button can be displayed on the timecode viewer title bar.
  If an offset is set to a timecode slot, the entire timecode show will be forwarded to match that offset.
  For example, if an event in the timeline is added at 1m00.0, and an offset of 1h00m00.0 is set to the selected timecode slot, the event will be triggered when the incoming timecode signal reaches 1h01m00.0.
  If programming a one-hour show, such as a musical, it is a good habit to have a timecode show for each act or number instead of a one-hour timecode show.
  For instance, storing a timecode show for the first musical number and offset the timecode slot at 1h00m00.0, then store a second timecode show for the second number and offset the timecode slot at 2h00m00.0, and so on.
- **Duration**:
  Tap to open the calculator, then set the duration of the timecode show recording. Setting the duration to match the actual show is a good practice when using **Auto Start** and **Auto Stop** functions. Read about these functions below. Having a duration prevents the auto start and stop of shows outside the received time range.
- **Cursor**:
  Displays the cursor at its current position on the timeline. Tap to open the calculator and enter a new position.

	Hint:
	The selected timecode slot appearance defines the cursor button background color. To learn more about timecode slots, read the **What are timecode slots** topic.

Playback

This is a short description of the settings in the **Playback** section.

- **Auto Start**:
  Starts the timecode show automatically when a time signal is received within the time range of the show.
- **Auto Stop**:
  Automatically stops the timecode show when the time signal is stopped.

- **Loop Mode**:
  The loop mode defines what happens with a show using Internal (Read about the **TC Slot** setting above) time, and the end of the timecode show is reached.
  - **Loop**:
    The timecode show will repeat playing from the beginning to the end. Press the stop button in the playback bar to stop the timecode show.
  - **Pause**:
    The timecode show will pause at the end. Press the start button in the playback bar to restart the timecode show from the beginning.
  - **Off**:
    The timecode show will not loop or pause.
- **Switch Off**:
  This defines what should happen to the playbacks when the timecode show is switched Off.
  - **Playbacks Off**:
    Switches off all playbacks started by the timecode show.
  - **Keep Playbacks**:
    Do not switch off the playbacks that were started by the timecode show.
- **Assert Previous Events**:
  When set to **Yes**, events preceding the cursor will be asserted.

Display

This is a short description of the settings in the **Display** section.

- **Time Display Format**:
  This can be used to define how the time should be displayed. The following options exist:
  - **Default**:
    This follows the user profile setting called **Time Readout**. Learn more in the **User Settings topic**.
  - **10d11h23m45**:
    The time is separated into days, hours, minutes, and seconds using letters as separators.
  - **251h23m45**:
    This is separated into hours, minutes, and seconds using letters as separators. The hour number can become more than 24 if time is more than a day.
  - **10.11:23:45**:
    The time is separated into days, hours, minutes, and seconds using a dot and colons as separators.
  - **251:23:45**:
    This is separated into hours, minutes, and seconds using colons as separators. The hour number can become more than 24 if time is more than a day.
- **Frame Readout**:
  This defines how the frame read is shown. The options are:
  - **Default**:
    This follows the user profile setting called **Frame Readout**. Learn more in the **User Settings topic**.
  - **Seconds**:
    This will show the frames as fractions of a second.
  - **24 fps**, **25 fps**, **30 fps**, and **60 fps**:
    This can be used to match the displayed frames per second to the external source. This is only a readout, and it does not affect the actually received signal.

	Hint:
	If frame readouts (24, 25, 30, 60 fps) are used, fractions are separated by a colon, and fractions of seconds are separated by dots.

Other

This is a short description of the settings in the **Other** section.

- **User bits**:
  Besides the 32 Bit for 8-digit timecode time, timecode executes 32 User Bits (8-digit) per frame.
  Use User Bits to mark a timecode signal. For example, use User Bit 1 for light and User Bit 2 for pyro. This is not available when using an internal timecode source.

	Important:
	Do not change the user bits unless you are told to do so by the timecode supplier. If the user bits are changed and the timecode supplier does not know, the timecode signal will not be received anymore.

- **Lock**:
  When enabled, the timecode show can not be edited or deleted; it can only be played back or copied.

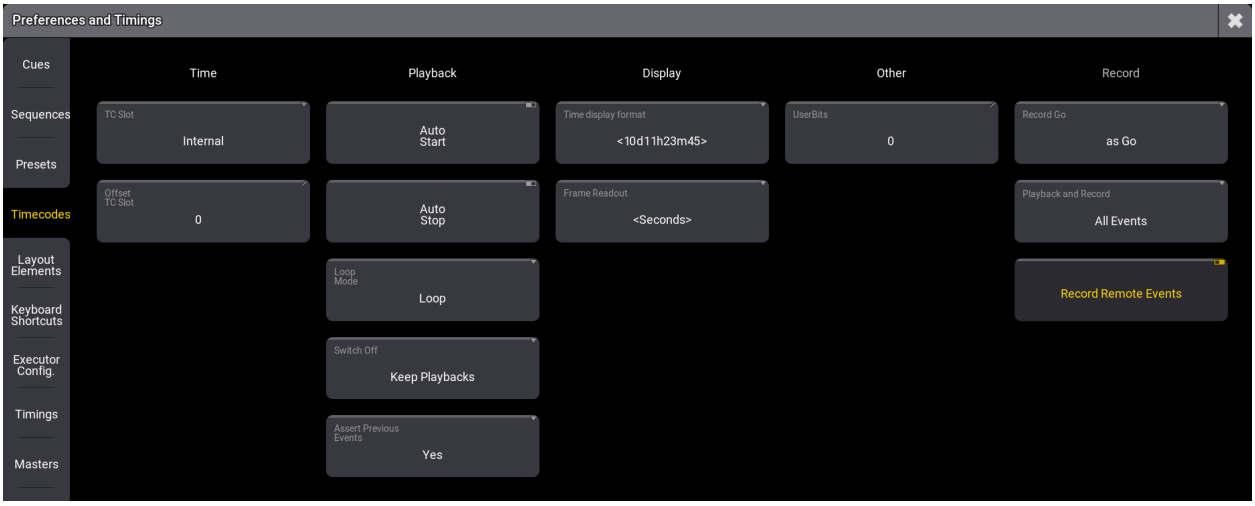
Record

This is a short description of the settings in the **Record** section.

- **Record Go**:
  This defines how Go actions are recorded when the timecode show is recording the actions. This is not relevant when manually adding events.
  - **as Go**:
    Events are recorded as **Go+** action that triggers a cue but does not assert the cue.
  - **as Goto(Status)**:
    Events are recorded as **Goto** action, which asserts the cue.
- **Playback and Record**:
  Defines which type of events will be played back and recorded.
  - **Manual Events**:
    Only actions the user triggers are recorded and played back. Follow cues, for example, are not recorded nor played back. Changes made to commands in cues after the timecode show was recorded are respected when playing back the timecode show. This is the default value for Playback and Record.
  - **All Events**:
    All events will be recorded and played back. The cells in the **Execute Command** column of the Text and Both View Mode display "---" to indicate that the execution of cue commands within the timecode show can not be overwritten. Cue commands are recorded, but changes made to commands in cues after the timecode show was recorded will not be part of the timecode show.
- **Record Remote Events**:
  This defines whether playback actions that are triggered by remote actions are recorded or not. When this setting is enabled, all remote events are recorded, regardless of the status of the **Timecode Events Recording From** setting found in the Global settings.

Timecode Settings in Preference and Timing

The **Preference and Timing** menu has many of the settings described above.



Timecodes section in the Preference and Timings menu. The menu can be opened by tapping the gear icon (⚙) on the **Control Bar** or by pressing Menu. Then, tap Preference and Timing, followed by Timecode.

The settings found here are the defaults used when a new timecode show is created.

Learn about the settings by reading **above**.

Global Settings in Preference and Timing

The menu can be opened by tapping the gear icon (⚙) on the **Control Bar** or by pressing Menu. Then, tap Preference and Timing, followed by Global.

The Control Bar can be used to access menus and functions. It is often found on the left side of the display in grandMA3 onPC system. Learn more about it in the **Control Bar topic**.

This menu only has one setting called **Timecode Events Recording From**. It has two options:

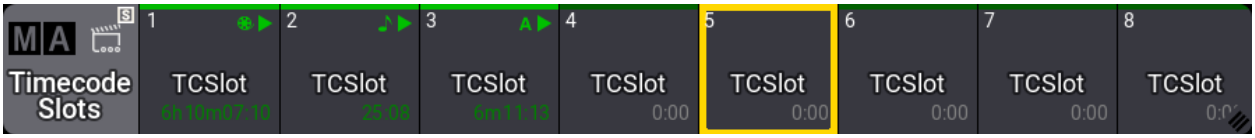
- **All Users**:
  This records the playback actions from all users in the session.
- **Single User**:
  This records only the playback actions from the user who started the timecode recording. This is the default.

The Control Bar can be used to access menus and functions. It is often found on the left side of the display in grandMA3 onPC system. Learn more about it in the Control Bar topic.

1.42.10. External Connections

External connections allow the timecode slots to receive a time source from SMPTE/LTC, MIDI timecode, and ArtTimeCode. It is also possible to transmit SMPTE/LTC and MIDI timecode from the timecode slot generator.

When a received signal is correctly configured and accepted, the timecode slots indicate the signal type with a small icon and a play icon.



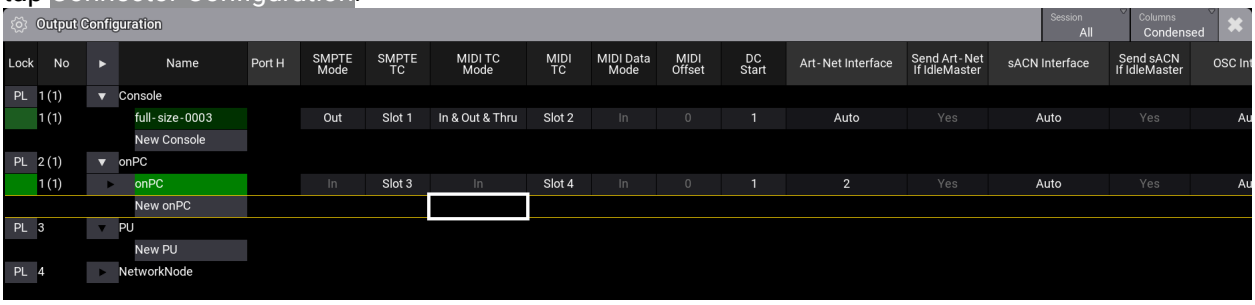
Timecode slot pool receiving three different external timecode signals. The example above shows the timecode slot pool with an SMPTE signal in slot 1, a MIDI signal in slot 2, and an ArtTimeCode signal in slot 3. The received times are displayed in green at the bottom of the pool objects.

Set Up SMPTE/LTC and MIDI Timecode

Physical connections must be made for the time signals to be received or transmitted. Learn how in the **Connect LTC** and **Connect MIDI** topics.

With connections, the next step is to access the **Output Configuration**.

The menu can be opened by tapping the gear icon ( ) on the **Control Bar** or by pressing Menu. Then, tap Connector Configuration.



Output Connection menu showing the timecode relevant columns. Four columns are relevant for the timecode in this menu.

- **SMPTE Mode**:
  This mode has two possible options that can be toggled:
  - **In**:
    The 3-pin female XLR connector marked LTC on the station is used to receive a SMTPE timecode signal.
  - **Out**:
    The 3-pin female XLR connector marked LTC on the station is used to transmit a SMPTE time signal.
- **SMPTE TC**:
  This is used to select the timecode slot to which the physical port relates.

- **MIDI TC Mode**:
  This mode has three options that can be edited and selected using the small Select MIDI TC Mode pop-up.
  - **In**:
    The 5-pin DIN connector marked MIDI In on the station receives the incoming MIDI timecode signal.
  - **Out**:
    The 5-pin DIN connector marked MIDI Out on the station transmits the outgoing MIDI timecode signal.
  - **In & Out & Thru**:
    The MIDI In connector can receive a MIDI timecode signal, which is then output on the MIDI Out port. The system can also generate a MIDI timecode signal transmitted from the MIDI Out port.
- **MIDI TC**:
  This is used to select the timecode slot to which the physical port relates.

Several stations can receive a signal related to the same Timecode Slot. The device receiving the signal first is the one the slot listens to. When multiple stations are in Out mode, the master transmits the time signal.

The **Timecode Slot Settings** display the station's **Source IP** address that provides the time signal for the slot.

The text "No Cable" in the Source IP button can be an indicator that:

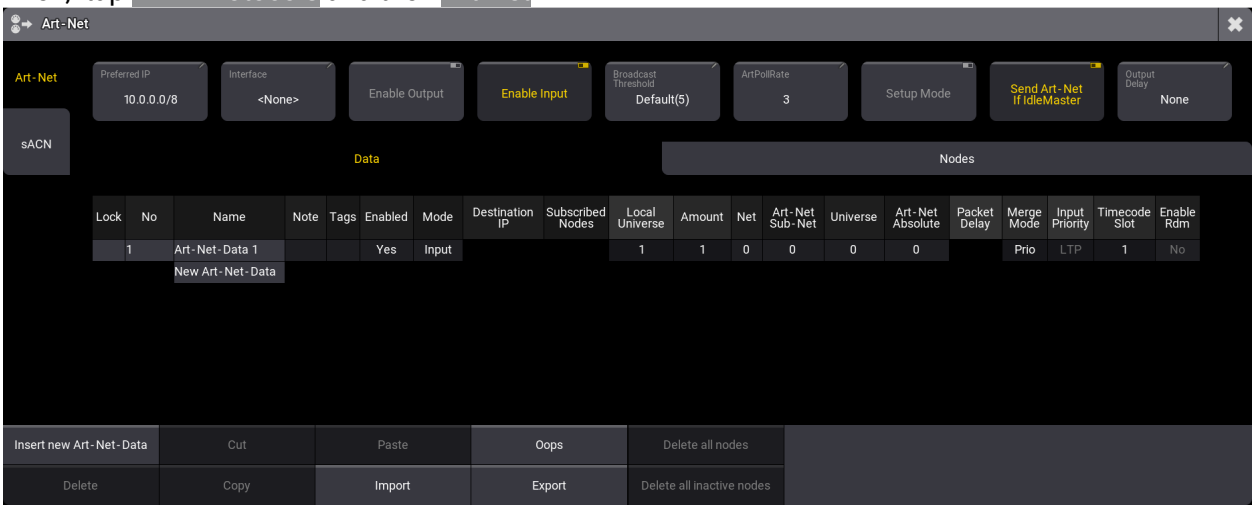
- Indicates that no network cable is connected to the selected MA-Net-Interface.
- No timecode show is currently running on the selected TCSlot.
- The time is running on another timecode slot.

Set Up ArtTimeCode

ArtNet can be used to transmit timecode using ArtTimeCode.

This can be set up in the DMX Protocols menu.

The menu can be opened by tapping the gear icon () on the **Control Bar** or by pressing Menu.
Then, tap DMX Protocols and then Art-Net.



The Art-Net menu with timecode columns. The following elements must be set up to receive ArtTimeCode:

1. The correct Interface must be selected to match the interface that connects to the source.
2. The Enable Input must be active.
3. There must be a row with Art-Net-Data.
4. The row must be **Enabled,** and the **Mode** must be **Input**.
5. The Timecode Slot must be set to the correct slot number.

When all these elements are set up, the ArtTimeCode received will set the time for the related slot.

ArtTimeCode cannot be generated by the grandMA3 system.

The Control Bar can be used to access menus and functions. It is often found on the left side of the display in grandMA3 onPC systems. Learn more about it in the **Control Bar topic**.

The Control Bar can be used to access menus and functions. It is often found on the left side of the display in grandMA3 onPC systems. Learn more about it in the **Control Bar topic**.

1.43. Layouts

Layouts are two-dimensional drafts where it is possible to arrange fixtures, macros, groups, and other pool objects.

- Layouts are created in the Layouts pool.
- Layouts are displayed and edited in the Layout Viewer.

	Restriction:
	The maximum number of elements per layout is 10 000.

To set the defaults for new layout elements, press Menu, tap Preferences and Timings, then tap Layout Elements on the left side of the window. For more information, see **Layout Element Defaults**.

Subtopics

- **Create a Layout**
- **Assign Multipatch Fixture**
- **Edit Layout**
- **Layout View Settings**
- **Edit Layout View**
- **Edit Layout Elements**
- **Layout Encoder Bar**

1.43.1. Create a Layout

Layouts are created in the Layout pool and can be edited in the Layout window or the Layout editor.

Arrange the layout pool and the layout window on one screen to operate the layouts faster and to have them all at a glance. See

dd fixtures to a layout

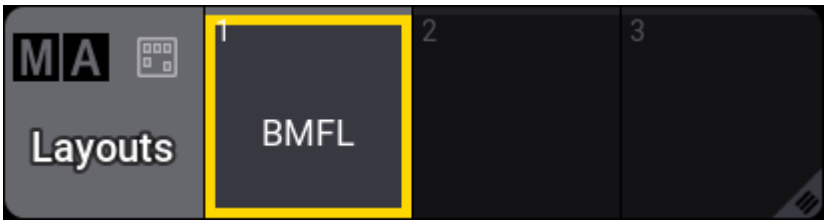
Requirement:

1. Open the Add Window dialog. Tap Data Pools and then Layouts. The Layouts window opens.
2. Select the fixtures you want to assign to the layout.

	Hint:
	The selection grid position of the selected fixtures is assigned to layouts. For more information, see **Selection Grid**.

Fixture Sheet, Layouts Pool, and Groups Pool

3. To add the selected fixtures to a layout, press Assign and tap a cell in the layout pool, or tap the area in the layout window where you would like to assign them. The fixtures are then added to the layout.



Layout pool window

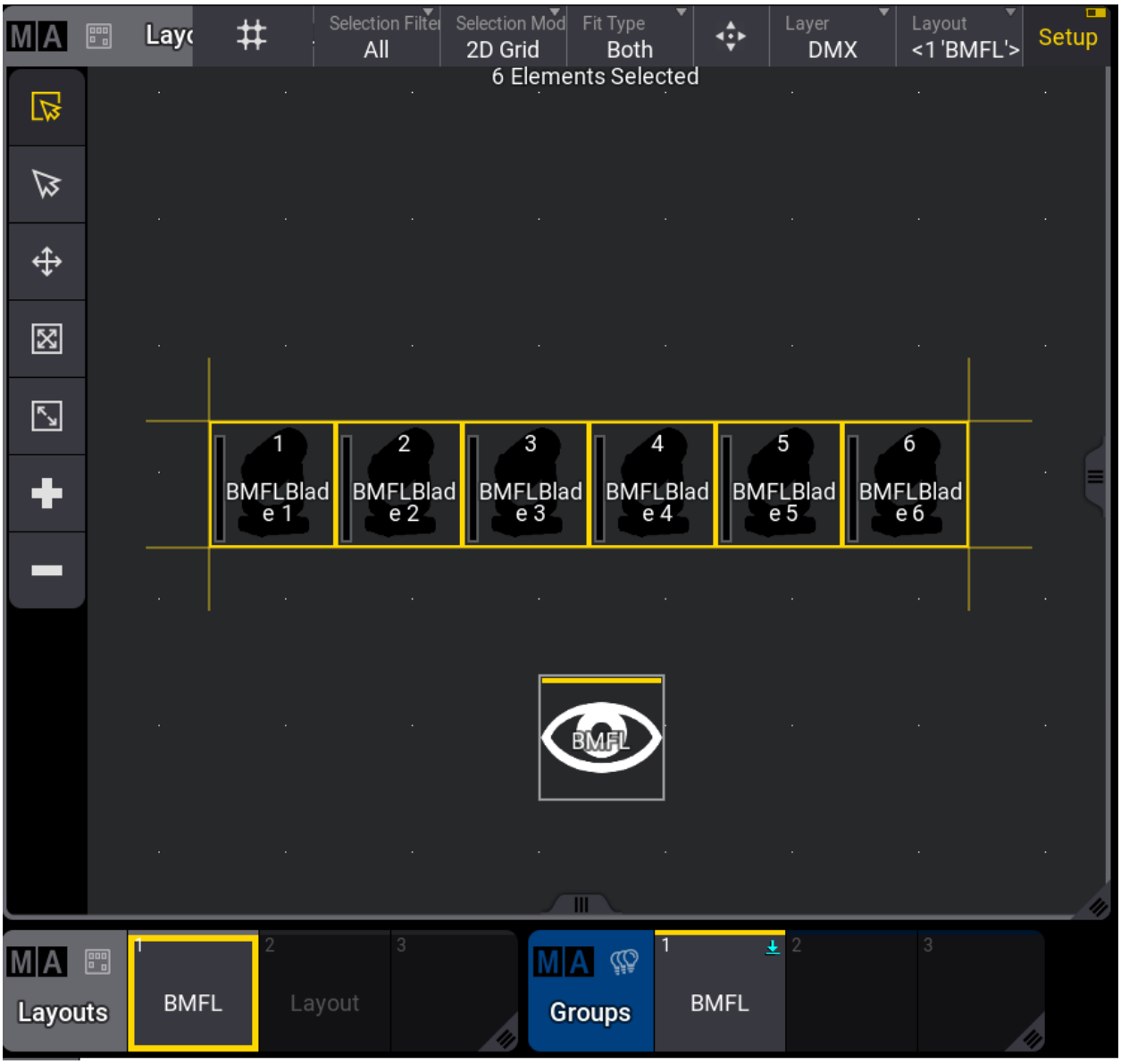
	Hint:
	Additionally, select sub-fixtures to assign them to a layout.

Pool Elements

To add pool elements to the layout window, press Assign Group 3 and then tap the area of the layout window where you would like to assign this group. The pool element is added to the layout window.

To set up defaults for various layout elements, see **Layout Element Defaults**.

	Hint:
	Arrange the layout pool and the layout window on one screen to operate the layouts faster and to have them all at a glance.



Layout view and layout pool

Layout View Toolbar

Tap Setup in the title bar to enable the toolbar on the left side of the layout view.

	Hint:
	When Setup is enabled, the layout encoder bar is also enabled. See, **Layout encoder bar**.

Select an Element

1. To select an element, tap .
2. Tap the element you wish to select or use the lasso command to select multiple elements.
3. The element is selected.

Add an Element

The upper left corner of a layout element is set to the insertion point.

1. To add an element in the layout window, tap  .
2. Tap an empty space in the layout window.
3. The element is added.
4. To add an object to the element, see **Edit layout view**.

Delete an Element

1. To delete an element in the layout window, tap .
2. Tap the element you want to delete.
3. The element is deleted.

Move an Element

1. To move an element within the layout window, tap  .
2. Tap, hold and drag the element to move it.
   The element is moved.

Resize an Element

1. To resize an element, tap  .
2. Select the element you would like to resize.
3. Tap, hold, and drag to resize.
   The element is resized.

Resize an Element with Fixed Ratio

1. To resize an element and preserve the actual size ratio, tap  .
2. Select the element you would like to resize.
3. Tap, hold, and drag to resize.
   The element is resized with a fixed ratio.

Auto Tool

1. To use the auto tool, tap  .
2. Tap the element you wish to select or use the lasso command to select multiple elements. A yellow responsive frame appears around the selected items.

The following functions can be used depending on where the layout element is tapped:

- Tap, hold, and drag inside a layout element to move it.
- Tap, hold and drag one of the outer edges to change the size of the layout element for that particular edge.
- Tap, hold, and tap one of the outside corners to resize that layout element without taking proportions into account. Pressing MA additionally takes proportions into account.

Layout View Fit Options

In the title bar, tap Fit Type and select the fit option you wish to perform.

- Select Elements to fit all layout elements in the window.
- Select Canvas to fit the entire canvas in the window. If some elements are assigned far outside the canvas, they will not be displayed.
- Selecting Both will fit the canvas and all elements.
- Tap to execute "Zoom to Fit" depending on the selected Fit Type.

	Hint:
	The canvas fit mode can be modified. For more information, see **Layout View Settings**.

- Create a layout with the command line using the layout keyword. For more information, see **Layout keyword** and **Assign keyword**.

```
User name[Fixture]>Assign Layout 1
```

Cloning in Layouts

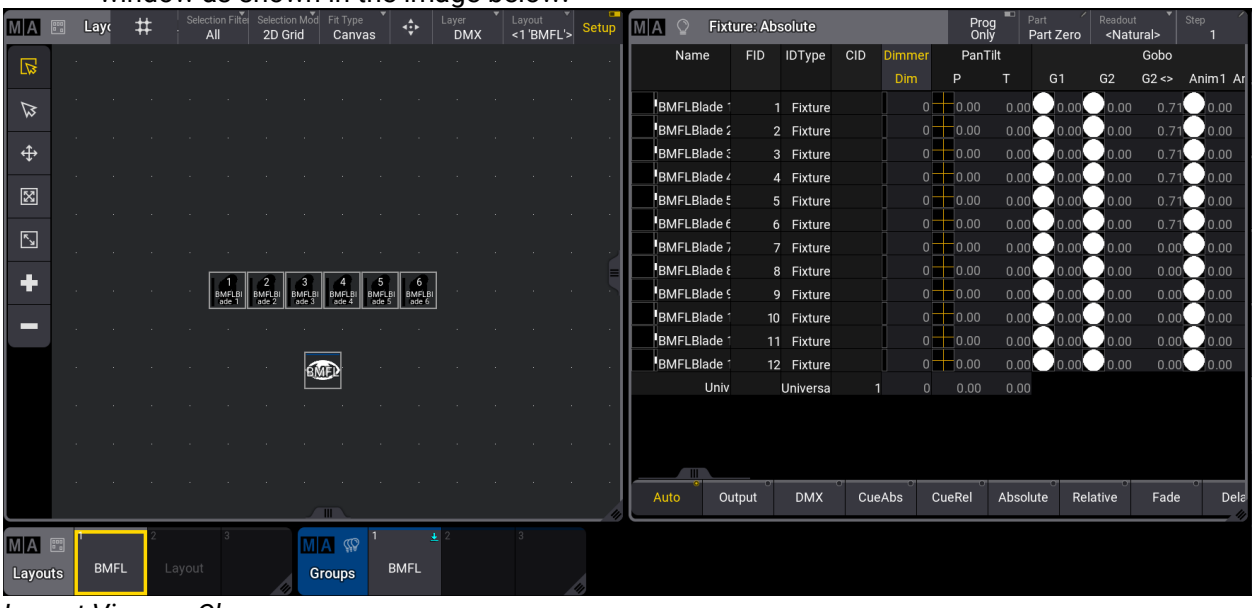
It is possible to clone fixtures into a layout using a clone syntax. For more information about cloning, see **Clone keyword** and **Clone** topic.

Requirements:

- Some patched fixtures. For more information, see **Patch and Fixture Setup**.
- Layout Viewer window, Fixture Sheet and Layout Pool are open.

To clone fixtures into a layout, follow the steps in the example below:

1. Tap the BMFL pool in the Layout Pool window. The BMFL layout is shown in the Layout Viewer window as shown in the image below:

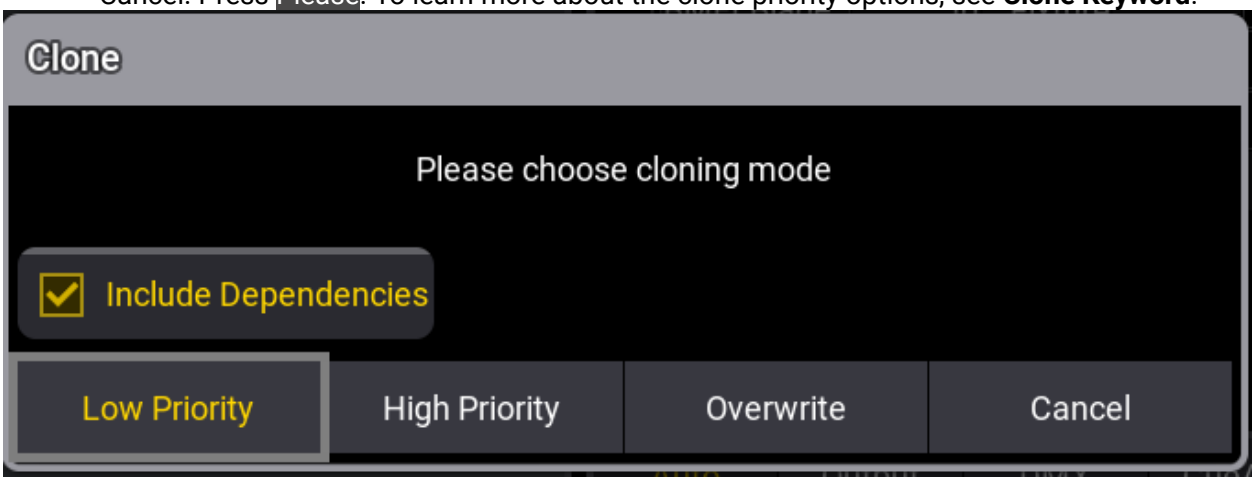


Layout Viewer - Clone

2. To clone new fixtures into the first layout pool, type the following command into the command line:

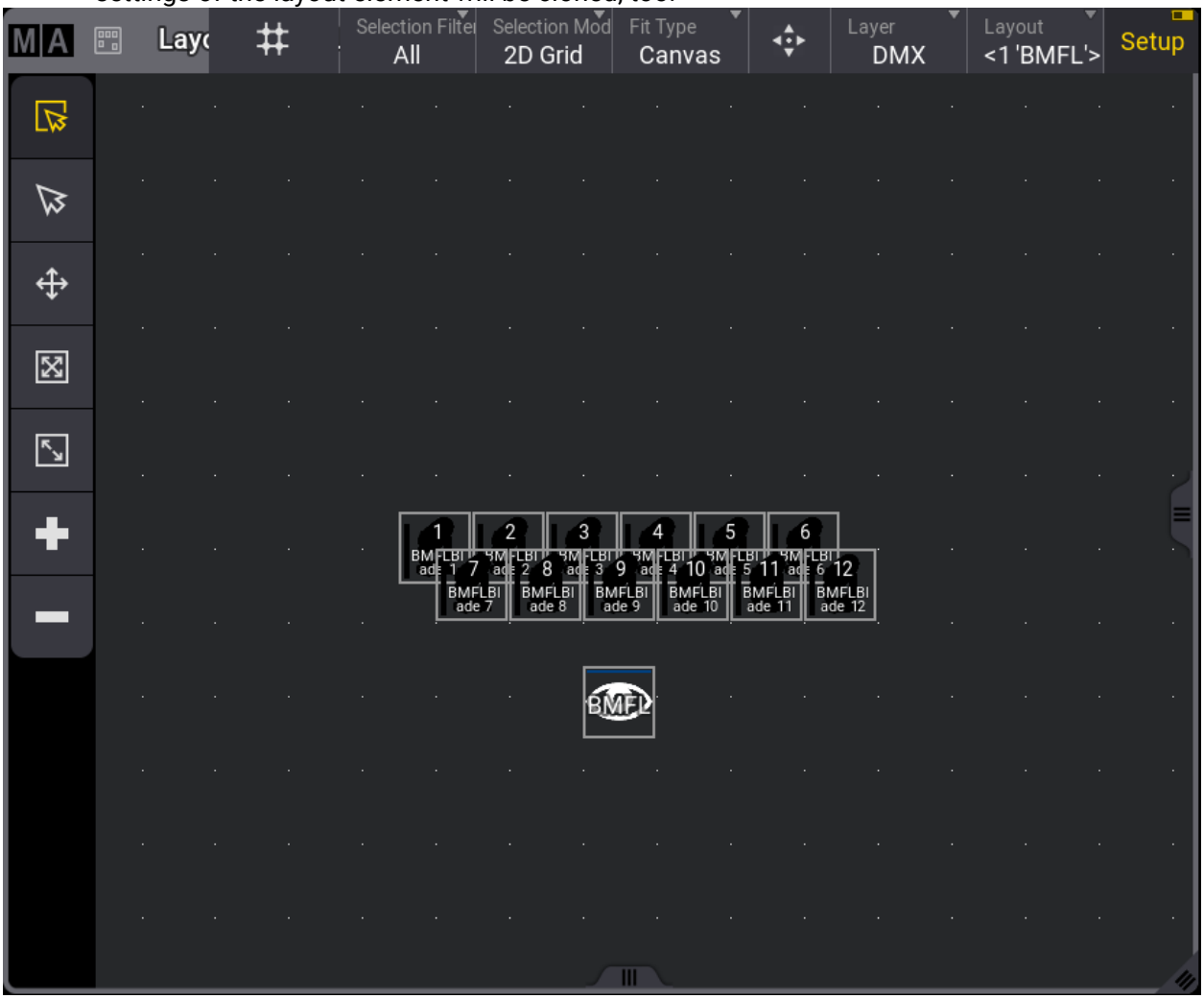
```
User name[Fixture]>Clone Fixture 1 Thru 6 At Fixture 7 Thru 12 If Layout 1
```

3. The Clone pop-up appears, offering the choices of Low Priority, High Priority, Overwrite, and Cancel. Press Please. To learn more about the clone priority options, see **Clone Keyword**.



Clone pop-up

4. The new layout elements will be placed with a small offset to the existing layout elements. All settings of the layout element will be cloned, too:



Fixtures are placed with an offset in the Layout Viewer window

1.43.2. Assign Multipatch Fixture

Requirement:

Patched multipatch fixtures. See **Add Multipatch Fixtures**.

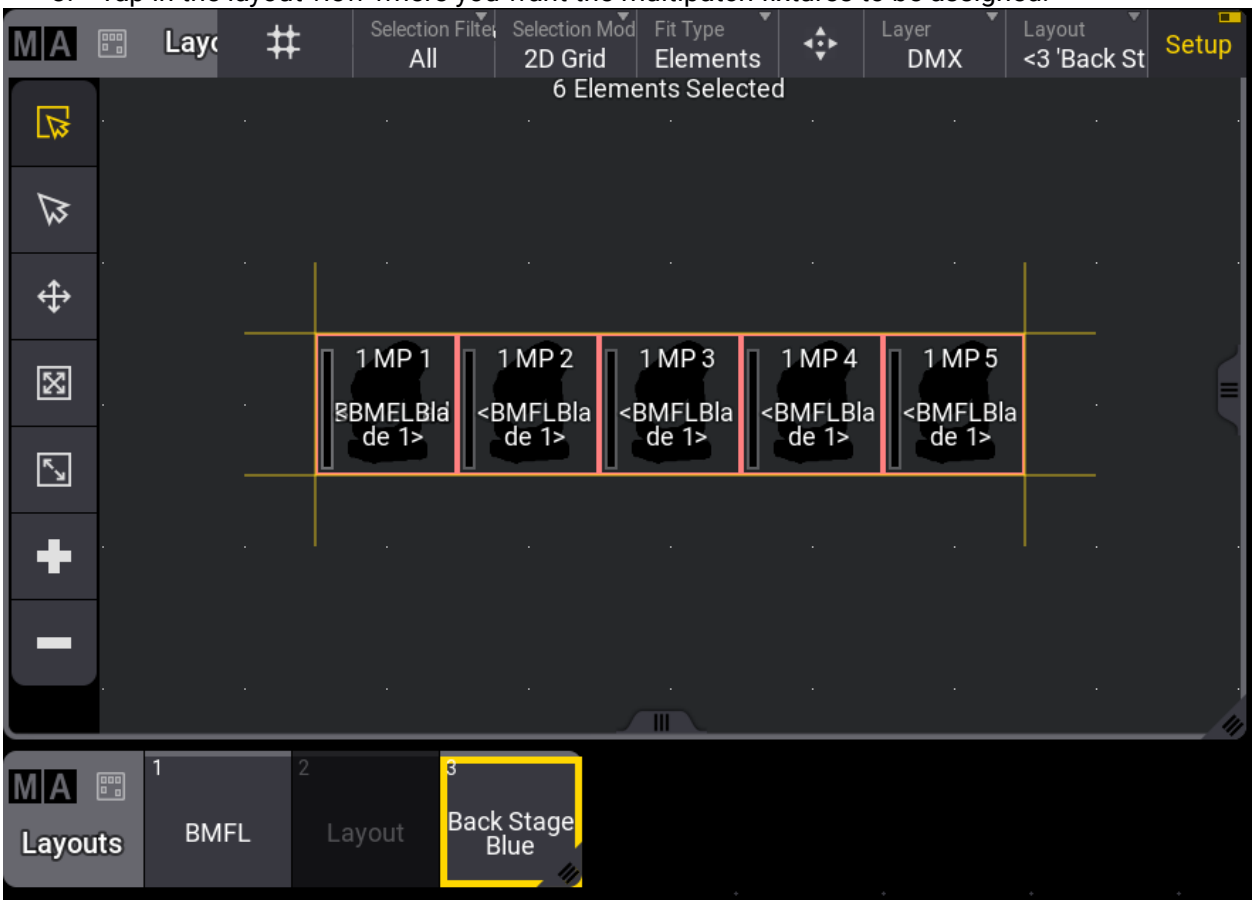
Multipatch fixtures can be selected via the command line by typing their CID number or global position order in the patch. See **Add Multipatch Fixtures**.

The following is an example of how to assign primary fixture 1 and its multipatch fixtures to a layout. Before doing so, make sure to read **Create a Layout**.

To assign fixture 1 multipatch fixtures 1 thru 5:

1. Select the new layout pool object you have created.
2. Write the following syntax in the command line.

   User name[Fixture]>Assign Fixture 1 Multipatch 1 Thru 5
3. Tap in the layout view where you want the multipatch fixtures to be assigned.



Layout 3

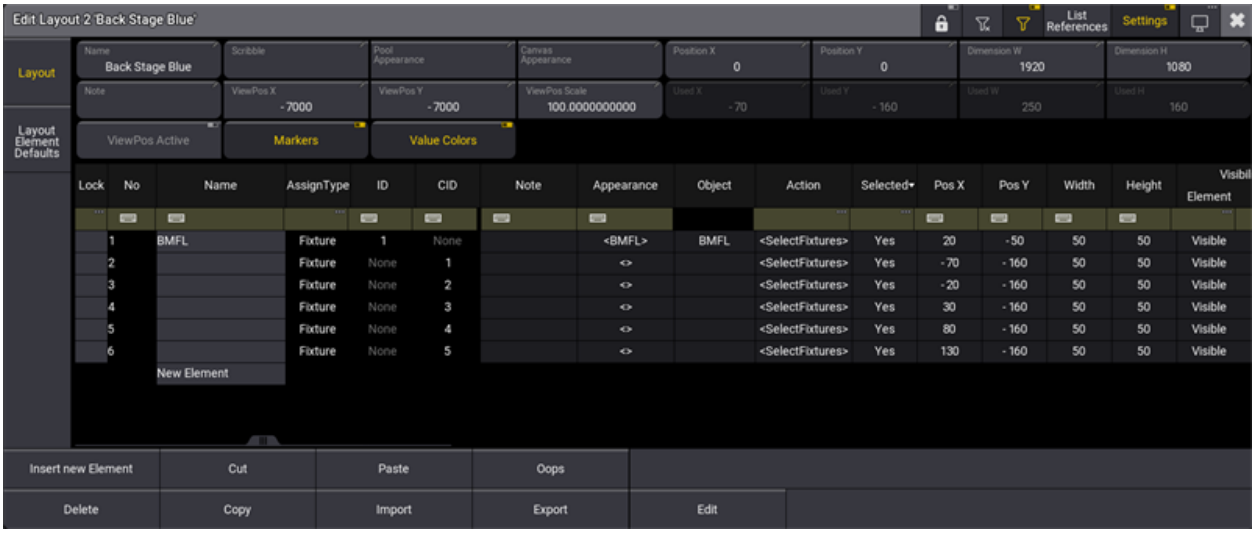
The selection color is light red instead of yellow (the default color). When a multipatch fixture is selected, the primary ID will flash between yellow and light red in the fixture sheet.

1.43.3. Edit Layout

Requirements:
A created layout. For more information, see **Create a Layout**.

- To edit a layout, use the swipey command or press Edit, then tap the layout pool object.

The Edit Layout overlay opens:



Edit layout window

The following is a description of the buttons in the Edit Layout window:

- To rename the layout, tap Name, enter the new name, tap Enter on the virtual keyboard, or press Please. For more information, see **Label pool objects**.
- To add a scribble to the layout pool object, tap Scribble. For more information, see **Scribbles**.
- To add an appearance to the layout pool object, tap Pool Appearance. For more information, see **Appearances**.
- To add a canvas appearance to the layout, tap Canvas Appearance. This will apply to the layout window background. For more information, see **Appearances**.

	Hint:
	If no dedicated canvas appearance is set, the pool appearance will be taken as canvas appearance.

Tap the following buttons to set the position and size of the canvas:

- To set the start X position, tap PositionX.
- To set the start Y position, tap PositionY.
- To set the width dimension, tap DimensionW.
- To set the horizontal dimension, tap DimensionH.

	Hint:
	This feature allows you to store two layouts with the same fixtures but with different positions and zoom values.

The following buttons can be toggled:

- When ViewPosActive is switched on, loading a layout again will recall the previously stored view zoom and position values. This button is switched off by default.
- The layout property Markers allows to display the layer markers of fixtures at their corresponding layout elements when it is enabled.
- Enabling Value Colors in the layout settings displays the dimmer values of fixtures in their corresponding layout elements in the same colors as in the fixture sheet. When the background of the dimmer attribute cell in the fixture sheet is colored, the shadow of the value in the layout element gets colored equally.

	Hint:
	To see the same colors for values in a layout element as in the fixture sheet, the layout element needs to have a fixture with a dimmer assigned. The visibility fo the value needs to be enabled for the layout element, too.

ViewPosX, ViewPosY, and ViewPosScale are the stored values for the layout view position and zoom.

Layout Element Defaults

In the Layout Elements defaults tab, it is possible to change templates for layout elements such as groups, worlds, or sequences.

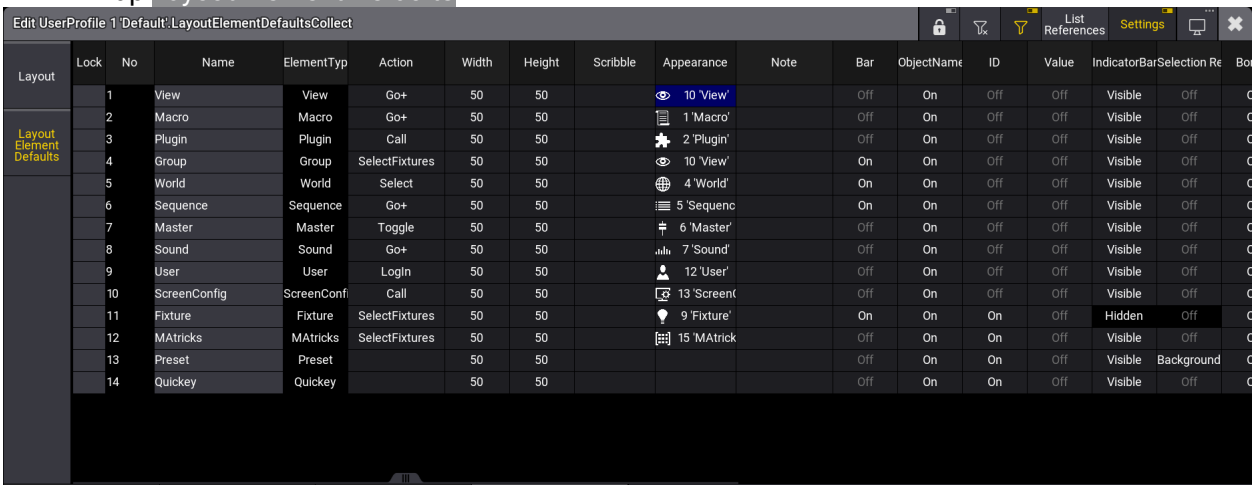
Layout element defaults are part of the user profile and will apply as a default template when assigning an object to a layout.

To use a custom appearance for layout elements, two finger tap in the Appeareance column and select a new one.

1. To set the default values for layout elements, press Menu, then tap Preferences and Timings. On the left-hand side, tap Layout Elements

or

1. Press Edit and then tap a layout pool object.
2. Tap Layout Element Defaults.



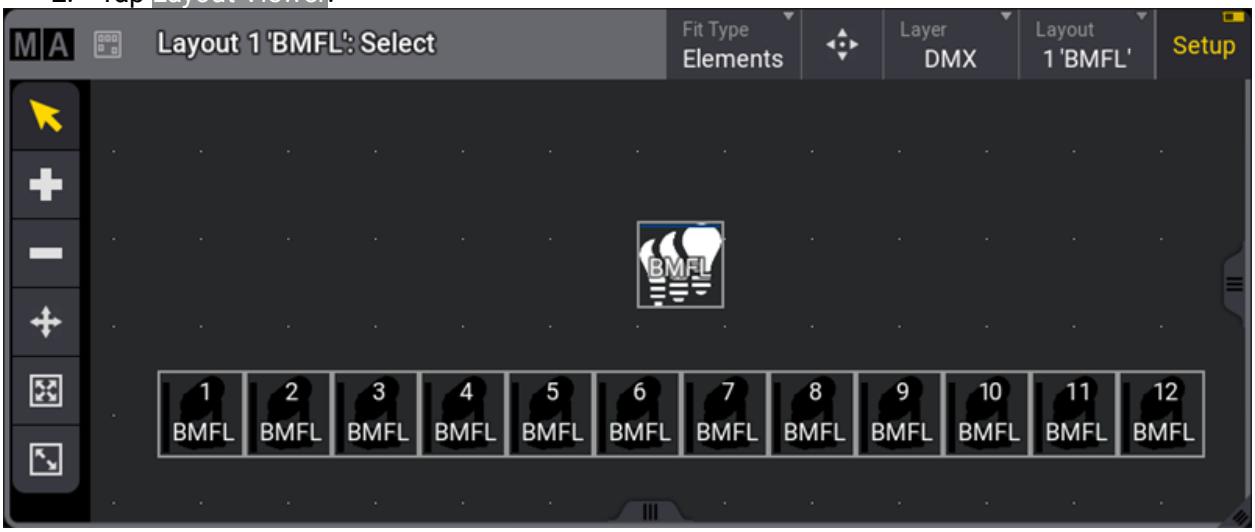
Layout Element Defaults

1.43.4. Layout View Settings

To open the Layout View window, follow the instructions under **Add windows**.

In the Add Window pop-up:

1. Tap Tools.
2. Tap Layout Viewer.



Layout View window

Open Layout View Settings

- To open the Layout View Settings, tap MA in the upper left corner of the layout window.

The Layout View Settings pop-up opens.

Layout View Settings pop-up

This is a description of each button in the layout view settings.

Display Tab

- **Layout**:
  Link selected, select an existing layout or create a new one.

- **Setup:**

- **Layer**:
  Select the layer readout between **DMX**, **Value**, or **Output**.

- **Lasso Filter:**
  Specifies which type of layout element will be selected when a lasso selection is made.
  - **All**: All layout elements will be selected.
  - **Fixtures**: Only fixtures will be selected.
  - **Others**: Everything but fixtures will be selected.

- **Show Selection:**
  Displays the selected fixtures by a yellow frame in the layout view or not. Show Selection is not available in the Setup mode.

- **Right Click To Edit**:
  When this button is toggled on, it will allow editing of objects assigned to layout elements by right-clicking or using the two-finger edit gesture.
  Toggle this button off to disable the right-click and two-finger edit gesture to prevent accidental editing of objects.

	Hint:

	To edit an object when this button is toggled off, press Edit and tap the assigned object you wish to edit.

- **Lock Position**:
  Toggle on to protect the layout arrangement.

- **Selection Mode:**
  This defines how the fixtures are selected and positioned in the **Selection Grid**. The selection mode has two different options:
  - **2D Grid**: Using the lasso selection adds them as a two-dimensional selection to the Selection Grid.
  - **Linearize:** The selection is linearized to only the X-axis of the Selection Grid.

- **Auto Fit**:
  Toggle on to fit elements according to the window size.

- **Canvas Fit Mode**:
  There are 3 fit modes, **Bar** will fit the entire canvas, **Crop** fits the canvas to its total width, and **Stretch** fits the canvas to its total height.

- **Fit Type**:
  Toggle to fit the **Elements**, the **Canvas**, or **Both**.

- **Snap To Grid**:
  Toggle on for objects to snap to the grid when in setup mode.

- **Snap Grid**:
  Tap to open the calculator, then enter a snap value between 1 and 14 000 pixels.
  Make this number smaller to be as precise as possible when moving elements in the layout view.

- **Visible Grid**:
  Tap to open the calculator, then enter a value for the visible grid size.

- **Grid Color**:
  Use the edit grid color popup to create the desired grid color.

- **Grid Style**:
  Select the grid style between **Off**, **Lines**, or **Dots**.

- **Scale**:
  When Auto Fit is off, the fader can scale the layout.

- **Show Title Bar:**

Mask Tab

- **Filter**:

Layout View Title Bar

- #:
  Transfers the currently selected arrangement of fixture positions in the layout to the Selection Grid.
- **Zoom to fit**:
  Tap to zoom out to see the entire canvas.

	**Hint:**
	If no values are set for Dimension H and Dimension W, Zoom to fit is unavailable.

1.43.5. Edit Layout View

To open the layout view, follow the instructions under **Add windows**.

In the Add Window pop-up:

1. Tap Tools.
2. Tap Layout Viewer.
   The layout view opens.

	Important:
	When Setup is enabled, it allows the editing of single layout elements. When disabled, it allows editing as in the programmer.

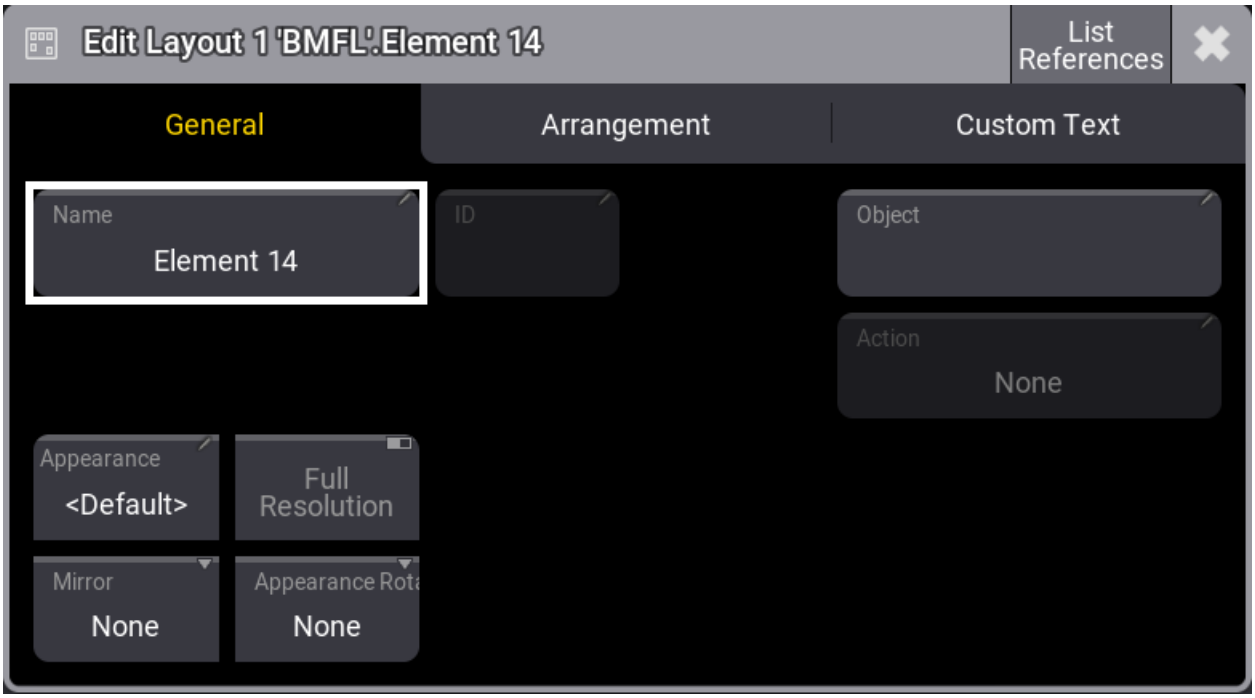
Add a Layout Element

In the layout viewer window title bar, tap Setup to enable the setup mode.

- To add a layout element in the layout window, tap +.
- Tap an empty area in the window, and a new empty layout element is added. (Empty yellow square)

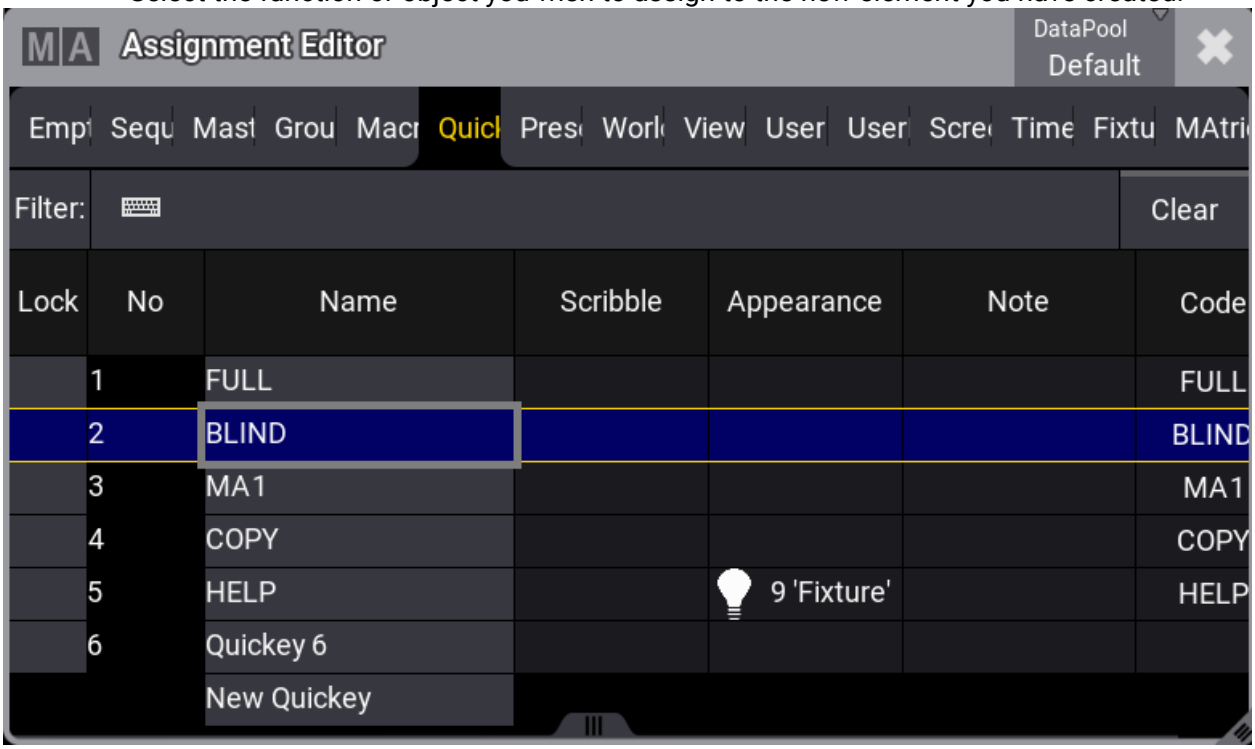
Layout View

- To assign a function/object to the element you have created in the layout, tap in the toolbar and open the new element with the **2-finger edit** or press Edit and tap the new element.
  The edit layout element pop-up opens.



Edit Layout pop-up

- Under the General tab, tap Object, and the assignment editor opens.
- Select the function or object you wish to assign to the new element you have created.



Assignment Editor pop-up

Add an Element Using the GUI

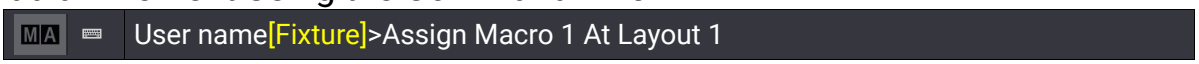
The group pool and the Layout window should be visible on the screen for this.

For example, add group 1 to layout 1:

1. Press Assign.
2. Tap group 1 in the group pool.
3. Tap an empty area in the layout window.

Hint:
This can be done in setup mode and normal mode.

Add an Element Using the Command Line



Layout Toolbar

Use the toolbar to edit elements. For more information, see **Create a layout**.

1.43.6. Edit Layout Elements

To edit a layout element:

1. Tap Setup to activate the setup mode in the Layout View window.
2. Tap and open the element with the **2-finger edit**.

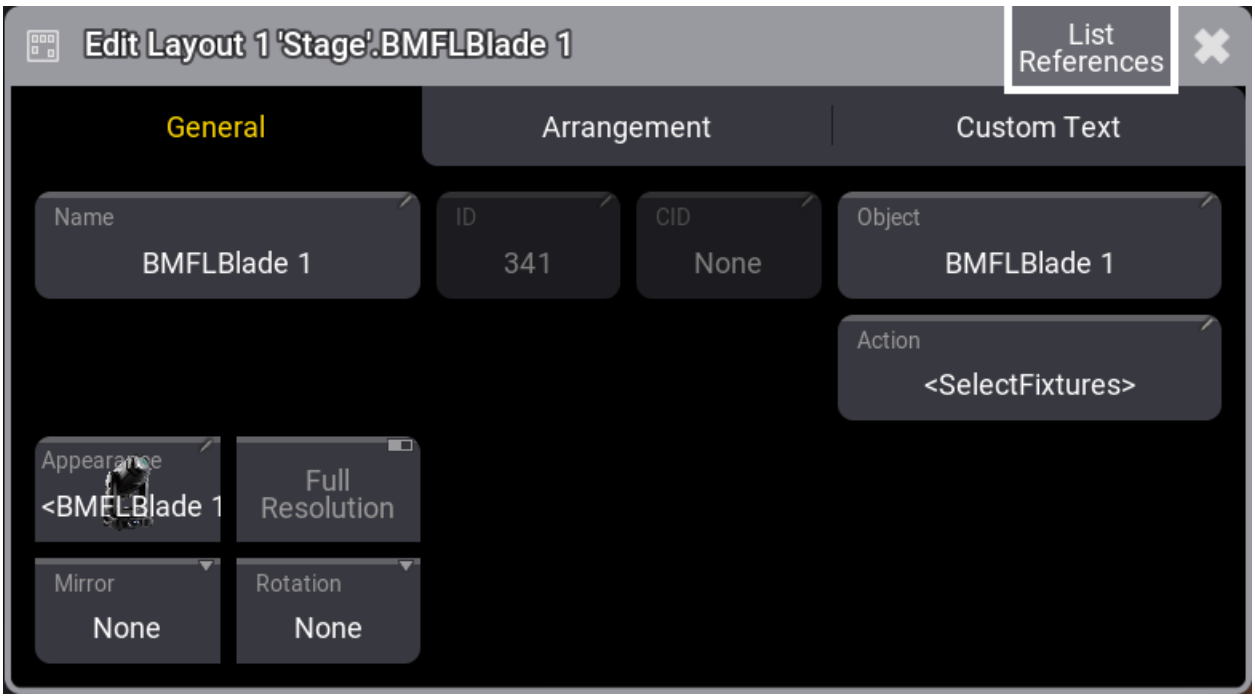
-Or-

1. Tap Setup to activate the Layout Encoder Bar.
2. Select the layout elements to edit in the Layout View window.
3. Tap Edit Selected in the Layout Encoder Bar to open the layout element editor.

The Edit Layout window is separated into three tabs. General, Arrangement, and Custom Text.

General Tab

Manages all properties regarding the object assigned to the layout element.



General tab

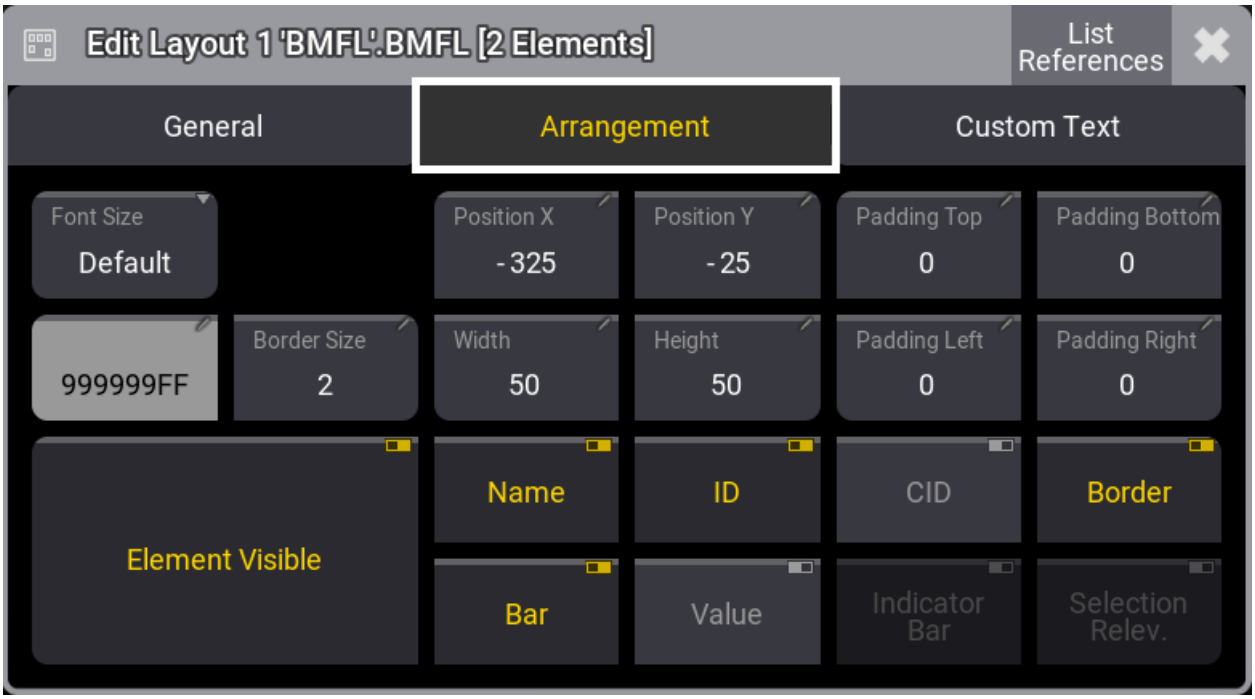
- **Name:** Opens the virtual keyboard to enter a name for the selected element.
- **ID:** When a fixture is assigned to a layout, the fixture ID is displayed here. For more information, see **What are fixtures**. When a pool object is assigned, ID represents the pool object number.
- **CID:** If the **IDType** is something different than "Fixture", the CID is displayed here. For more information, see **Add fixtures to the show**.
- **Object**: Opens the **Assignment Editor** pop-up.

- **Action**: Opens the **Select Action** pop-up. Select an action to be performed when tapping the element. Selecting <Default>, will set the default action for the assigned object. For example, if the assigned object is a fixture, the default action will be **<SelectFixture>**.
- **Appearance:** Opens the **Select Appearance** pop-up. <Default> will use the appearance of the assigned object. For more information, see **Appearances**.
- **Full Resolution:** Defines if an image is displayed with its full resolution (enabled) or reduced quality (disabled).
- **Mirror:** Mirrors the displayed appearance on a layout element horizontally, vertically, or equally.
- **Appearance Rotation:** Rotates the displayed appearance to 90°, 180°, or 270° degrees.

	Important:
	Having a lot of high-resolution images displayed may slow down the graphical user interface. In this case, the software falls back, rendering images with a lower quality. Images with 150 pixels or below will always be displayed with a lower quality.

Arrangement Tab

Manages the position and look of a layout element.

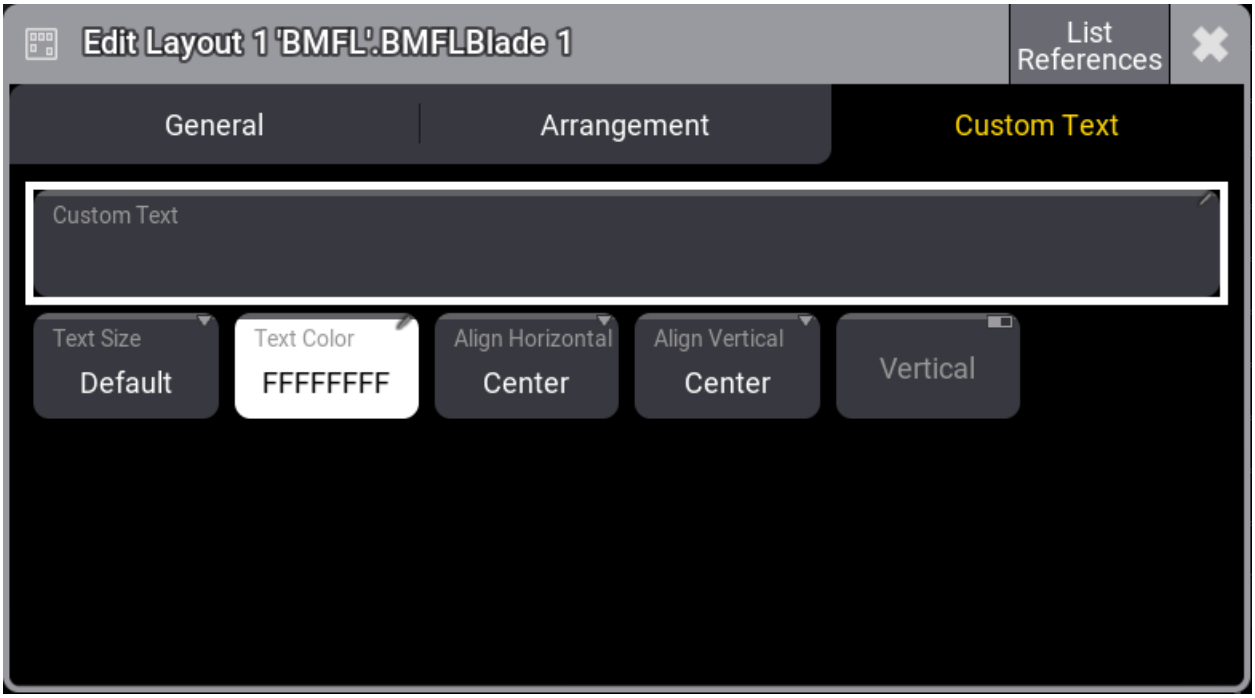


Arrangement tab

- **Font Size**: Use these buttons to personalize the custom text alignment. Setting layout elements to a dedicated font size displays the text in the set size.
- **Position X** and **Position Y**: Specify the position along X and Y in the layout.
- **Padding Top, Padding Bottom, Padding Left, Padding Right**: Padding is the distance between the edge of the element and the content inside the element.
- **Border Color** and **Border Size**: Adjust the outline color and size.
- **Width** and **Height**: This is to adjust the dimensions of the element.
- **Element Visible**: Toggle to display or hide the element.

- **Name**, **ID**, **CID**, **Boarder**, **Bar,** and **Value**: Toggle to display or hide the name, the fixture ID, the custom ID, the dimmer bar, the output value, and the border of the layout element.
- **Indicator Bar**: When a pool object is assigned to a layout (e.g., a group), it defines if the indicator bar will be displayed on top of the element.
- **Selection Relevance**: When enabled, the image of the layout element will be colored with the For All or For Some color. For more information, see **Window settings, Pool Settings**.

Custom Text Tab



Custom Text tab

- **Custom Text:** Add a custom label to the element. Text on layout elements that is too long to display is cut off.
- **Text Size** and **Text Color**: Use these buttons to personalize the custom text size and color.
- **Align Horizontal**: The horizontal custom text alignment.
    - **Center**: Will align the text to the horizontal center of the layout element.
    - **Left**: The left end of the text will be aligned with the left edge of the layout element.
    - **Right**: The right end of the text will be aligned with the right edge of the layout element.
    - **OutsideLeft**: Places the custom text outside of the layout element on the left.
    - **OutsideRight**: Places the custom text outside of the layout element on the right.
- **Align Vertical:** The vertical custom text alignment.
    - **Center**: Will align the text to the vertical center of the layout element.
    - **Top**: Places the text within the layout element at the top edge.
    - **Bottom**: Places the text within the layout element at the bottom edge.

- **Above**: Places the text outside the layout element at the top edge.
- **Below**: Places the text outside of the layout element below the layout element.

- **Vertical**: Displays the custom text in vertical direction when enabled.

	**Hint:**
	Multiple selected layout elements can be edited simultaneously.
	**Hint:**
	The layout element selection order will determine the alignment order.
	**Hint:**
	When toggling layout element properties while having multiple layout elements selected with different property values, they will switch to the same value and change their value together.

Example

To edit multiple selected layout elements simultaneously:

1. Tap Setup in the title bar to enable **Setup**.
2. Select multiple fixtures in the Layout Viewer window.
3. Tap Edit Selected in the Layout Encoder Bar to open the layout element editor. For more information, see **Layout encoder bar**.
4. Tap the Name button; this opens the virtual keyboard. Enter a new name and tap Enter.
5. Close the Edit Layout dialogue.

All selected fixtures are renamed simultaneously

1.43.7. Layout Encoder Bar

Use the layout encoder bar to position and arrange layout elements.

Requirements:

- A visible layout pool and a visible layout window.

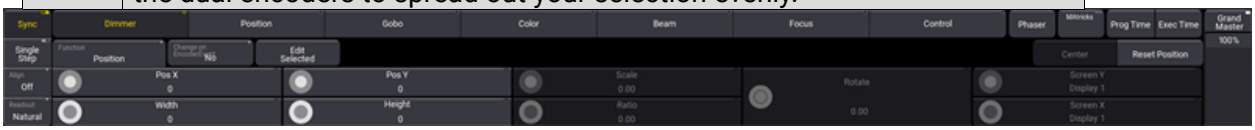
For more information on how to add a window, see **Add windows**.

There are two function settings for the layout encoder bar: **Position** and **Arrangement**.

Layout Encoder Bar Function Set to Position

1. Create a layout and select it. See **Create a layout**.
2. Tap Setup in the title bar to enable **Setup**; the layout encoder bar is displayed.
3. Tap Function in the layout encoder bar until it is set to **Position**.
4. Select the layout element (e.g., fixture) you wish to position.
5. Use the encoders to position and/or resize the element.

	Hint:
	When multiple elements are selected, use the **Align** function combined with the dual encoders to spread out your selection evenly.



Layout Encoder Bar

- **Pos X**: Rotate the inner ring of the dual encoder to move the selected element left and right.
- **Pos Y**: Rotate the inner ring of the dual encoder to move the selected element up and down.
- **Width**: Rotate the outer ring of the dual encoder to set the width of the selected element.
- **Height**: Rotate the outer ring of the dual encoder to set the height of the selected element.
- **Scale:** Scales the arrangement of the selected elements in the X and Y direction.
- **Ratio:** Scales the arrangement of the selected elements in the X direction.
- **Rotate:** Rotates the selected fixtures on the canvas and changes their position as a group. When only one fixture is selected, the encoder is grayed out.
- **Change on EncoderEvent**: When set to **Yes**, encoder changes are instantly transmitted to the network. If set to **No**, after stopping to turn an encoder the actual state will be transmitted after 2s.

- **Reset Position**: Tap to position the bottom left corner of the selected elements to Pos X and Pos Y zero.

- **Edit Selected:** Tap to enter the layout element editor.

Layout Encoder Bar Function Set to Arrangement

1. Create a layout and select it.
2. Tap Mode in the title bar to enable **Setup**; the layout encoder bar is displayed.
3. Tap Function in the layout encoder bar until it is set to **Arrangement**.
4. Select the layout elements (e.g., fixtures) you wish to position.
5. Tap and hold LayoutType, the Select Layout Type drop-down menu opens.
6. Select one of the four arrangement types: **Line**, **Grid**, **Circle**, or **Camera**.

For more information on **Line**, **Grid**, and **Circle** arrangements, see **Position fixtures in the 3D space**.

Camera Arrangement

Arrange your current selection similar to your camera view in the 3D window. Use the encoders to select a camera, set the scale and ratio factor, and move the selected elements.

Layout encoder bar function set to Arrangement

To do so:

1. Tap Function until it is set to **Arrangement**.
2. Tap Layout Type until it is set to **Camera**.

Set the following parameters using the dual encoders:

- **Camera**: Select the desired camera.
- **Scale**: Scales the whole selection in X and Y direction.
- **Ratio**: Defines the ratio of the selection.
- **Move X**: Move the whole selection in the X direction for a more convenient position.
- **Move Y**: Move the whole selection in the Y direction for a more convenient position.

	Hint:
	Press the first dual encoder inner ring or the first dual encoder key to open the camera calculator. Select a camera from the list or enter a camera number.

3. When you're satisfied with your layout setup, tap Apply to confirm.

	Hint:
	When Apply on change is active, Apply is disabled, and changes are made in real-time.

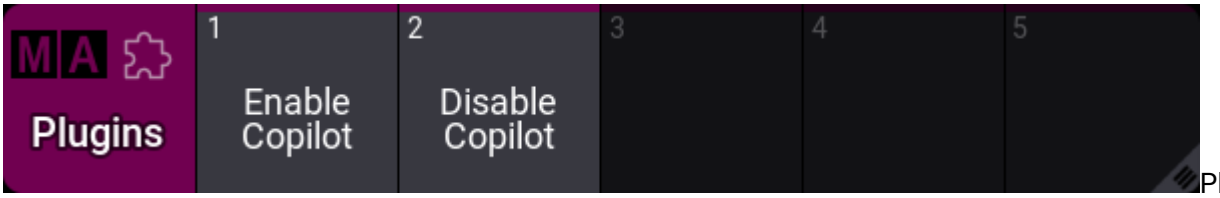
1.44. Plugins

Plugins are pieces of Lua code that can add features or functionality to the grandMA3.

With plugins, it is possible to do more than what can be achieved with macros.

	Important:
	The creation and use of plugins can go deeper into the system than the "normal" usage of a console. Therefore the technical support team of MA Lighting may not be able to help you in all circumstances when using complex plugins, and plugins might have to be rewritten when migrating show files to future grandMA3 software versions.

The plugins are stored in the Plugins Pool.



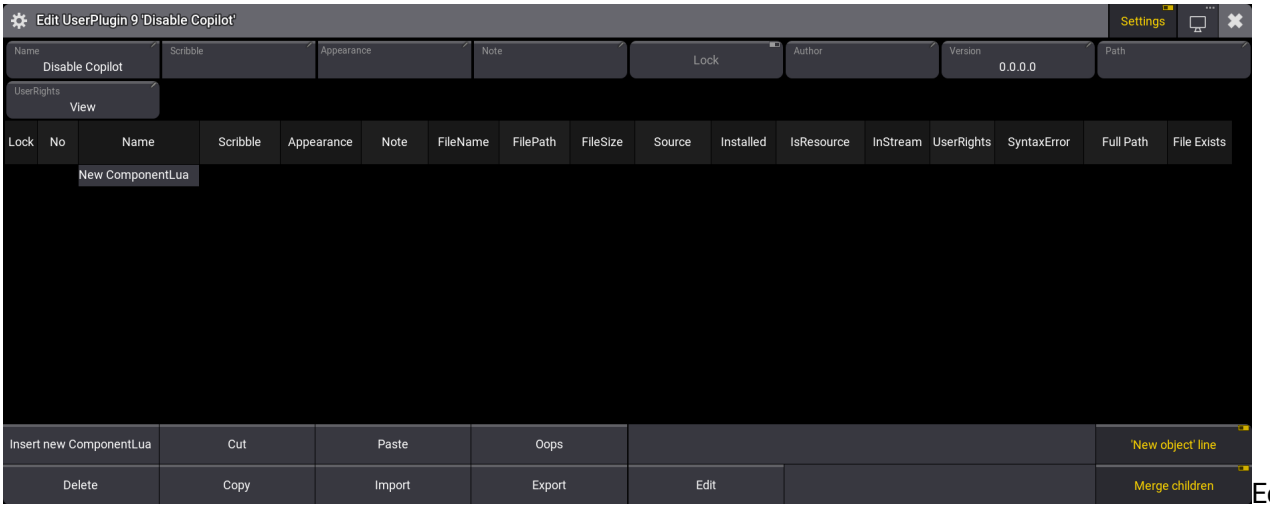
Plugin pool window

The pool can be created like any other using the **Add Window pop-up**.

Create a New Plugin

To create or add a plugin to the plugin pool, edit an empty pool object using any edit method.

The **Edit UserPlugin** pop-up opens:



Edit Plugin window

This editor can show the settings for the plugin if Setting is On in the title bar.

These are the settings:

- **Name**:
  The name of the plugin. Tap this to edit the name.
- **Scribble**:
  The assigned scribble. Tap this to select a scribble or open the editor to create a new one.
- **Appearance**:
  The assigned appearance. Tap this to select an appearance or open the editor to create a new one.
- **Note**:
  A note can be added to the plugin.
- **Lock**:
  Toggle this On to lock the plugin from changes.
- **Author**:
  This can be used to add the name of the plugin author.
- **Version**:
  This custom field can be used to indicate a version of the plugin.
- **Path**:
  This is a path (sub-folders) for the files inside the plugin folder (gma3_library/datapools/plugins).
- **UserRights**:
  This can be used to set the needed user rights for running this plugin.

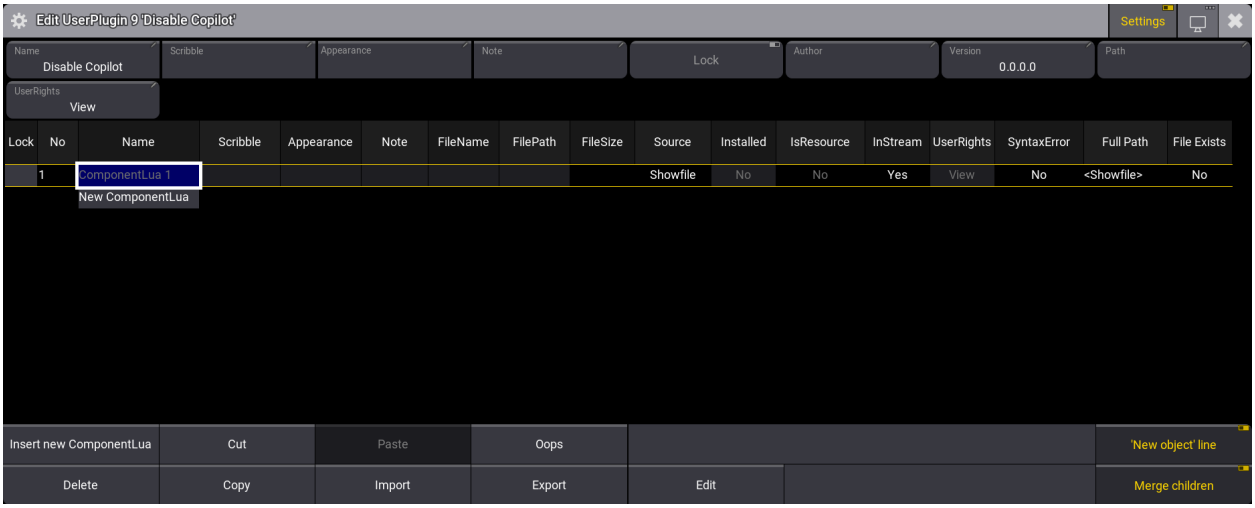
A plugin can contain several Lua components but should at least have 1.

Create and Edit a Component

A Lua component contains the Lua code. Each component is usually one file.

Add the component by pressing and holding the New ComponentLua.

A new component can also be added by selecting a line in the component list and tapping Insert new ComponentLua. This creates a new line above the selected line.



Plugin with 1 added Lua component

Each component has some settings and some information. It is the different columns in the list.

- **Name**:
  This is the name of the component.
- **Scribble**:
  This is the assigned scribble - this is not displayed anywhere.
- **Appearance**:
  This is the assigned appearance - this is not displayed anywhere.
- **Note**:
  A note can be added to the component.
- **FileName**:
  This is the file name for the Lua component file. A file name is needed if the plugin is exported. Giving the component a name also creates a file name.
- **FilePath**:
  This is a sub-folder path for the Lua file inside the plugin folder (gma3_library/datapools/plugins).
- **FileSize** (Information only):
  This displays the file size of the Lua component file in bytes.
- **Source** (Information only):
  This is information only. It indicates whether the component is located in the showfile or the library.
- **Installed**:
  If this is set to Yes, the Lua components will be updated from the file archive they were imported from. This is useful when Lua files are edited and copied into the folder using an external editor.
  To update any changes in these Lua components, the **ReloadAllPlugins keyword** must be executed.
- **IsResource** (Information only):
  This indicates if the Lua file is an internal resource stored in an internal library. These files are not stored in the showfile and are not streamed.
- **InStream** (Information only):
  Yes means that the Lua code is saved in the show file and streamed in the session but stays as saved in the show file.
  No means that the Lua code is locally saved on the hard drive. The content of this Lua code can be updated using the **ReloadAllPlugins keyword**.
  Setting **Installed** to Yes will set InStream to No.
- **UserRights**:
  This can be used to set user rights on specific components.
- **SyntaxError** (Information only):
  If syntax errors prevent the Lua component from being loaded into the Lua engine, the SyntaxError property turns to Yes, and the component line turns red.
- **Full Path** (Information only):
  This shows "<showfile>" or the full path for the file.
- **File Exists** (Information only):
  This is information only. It indicates "Yes" if the file is found or "No" if it is not found. This is only relevant if the source is the Library.

Select the component that needs to be edited and tap Edit.

This opens the **Lua Input** pop-up:

Empty Lua Input pop-up

Text can be copied into the pop-up from an external editor (when using the grandMA3 onPC), or it can be written directly in the pop-up.

Editing existing code can also be done in the Lua Input pop-up.

Line numbers help when troubleshooting code or as general help (the numbers are not part of the code). The number of the active line is in white color, and the others are in gray color.

When Tab is used to indent text, a gray ">" appears where the tab was pressed.

When the desired code is input, it needs to be saved by tapping Save in the title bar.

The title bar has other buttons:

- **(Virtual Keyboard)**:
  This toggles the on-screen virtual keyboard.
- **Lua Code Templates**:
  This toggles the Lua Code Templates that can be used as templates for writing new code. Learn more about this **below**.
- **API Description**:
  This toggles the grandMA3-specific Lua API description on the right-hand side of the Lua Input. Learn more about this **below**.
- **(Undo)**:
  This undoes the last edit.
- **(Redo)**:
  Thie redoes undoed actions.

After the code is saved, the Lua Input pop-up can be closed by tapping the in the upper right corner.

Run a Plugin

The plugins can be run by tapping them in the pool or using the **Plugin keyword**.

Running the plugin will execute the first component. This component needs to call other components, or specific components can be called directly using this syntax:

Plugin [Plugin_Number or "Plugin_Name"].[Component_Number or "Component_Name"]

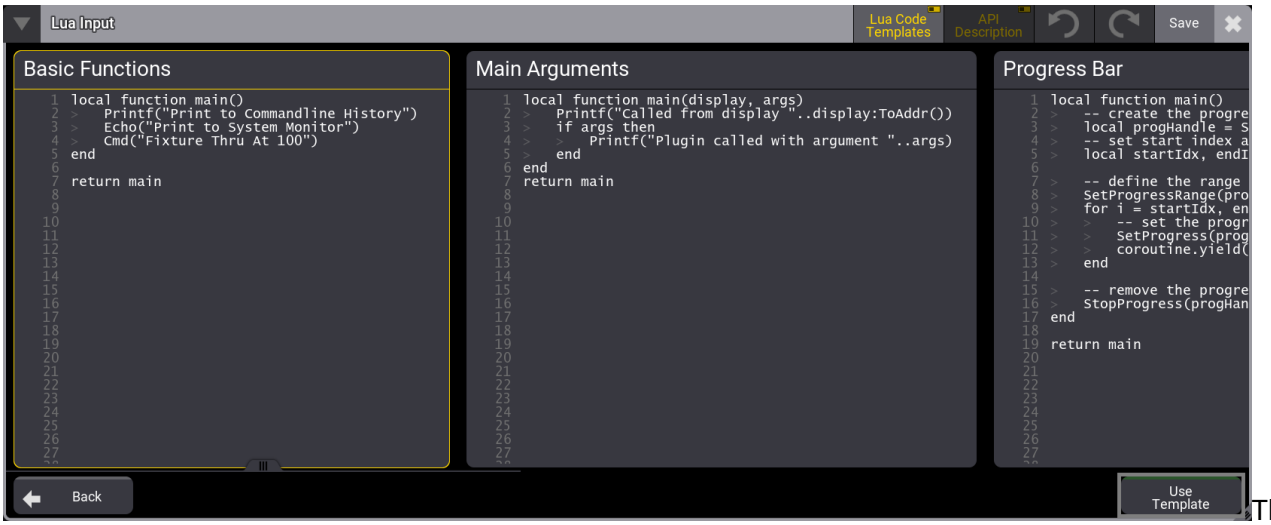
Plugins can also be assigned to executors and view buttons. Learn how in the **Assign Object to an Executor topic**.

Lua Code Templates

The templates are example codes that can be copied into the editor.

	Restriction:
	Using a template will delete any existing code.

Tap Lua Code Templates in the title bar to show the templates.



The Lua code templates

The different templates are shown next to each other, and there is a horizontal scroll bar, making it possible to scroll through the different templates.

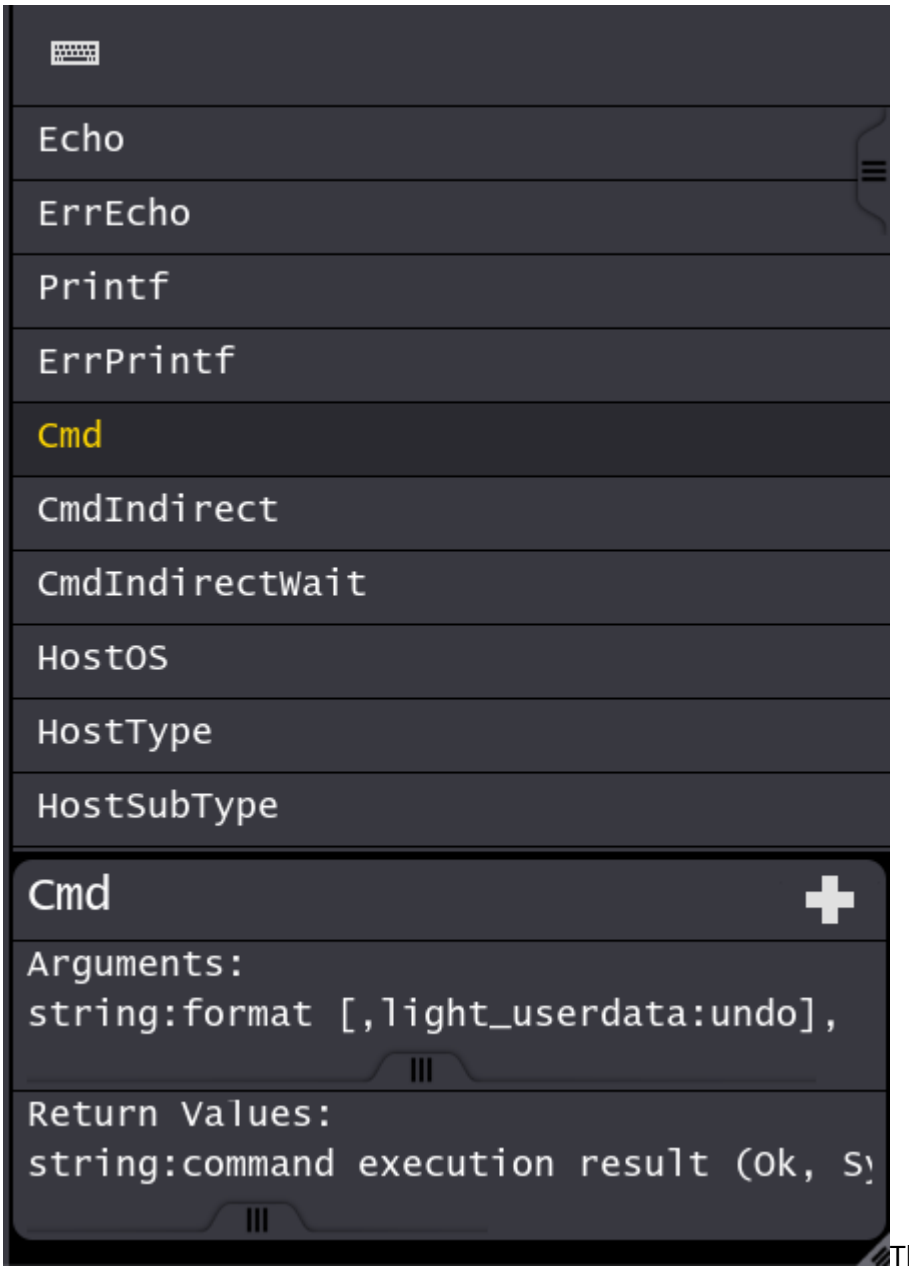
To import one of the templates, select it (yellow frame) and tap Use Template.

The templates can be closed without importing a template by tapping Lua Code Templates in the title bar or the Back in the lower left corner.

API Description

The API description can be shown on the right side of the Lua Input pop-up. It can be toggled On or Off by tapping API Description in the title bar.

It lists all the grandMA3-specific Lua functions and shows a short explanation of the required arguments and the return values.



The API description

There are three elements to the description. At the top is a search input field where text can be input to search the list of functions.

Below is the actual list of functions in alphabetic order. Selecting one shows the description of the selected function below the list.

Tapping the ✚ in the description area adds the function to the Lua input code, with the arguments, at the cursor's current location.

Export a Plugin

The plugin can be exported from the **Edit UserPlugin** by tapping Export.

This creates an XML file for the plugin.

If the Lua components have a file name ending in ".lua", an extra LUA file is exported for the component. If there is no valid file name for the component, it is then stored in the XML file, coded in the "base64" format.

	Restriction:
	The Lua files are only exported if the Lua component has "Yes" in the "InStream".

Import a Plugin

All the required files are needed to import a plugin successfully. The XML file is always needed; if the Lua components are stored as extra Lua files, they are also needed. These files must be in the ../gma3_library/datapools/plugins folder on the desired drive.

1. Create a new plugin.
2. Tap Import.
3. Select the desired drive and XML file.
4. Tap Import.
5. The plugin is imported, and the editor can be closed.

Subtopics

- **What is Lua**
- **Handle - light_userdata**
- **Interface Functions**
- **Variable Functions**
- **Lua Functions - Object-Free API**
- **Lua Functions - Object API**

1.44.1. What is Lua

Lua is a scripting language designed to support general procedural programming. It offers support for object-oriented programming, functional programming, and data-driven programming. Lua is implemented as a library, written in *clean* C (a common subset of ANSI C and C++).

For more information on scripting with Lua, see **www.lua.org**.

The grandMA3 software supports Lua version 5.4.6 and all the built-in standard libraries.

Use the **Version keyword** in the command line to check which Lua version your device is currently running.

Besides the standard Lua libraries, there are some custom grandMA3 Functions.

A list of all the current grandMA3 functions can be created by using the **HelpLua keyword**. This keyword creates a file called "grandMA3_lua_functions.txt" in the **gma3_library** folder. Learn more about how to find this in the **Folder Structure topic**.

This list separates the functions into two categories: **Object-Free API** and **Object API**. Individual functions are described in sub-topics in these two topics.

Lua code can be executed from plugins or directly in the command line using the **Lua keyword**.

1.44.2. Handle - light_userdata

Many functions require a handle as an argument or return a handle.

The handle is a custom data type called "light_userdata".

The handle is a unique identifier that refers to a grandMA3 object, for instance, a specific sequence, cue, preset, or fixture.

The object the handle refers to has some properties. Some can be changed, and some are read-only. The object might also have child objects. The object always has **Name**, **Class**, and **Path** information.

The path is the same as an **address** that identifies where the object exists in the structure of the showfile.

Related Functions

- **Addr()** - Converts a handle into a numbered address path.
- **AddrNative()** - Converts a handle into a named address path.
- **FromAddr()** - Converts a numbered or named address into a handle.
- **HandleToStr()** - Converts a handle into a string with a hexadecimal number.
- **HandleToInt()** - Converts a handle into an integer.
- **StrToHandle()** - Converts a hexadecimal number string into a handle.
- **IntToHandle()** - Converts an integer into a handle.
- **ToAddr()** - Object-free version - Converts a handle into an address string.
- **ToAddr()** - Object version - Converts a handle into an address string.

1.44.3. Interface Functions

There are several Lua functions related to creating and displaying interface elements.

They are often some of the most complex functions that can need a lot of different input elements that allow adjustment to specific needs.

Some of them can also return user input to Lua functions.

Individual functions are described like all others, but there are sub-topics here to explain further and provide a collection of functions.

	Known Limitation:
	Not all Lua functions are described yet. This part of the manual will expand in the future.

Subtopics

- **Progress Bar**

1.44.3.1. Progress Bar

The Progress Bar is a Lua function that can create a moving progress bar on the screens.

There are several Lua functions connected to creating and running a progress bar. See links to the topics below the example.

Example

This example uses all the progress bar functions:

Lua

```lua
return function()
    -- create the progress bar
    local progressBarHandle = StartProgress("myProgressTitle")

    -- set start index and end index of the progress bar
    local progressRangeStart, progressRangeEnd = 1, 10

    -- Define the range of the progress bar
    SetProgressRange(progressBarHandle, progressRangeStart, progressRangeEnd)
    -- Define the text of the progress bar
    SetProgressText(progressBarHandle, "This is my ProgressBar Text")
    -- Set the progress bar value to the start of range
    SetProgress(progressBarHandle, progressRangeStart)

    -- Loop that goes through the progress bar
    for i = progressRangeStart, progressRangeEnd do
        -- Add a yield to allow other functions and delay the progress
        coroutine.yield(1)
        -- Increment the progress state of the progress bar
        IncProgress(progressBarHandle, 1)
    end

    -- remove the progress bar:
    StopProgress(progressBarHandle)

end
```

Related Functions

- **StartProgress**
- **SetProgressRange**
- **SetProgress**
- **SetProgressText**
- **IncProgress**
- **StopProgress**

1.44.4. Variable Functions

This topic is not about local variables in Lua plugins. This is about interactions with variables outside the plugins. For instance, the same user and global variables can be used in macros and the regular command line. Read more about these in the **Variables** topic in the Macro section.

There are two different sets of variables.

The sets are:

- **UserVars()** - These are the variables stored as User variables.
- **GlobalVars()** - These are the variabels stored as Global variabels.

There are three functions regarding interaction with variables in a set:

- **GetVar()** - Gets the value from a specific variable in one of the sets.
- **SetVar()** - Sets a value in a specific variable in one of the sets. This also creates the variable if it does not exist.
- **DelVar()** - Deletes a specific variable in a set of variables.

Common for the three functions is a need to know the variable's name. The user and global variables can be listed using the **GetUserVariable** and **GetGlobalVariable** keywords combined with an asterisk wildcard.

1.44.5. Lua Functions - Object-Free API

Object-Free API means Lua functions that are not functions/methods of objects.

Subtopics

- **GetAttributeCount()**
- **GetAttributeIndex(string)**
- **GetButton(handle)**
- **GetChannelFunction(integer, integer)**
- **GetChannelFunctionIndex()**
- **GetClassDerivationLevel(string)**
- **GetCurrentCue()**
- **GetDebugFPS()**
- **GetDisplayByIndex(integer)**
- **GetDisplayCollect()**
- **GetDMXUniverse(integer[, boolean])**
- **GetDMXValue(integer[, integer, boolean])**
- **GetExecutor(integer)**
- **GetFocus()**
- **GetFocusDisplay()**
- **GetObjApiDescriptor()**
- **GetPath(string[, boolean] | integer)**
- **GetPathOverrideFor(string|integer, string[, boolean])**
- **GetPathSeparator()**
- **GetPathType(handle[, integer])**
- **GetRTChannel(integer)**
- **GetPresetData(handle[, boolean[, boolean]])**
- **GetRTChannelCount()**
- **GetRTChannels(integer[,boolean] OR handle[,boolean])**
- **GetSample(string)**
- **GetScreenContent(handle)**
- **GetSelectedAttribute()**
- **GetShowFileStatus()**
- **GetSubfixture(integer)**
- **GetSubfixtureCount()**
- **GetTokenName(string)**
- **GetTokenNameByIndex(int)**
- **GetTopModal()**
- **GetTopOverlay()**
- **GetUIChannelCount()**
- **GetUIChannelIndex(integer, integer)**
- **GetUIChannels(integer[,boolean] OR handle[,boolean])**
- **GetUIObjectAtPosition(integer, table)**
- **GetVar(handle, string)**
- **GlobalVars()**
- **HandleToInt(handle)**
- **HandleToStr(handle)**
- **HookObjectChange(function, handle, handle[, handle])**
- **HostOS()**
- **HostSubType()**
- **HostType()**
- **Import(string)**
- **IncProgress(handle, integer)**
- **IntToHandle(integer)**
- **IsClassDerivedFrom(string, string)**

- **IsObjectValid(handle)**
- **KeyboardObj()**
- **MasterPool()**
- **MessageBox(table)**
- **MouseObj()**
- **NeedShowSave()**
- **ObjectList(string[, table])**
- **Patch()**
- **Printf(string)**
- **Programmer()**
- **ProgrammerPart()**
- **Pult()**
- **ReleaseType()**
- **Root()**
- **SelectedFeature()**
- **SelectedLayout()**
- **SelectedSequence()**
- **SelectedTimecode()**
- **SelectedTimer()**
- **Selection()**
- **SelectionCount()**
- **SelectionFirst()**
- **SelectionNext()**
- **SerialNumber()**
- **SetBlockInput(boolean)**
- **SetLED(handle,table)**
- **SetProgress(handle, integer)**
- **SetProgressRange(handle, integer, integer)**
- **SetProgressText(handle, string)**
- **SetVar(handle, string, value)**
- **ShowData()**
- **ShowSettings()**
- **StartProgress(string)**
- **StopProgress(handle)**
- **StrToHandle(string)**
- **TextInput([string[, string[, integer[, integer]]]])**
- **Time()**
- **Timer(string, integer, integer[, string[, handle]])**
- **ToAddr(handle[, boolean])**
- **TouchObj()**
- **Unhook(integer)**
- **UnhookMultiple(function, handle, handle)**
- **UserVars()**
- **Version()**

1.44.5.1. AddFixtures(table)

Description

The **AddFixture** Lua function adds fixtures to the patch. The argument for the function is a table, which must contain valid data for the function to succeed. The function returns a "true" boolean value if the addition was a success. The function must be run with the command line in the correct patch destination.

Arguments

- **Table**:
  The table must contain valid data. This is a list of possible table elements. It is not necessary to add all elements.
  - **mode**:
    This must be a **handle** to a valid "dmx_mode". This defines a specific fixture type in a specific mode.
  - **amount**:
    This is an **integer** number that defines how many fixtures should be added.
  - **name** (optional):
    This is a string with the name of the (first) fixture.
  - **fid** (optional):
    This is a **string** with the fixture's FID.
  - **cid** (optional):
    This is a **string** with the CID for the fixture. This table field is only valid if the "idtype" is not "Fixture".
  - **idtype** (optional):
    This is a **string** with the name of the ID Type. This is only needed if the type is different than "Fixture".
  - **patch** (optional):
    This is a **table** with up to eight **strings**. The string must indicate a universe and a start address in the universe. The two must be separated by a dot. Each table element is used for the up to eight DMX breaks in the patch.
  - **layer** (optional):
    This is a **string** with the layer name.
  - **class** (optional):
    This is a **string** with the class name.
  - **parent** (optional):
    This is a **handle** of the parent fixture. It is only needed if the fixture should be a sub-fixture of an existing fixture.
  - **insert_index** (optional):
    This is an **integer** indicating an insert index number.
  - **undo** (optional):
    This is a **string** with an undo text.

Return

- **Boolean** or **nil**:
  The returned table contains key value pairs with configuration information. See the example below.

Example

This example adds a dimmer fixture with FID and CID 301 and patch address "10.001". It is a requirement that the generic dimmer type is already added to the show and that the ID and patch address are available. The example does not perform any check for availability.

Lua

```
return function()
    -- Change the command line destination to the correct patch location.
    Cmd("ChangeDestination Root")
    -- Enter the "ShowData".
    Cmd('ChangeDestination 13')
    -- Enter the "Patch".
    Cmd("ChangeDestination 9")
    -- Enter the "Stages".
    Cmd("ChangeDestination 6")
    -- Enter the first stage object.
    Cmd("ChangeDestination 1")
    -- Enter the "Fixtures" part of the stage.
    Cmd("ChangeDestination 2")

    -- Create a table.
    local myAddFixtureTable = {}
    -- Set the mode to a 8-bit Dimmer fixture type.
    myAddFixtureTable.mode = Patch().FixtureTypes.Dimmer.DMXModes["Mode 0"]
    -- Set the amount of fixtures.
    myAddFixtureTable.amount = 1
    -- Set the FID for the fixture.
    myAddFixtureTable.fid = "301"
    -- Set the IdType - it is not needed if the type is "Fixture".
    myAddFixtureTable.idtype = "Channel"
    -- Set the CID - Use only this when the "idtype" is different than
Fixture.
    myAddFixtureTable.cid = "301"
    -- Set the name of the fixture.
    myAddFixtureTable.name = "Dimmer 301"
    -- Create a patch table with an address.
    myAddFixtureTable.patch = {"10.001"}

    -- Add the fixture to the patch using the table data. Store the result in
a local variable.
    local success = AddFixtures(myAddFixtureTable)

    -- Provide some feedback.
    if success ~= nil then
        Printf("Fixture " .. myAddFixtureTable.fid .. " is added with patch
address " .. myAddFixtureTable.patch[1])
    else
        Printf("AddFixture failed!")
    end

    -- Return the command line to the root destination.
```

```
    Cmd("ChangeDestination Root")
end
```

1.44.5.2. AddonVars(string)

Description

The **AddonVars** function returns a handle to the set of variables connected to a specific addon.

	Restriction:
	The addon variable set is not helpful at the moment.

Arguments

- **String**:
  The string needs to be the name of the addon.

Return

- **Handle**:
  The function returns a handle of the set of variables.

Example

This example prints information connected to the "Demo" addon variable set. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Stores the handle to a variable set connected to the addon named
'Demo'.
    local variableSet = AddonVars("Demo")
    -- Check if the return is nil and print an error message
    if variableSet == nil then
        ErrPrintf("The variable set does not exists")
        return
    end
    Printf("=============== START OF DUMP ===============")
    variableSet:Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.3. BuildDetails()

Description

The BuildDetails function returns a table with key-value pairs about the software build.

Arguments

This function does not accept any arguments.

Return

- **Build details**:
  This is the table with key-value pairs. These are the possible keys in the table:
  - **GitDate**: String with the date for the repository branch of the software.
  - **GitHead**: String with the branch of the repository.
  - **GitHash**: String with the hash for the repository.
  - **CompileDate**: String with the date for the compile.
  - **CompileTime**: String with the time for the compile.
  - **BigVersion**: String indicating the software version.
  - **SmallVersion**: String with the small version number of the software. Devices that only listen to a DMX data stream need to have this version to "understand" the streaming data.
  - **HostType**: String with the host type, for instance, "Console" or "onPC".
  - **HostSubType**: String with the host sub-type, for instance, "FullSize" or "Light".
  - **CodeType**: String showing the type of code, for instance, "Release".
  - **IsRelease**: Boolean indicating if the software is a release version.

Example

This example prints the content of the BuildDetails table:

```lua
return function()
    --Store the build detials table
    local myBuild = BuildDetails()
    --Print the content of the table
    Printf("GitDate: " .. myBuild.GitDate)
    Printf("GitHead: " .. myBuild.GitHead)
    Printf("GitHash: " .. myBuild.GitHash)
    Printf("CompileDate: " .. myBuild.CompileDate)
    Printf("CompileTime: " .. myBuild.CompileTime)
    Printf("BigVersion: " .. myBuild.BigVersion)
    Printf("SmallVersion: " .. myBuild.SmallVersion)
    Printf("HostType: " .. myBuild.HostType)
    Printf("HostSubType: " .. myBuild.HostSubType)
    Printf("CodeType: " .. myBuild.CodeType)
    Printf("IsRelease: " .. tostring(myBuild.IsRelease))
end
```

Related Lua Functions

- **Version()**
- **HostType()**
- **HostSubType()**

1.44.5.4. CheckDMXCollision(handle, string[, integer[, integer]])

Description

The **CheckDMXCollision** Lua function checks if a specific DMX address range is available or already used.

It uses the number of DMX channels in a specific "DMX mode" of a fixture type to calculate the number of DMX channels that should be available from a specified DMX start address.

All fixture types have at least one defined DMX mode. But fixtures often have more than one mode. This Lua function uses a specific DMX mode of a fixture type.

Arguments

- **Handle**:
  The handle must be for a "DMX mode". This is used to calculate how many DMX channels should be available in the range.
- **String**:
  This must be a DMX address expressed as a string. This defines the start of the range to be checked.
- **Integer** (optional)|**nil**:
  This optional integer is a count of subsequent "DMX Modes" that should also be checked. The default value is **1**.
  For instance, if the provided "DMX Mode" uses 10 DMX channels and the count is set to 5, then there must be 50 unpatched DMX channels from the start address for a positive result.
- **Integer** (optional):
  This optional integer indicates the break_index. The default value is **0**, which is the first "DMX break" defined for the "DMX mode". All fixture types have at least one defined "DMX break".

Return

- **Boolean**:
  The function returns a boolean.
  - **True**:
    The DMX address is available as a start address.
  - **False**:
    The DMX address is unavailable as a start address for the calculated number of DMX channels.

Example

This example prints feedback to the DMX collision check based on a DMX address of "1.001" and the DMX mode of the first fixture in the current selection:

Lua

```
return function()
    -- Set the DMX universe - range 1-1024.
    local myDMXUniverse = 1
```

```
    -- Set the DMX address in the universe - range 1-512.
    local myDMXAddress = 1
    -- Set the optional count for the number of fixtures (break_index channel
amount) to check.
    local myCount = 1
    -- Set the optional break_index number for fixtures with multiple breaks.
    -- Default value is 0 to indicate the first break.
    local myBreakIndex = 0

    -- Creates the string used for the DMX address.
    local startOfRange = string.format("%d.%03d", myDMXUniverse,
myDMXAddress)

    -- Check if there is a selection and exit if there isn't.
    if SelectionFirst() == nil then
        Printf("Please make a selection and try again.")
        return
    end
    -- This gets the handle for the first fixture a patched generic Dimmers
8-bit mode.
    local myDmxMode = GetSubfixture(SelectionFirst()).ModeDirect

    if myDmxMode == nil then
        -- Exit the function if the DMX mode returns nil.

    else
        -- Do the actual collision check and provide useful feedback.
        if CheckDMXCollision(myDmxMode, startOfRange, myCount, myBreakIndex)
then
            Printf("The DMX address " .. startOfRange .. " is available.")
            return
        else
            Printf("The DMX address " .. startOfRange .. " cannot be used as
a start address for this patch.")
            return
        end
    end
end
```

1.44.5.5. CheckFIDCollision(integer[, integer[, integer]])

Description

The **CheckFIDCollision** Lua function checks if a specific (range of) ID is available or already used. It can be used to check FID and any type of CID by adding a type integer.

Arguments

- **Integer**:
  The first integer is the ID that should be checked.
- **Integer** (optional):
  This optional integer is a count of subsequent IDs that should also be checked. The default value is **1**. For instance, if FID 21 to 25 should be checked, then the count integer should be **5**.
- **Integer** (optional):
  This optional integer indicates the IDType. The default value is **0**, which is the "Fixture" ID Type. See the example below for other valid integers.

Return

- **Boolean**:
  The function returns a boolean.
  - **True**:
    The ID is available.
  - **False**:
    The ID is already used.

Example

This example prints feedback to the FID check:

Lua

```
return function()
    -- Create a variable with the FID you want to check.
    local myFID = 2001
    -- Create a variable with the number of subsequent ID's to also check.
    local myCount = 10
    -- Create a variable with the IDType you want to check.
    -- Default value is 0. This is the "Fixture" type.
    -- Valid integers are:
    --- 0 = Fixture
    --- 1 = Channel
    --- 2 = Universal
    --- 3 = Houseligths (default name)
    --- 4 = NonDim (default name)
    --- 5 = Media (default name)
    --- 6 = Fog (default name)
    --- 7 = Effect (default name)
    --- 8 = Pyro (default name)
    --- 9 = MArker
    --- 10 = Multipatch
```

```
    local myType = 0

    -- Check if the count is more than one.
    if myCount > 1 then
        -- Check if there is a collision and print valid feedback.
        if CheckFIDCollision(myFID, myCount, myType) then
            Printf("The FID " .. myFID .. " to " .. (myFID + myCount) .. " is
available.")
            return
        else
            Printf("The FID " .. myFID .. " to " .. (myFID + myCount) .. "
gives an FID collision.")
            return
        end
    else
        if CheckFIDCollision(myFID, nil, myType) then
            Printf("The FID " .. myFID .. " is available.")
            return
        else
            Printf("The FID " .. myFID .. " gives an FID collision.")
            return
        end
    end
end
```

1.44.5.6. ClassExists(string)

Description

The **ClassExists** Lua function returns a boolean indicating whether the provided string is a class.

Arguments

- **String**:
  A string containing a single word that could be a class.

Return

- **Boolean**:
  The function returns a boolean.
  - **True**:
    The provided word is a class.
  - **False**:
    The provided input is not a class.

Example

This example asks if the word "Display" is a class and returns proper feedback.

Lua

```
return function()
    -- Store a string with the class name
    local className = "Display"
    -- Check if the class exists and then provide proper feedback
    if ClassExists(className) then
        Printf("The class '%s' exists", className)
    else
        Printf("The class '%s' does not exists", className)
    end
end
```

1.44.5.7. CloseAllOverlays()

Description

The **CloseAllOverlays** function closes any pop-ups or menus (overlays) open on any screen.

Arguments

This function does not accept any arguments.

Return

This function does not return anything.

Example

This example simply closes any overlay.

Lua

```
return function()
    CloseAllOverlays()
end
```

1.44.5.8. CloseUndo(handle)

Description

The **CloseUndo** Lua function closes an open undo list. The function returns a boolean indicating if the function succeeds.

Undo lists need to be created to be closed. See more about this in the **CreateUndo** function.

Arguments

- **Handle**:
  The handle of a created undo list.

Return

- **Boolean**:
  - True: The undo list was closed.
  - False: The undo list is still in use and cannot be closed.

Example

This example creates an undo list, performs a series of commands that are added to the undo list, and closes the undo list. Now the series of commands can be oopsed with one oops command.

Lua

```lua
return function()
    --Create the undo object
    local MyNewUndo = CreateUndo("MySelection")
    --Create command actions connected to the undo object
    Cmd("ClearAll", MyNewUndo)
    Cmd("Fixture 1", MyNewUndo)
    Cmd("Fixture 2", MyNewUndo)
    Cmd("Fixture 5", MyNewUndo)
    Cmd("Fixture 7", MyNewUndo)
    --Close the undo group and store it's return in a variable
    local closeSuccess = CloseUndo(MyNewUndo)
    --Print the feedback from the closing action - 1 = Success / 0 = Failure.
    if closeSuccess == false then
        ErrPrintf("The CloseUndo was not successful")
    elseif closeSuccess == true then
        Printf("The CloseUndo was successful")
    else
        Printf("The CloseUndo did not return a meaningful result")
    end
end
```

1.44.5.9. Cmd(string[, handle])

Description

The **Cmd** Lua function executes a command in the grandMA3 command line. It is executed in a Lua task - not the Main task (standard typed commands are run in the Main task). It is executed synchronously, and it blocks the Lua task while executing. This means that a bad command has the potential to block the system.

Alternative functions are **CmdIndirect()** and **CmdIndirectWait()**.

Argument

- **String**:
  A string with the command to be executed in the command line. Do not add a please or enter to execute the command.
- **Handle** (optional):
  A handle to an undo (oops) list. Learn more in the **CreateUndo topic**.
- **...** (optional):
  Additional arguments relevant for the command.

Return

- **String**:
  A string is returned with the execution feedback known from the command line feedback
  - **OK**:
    Command executed.
  - **Syntax Error**:
    The command was not executed because of a syntax error.
  - **Illegal Command**:
    Command not executed because of some illegal command or action.

The returned string does not need to be used.

Example

This example executes the command "ClearAll" in the command line.

Lua

```
return function()
    --Execute the command directly
    Cmd("ClearAll")
end
```

1.44.5.10. CmdIndirect(string[, handle[, handle]])

Description

The **CmdIndirect** Lua function executes a command within the grandMA3 command line. It is executed asynchronously in the Main task. It does not block the Lua execution since it is not executed in the Lua Task.

Arguments

- **String**:
  A string with the command to be executed in the command line. Do not add a please or enter to execute the command.
- **Handle** (optional):
  A handle to an undo (oops) list. Learn more in the **CreateUndo topic**.
- **Handle** (optional):
  This is a handle for the target for the command. The target can be a specific screen. See the example below.

Return

This function does not return anything.

Example

This example prints "1" and "2" in the Command Line History and let the main task open the **Configure Display pop-up** on screen 2.

```lua
return function()
    --Print something
    Printf("1")
    --Use the 'CmdIndirect' to open a pop-up
    CmdIndirect("Menu DisplayConfig", nil, GetDisplayByIndex(2))
    --Print something else
    Printf("2")
end
```

The Command Line History shows:

```
OK :  Call Plugin 49
1
2
OK :  Menu "DisplayConfig"
```

1.44.5.11. CmdIndirectWait(string[, handle[, handle]])

Description

The **CmdIndirectWait** Lua function executes a command within the grandMA3 command line. It does not block the Lua execution and is executed synchronously in the main task. Synchronous commands wait for the command to be executed before executing any following command.

Arguments

- **String**:
  A string with the command to be executed in the command line. Do not add a please or enter to execute the command.
- **Handle** (optional):
  A handle to an undo (oops) list. Learn more in the **CreateUndo** topic.
- **Handle** (optional):
  This is a handle for the target for the command. The target can be a specific screen. See the example below.

Return

This function does not return anything.

Example

This example prints "1" and "2" in the Command Line History and lets the main task open the **Configure Display pop-up** on screen 2.

```lua
return function()
    --Print something
    Printf("1")
    --Use the 'CmdIndirectWait' to open a pop-up
    CmdIndirectWait("Menu DisplayConfig", nil, GetDisplayByIndex(2))
    --Print something else
    Printf("2")
end
```

The Command Line History shows:

```
OK :  Call Plugin 50
1
OK :  Menu "DisplayConfig"
2
```

1.44.5.12. CmdObj()

Description

The **CmdObj** Lua function returns information about the command line object.

Argument

This function does not have any arguments.

Return

- **Handle**:
  The function returns a handle to the command line object.

Example

This example uses the **Dump() function** on the command object. It lists all the properties and lists the children and some extra examples of how the command line object can be used:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    --Store the handle to the command object
    local cmd = CmdObj()
    --Print all information about the command object
    Printf("=============== START OF DUMP ===============")
    cmd:Dump()
    Printf("================ END OF DUMP ================")
    --Print some selected elements from the command object - this is
currently not in the online manual
    Printf("Current text in the command line: " ..cmd.cmdtext)
    Printf("Current cmd edit object: " ..tostring(cmd.editobject and
cmd.editobject:ToAddr()))
    Printf("Current cmd destination: " ..tostring(cmd.destination and
cmd.destination:ToAddr()))
    Printf("Current user of the command line: " ..tostring(cmd.user and
cmd.user:ToAddr()))
    Printf("Current profile of the command line: " ..tostring(cmd.profile and
cmd.profile:ToAddr()))
    Printf("Current DMX readout: " ..cmd.dmxreadout)
    Printf("Current amount steps: " ..cmd.maxstep)
    Printf("Current selected object: " ..tostring(cmd:GetSelectedObject() and
cmd:GetSelectedObject():ToAddr()))
```

```
end
```

1.44.5.13. ConfigTable()

Description

The **ConfigTable** Lua function returns a table with some configuration information. These are information only. The function does not have any actual functions. The table is not sorted.

Arguments

This function does not accept any arguments.

Return

- **Table**:
  The returned table contains kay value pairs with configuration information. See the example below.

Example

This example prints the content of the returned table.

Lua

```
return function ()
    -- Prints the content of the ConfigTable
    for key,value in pairs(ConfigTable()) do
        Printf(key .. " : " .. value)
    end
end
```

1.44.5.14. Confirm(string[, string[, integer[, boolean]]])

Description

The **Confirm** Lua function provides a simple confirmation pop-up for a true/false query. It is part of the user interface functions.

Arguments

- **String**:
  This string is the title for the pop-up.
- **String** (optional):
  This string is the text in the pop-up.
- **Integer** (optional):
  This integer is not used since the pop-up appears on all screens. The value can be *nil*.
- **Boolean** (optional):
  This boolean defines if there is a Cancel button in the pop-up or not.
  - true: There is a Cancel button in the pop-up. This is the default option used if it is not defined.
  - false: There is only an OK button in the pop-up.

Return

- **Boolean**:
  - True / 1: The pop-up was confirmed with the OK.
  - False / 0: The pop-up was not confirmed with Cancel. This is only a possible option if the Cancel button is visible.

Example

This example creates a confirmation pop-up with printed feedback in the Command Line History:

Lua

```
return function()
    --Creates a pop-up asking to be confirmed and prints a useful text.
    if Confirm("Confirm me", "Tap OK or Cancel", nil, true) then
        Printf("Pop-up result: OK")
    else
        Printf("Pop-up result: Cancel")
    end
end
```

1.44.5.15. CreateMultiPatch({handles}, integer[, string])

Description

The **CreateMultiPatch** Lua function creates a series of multi patch fixtures to a table of fixtures.

Arguments

- **Table**:
  The table must contain **handles** to the fixtures who should have the multi patch fixtures.
- **Integer**:
  The number of multi patch fixtures to create.
- **String** (optional):
  The string is an optional undo text. It needs to be in quotation marks.

Return

- **Integer** | **nil**:
  The returned integer indicates the amount of multi patch fixtures created.

Example

This example creates two multi patch fixtures to the first fixture (excluding the "Universal" fixture) in the patch.

Lua

```
return function()
    -- Enter Patch.
    Cmd("ChangeDestination Root");
    -- Enter the SetupPatch.
    Cmd("ChangeDestination 'ShowData'.'Patch'");

    -- Get the handle for the first fixture in the patch.
    local myFixture = Patch().Stages[1].Fixtures[2]
    -- Add the handle a list element in an table.
    local myFixtureTable = {myFixture}
    -- Add a variable with the amount of multipatch fixtures needed.
    local multiPatchAmount = 2

    -- Count the number of elements in the fixture table and store in a
variable.
    local count = 0
    for _ in pairs(myFixtureTable) do
        count = count + 1
    end
    -- Create an unto text string.
    local undoText = string.format("Create %d multipatch fixtures for up to
%d fixtures", multiPatchAmount, count)

    -- Create the multipatch fixtures to the each fixture handle in the table
```

```
and store the returned value.
    local multiPatchAmount = CreateMultiPatch(myFixtureTable,
multiPatchAmount, undoText)
    if multiPatchAmount ~= nil then
        Printf(multiPatchAmount .. " multi patch objects was created")
    else
        Printf("An error occured")
    end

    -- Return the command line destination to the root.
    Cmd("ChangeDestination Root")
end
```

1.44.5.16. CreateUndo(string)

Description

The **CreateUndo** Lua function returns a handle to a list of commands and function calls grouped in the same oops action.

Functions can be executed with a reference to the undo handle. This adds the function to the undo list.

Undo lists need to be closed using the **CloseUndo** function.

Arguments

- **String**:
  A text string must be added. It can be used to identify the undo list.

Return

- **Handle**:
  The function returns the handle to the undo list.

Example

This example creates an undo list, performs a series of commands being added to the undo list, and closes the undo list. Now, the series of commands can be oopsed with one oops command.

Lua

```
return function()
    -- Create the undo group.
    local MyNewUndo = CreateUndo("MySelection")
    -- Make some command line actions linked to the undo.
    Cmd("ClearAll", MyNewUndo)
    Cmd("Fixture 1", MyNewUndo)
    Cmd("Fixture 2", MyNewUndo)
    Cmd("Fixture 5", MyNewUndo)
    Cmd("Fixture 7", MyNewUndo)
    -- Closing the undo group and store it's return in a variable.
    local closeSuccess = CloseUndo(MyNewUndo)
    -- Print the feedback from the closing action - 1 = Success / 0 =
Failure.
    if closeSuccess == false then
        ErrPrintf("The CloseUndo was not successful")
    elseif closeSuccess == true then
        Printf("The CloseUndo was successful")
    else
        Printf("The CloseUndo did not return a meaningful result")
    end
end
```

1.44.5.17. CurrentEnvironment()

Description

The **CurrentEnvironment** Lua function returns a handle to the current users' selected environment.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the environment.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Dumps information about the current environment
    Printf("=============== START OF DUMP ===============")
    CurrentEnvironment():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.18. CurrentExecPage()

Description

The **CurrentEnvironment** Lua function returns a handle to the current users' selected executor page.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the executor page.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Dumps information about the current executor page
    Printf("=============== START OF DUMP ===============")
    CurrentExecPage():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.19. CurrentProfile()

Description

The **CurrentProfile** Lua function returns a handle to the current users' profile.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the user profile.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Dumps information about the current executor page
    Printf("=============== START OF DUMP ===============")
    CurrentProfile():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.20. CurrentScreenConfig()

Description

The **CurrentScreenConfig** Lua function returns a handle to the current users' screen configuration.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the screen configuration.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Dumps information about the current screen configuration
    Printf("=============== START OF DUMP ===============")
    CurrentScreenConfig():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.21. CurrentUser()

Description

The **CurrentUser** Lua function returns a handle to the current user.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the user.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Dumps information about the current user
    Printf("=============== START OF DUMP ===============")
    CurrentUser():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.22. DataPool()

Description

The **DataPool** Lua function references the currently selected DataPool and is used to read or edit properties within the data pool.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle to the DataPool object.

Example

This example uses the **Dump() function** on the data pool object. Dump lists all the properties and lists the children. Finally, the example also prints the name of the first sequence in the data pool.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```lua
return function()
    -- Dumps information about the datapool object.
    Printf("=============== START OF DUMP ===============")
    DataPool():Dump()
    Printf("================ END OF DUMP ================")
    -- Prints the name of the first sequence.
    Printf("Name of sequence 1: " .. DataPool().Sequences[1].Name)
end
```

1.44.5.23. DefaultDisplayPositions()

Description

The **DefaultDisplayPositions** Lua function returns the handle of the conventional default display positions, which contains the first seven screens as children.

For example, whether the command line, view bar, and encoder/playback bar are displayed.

Argument

This function does not have any arguments.

Return

- **Handle**:
  The function returns a handle to the command line object.

Examples

This example prints all the information about display 1 (child 1 of the default displays) using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

```lua
return function()
    -- Store a handle to display 1 (child 1 of the default displays).
    local display1 = DefaultDisplayPositions():Children()[1]
    -- Dumps information about the display.
    Printf("=============== START OF DUMP ===============")
    display1:Dump()
    Printf("================ END OF DUMP ================")
end
```

This example toggles the **Control Bar** for display 1 with the help of the DefaultDisplayPositions object:

```lua
return function()
    -- Store a handle to display 1 (child 1 of the default displays).
    local display1 = DefaultDisplayPositions():Children()[1]
    -- Toggles the 'ShowMainMenu' setting.
    display1.ShowMainMenu = not display1.ShowMainMenu
```

```
end
```

1.44.5.24. DelVar(handle, string)

Description

The **DelVar** Lua function deletes a specific variable in a set of variables. To learn more about the variables in plugins, have a look at the **Variable Functions** topic.

Arguments

- **Handle**:
  The handle of variable set.
- **String**:
  The name of the variable. It needs to be in quotation marks.

Return

- **Boolean**:
  - True / 1: The variable was deleted.
  - False / 0: The variable was not deleted.

If the variable does not exist, then false is also returned.

Example

This example deletes a variable called "myUserVar" in the set of user variables.

Lua

```
return function()
    -- Deletes the variable called 'myUserVar' in the 'UserVars' variable
set.
    local success = DelVar(UserVars(), "myUserVar")
    -- Prints the outcome of the deletion outcome.
    if success then
        Printf("Variable is deleted.")
    else
        Printf("Variable is NOT deleted!")
    end
end
```

1.44.5.25. DeskLocked()

Description

The **DeskLocked** Lua function returns a boolean indicating if the station is locked.

Arguments

This function does not accept any arguments.

Return

- **Boolean**:
  The boolean indicates if the station is desk locked or not.
  - **True** (or 1): The station is locked.
  - **False** (or 0): The station is not locked.

Example

This example prints the boolean number indicating the "DeskLocked" status to the **Command Line History**.

Lua

```
return function()
    -- The DeskLocked() return is printed.
    Printf("The desk is locked: " .. tostring(DeskLocked()))
end
```

1.44.5.26. DeviceConfiguration()

Description

The **DeviceConfiguration** Lua function returns a handle to the DeviceConfiguration object.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the DeviceConfiguration.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

```lua
return function()
    -- This example dumps all information about the DeviceConfiguration object.
    Printf("=============== START OF DUMP ===============")
    DeviceConfiguration():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.27. DirList(string[, string])

Description

The **DirList** Lua function returns a table of files at a specified path. The returned list can be filtered using an optional filter argument.

Arguments

- **String**:
  The desired path in a string format.
- **String** (optional):
  The optional filter string. The * can be used as a wildcard in the string. See the example below.

Return

- **Table**:
  The returned table has elements of other tables. Each of these table elements has the following keys:
  - name: The name of the file. The value of name is returned as a string.
  - size: The size of the file in bytes. The value of size is returned as a number.
  - time: The timestamp for the file. The value of time is returned as a number.

Example

This example prints the show files in the showfile directory. It uses the **GetPath() function**.

The GetPath Lua function delivers a string with the path of a grandMA3 folder. Learn more in the **GetPath() topic**.

Lua

```
return function ()
    -- Get the path to the show files.
    local path = GetPath(Enums.PathType.Showfiles)
    -- Make a filter to only list .show files.
    local filter = "*show"
    -- Use the DirList function to get a table of the files.
    local returnTable = DirList(path, filter)

    -- Print the information of the files in the returned table.
    for _, value in pairs(returnTable) do
        Printf(value['name'] .. " - Size: " .. value['size'] .. " bytes -
Time: " .. os.date("%c", value['time']))
    end
end
```

1.44.5.28. DrawPointer(integer,table[,integer])

Description

The **DrawPointer** function draws a red pointer on the display. There can only be one pointer at a time on each station.

Arguments

- **Integer**:
  This integer is the display index where the pointer should be drawn.
- **Table**:
  This key-value table must have 'x' and 'y' keys with values indicating a position on the display. See the example below.
- **Integer** (optional):
  This optional integer defines a duration for the pointer in milliseconds. It fades out. If a duration is not set, then it stays visible.

Return

This function does not return anything.

Example

This example draws a pointer on display 1 for 5 seconds:

Lua

```lua
return function()
    --Set a display index
    local displayIndex = 1
    --Create and set the position in a table
    local position = {}
    position.x = 150
    position.y = 25
    --Set a 5 seconds duration - in milliseconds
    local duration = 5000
    --Draw the actual pointer
    DrawPointer(displayIndex,position,duration)
end
```

1.44.5.29. DumpAllHooks()

Description

The **DumpAllHooks** function prints a list of the hooks in the system. The list is only shown in the **System Monitor**.

Arguments

This function does not accept any arguments.

Return

The function does not return anything. It does print a list in the system monitor.

	**Hint:**
	See also these related functions: **HookObjectChange**, **Unhook**, **UnhookMultiple**.

Example

This example prints the list of hooks in the **system monitor**.

The system monitor shows what is happening at the station. This includes feedback on user commands. It logs the different things happening in the background. It also shows warnings, errors, and changes to the system. Learn more in the **System Monitor topic**.

Lua

```
return function()
    -- Dumps a list of all the hooks in the System Monitor.
    Printf("=============== START OF HOOK DUMP ===============")
    DumpAllHooks()
    Printf("================ END OF HOOK DUMP ================")
end
```

1.44.5.30. Echo(string)

Description

The **Echo** Lua function prints a string in the **System Monitor**.

Argument

- **String**:
  The string text to be printed to the System Monitor.

Return

This function does not return anything.

Example

This example prints "Hello World!" on the System Monitor:

```lua
return function()
    -- Prints 'Hellow World!' in the system monitor in yellow text.
    Echo("Hello World!")
end
```

1.44.5.31. ErrEcho(string)

Description

The **ErrEcho** Lua function prints a red error message on the **System Monitor**.

Argument

- **String**:
  The string text is to be printed to the System Monitor.

Return

This function does not return anything.

Example

This prints "This is a red error message!" on the system monitor:

```lua
return function()
    -- Prints an error message in the system monitor in red text.
    ErrEcho("This is an error message!")
end
```

1.44.5.32. ErrPrintf(string)

Description

The **ErrPrintf** Lua function prints a red error message in the **Command Line History** and **System Monitor**.

Argument

- **String**:
  The string text to be printed to the Command Line History.

Return

This function does not return anything.

Example

This example prints "This is a red error message!" in the Command Line History and System Monitor:

Lua

```lua
return function()
    -- Prints an error message in the command line feedback in red text.
    ErrPrintf("This is an error message!")
end
```

1.44.5.33. Export(filename, export_data)

Description

The object-free **Export** Lua function exports a Lua table in XML format.

This Lua function correlates with the **Import Lua function**.

There is a related object version of **Export**.

Arguments

- **Filename**:
  This is a string containing the file name of the exported file. It should contain the file name, including the entire path. See the example below.
- **Export_data**:
  This is the data that is going to be exported. It should be a table object.

Return

- **Boolean**:
  This function returns a boolean.
  - **True**:
    The export was a success
  - **False**:
    The export failed.

Example

To export the build details table, create a plugin with this code:

Lua

```
return function()
    -- 'BuildDetails()' creates a table with information about the software
build.
    local build = BuildDetails()
    --The path and filename is stored in a variable.
    local exportPath = GetPath(Enums.PathType.Library) .. "/BuildDetails.xml"
    --The actual export (in xml format) using the path and the table - the
result boolean stored in a variable.
    local success = Export(exportPath, build)
    --Print feedback about the export path.
    if success then
        Printf("The export was stored at: " .. exportPath)
    else
        Printf("The export failed")
    end
end
```

1.44.5.34. ExportCSV(filename, export_data)

Description

The object-free **ExportCSV** Lua function exports a Lua table in CSV format.

	Known Limitation:
	The output CSV file might not formatted correctly.

Arguments

- **Filename**:
  This is a string containing the file name of the exported file. It should contain the file name, including the entire path. See the example below.
- **Export_data**:
  This is the data that is going to be exported. It should be a table object.

Return

- **Boolean**:
  This function returns a boolean.
  - **True**:
    The export was a success.
  - **False**:
    The export failed.

Example

To export the build details table, create a plugin with this code:

```lua
return function()
    -- 'BuildDetails()' creates a table with information about the software build.
    local build = BuildDetails()
    --The path and filename is stored in a variable.
    local exportPath = GetPath(Enums.PathType.Library) .. "/BuildDetails.csv"
    --The actual export (in csv format) using the path and the table - the result boolean stored in a variable.
    local success = ExportCSV(exportPath, build)
    --Print feedback about the export path.
    if success then
        Printf("The export was stored at: " .. exportPath)
    else
        Printf("The export failed.")
    end
end
```

1.44.5.35. ExportJson(filename, export_data)

Description

The object-free **ExportJson** Lua function exports a Lua table in JSON format.

	Known Limitation:
	The JSON file might not be formatted in proper JSON format.

Arguments

- **Filename**:
  This is a string containing the file name of the exported file. It should contain the file name, including the entire path. See the example below.
- **Export_data**:
  This is the data that is going to be exported. It should be a table object.

Return

- **Boolean**:
  This function returns a boolean.
  - **True**:
    The export was a success.
  - **False**:
    The export failed.

Example

To export the build details table, create a plugin with this code:

Lua

```
return function()
    -- 'BuildDetails()' creates a table with information about the software
build.
    local build = BuildDetails()
    --The path and filename is stored in a variable.
    local exportPath = GetPath(Enums.PathType.Library) ..
"/BuildDetails.json"
    --The actual export (in JSON format) using the path and the table - the
result boolean stored in a variable.
    local success = ExportJson(exportPath, build)
    --Print feedback about the export path.
    if success then
        Printf("The export was stored at: " .. exportPath)
    else
        Printf("The export failed.")
    end
end
```

1.44.5.36. FileExists(string)

Description

The FileExists Lua function checks if a file exists and returns a boolean with the result.

Arguments

- **String**:
  The string must include the path and filename for the file that should be checked.

Return

- **Boolean**:
  - True / 1: The file exists.
  - False / 0: The file does not exist.

Example

This example returns feedback for the first file in the show file folder. The example uses the **GetPath()** and **DirList()** functions.

The **GetPath** Lua function delivers a string with the path of a grandMA3 folder.

Learn more in the **GetPath() topic**.

The **DirList** Lua function returns a table of files at a specified path.

Learn more in the **DirList() topic**.

Lua

```
return function ()
    -- Get the path to the show files.
    local path = GetPath(Enums.PathType.Showfiles)
    -- Get a table of files at the path.
    local dirTable = DirList(path)
    -- Get the file name for the first file.
    local firstFile = dirTable[1]['name']
    -- Create a string with the path and filename.
    local filepath = string.format("%s%s%s", path, GetPathSeparator(),
firstFile)

    -- Check if the file exist and return useful feedback.
    if FileExists(filepath) then
        Printf('The file "' .. firstFile .. '" exist at path "' .. path ..
'"')
    else
        Printf('The file "' .. firstFile .. '" does not exist')
    end
end
```

1.44.5.37. FindTexture(string)

Description

The **FindTixture** Lua function returns a handle to the texture matching the input text string - if the texture exists.

Arguments

- **String**:
  The text string must be the name of the texture without the file type. See the example below.

Return

- **Handle** | **nil**:
  The function returns the texture handle or nil if it does not exist.

Example

This example prints the information about the "button" texture. The example uses the **Dump() function**.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```lua
return function ()
    -- Set a texture name.
    local textureName = "button"
    -- Get the handle of the texture.
    local textureHandle = FindTexture(textureName)
    -- Check if textureHandle returned something and provide feedback.
    if textureHandle == nil then
        ErrPrintf("Texture does not exist.")
    else
        Printf("=============== START OF DUMP ===============")
        FindTexture(textureName):Dump()
        Printf("================ END OF DUMP ================")
    end
end
```

1.44.5.38. FirstDmxModeFixture(handle)

Description

The **FirstDmxModeFixture** Lua function returns a handle to the first fixture matching the supplied DMX mode.

Arguments

- **Handle**:
  This must be a handle to a DMX mode.

Return

- **Handle**:
  The returned handle to the first fixture matching the DMX mode.

Example

If it exists, this example prints the data connected to the first "Dimmer" fixture using "Mode 0" - if the fixture type exists in the show. It uses the **Dump()** functions:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- Get the handle for the Dimmer fixture.
    local fixtureTypeHandle = Patch().FixtureTypes['Dimmer']

    -- Check if fixture type returned something and provide feedback.
    if fixtureTypeHandle == nil then
        ErrPrintf("The fixture type does not exist in this show. Try adding
it or edit this plugin.")
    else
        -- Get the handle for the DMX mode of a Dimmer fixture.
        local fixtureDMXMode = fixtureTypeHandle.DMXModes["Mode 0"]

        -- Check if fixtureDMXMode returned something and provide feedback.
        if fixtureDMXMode == nil then
            ErrPrintf("The fixture type does not contain a 'Mode 0' DMX mode.
Try adding it or edit this plugin.")
        else
            -- Dumps information about the first fixture matching the DMX
mode.
            Printf("=============== START OF DUMP ===============")
```

```
            FirstDmxModeFixture(fixtureDMXMode):Dump()
            Printf("================ END OF DUMP ================")
        end
    end
end
```

1.44.5.39. FixtureType()

Description

The FixtureType Lua function returns a handle to the fixture type. The function does not accept any arguments, but the function must be executed when the command line destination is at a fixture type. If the command line destination is not a valid fixture type, then the function returns "nil".

Arguments

This function does not accept any arguments.

Return

- **Handle** or **nil**:
  The handle for the fixture type or nil.

Example

This example prints the information about the second fixture type in the show:

Lua

```
return function ()
    -- The function returns the handle to the fixture at the current command
line destination.
    -- Change to the "FixtureType" destination.
    Cmd("ChangeDestination FixtureType")
    -- Change to the second fixture type in the show.
    Cmd("ChangeDestination 2")
    -- Dump information about the Fixture Type handle.
    Printf("=============== START OF DUMP ===============")
    FixtureType():Dump()
    Printf("================ END OF DUMP ================")
    -- Return the command line destination to the Root.
    Cmd("ChangeDestination Root")
end
```

1.44.5.40. FromAddr(string[, handle])

Description

The **FromAddr** Lua function converts a numbered string address into a handle that can be used in commands.

Arguments

- **String**:
  A text string identifying an object. It can be a numbered or named address.
- **Handle** (optional):
  The default is to write the address from the root location. This optional handle can specify a different base location. It still needs to be a base location in the address path from the root to the object.

Return

- **Handle**:
  The handle for the addressed object.

Example

This example prints the address of the first sequence:

Lua

```
return function()
    -- Converts the string to a handle and store in a variabel.
    local mySequenceHandle = FromAddr("13.13.1.5.1")
    -- Converts the handle back to a numbered string and prints it.
    Printf("The address is: " ..mySequenceHandle:Addr())
    -- Converts the handle to a named string and prints it.
    Printf("The address is: " ..mySequenceHandle:AddrNative())

    -- Store the handle of the selected datapool.
    local myDataPool = DataPool()
    -- Prints the address of the selected datapool.
    Printf("The datapool address is: " ..myDataPool:Addr())

    --- The follwoing example uses the name of a sequence in the sequence
pool.
    --- Please adjust the "Default" name in the next line to match an
existing named sequence.
    -- Finds the address based on the base location and a text string with
names.
    local alsoMySequenceHandle = FromAddr("Sequences.Default", myDataPool)
    -- Converts the handle back to a numbered string and prints it.
    Printf("The address is: " ..alsoMySequenceHandle:Addr())
    -- Converts the handle to a named string and prints it.
    Printf("The address is: " ..alsoMySequenceHandle:AddrNative())
end
```

1.44.5.41. GetApiDescriptor()

Description

The **GetApiDescriptor** Lua function returns a table with a description of all the object-free Lua functions. These are descriptions only. The function does not have any actual functions. The table is not sorted.

Arguments

This function does not accept any arguments.

Return

- **Table**:
  The returned table contains elements with other tables.
  - **String**:
    This is the Api function name.
  - **String**:
    This is the description of the Api arguments.
  - **String**:
    This is the description of the Api returns.

Example

This example prints the content of the returned table.

Lua

```
return function ()
    -- This returns information about all the Lua "object-free" functions.
    -- GetApiDescriptor() returns a table with all the functions.
    -- Each table element is another table with the name, argument
description, and return description.
      for key,value in ipairs(GetApiDescriptor()) do
        if value[1] ~= nil then
          Printf("Api " .. key .. " is: " .. value[1])
        end
        if value[2] ~= nil then
          Printf("Arguments: " .. value[2])
        end
        if value[3] ~= nil then
          Printf("Returns: " .. value[3])
        end
        Printf("---------------------------------------")
      end
  end
```

1.44.5.42. GetAttributeByUIChannel(integer)

Description

The **GetAttributeByUIChannel** Lua function returns the handle to an attribute based on a "UI Channel Index". The index number can be found in the **Parameter List**.

Arguments

- **Integer**:
  The integer is the UI Channel index number.

Return

- **Handle**:
  The handle to the attribute connected to the UI Channel.

Example

This example prints the "native" address to the first attribute of the first fixture in the current selection:

```lua
return function()
    -- Get a handle to the first fixture in the current selection
    local fixtureIndex = SelectionFirst()
    -- Get the UI Channel Index number for the first attribute for the
fixture
    local channelIndex = GetUIChannelIndex(fixtureIndex,0)
    -- Print the native address for the attribute with the handle
    Printf("The native addr for the attribute is:
%s",GetAttributeByUIChannel(channelIndex):AddrNative())
end
```

1.44.5.43. GetAttributeCount()

Description

The **GetAttributeCount** Lua function returns the total number of attribute definitions in the show.

Arguments

This function does not accept any arguments.

Return

- **Integer**:
  The returned integer number represents the total amount of attribute definitions in the show file.

Example

This example prints the returned number in the Command Line History.

```Lua
return function()
    Printf("Attribute count is %i", GetAttributeCount())
end
```

1.44.5.44. GetAttributeIndex(string)

Description

The **GetAttributeIndex** Lua function returns the (0 based) index number of the attribute definition based on the system name of the attribute.

Arguments

- **String**:
  The string text of the attribute system name.

Return

- **Integer**:
  The returned integer number represents the total amount of patched fixtures on all the stages in the show file.

Example

This example prints the index number of the attribute in Command Line History if it exists:

Lua

```
return function()
    -- store the returned index or nil of "Gobo1"
    local attributeIndex = GetAttributeIndex("Gobo1")
    -- Check if the returned value is not nil and print a useful feedback
    if attributeIndex~=nil then
        Printf("Attribute is index number %i", attributeIndex)
    else
        Printf("The attribute is not found")
    end
end
```

1.44.5.45. GetButton(handle)

Description

The **GetButton** Lua function returns a key-value pairs table indicating, with a boolean value, whether a button is pressed on an MA3Module.

Below the example is a table listing all the grandMA3 hardware modules and which index number matches which button on the hardware module.

Arguments

- **Handle:**
  The handle for the MA3 module.

Return

- **Table**:
  The returned table is a key-value pairs table with a set of 512 pairs with a boolean value. A **true** boolean value indicates that the button is pressed or the fader is touched. The table key is 1-indexed.

Example

This example requests the buttons states on the master module on a grandMA3 full-size console:

Lua

```
return function()
    --- grandMA3 full-size modules are:
    --- Master Module (MM): "UsbDeviceMA3 2"
    --- Fader Module Encoder (MFE): "UsbDeviceMA3 3"
    --- Fader Module Crossfader (MFX): "UsbDeviceMA3 4"

    -- Get a handle to the Master Module on a grandMA3 full-size.
    local usbDeviceHandle = Root().UsbNotifier.MA3Modules["UsbDeviceMA3 2"]
    -- Create a table with the button status.
    local buttonTable = GetButton(usbDeviceHandle)
    -- Check if the table is nil and then print an error.
    if buttonTable == nil then
        ErrPrintf("nil")
        return
    end
    -- If the table is not nil, then print a usefull feedback about pressed
buttons.
    for key,value in pairs(buttonTable) do
        if tostring(value) == "true" then
            Printf("The button with the index " .. key .. " is pressed.")
        end
    end
end
```

Hardware Modules Button Table

This table provides a list of index numbers (0-indexed and 1-indexed). The numbers are matched with elements on the three different kinds of hardware modules. Notice that the naming here comes from the internal hardware definition and might not match exactly the print on the keys or the official name.

0-Index	1-Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
0	1			
1	2			
2	3	ENCODER_INSIDE4	EXEC_108	EXEC_108
3	4	ENCODER_OUTSIDE3	EXEC_110	EXEC_110
4	5			
5	6			
6	7			
7	8			
8	9			
9	10	ENCODER_INSIDE2	EXEC_211	EXEC_211
10	11	EXEC_GrandKnob	EXEC_212	EXEC_212
11	12	MENU	EXEC_213	EXEC_213
12	13		EXEC_214	EXEC_214
13	14	ENCODER_OUTSIDE2	EXEC_215	EXEC_215
14	15	ENCODER_INSIDE1		
15	16	ENCODER_OUTSIDE1		
16	17		EXEC_209	EXEC_209
17	18		EXEC_210	EXEC_210
18	19		EXEC_208	EXEC_208
19	20			
20	21			
21	22			DEF_GO
22	23		EXEC_115	EXEC_115
23	24		EXEC_114	EXEC_114
24	25		EXEC_113	EXEC_113
25	26		EXEC_112	EXEC_112
26	27		EXEC_207	EXEC_207
27	28		EXEC_206	EXEC_206
28	29		EXEC_205	EXEC_205
29	30			
30	31			
31	32			
32	33		EXEC_105	EXEC_105
33	34	ENCODER_OUTSIDE4	EXEC_106	EXEC_106
34	35		EXEC_107	EXEC_107
35	36			
36	37			
37	38		EXEC_109	EXEC_109
38	39		EXEC_204	EXEC_204
39	40		EXEC_203	EXEC_203
40	41		EXEC_202	EXEC_202
41	42		EXEC_201	EXEC_201
42	43			EXEC_XFade2Btn

0-Index	1-Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
43	44			DEF_PAUSE
44	45			EXEC_XFade1Btn
45	46		EXEC_111	EXEC_111
46	47	ENCODER_OUTSIDE5	EXEC_101	EXEC_101
47	48	ENCODER_INSIDE5	EXEC_102	EXEC_102
48	49		EXEC_103	EXEC_103
49	50		EXEC_104	EXEC_104
50	51	ENCODER_INSIDE3		DEF_GOBACK
51	52			
52	53			
53	54			
54	55			
55	56			
56	57			
57	58			
58	59		FADER_211	FADER_211
59	60		FADER_212	FADER_212
60	61		FADER_213	FADER_213
61	62		FADER_214	FADER_214
62	63		FADER_215	FADER_215
63	64			FADER_XFade1
64	65			
65	66			
66	67	ESC	FADER_312 (Disabled)	FADER_312 (Disabled)
67	68	CLEAR	FADER_311 (Disabled)	FADER_311 (Disabled)
68	69			
69	70			
70	71			
71	72			
72	73			
73	74	HELP	FADER_413 (Disabled)	FADER_413 (Disabled)
74	75		EXEC_411	EXEC_411
75	76	GOTO	EXEC_412	EXEC_412
76	77			
77	78	ALIGN	FADER_414 (Disabled)	FADER_414 (Disabled)
78	79	COPY	FADER_411 (Disabled)	FADER_411 (Disabled)
79	80	OFF	FADER_412 (Disabled)	FADER_412 (Disabled)
80	81	FULL	EXEC_415	EXEC_415
81	82		EXEC_414	EXEC_414
82	83	PLEASE	EXEC_413	EXEC_413
83	84	NUM4		
84	85	NUM5		
85	86	UNDO		
86	87	GROUP		
87	88	THRU		
88	89	NUM6		
89	90	NUM2		
90	91			
91	92	STORE		
92	93	ASSIGN		

0-Index	1-Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
93	94			
94	95			
95	96			
96	97	AT	EXEC_312	EXEC_312
97	98	MA1	FADER_313 (Disabled)	FADER_313 (Disabled)
98	99	SLASH	EXEC_311	EXEC_311
99	100	NUM1		
100	101	CUE		
101	102	TIME		
102	103			
103	104	SEQUENCE		
104	105			
105	106	CHANNEL		
106	107	NUM7		
107	108	NUM8		
108	109	NUM9		
109	110	NUM3	EXEC_313	EXEC_313
110	111	MINUS	FADER_315 (Disabled)	FADER_315 (Disabled)
111	112	NUM0	FADER_314 (Disabled)	FADER_314 (Disabled)
112	113	DOT	EXEC_314	EXEC_314
113	114	IF	EXEC_315	EXEC_315
114	115	PLUS	FADER_415 (Disabled)	FADER_415 (Disabled)
115	116			
116	117			
117	118			
118	119			
119	120			
120	121		FADER_209	FADER_209
121	122		FADER_210	FADER_210
122	123			FADER_XFade2
123	124			
124	125			
125	126			
126	127			
127	128			
128	129			
129	130			
130	131		FADER_307 (Disabled)	FADER_307 (Disabled)
131	132	LEARN	FADER_306 (Disabled)	FADER_306 (Disabled)
132	133			
133	134			
134	135			
135	136			
136	137			
137	138	FADER_297 (Disabled)	FADER_408 (Disabled)	FADER_408 (Disabled)
138	139	X5	EXEC_406	EXEC_406
139	140	X6	EXEC_407	EXEC_407
140	141			
141	142	FADER_298 (Disabled)	FADER_409 (Disabled)	FADER_409 (Disabled)
142	143	FADER_295 (Disabled)	FADER_406 (Disabled)	FADER_406 (Disabled)

0-Index	1-Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
143	144	FADER_296 (Disabled)	FADER_407 (Disabled)	FADER_407 (Disabled)
144	145	GOFAST	EXEC_410	EXEC_410
145	146	X13	EXEC_409	EXEC_409
146	147	X14	EXEC_408	EXEC_408
147	148			
148	149			
149	150			
150	151			
151	152			
152	153			
153	154	DELETE		
154	155	X15		
155	156	GOBACKFAST		
156	157	X8		
157	158			
158	159			
159	160			
160	161	STOMP	EXEC_307	EXEC_307
161	162	SELECT	FADER_308 (Disabled)	FADER_308 (Disabled)
162	163		EXEC_306	EXEC_306
163	164			
164	165			
165	166			
166	167			
167	168			
168	169	X16		
169	170	X7		
170	171			
171	172	ON		
172	173	MOVE		
173	174	FIXTURE	EXEC_308	EXEC_308
174	175	PRESET	FADER_310 (Disabled)	FADER_310 (Disabled)
175	176	EDIT	FADER_309 (Disabled)	FADER_309 (Disabled)
176	177	UPDATE	EXEC_309	EXEC_309
177	178		EXEC_310	EXEC_310
178	179		FADER_410 (Disabled)	FADER_410 (Disabled)
179	180			
180	181			
181	182			
182	183			
183	184			
184	185			
185	186			
186	187		FADER_203	FADER_203
187	188		FADER_204	FADER_204
188	189		FADER_205	FADER_205
189	190		FADER_206	FADER_206
190	191		FADER_207	FADER_207
191	192		FADER_208	FADER_208
192	193			

0-Index	1-Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
193	194			
194	195	PAUSE	FADER_302 (Disabled)	FADER_302 (Disabled)
195	196	GOBACK	FADER_301 (Disabled)	FADER_301 (Disabled)
196	197			
197	198			
198	199			
199	200			
200	201			
201	202	FADER_293 (Disabled)	FADER_403 (Disabled)	FADER_403 (Disabled)
202	203	SOLO	EXEC_401	EXEC_401
203	204	HIGHLIGHT	EXEC_402	EXEC_402
204	205			
205	206	FADER_294 (Disabled)	FADER_404 (Disabled)	FADER_404 (Disabled)
206	207	FADER_291 (Disabled)	FADER_401 (Disabled)	FADER_401 (Disabled)
207	208	FADER_292 (Disabled)	FADER_402 (Disabled)	FADER_402 (Disabled)
208	209	GO	EXEC_405	EXEC_405
209	210	LIST	EXEC_404	EXEC_404
210	211	PAGE_DOWN	EXEC_403	EXEC_403
211	212			
212	213			
213	214			
214	215			
215	216	SELFIX		
216	217	MA2		
217	218			
218	219	PAGE_UP		
219	220	XKEYS		
220	221	BLIND		
221	222			
222	223			
223	224			
224	225	X10	EXEC_302	EXEC_302
225	226	X11	FADER_303 (Disabled)	FADER_303 (Disabled)
226	227	X12	EXEC_301	EXEC_301
227	228			
228	229			
229	230			
230	231			
231	232	PREVIEW		
232	233	FREEZE		
233	234	DOWN		
234	235	PREV		
235	236	SET		
236	237	UP		
237	238	X1	EXEC_303	EXEC_303
238	239	X2	FADER_305 (Disabled)	FADER_305 (Disabled)
239	240	X3	FADER_304 (Disabled)	FADER_304 (Disabled)
240	241	X4	EXEC_304	EXEC_304
241	242	X9	EXEC_305	EXEC_305
242	243	NEXT	FADER_405 (Disabled)	FADER_405 (Disabled)

0-Index	1-Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
243	244			
244	245			
245	246			
246	247			
247	248			
248	249		FADER_201	FADER_201
249	250		FADER_202	FADER_202
250	251			
251	252			
252	253			
253	254			
254	255			
255	256			
256	257			
257	258			
258	259		EXEC_RateBtn2	
259	260		EXEC_ExecBtn1	
260	261			
261	262			
262	263			
263	264			
264	265			
265	266			
266	267			
267	268			
268	269			
269	270			
270	271		EXEC_ProgEncoder	
271	272		EXEC_ExecEncoder	
272	273			
273	274			
274	275			
275	276			
276	277			
277	278			
278	279			
279	280			
280	281			
281	282			
282	283			
283	284			
284	285			
285	286			
286	287			
287	288			
288	289		EXEC_SpeedBtn1	
289	290		EXEC_RateBtn1	
290	291		EXEC_SpeedBtn2	
291	292			
292	293			

0-Index	1-Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
293	294			
294	295			
295	296			
296	297			
297	298			
298	299		EXEC_ProgBtn1	
299	300		EXEC_ProgBtn2	
300	301		EXEC_ProgBtn3	
301	302			
302	303			
303	304			
304	305		EXEC_ExecBtn3	
305	306		EXEC_ExecBtn2	
306	307			
307	308			
308	309			
309	310			

1.44.5.46. GetChannelFunction(integer, integer)

Description

The **GetChannelFunction** Lua function returns a handle to a channel function based on two index inputs.

Arguments

- **Integer**:
  The first integer is a UI Channel Index. This can be found in the **Parameter List** or by the **GetUIChannelIndex()** Lua function.
- **Integer**:
  This integer is an Attribute Index (0-based). This can be found in the **Attribute Definitions** or by the **GetAttributeIndex()** Lua function.

Return

- **Handle**:
  The returned handle to the channel function.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Select the first fixture in the current selection.
    local subfixtureIndex = SelectionFirst()
    -- End the function if there is no selection.
    if subfixtureIndex == nil then
        ErrPrintf("Please select a fixture with a Dimmer")
        return
    end
    -- Get the Attribute index and UIChannel index.
    local attributeIndex = GetAttributeIndex("Dimmer")
    local uiChannelIndex = GetUIChannelIndex(subfixtureIndex,attributeIndex)
    Printf("The UIChannel Index is: %i. The Attribute Index is: %i.
",uiChannelIndex, attributeIndex)
    -- End the function if any of the index return nil.
    if (attributeIndex == nil or uiChannelIndex == nil) then
        ErrPrintf("Something wrong happened, maybe your first selected
fixture don't have a Dimmer - Please try again")
```

```
        return
    end
    -- The following prints the dump for the dimmer channel function.
    Printf("================ START OF DUMP ================")
    GetChannelFunction(uiChannelIndex,attributeIndex):Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.47. GetChannelFunctionIndex()

Description

The **GetChannelFunctionIndex** Lua function returns the integer matching a channel function based on two index inputs.

Arguments

- **Integer**:
  The first integer is a UI Channel Index. This can be found in the **Parameter List** or by the **GetUIChannelIndex()** Lua function.
- **Integer**:
  This integer is an Attribute Index (0-based). This can be found in the **Attribute Definitions** or by the **GetAttributeIndex()** Lua function.

Return

- **Integer**:
  The returned integer to a channel function.

Example

This example prints the indexes based on the fixture selection and the "Dimmer" attribute.

Lua

```
return function()
    -- Get the Attribute index and UIChannel index.
    local attributeIndex = GetAttributeIndex("Dimmer")
    local uiChannelIndex = GetUIChannelIndex(SelectionFirst(),attributeIndex)
    -- End the function if any of the index return nil.
    if (attributeIndex == nil or uiChannelIndex == nil) then
        ErrPrintf("Something wrong happened, maybe your first selected
fixture don't have a Dimmer - Please try again")
        return
    end
    -- Get the Channel Function Index and store it in a variable.
    local channelFunctionIndex =
GetChannelFunctionIndex(uiChannelIndex,attributeIndex)
    Printf("The UIChannel Index is: %i. The Attribute Index is: %i. The
Channel Function Index is: %i", uiChannelIndex, attributeIndex,
channelFunctionIndex)
end
```

1.44.5.48. GetClassDerivationLevel(string)

Description

The **GetClassDerivationLevel** Lua function returns an integer indicating the derivation level index for a class based on a class name.

Arguments

- **String**:
  This string needs to be the name of a class.

Return

- **Integer**:
  The returned integer indicates the class derivation level.

Example

This example prints the index integer for the Pool class in the Command Line History:

```lua
return function()
    -- Get the index integer for the "Pool" class.
    local classDerivationLevel = GetClassDerivationLevel("Pool")
    -- Create a valid Printf return.
    if classDerivationLevel == nil then
        Printf("The return is nil")
    else
        Printf("The ClassDerivationLevel index for 'Pool' is: %i",
classDerivationLevel)
    end
end
```

1.44.5.49. GetCurrentCue()

Description

The **GetCurrentCue** Lua function returns a handle to the last activated cue in the selected sequence.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the cue.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Dumps information about the last activated cue in the selected
sequence
    Printf("=============== START OF DUMP ===============")
    GetCurrentCue():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.50. GetDebugFPS()

Description

The **GetDebugFPS** Lua function returns a float number with the frames per second.

Arguments

This function does not accept any arguments.

Return

- **Number**:
  The returned number indicates the current frames per second.

Example

This example prints the FPS number:

```Lua
return function ()
    -- Prints the current frames per second.
    Printf("Current FPS: " .. GetDebugFPS())
end
```

1.44.5.51. GetDisplayByIndex(integer)

Description

The **GetDisplayByIndex** Lua function returns a handle to the display object matching the provided index number.

Arguments

- **Integer**:
  This function needs an index number for one of the displays.

Return

- **Handle**:
  The returned handle to the display object.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Get the index number for "Display 1"
    local displayIndex = GetDisplayCollect()["Display 1"].INDEX
    -- return error text in case the index number is nil
    if displayIndex == nil then
        ErrPrintf('Something went wrong. It appears that there is no "display
1"')
        return
    end
    -- Dump all information about the display with the index number
    Printf("=============== START OF DUMP ===============")
    GetDisplayByIndex(displayIndex):Dump()
    Printf("================ END OF DUMP ================")
end
```

Hint:

The example uses the **GetDisplayCollect() function** to get the index number. The displays are children of the Display Collect, and this function can be used to access the same information using:

GetDisplayCollect()

	The **GetDisplayCollect** Lua function returns a handle to the DisplayCollect object. Learn more in the **GetDisplayCollect() topic**. GetDisplayCollect()["Display 1"]

1.44.5.52. GetDisplayCollect()

Description

The **GetDisplayCollect** Lua function returns a handle to the DisplayCollect object.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the DisplayCollect.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

```lua
return function()
    -- This example dumps all information about the DisplayCollect object.
    Printf("=============== START OF DUMP ===============")
    GetDisplayCollect():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.53. GetDMXUniverse(integer[, boolean])

Description

The **GetDMXUniverse** Lua function returns a table with the DMX channels and their current value.

Arguments

- **Integer**:
  The integer is the universe number. The valid range is 1 to 1024.
- **Boolean** (optional):
  The boolean indicates if the returned value is in percent or DMX value.
  - True:
    The returned value is in percent. The range is 0 to 100.
  - False:
    The returned value is in DMX value. The range is 0 to 255.

Return

- **Table**:
  The returned table lists all the DMX addresses and the corresponding values.

-- OR --

- **Nil**:
  Nil is returned if the universe is not granted or the input value is out of range.

Example

This example prints the table in a list for DMX universe 1 (if it is granted):

Lua

```
return function()
    -- This gets a table for universe 1 with the returned value in percent.
    local tableDMXUniverse = GetDMXUniverse(1,true)
    -- Check the returned table and print information if nil.
    if tableDMXUniverse == nil then
        Printf("No value is returned. The univer is not granted or input is
out of range")
        return
    end
    -- Prints the table if not nil.
    for addr, value in ipairs(tableDMXUniverse) do
        Printf("DMX Addr: %i - DMX value : %i", addr, value)
    end
end
```

1.44.5.54. GetDMXValue(integer[, integer, boolean])

Description

The **GetDMXValue** Lua function returns a number indicating the DMX value of a specified DMX address.

Arguments

- **Integer**:
  The integer is the DMX address. This value should be from 1 to 512 if a universe integer is provided. If a universe is not provided, this should be the absolute DMX address ranging from 1 to 524 288.
- **Integer** (optional):
  The integer is the universe number.
- **Boolean** (optional):
  The boolean indicates if the returned value is in percent or DMX value.
  - True:
    The returned value is in percent. The range is 0 to 100.
  - False:
    The returned value is in DMX value. The range is 0 to 255.

Return

- **Integer** or **nil**:
  The returned integer value corresponds with the value of the selected DMX address or nil if the DMX address is not granted.

Example

This example prints the value for DMX address 1 in Universe 4 (if it is granted):

Lua

```
return function()
    -- This prints the value of DMX address 1 in universe 4 in a range of 0
to 255
    local address = 1       -- The DMX address
    local universe = 4      -- The DMX universe
    local percent = false   -- Readout in percent or DMX value
    local value = GetDMXValue(address, universe, percent)
    Printf("DMX address %i.%03d is %03d", universe, address, value)
end
```

1.44.5.55. GetExecutor(integer)

Description

The **GetExecutor** Lua function returns the handles of the executor and the page based on the executor number.

Arguments

- **Integer**:
  The integer number for the executor.

Return

- **Handle - Executor**:
  The returned handle to the executor.
- **Handle - Page**:
  The returned handle to the page.

Example

This example stores the handles for executor number 201. It then uses the **Dump() function** to show the data for the two handles.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- This saves the handles for executor 201 on the selected page.
    local executorHandle, pageHandle = GetExecutor(201)
    -- exit the function and print an error message if any of the handles are
nil.
    if executorHandle == nil or pageHandle == nil then
        ErrPrintf("There is not a valid object on executor 201, please assign
something and try again.")
        return
    end
    -- The following prints the dumps of the two handles.
    Printf("============ START OF EXEC DUMP =============")
    executorHandle:Dump()
    Printf("================ END OF DUMP ================")
    Printf("============ START OF PAGE DUMP =============")
    pageHandle:Dump()
    Printf("================ END OF DUMP ================")
```

```
end
```

1.44.5.56. GetFocus()

Description

The **GetFocus** Lua function returns a handle to the object that currently has focus in the UI.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the object.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- This example dumps all information about the object who currently got
focus.
    Printf("=============== START OF DUMP ===============")
    GetFocus():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.57. GetFocusDisplay()

Description

The **GetFocusDisplay** Lua function returns a handle to the display object that currently has focus in the UI.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the display object.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- This example dumps all information about the display object who
currently got focus.
    Printf("================ START OF DUMP ================")
    GetFocusDisplay():Dump()
    Printf("================= END OF DUMP =================")
end
```

1.44.5.58. GetObjApiDescriptor()

Description

The **GetObjApiDescriptor** Lua function returns a table with a description of all the object Lua functions. These are descriptions only. The function does not have any actual functions. The table is not sorted.

Arguments

This function does not accept any arguments.

Return

- **Table**:
  The returned table contains elements with three values.
  - **String**:
    This is the API function name.
  - **String**:
    This is the description of the API arguments.
  - **String**:
    This is the description of the API returns.

Example

This example prints the content of the returned table.

Lua

```
return function ()
  -- This returns information about all the Lua "object" functions.
  -- GetObjApiDescriptor() returns a table with all the functions.
  -- Each table element is another table with the name, argument description,
and return description.
    for key,value in ipairs(GetObjApiDescriptor()) do
      if value[1] ~= nil then
        Printf("Api " .. key .. " is: " .. value[1])
      end
      if value[2] ~= nil then
        Printf("Arguments: " .. value[2])
      end
      if value[3] ~= nil then
        Printf("Returns: " .. value[3])
      end
      Printf("---------------------------------------")
      end
end
```

1.44.5.59. GetPath(string[, boolean] | integer)

Description

The **GetPath** Lua function returns a string with the path of a grandMA3 folder.

The function has two possible argument types - use one of them with each function call.

Argument

- **String**:
  A text string with the folder name.
- **Boolean** (optional with string):
  If this boolean is true, then the folder at the path is created if it does not exist.

  - OR -

- **Integer**:
  An integer identifying an index in the "Enum.PathType" table.

	Restriction:
	Folder creation only works with string arguments.

Return

- **String**:
  The returned string is the *first found* full path related to the provided argument.

Example

This example prints the paths of the show folder on the system monitor twice. It demonstrates the two different input types:

Lua

```
return function()
    -- This prints a path based on a string input and it creates the folder
if it does not exists.
    Printf("Path of show files (string) is: " .. GetPath("shows", true))
    -- This prints the path based on an integer. The integer is looked-up
using the 'PathType' enum.
    Printf("Path of show files (integer) is: " ..
GetPath(Enums.PathType.Showfiles))
end
```

1.44.5.60. GetPathOverrideFor(string|integer, string[, boolean])

Description

The **GetPathOverrideFor** Lua function delivers a string with the path of a grandMA3 folder. The function is relevant when the path should be on a removable drive connected to a console.

Argument

- **String**:
  A text string with the folder name.
- **String**:
  The base path in a string format.
- **Boolean** (optional with string):
  If this boolean is true, then the folder at the path is created if it does not exist.

  - OR -

- **Integer**:
  An integer identifying an index in the "Enum.PathType" table.
- **String**:
  The base path in a string format.
- **Boolean**:
  If this boolean is true, then the folder at the path is created if it does not exist.

Return

- **String**:
  The returned string is the *first found* full path related to the provided argument.

Example

This example prints the override path of the macro folder on the system monitor. It should be run on a console with a removable drive connected.

Lua

```
return function ()
    -- Set a path for the first removable media.
    -- Set the initial value to nil.
    local myBasePath = nil
    -- Itereate the drives and find the first 'Removeable' drive and store
the path.
    for _, value in ipairs(Root().Temp.DriveCollect) do
        local driveType = value.drivetype
        if driveType == "Removeable" then
            myBasePath = value.path
            break
        end
    end
    -- If no removeable drive was found, then provide feedback and exit the
```

```
function.
    if myBasePath == nil then
        ErrPrintf("No removeable drive could be found. Please insert one and
try again")
        return
    end

    -- Get the integer for the UserMacros path type.
    local myPathType = Enums.PathType.UserMacros

    -- Gey the string for the path override.
    local myOverridePath = GetPathOverrideFor(myPathType, myBasePath)
    -- Print the returned string.
    Printf("The path is: " .. myOverridePath)
end
```

1.44.5.61. GetPathSeparator()

Description

The **GetPathSeparator** function returns a string with the path separator for the operating system.

Arguments

This function does not accept any arguments.

Return

- **String**:
  The string is a single character indicating the path separator based on the operating system.

Example

This example prints the path separator:

```lua
return function()
    --- This prints the path seperator. It is different between a Linux and macOS (/) and a Windows (\) operating system.
    Printf("The path seperator is " .. GetPathSeparator())
end
```

1.44.5.62. GetPathType(handle[, integer])

Description

The **GetPathType** Lua function returns a string with a name for the path type. This function can be useful when importing objects.

Argument

- **Handle**:
  The handle should match the object type for which the path type is needed.
- **Integer** (optional):
  The optional integer can be used to specify if the returned string should match the user path type or the system path type. See the example below.
  The Enums.PathContentType can be used, or just use **0** for the system path and **1** for the user path.

Return

- **String**:
  The returned string is the name of the path type.

Example

This example prints the path type name for the first macro object - if it exists:

Lua

```
return function ()
    -- Get a handle to the first Macro.
    local myMacro = DataPool().Macros[1]
    if myMacro == nil then
        ErrPrintf("An error occurred, possibly because the first macro does
not exist.")
        ErrPrintf("Please create one and try again.")
        return
    end
    -- Get the user name of the path type.
    local myPathTypeNameUser = GetPathType(myMacro,
Enums.PathContentType.User)
    if myPathTypeNameUser ~= nil then
        Printf("The user name of the path type is: " .. myPathTypeNameUser)
    else
        ErrPrintf("There was an error getting the path type.")
    end

    -- Get the system name of the path type.
    local myPathTypeNameSystem = GetPathType(myMacro,
Enums.PathContentType.System)
    if myPathTypeNameSystem ~= nil then
        Printf("The system name of the path type is: " ..
myPathTypeNameSystem)
    else
```

```
        ErrPrintf("There was an error getting the path type.")
    end
end
```

1.44.5.63. GetRTChannel(integer)

Description

The **GetRTChannel** Lua function returns a table with information about the related RT Channel.

Arguments

- **Integer**:
  The integer should be the index number for an RT Channel.

Return

- **Table**:
  The returned table contains related numbers, tables, and handles with a named identifying key:
  - handle "fixture"
  - handle "subfixture"
  - handle "dmx_channel
  - integer "dmx_default"
  - integer "dmx_highlight"
  - integer "dmx_lowlight"
  - integer "ui_index_first"
  - integer "rt_index"
  - integer "freq"
  - table "info"
  - table "patch"

Example

This example prints all information related to the first RT Channel for the first fixture in the selection:

Lua

```
return function()
    -- Get the index number for the first RT Channel for the first fixture in
the current selection
    local channelRTIndex = GetRTChannels(SelectionFirst())[1]
    -- Print an error message if returnd index is nil
    if channelRTIndex == nil then
        ErrPrintf("Please select a fixture and try again")
        return
    end
    -- Print all information about the RT Channel if it does not return nil
    local rtChannel = GetRTChannel(channelRTIndex)
    if rtChannel == nil then
        Printf("An RTChannel could not be found. Please try to selct a
different fixture and try again.")
        return
    end
    Printf("================ RT CHANNEL ================")
    Printf("ui_index_first = " .. rtChannel["ui_index_first"])
    Printf("dmx_lowlight = " .. rtChannel["dmx_lowlight"])
```

```
    Printf("dmx_highlight = " .. rtChannel["dmx_highlight"])
    Printf("dmx_default = " .. rtChannel["dmx_default"])
    Printf("freq = " .. rtChannel["freq"])
    Printf("rt_index = " .. rtChannel["rt_index"])
    Printf("========== RELATED DMX CHANNEL DUMP ==========")
    rtChannel["dmx_channel"]:Dump() -- Handle for relevant DMX channel
    Printf("============ RELATED FIXTURE DUMP ============")
    rtChannel["fixture"]:Dump() -- Handle for relevant fixture
    Printf("========== RELATED SUBFIXTURE DUMP ===========")
    rtChannel["subfixture"]:Dump() -- Handle for relevant subfixture
    Printf("==================== INFO ====================")
    Printf("normed_phaser_time = " ..
rtChannel["info"]["normed_phaser_time"])
    Printf("================ INFO FLAGS ==================")
    Printf("group_master = " .. rtChannel["info"]["flags"]["group_master"])
    Printf("additive_master = " ..
rtChannel["info"]["flags"]["additive_master"])
    Printf("solo = " .. rtChannel["info"]["flags"]["solo"])
    Printf("highlight = " .. rtChannel["info"]["flags"]["highlight"])
    Printf("lowlight = " .. rtChannel["info"]["flags"]["lowlight"])
    Printf("=================== PATCH ====================")
    Printf("break = " .. rtChannel["patch"]["break"])
    Printf("coarse = " .. rtChannel["patch"]["coarse"])
    Printf("fine = " .. rtChannel["patch"]["fine"])
    Printf("ultra = " .. rtChannel["patch"]["ultra"])
end
```

1.44.5.64. GetPresetData(handle[, boolean[, boolean]])

Description

The **GetPresetData** Lua function returns a table with the preset data based on the preset handle.

The returned table is quite complex and has tables inside the table.

Arguments

- **Handle**:
  The handle of the preset from which the data will be collected.
- **Boolean** | **nil** (optional):
  This boolean determines whether the returned table should only contain phaser data. The default value is "false".
- **Boolean** (optional):
  This boolean defines if there should be an extra object in the returned table. The default value is "true". The extra table object has the key "by_fixtures", and it contains the same table content as the returned table, but the keys are the fixture ID number instead of the UI Channel Index.

Return

- **Table** | **nil**:
  The returned table contains the preset data. It has multiple levels of tables.

Example

This example prints information about the first level table in the preset data and the first level of the first fixture in the preset. It uses dimmer preset 1, which must exist.

Lua

```
return function()
    -- Get the handle for the first Dimmer preset.
    local myPreset = DataPool().PresetPools[1][1]
    -- Get the Preset Data of the handle.
    local myPresetData = GetPresetData(myPreset, false, false)
    -- Check if the GetPresetData returns something.
    if myPresetData == nil then
        ErrPrintf("Dimmer preset 1 does not exist. Please create one and try
again.")
        return
    end

    -- Print the myPresetData table.
    for Key, value in pairs(myPresetData) do
      if type(value) == "table" then
        Printf("Key: " .. Key .. " ; Value type is: " .. type(value))
      else
        Printf("Key: " .. Key .. " ; Value type is: " .. type(value) .. " ;
Value: " .. value)
      end
```

```
    end

    -- Create a table object to hold all the integer keys in the myPresetData
table.
    local myIntegerTableKeys = {}
    -- Fill the table.
    for key,_ in pairs(myPresetData) do
        if type(key) == "number" then
            table.insert(myIntegerTableKeys, key)
        end
    end
    -- Sort the table
    table.sort(myIntegerTableKeys)

    -- Print the elements of the fixture with the lowest ui_channel_index in
the preset.
    local tableIndex = myIntegerTableKeys[1]
    if tableIndex ~= nil then
        Printf("=============== TABLE CONTENT START - Table Key: " ..
tableIndex .." ===============")
        for Key, value in pairs(myPresetData[tableIndex]) do
          if type(value) == "table" then
            Printf("Key: " .. Key .. " ; Value type is: " .. type(value))
          else
            Printf("Key: " .. Key .. " ; Value type is: " .. type(value) .. "
; Value: " .. tostring(value))
          end
        end
        Printf("================ TABLE CONTENT END - Table Key: " ..
tableIndex .." ================")
    end
end
```

1.44.5.65. GetRTChannelCount()

Description

The **GetRTChannelCount** Lua function returns a number indicating the total amount of RT channels.

Arguments

This function does not accept any arguments.

Return

- **Integer**:
  The function returns an integer number depicting the total amount of RT channels.

Example

This example prints the number of RT channels to the Command Line History:

```Lua
return function()
    Printf("The number of RT channels is " .. GetRTChannelCount())
end
```

1.44.5.66. GetRTChannels(integer[,boolean] OR handle[,boolean])

Description

The **GetRTChannels** Lua function returns a table with RT Channel indexes or a table with handles to the RT Channel objects. There are two different types of arguments for this function.

Arguments

- **Integer**:
  The integer should be the index number for a (sub)fixture.
- **Boolean** (Optional):
  - **True**:
    The returned table contains handles for RT Channel objects.
  - **False** (default):
    The returned table contains integers index values to the RT Channel objects.

- OR -

- **Handle**:
  The handle should relate to a (sub)fixture object.
- **Boolean** (Optional):
  - **True**:
    The returned table contains handles for RT Channel objects.
  - **False** (default):
    The returned table contains integers index values to the RT Channel objects.

Return

- **Table**:
  The returned table can be a list of RT Channel indexes or handles to the same RT Channels.

Examples

Example 1

This example prints a list of RT Channel indexes for the first fixture in the selection. It uses an index number as input:

Lua

```
return function()
    -- Get the index number for the first fixture in the current selection
    local fixtureIndex = SelectionFirst()
    -- Get the indexes of the RT channels
    local rtChannels = GetRTChannels(fixtureIndex, false)
    -- Print an error message if returnd table is nil
    if rtChannels == nil then
        ErrPrintf("Please select a fixture and try again")
        return
    end
```

```
    -- Print the table content
    for key,value in ipairs(rtChannels) do
        Printf("List index number ".. key .." : RTChannel index number = ".. value)
    end
end
```

Example 2

This example prints a list of RT Channel indexes and attributes for the first fixture in the selection. It uses a handle as the input:

Lua

```
return function()
    -- Get a handle to the first fixture in the current selection
    local fixtureHandle = GetSubfixture(SelectionFirst())
    if fixtureHandle == nil then
        ErrPrintf("Please select a fixture and try again")
        return
    end
    -- Creates a table of handles to the RT channels of the first selected fixture.
    local rtChannels = GetRTChannels(fixtureHandle, true)
    if rtChannels == nil then
        ErrPrintf("Please select a fixture and try again")
        return
    end
    -- Print DMX addresses of the RT Channels for the fixture
    for key,value in ipairs(rtChannels) do
        Printf("List index number ".. key .. ": RTChannel Index = %i, Coarse DMX addr. = %s, Fine DMX addr. = %s", value.INDEX, value.COARSE, value.FINE)
    end
end
```

1.44.5.67. GetSample(string)

Description

The **GetSample** Lua function returns a number representing a percentage usage based on a string input.

Arguments

- **String**:
  Only a specific list of strings can be input:
  - MEMORY
  - CPU
  - CPUTEMP
  - GPUTEMP
  - SYSTEMP
  - FANRPM

Return

- **Number**:
  A number (float) is returned.

Example

This example stores the different samples in a table and then prints the content of the table:

Lua

```
return function()
    -- Gather the sample information in a table
    local sample = {}
    sample["MEMORY"] = GetSample("MEMORY")
    sample["CPU"] = GetSample("CPU")
    sample["CPUTEMP"] = GetSample("CPUTEMP")
    sample["GPUTEMP"] = GetSample("GPUTEMP")
    sample["SYSTEMP"] = GetSample("SYSTEMP")
    sample["FANRPM"] = GetSample("FANRPM")
    -- Print the collected data
    Printf("Memory ; ".. sample["MEMORY"])
    Printf("CPU ; ".. sample["CPU"])
    Printf("CPU temperature ; ".. sample["CPUTEMP"])
    Printf("GPU temperature ; ".. sample["GPUTEMP"])
    Printf("System temperature ; ".. sample["SYSTEMP"])
    Printf("Fan RPM ; ".. sample["FANRPM"])
end
```

1.44.5.68. GetScreenContent(handle)

Description

The **GetScreenContent** Lua function returns a handle to the screen content based on a provided handle to a screen configuration.

Arguments

- **Handle**:
  This must be a handle to a screen configuration.

Return

- **Handle**:
  The returned handle to the screen content.

Example

This example prints the data connected to the screen content handle. It uses the **CurrentScreenConfig()** and **Dump()** functions:

The **CurrentScreenConfig** Lua function returns a handle to the current users' screen configuration. Learn more in the **CurrentScreenConfig topic**.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Create a handle for the current screen configuration.
    local myCurrentScreenConfig = CurrentScreenConfig()
    -- Create a handle for the screen content based on the screen
configuration.
    local myScreenContent = GetScreenContent(myCurrentScreenConfig)
    -- Print the Dump of the handle.
    Printf("=============== START OF DUMP ===============")
    myScreenContent:Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.69. GetSelectedAttribute()

Description

The **GetSelectedAttribute** Lua function returns a handle to the currently selected attribute.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the selected attribute.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- This example dumps all information about the currently selected
attribute
    Printf("=============== START OF DUMP ===============")
    GetSelectedAttribute():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.70. GetShowFileStatus()

Description

The **GetShowFileStatus** Lua function returns a string with the current device's show file status, for example, "NoShow", "ShowLoaded", "ShowDownloaded", "ShowSaving", and "DataNegotiationActive".

Arguments

This function does not accept any arguments.

Return

- **String**:
  The returned string is the enum string from "Enums.ShowFileStatus" that matches the current status.

Example

This example prints the current device's show file status in the Command Line History:

```lua
return function ()
    -- Prints the current showfile status
    Printf("ShowfileStatus: "..GetShowFileStatus())
end
```

1.44.5.71. GetSubfixture(integer)

Description

The **GetSubfixture** Lua function returns the handle of the fixture specified by its patch index number.

Arguments

- **Integer**:
  The patch index number for a fixture. This is also known as the "subfixtureindex".

Return

- **Handle**:
  The function returns a handle to the fixture object matching the provided index number.

Example

This example uses a fixture selection to print all the information (in the Command Line History) about the first fixture in the selection using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- Check for a fixture selection, by returning an index for the first
fixture
    if (SelectionFirst()) then
        -- There is a fixture selection, store the index for the first
fixture
        local fixtureIndex = SelectionFirst()
        -- Dump all information about the fixture
        Printf("=============== START OF DUMP ===============")
        GetSubfixture(fixtureIndex):Dump()
        Printf("================ END OF DUMP ================")
    else
        -- There needs to be a selection of at least one fixture
        Printf("Please select a fixture")
    end
end
```

Related Functions

- **SelectionFirst**
- **SelectionNext**

- **GetSubfixtureCount**

1.44.5.72. GetSubfixtureCount()

Description

The **GetSubfixtureCount** Lua function returns the total number of fixtures that are patched within the show file.

Arguments

This function does not accept any arguments.

Return

- **Integer**:
  The returned integer number represents the total amount of patched fixtures on all the stages in the show file.

Example

This example prints the total number of patched fixtures in the Command Line History:

```lua
return function ()
    Printf('Total number of patched fixtures: %i', GetSubfixtureCount())
end
```

1.44.5.73. GetTokenName(string)

Description

The **GetTokenName** Lua function returns a string with the full keyword based on the short version string input or nil if there is no corresponding keyword.

Arguments

- **String**:
  The string input should correspond to a short version of a keyword.

Return

- **String**:
  A string with the full keyword is returned.

- OR -

- **Nil**:
  If there is no corresponding keyword, then nil is returned.

Example

This example returns the full keyword matching the short "seq" string:

Lua

```
return function()
    -- Store a short string to be used as input
    local shortToken = 'seq'
    -- Get the full token name
    local tokenName = GetTokenName(shortToken)
    -- Print useful output if nil is not returned
    if tokenName ~= nil then
        Printf("The full version of '".. shortToken .. "' is '" .. tokenName
.. "'")
    end
end
```

1.44.5.74. GetTokenNameByIndex(int)

Description

The **GetTokenNameByIndex** Lua function returns a string with the keyword based on the index number provided.

Each keyword is described in the **Command Syntax and Keywords section**.

Arguments

- **Integer**:
  The integer input is the index number for a corresponding keyword. There is no apparent logic to the index number and the keyword.

Return

- **String**:
  A string with the full keyword is returned.

- OR -

- **Nil**:
  If there is no corresponding keyword, then nil is returned.

Example

If the keyword exists, this example returns the keywords matching the first 443 index numbers:

Lua

```lua
return function()
    -- Create a variable to hold the keyword string
    local tokenName = ""
    -- Print the keywords to the first 443 indexes if possible
    for index = 1, 443, 1 do
        tokenName = GetTokenNameByIndex(index)
        if tokenName ~= nil then
            Printf("Token index " .. index .. " = " .. tokenName)
        end
    end
end
```

1.44.5.75. GetTopModal()

Description

The **GetTopModal** Lua function returns a handle for the modal at the top. Modal is the internal name for pop-ups that interrupt the system's normal operation. A modal blocks other UI elements from being used while it is open.

For example, when opening a window's settings pop-up, it is not possible to use the command line. The settings pop-up is a modal. Modals can also be identified by the rest of the UI, which darkens a bit when it is open.

Argument

This function does not have any arguments.

Return

- **Handle** | **nil**:
  The function returns a handle to the top modal UI object if there is one.

Example

This example uses the **Dump() function** to show information about the StagePopup selection pop-up.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Open a Modal / Pop-up.
    Cmd('Menu "StagePopup"')
    -- Add a small wait.
    coroutine.yield({activeShowstatus=0.2})
    -- Get the handle for the modal / pop-up.
    local modalHandle = GetTopModal()
    -- If there is a handle then dump all information else print en error
feedback.
    if modalHandle ~= nil then
        Printf("=============== START OF DUMP ===============")
        modalHandle:Dump()
        Printf("================ END OF DUMP ================")
    else
        ErrPrintf("The Modal UI object could not be found.")
```

```
    end
    -- Close the modal / pop-up by pressing the Escape key.
    Keyboard(1,'press','Escape')
    Keyboard(1,'release','Escape')
end
```

1.44.5.76. GetTopOverlay()

Description

The **GetTopOverlay** Lua function returns a handle for the overlay at the top of the display with the provided index number. Overlay is the internal name for what is called pop-ups or menus in the rest of this manual.

Argument

This function does not have any arguments.

Return

- **Handle** | **nil**:
  The function returns a handle to the top overlay UI object if there is one.

Example

This example uses the **Dump() function** to show information about the MenuSelector pop-up - it is the one opening when pressing the **Menu key**.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```lua
return function()
    -- Open the MenuSelector overlay.
    Cmd('Menu "MenuSelector')
    -- Add a small delay.
    coroutine.yield({activeShowstatus=0.2})
    -- Get the handle for the overlay on the display with index 1.
    local overlayHandle = GetTopOverlay(1)
    -- Add a small delay.
    coroutine.yield({activeShowstatus=0.2})
    -- Close the MenuSelector overlay.
    Cmd('Menu "MenuSelector')
    -- Check if there is a handle and print appropriate feedback.
    if overlayHandle ~= nil then
        Printf("=============== START OF DUMP ===============")
        overlayHandle:Dump()
        Printf("================ END OF DUMP ================")
    else
        ErrPrintf("The Overlay UI object could not be found.")
    end
```

```
end
```

1.44.5.77. GetUIChannelCount()

Description

The **GetUIChannelCount** Lua function returns a number indicating the total amount of UI channels.

Arguments

This function does not accept any arguments.

Return

- **Integer**:
  The function returns an integer number depicting the total amount of UI channels.

Example

This example prints the number of UI channels to the Command Line History:

```lua
return function()
    Printf("The number of UI channels is " .. GetUIChannelCount())
end
```

1.44.5.78. GetUIChannelIndex(integer, integer)

Description

The **GetUIChannelIndex** Lua function returns the index integer matching a UI channel based on two index inputs.

Arguments

- **Integer**:
  The first integer is the patch index of a fixture.
- **Integer**:
  This integer is an attribute index (0-based). This can be found in the **Attribute Definitions** or by the **GetAttributeIndex() Lua function**.

Attribute Definition

Attributes are the building blocks of fixture types. The same building blocks are used throughout the console and they are what is controlled using the Encoder bar when operating fixtures.

Attributes definitions describe the relation between Main Attributes and sub-attributes.

Learn more in the **Attribute Definition topic**.

GetAttributeIndex()

The **GetAttributeIndex** Lua function returns the (0 based) index number of the attribute definition based on the system name of the attribute.

Learn more in the **GetAttributeIndex() topic**.

Return

- **Integer**:
  The returned integer to a channel function.

Example

This example prints the UI channel index of the "Dimmer" attribute of the first fixture in the current selection:

Lua

```
return function()
    -- Get the Attribute index and UIChannel indexes
    local attributeIndex = GetAttributeIndex("Dimmer")
    local uiChannelIndex = GetUIChannelIndex(SelectionFirst(),attributeIndex)
    -- End the function if any of the index return nil
```

```
    if (attributeIndex == nil or uiChannelIndex == nil) then
        ErrPrintf("Something went wrong, maybe your first selected fixture
don't have a Dimmer - Please try again")
        return
    end
    Printf("The UI Channel Index is " .. uiChannelIndex)
end
```

1.44.5.79. GetUIChannels(integer[,boolean] OR handle[,boolean])

Description

The **GetUIChannels** Lua function returns a table with UI Channel indexes or a table with handles to the UI Channel objects. There are two different types of arguments for this function.

Arguments

- **Integer**:
  The integer should be the index number for a (sub)fixture.
- **Boolean** (Optional):
  - **True**:
    The returned table contains handles for UI Channel objects.
  - **False** (default):
    The returned table contains integer index values to the UI Channel objects.

- OR -

- **Handle**:
  The handle should relate to a (sub)fixture object.
- **Boolean** (Optional):
  - **True**:
    The returned table contains handles for UI Channel objects.
  - **False** (default):
    The returned table contains integer index values to the UI Channel objects.

Return

- **Table**:
  The returned table can be a list of UI Channel indexes or handles to the same UI Channel indexes.

Examples

Example 1

This example prints a list of UI Channel indexes for the first fixture in the selection. It uses an index number as input:

Lua

```
return function()
    -- Creates a table of indexes of the UI channels of the first selected
fixture.
    local uiChannels = GetUIChannels(SelectionFirst())
    if uiChannels == nil then
        ErrPrintf("Please select a fixture and try again")
        return
    end
    for key,value in ipairs(uiChannels) do
```

```
        Printf("List index number ".. key .. " :  UIChannel Index = " ..
value)
    end
end
```

Example 2

This example prints a list of UI Channel indexes and attributes for the first fixture in the selection. It uses a handle as the input:

Lua

```
return function()
    local fixtureHandle = GetSubfixture(SelectionFirst())
    -- Creates a table of handles to the UI channels of the first selected
fixture.
    local uiChannels = GetUIChannels(fixtureHandle, true)
    if uiChannels == nil then
        ErrPrintf("Please select a fixture and try again")
        return
    end
    for key,value in pairs(uiChannels) do
        Printf("List index number ".. key .. ": UIChannel Index = %i,
(Sub)Attribute = %s", value.INDEX-1, value.SUBATTRIBUTE)
    end
end
```

1.44.5.80. GetUIObjectAtPosition(integer, table)

Description

The **GetUIObjectAtPosition** Lua function returns the handle of the UI Object at a specified position on a specified display.

Argument

- **Integer**:
  The integer should be the index number of the display with the UI object.
- **Table**:
  The table must have two elements with the following keys:
  - x: This is the X position on the display. The value must be a number indicating the desired pixel position. It is counted from the left side of the display.
  - y: This is the Y position on the display. The value must be a number indicating the desired pixel position. It is counted from the top of the display.

Return

- **Handle** | **nil**:
  If a UI object is at the provided position, then the handle to the object is returned. Otherwise, it returns nil.

Example

This example prints the **Dump** of the UIObject at a specific position on display 1. It also uses the **DrawPointer** function to draw a red pointer at the position.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

The DrawPointer function draws a red pointer at a display. Learn more about it in the **DrawPointer() topic**.

Lua

```lua
return function()
    -- Get the index number for "Display 1"
    local displayIndex = GetDisplayCollect()["Display 1"].INDEX
    -- Create a table with X and Y position
    local positionTable = {}
    positionTable.x = 1000
    positionTable.y = 500
    -- Get the UI object handle
```

```
    local uiObjectAtPositionHandle =
GetUIObjectAtPosition(displayIndex,positionTable)
    -- Dump all information about the display with the index number if not
nil
    if uiObjectAtPositionHandle == nil then
        Printf("The returned value was not a valid handle.")
        return
    end
    -- Draw a pointer at the posiiton for 5 seconds
    DrawPointer(displayIndex,positionTable,5000)
    --Dump of the UIObject
    Printf("=============== START OF DUMP ===============")
    uiObjectAtPositionHandle:Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.81. GetVar(handle, string)

Description

The GetVar Lua function returns the value of a specific variable in a set of variables. To learn more about the variables in plugins, look at the **Variable Functions** topic.

Arguments

- **Handle**:
  The handle of variable set.
- **String**:
  The name of the variable. It needs to be in quotation marks.

Return

- **Value**:
  This is the value of the variable.

If the variable does not exist, then nil is returned.

Example

This example returns the value of a variable called "myUserVar" in the set of user variables if it exists:

Lua

```
return function()
    -- Get the value from a user variable called "myUserVar" - assuming it
already exists
    local varValue = GetVar(UserVars(), "myUserVar")
    -- Print en error feedback or the value of the variable
    if varValue == nil then
        Printf("Variable returns nothing!")
    else
        Printf("Variable value is: " .. varValue)
    end
end
```

1.44.5.82. GlobalVars()

Description

The GlobalVars function returns a handle to the set of global variables. Read more about these in the **Variables** topic in the Macro section.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle of the set of global variables.

Example

This example sets, gets, and deletes a global variable:

Lua

```
return function()
    -- Stores a local Lua variable with the handle for the global variable
set.
    local variableSet = GlobalVars()
    -- Sets a global variable with an integer value using the SetVar()
function.
    SetVar(variableSet, "myGlobalVar", 42)
    -- Prints the global variable using the GetVar() function.
    Printf("The value  of myGlobalVar is: " .. GetVar(variableSet,
"myGlobalVar"))
    -- Deletes the global variable using the DelVar() function.
    DelVar(variableSet, "myGlobalVar")
end
```

1.44.5.83. HandleToInt(handle)

Description

The **HandleToInt** Lua function converts a handle into an integer format.

See the **Handle topic** for more info regarding handles and links to other related functions.

Arguments

- **Handle**:
  The handle of the object.

Return

- **Integer**:
  The returned integer is the handle converted to an integer.

Example

This example prints the handle integer number for the selected sequence. It also converts the integer back to a handle and uses this to print the name of the sequence:

```
Lua
return function()
    Printf("The integer number for the handle of the selected sequence: %i",
HandleToInt(SelectedSequence()))
end
```

1.44.5.84. HandleToStr(handle)

Description

The **HandleToStr** Lua function converts a handle into a string in a hexadecimal number format.

See the **Handle topic** for more info regarding handles and links to other related functions.

Arguments

- **Handle**:
  The handle of the object.

Return

- **String**:
  The returned string is the handle number converted to a hexadecimal format.

Example

This example prints the handle hex number for the selected sequence. It also converts the string back to a handle and uses this to print the name of the sequence:

```Lua
return function()
    Printf("The string (in hex format with 'H#' in front) for the handle of the selected sequence: %s",HandleToStr(SelectedSequence()))
end
```

1.44.5.85. HookObjectChange(function, handle, handle[, handle])

Description

The **HookObjectChange** Lua function automatically calls a function when a grandMA3 object changes.

Arguments

- **Function**:
  This must be the name of a function. This function is triggered every time the provided grandMA3 object changes.
- **Handle**:
  This is the handle for the grandMA3 objects that should be monitored for changes. The triggered function passes this handle on as the first argument.
- **Handle**:
  The handle must be for the plugin creating this HookObjectChange - it is the handle for "this" plugin.
- **Handle** (optional):
  This optional handle is for an object that will be passed on to the triggered function (as the third argument).

Return

- **Integer**:
  The function returns an integer identifying the hook. This can be saved to unhook the object later.

	**Hint:**
	See also these related functions: **DumpAllHooks**, **Unhook**, **UnhookMultiple**.

Example

To call a function every time the content of the sequence pool changes, create a plugin with this code:

Lua

```
-- Get the handle to this Lua component.
local luaComponentHandle = select(4,...)

function Main()
    -- Get a handle to the sequence pool.
    local hookObject = DataPool().Sequences
    -- Get a handle to this plugin.
    local pluginHandle = luaComponentHandle:Parent()
    -- Create the hook and save the Hook ID.
    SequenceHookId = HookObjectChange(MySequencePoolCallback, hookObject,
pluginHandle)
    -- Print the returned Hook ID.
    Printf("HookId: " .. SequenceHookId)
end

-- This function is called when there are changes in the sequence pool.
```

```
function MySequencePoolCallback(obj)
    Printf(tostring(obj.name) .. " changed!")
end

return Main
```

1.44.5.86. HostOS()

Description

The **HostOS** Lua function returns a string with the type of operating system of the device where the plugin is executed (for instance, "Windows", "Linux", or "Mac").

Arguments

This function does not accept any arguments.

Return

- **String**:
  The returned string is the operating system of the grandMA3 hardware or grandMA3 onPC computer.

Example

This example prints the operating system of the device in the Command Line History:

```lua
return function()
    Printf("The HostOS is "..HostOS())
end
```

1.44.5.87. HostSubType()

Description

The **HostSubType** Lua function returns a string with the host sub type of the station where the plugin is executed (for example, "FullSize", "Light", "RPU", "onPCRackUnit", "Undefined").

Arguments

This function does not accept any arguments.

Return

- **String**:
  The returned string is the host sub-type of the device.

Example

This example prints the host sub-type of the station in the Command Line History:

```lua
return function()
    Printf("The HostSubType is "..HostSubType())
end
```

1.44.5.88. HostType()

Description

The **HostType** Lua function returns a string with the host type of the device where the plugin is executed (for example, "Console" or "onPC").

Arguments

This function does not accept any arguments.

Return

- **String**:
  The returned string is the host type of the device.

Example

This example prints the host type of the device in the Command Line History:

```Lua
return function()
    Printf("The HostType is "..HostType())
end
```

1.44.5.89. Import(string)

Description

The object-free **Import** Lua function imports a Lua table in XML format.

This function correlates to the **Export function**.

Arguments

- **String**:
  This is a string containing the file name of the desired imported file. It should contain the file name, including the entire path. See the example below.

Return

- **Table**:
  This is the imported table.

Example

This example imports the table exported using the example in the **Export() function topic** - please run that example before running this example.

Lua

```
return function ()
    -- Get the path for the exported table.
    local importPath = GetPath(Enums.PathType.Library) .. "/BuildDetails.xml"
    -- Check if the file exist and print relevant feedback.
    if importPath == nil then
        -- File does not exist.
        ErrPrintf("The desired file does not exist. Please add it or adjust
the requested file name.")
    else
        -- Import the table.
        local importedTable = Import(importPath)
        -- Check if the import returned something and print relevant
feedback.
        if importedTable == nil then
            -- Import didn't return anything.
            ErrPrintf("The import failed.")
        else
        -- Print some of the table content.
        Printf("CompileDate: " .. importedTable.CompileDate)
        Printf("CompileTime: " .. importedTable.CompileTime)
        Printf("BigVersion: " .. importedTable.BigVersion)
        Printf("HostType: " .. importedTable.HostType)
        Printf("HostSubType: " .. importedTable.HostSubType)
        Printf("CodeType: " .. importedTable.CodeType)
        end
    end
end
```

1.44.5.90. IncProgress(handle, integer)

Description

The IncProgress Lua function changes the value on the range for a progress bar using an integer input. A handle input argument defines the progress bar. The progress br needs to be created using the **StartProgress() function.**

StartProgress()

The StartProgress Lua function creates and displays a progress bar on all screens.

Learn more in the **StartProgress() topic**.

See the **ProgressBar topic** for more info regarding progress bars and links to other related functions.

Arguments

- **Handle**:
  The handle for the progress bar.
- **Integer**:
  The desired value for the range. This can be a negative value to decrease the value.

Return

This function does not return anything.

Examples

These two examples increase and decrease the range value for the progress bar created using the example in the StartProgress topic (link above):

Lua
```
return function()
    -- Increase the current value for a progress bar with the matching
handle.
    IncProgress(progressHandle, 1)
end
```
Lua
```
return function()
    -- Decrease the current value for a progress bar with the matching
handle.
    IncProgress(progressHandle, -1)
end
```

1.44.5.91. IntToHandle(integer)

Description

The **IntToHandle** Lua function converts an integer number into a handle. The integer needs to correlate with an actual handle.

See the **Handle topic** for more info regarding handles and links to other related functions.

Arguments

- **Integer**:
  The integer that correlates to an object's handle.

Return

- **Handle**:
  The returned handle of the object correlates with the integer.

Example

This example prints the handle integer number for the selected sequence. It also converts the integer back to a handle and uses this to print the name of the sequence:

Lua

```
return function()
    -- Convert the handle of the currently selected sequence to an integer
    local handleInt = HandleToInt(SelectedSequence())
    -- Print the handle integer
    Printf("The handle integer number of the selected sequence: %i",
HandleToInt(SelectedSequence()))
    -- Convter the integer back to a hanndle and use it to get the sequence
name
    Printf("The name of the selected sequence is: %s",
IntToHandle(handleInt).name)
end
```

1.44.5.92. IsClassDerivedFrom(string, string)

Description

The **IsClassDerivedFrom** Lua function returns a boolean indicating if a class is derived from a different class.

Arguments

- **String**:
  This string needs to be the name of the class that might be derived from a different class.
- **String**:
  This string needs to be the name of the class that might be the base class.

Return

- **Boolean**:
  The returned boolean indicates if the class is derived from the base class.

Example

This example checks if a class is derived from a different class and returns useful feedback.

Lua

```
return function()
    -- Set the value of the two strings.
    local derivedName = "World"
    local baseName = "Group"
    -- Check if the derivedName is the name of a class derived from the
baseName class.
    local isDerived = IsClassDerivedFrom(derivedName, baseName)
    -- Provide feedback.
    if isDerived then
        Printf(derivedName .. " is derived from " .. baseName)
    else
        Printf(derivedName .. " is not derived from " .. baseName)
    end
end
```

1.44.5.93. IsObjectValid(handle)

Description

The **IsObjectValid** function returns a boolean true or nil depending on whether the supplied argument is a valid object.

Arguments

- **Handle**:
  The argument should be the handle to a possible object.

Return

- **Boolean or nil**:
  The returned value is a boolean True if the handle is a valid object or it returns nil if it is not a valid object.

Example

This example below examines if "Root()" is a valid object and prints meaningful feedback:

Lua

```
return function()
    --Create a variable with the possible object
    local myObject = Root()
    --Check if it is an object
    local myReturn = IsObjectValid(myObject)
    --Print the result
    if myReturn == nil then
        ErrPrintf("It is not a valid object")
    else
        Printf("It is an object")
    end
end
```

1.44.5.94. KeyboardObj()

Description

The **KeyboardObj** function returns the handle to the first found keyboard object.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns the handle to the keyboard object.

Example

This example prints the information of the keyboard object. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Print all informatin about the KeyboardObj object
    Printf("=============== START OF DUMP ===============")
    KeyboardObj():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.95. MasterPool()

Description

The **MasterPool** Lua function returns the handle to the masters.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns the handle to the pool of masters.

Example

This example prints the information of the MasterPool object. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

```lua
return function()
    -- Print all informatin about the MasterPool object
    Printf("=============== START OF DUMP ===============")
    MasterPool():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.96. MessageBox(table)

Description

The MessageBox Lua function is used to create pop-up message boxes. These can be simple or complex information pop-ups with many different options and user inputs.

The message box contains multiple elements that must be defined in a table. This table is the single input argument to the function.

The elements in the message box are displayed in alphabetical order.

This function is part of the User Interface functions.

Arguments

- **Table**:
  The input to the function must be formatted as a table using key-value pairs. The needed elements have default values that will be used if not overwritten. The values can be defined in any order if the key is defined.
  The table can have the following elements:
  - **title**:
    This is the title of the pop-up message box.
  - **titleTextColor**:
    This is the text color for the title text. The value is a number or string that refers to the **UI Colors** in the color theme. See the link below.
  - **backColor**:
    This is the color of the frame or border of the pop-up. The value is a number or string that refers to the **UI Colors** in the color theme. See the link below.
  - **icon**:
    This is an icon that can be shown in the upper left corner of the pop-up. The value can be an integer or a string that refers to the number or name of a texture image (without the file format).
    The icons can be listed by navigating to the texture folder **ChangeDestination GraphicsRoot/TextureCollect/Textures** and then doing a **List** command.
    This shows a long list with numbers and names of all the textures in the Command Line History window.
  - **message**:
    This message text string is displayed in the main part of the pop-up. A new line can be created by adding a "\n" in the text.
  - **messageTextColor**:
    This is the text color for the message text. The value is a number or string that refers to the **UI Colors** in the color theme. See the link below.
  - **autoCloseOnInput**:
    This option defines if a Please (or Enter) from an input field closes the message box pop-up. The default value is **true**. Setting this to **false** keeps the message box open until it is explicitly closed. See example 5 below.

- **timeout**:
  The timeout value is an integer that indicates how long the message box is displayed in milliseconds. It will show a countdown timer at the top of the message area. When the countdown runs out, it will close the pop-up.
  When the timeout closes the pop-up, it returns a **success** element with a **true** value (see more about the return below). It was closed "normally".
- **timeoutResultCancel**:
  This element can change the return to mimic a cancel of the pop-up, which returns a **false** instead of a **true** for the **success** element.
- **timeoutResultID**:
  When the timeout closes the pop-up with the **success** value = **true**, a special return result can be defined using this, for instance, setting the value to 99 - then 99 is returned when the message box is closed by the timeout.
- **commands**:
  The commands are buttons at the bottom of the message box pop-up. The input here is a table of objects using the following structure:
  - **value**:
    This integer value will be returned as the result value - see more about the return below.
  - **name**:
    This is a string which will be shown on the button.
- **inputs**:
  The inputs are user input fields where text or numbers can be input. The input fields will be displayed in alphabetical order.
  The input fields are defined using an table with the following structure (see example 5 below):
  - **name**:
    This is a string value - the text will be shown as a label for the input field.
  - **value**:
    This is a string value - it is a default input value for the input field.
  - **blackfilter**:
    This is a string value - it defines input characters that are not allowed.
  - **whitefilter**:
    This is a string value - it defines which input characters are allowed.
  - **vkPlugin**:
    This is a string value - it is the name of the input pop-up, which is opened if the on-screen keyboard icon (⌨) is tapped in the pop-up. Example 5 below has a list of possible vkPlugin values.
  - **maxTextLength**:
    This is an integer value - it defines the maximum number of characters for the input.
- **states**:
  The states are buttons in the pop-up. State buttons have a small checkbox and can have a true or false state. The buttons will be displayed in alphabetical order.
  The buttons are defined using a table with the following structure (see example 4 below):
  - **name**:
    This is a string value - the text will be shown on the button.
  - **state**:
    This is the initial state of the button checkbox.

- **selectors**:
  Selector buttons are two different types of buttons. Each type can have a selected value based on a list of available values. The two types are **Swipe** button (type 0) and **Radio** button (type 1).
  The buttons are defined in a table with the following structure:
  - **name**:
    This is a string value - the text will be shown on the button (swipe button) or as a label above the buttons (radio buttons).
  - **selectedValue**:
    This is an integer value - it defines the default selected value
  - **type**:
    This is an integer value - it defines the type of selector button. The options are:
    - **0**:
      This defines the button as a swipe button.
    - **1**:
      This defines the button as a radio button.
  - **values**:
    This is another table containing the different values available for the selector button. Each value element in the table has the following structure: **["string"]=integer**
    The string is the name displayed for the value. The integer is the value returned and the one used for **selectedValue** (see above). See example 6 below for an example of use.

The table can contain some or all of the elements described above.

The colors mentioned above can be a string or number value. It refers to a defined UI Color in the color theme, for instance, **"Global.Text"** or **1.27**. See more in the **Color Theme topic**.

A message box pop-up should have at least a title, message, and *either* a timeout *or* some basic command buttons. See the first three examples below.

Return

- **Table**:
  The return from a message box pop-up is formatted in a table. The returned table adjusts to match the elements of the message box. For instance, if there are selection buttons in the message box, then there is another table inside the result table containing the return from the selection buttons. See the examples for details on how to retrieve the results. The table can contain the following elements:
  - **success**:
    This is a boolean - it returns true if the message box was closed by tapping a command button or by a timeout (see **timeoutResultCancel** above for exceptions).
  - **result**:
    This is an integer - it returns the value of the tapped command button or the timeout result ID.
  - **inputs** (only if the message box has inputs fields):
    This is a table with a list of the input fields' name and string value in a key-value pair table.

- **states** (only if the message box has state buttons):
  This is a table with a list of the state buttons' name and boolean value in a key-value pair table.
- **selectors** (only if the message box has selector buttons):
  This is a table with a list of the selector buttons' name and integer value in a key-value pair table.

An error is thrown if the message box does not have inputs, states, and selectors, but the script tries to use the table elements.

See the examples to see how to extract the results.

Examples

There are six different examples demonstrating different elements of the message box. The elements can be combined, but the examples highlight different functions.

Example 1

A simple message box pop-up that shows a single confirm button:

```lua
return function ()
    -- This creates a small pop-up with a single button.
    local returnTable = MessageBox(
        {
            title = "Please confirm This",
            commands = {{value = 1, name = "Confirm"}}
        }
    )

    -- Print the content of the returned table.
    Printf("Success = "..tostring(returnTable.success))
    Printf("Result = "..returnTable.result)
end
```

Example 2

This example opens a pop-up with some text and two command buttons:

```lua
return function ()
    -- A table with two default buttons for the pop-up
    local defaultCommandButtons = {
        {value = 2, name = "OK"},
        {value = 1, name = "Cancel"}
    }

    -- A table with the elements needed for the pop-up
    local messageTable = {
        icon = "object_smart",
        backColor = "Window.Plugins",
        title = "This is the title",
        message = "This is a message\nThat can have multiple lines",
```

```
        commands = defaultCommandButtons,
    }

    -- The creation on the actual pop-up with the result stored in a variable
    local returnTable = MessageBox(messageTable)

    -- Print the content of the returned table
    Printf("Success = "..tostring(returnTable.success))
    Printf("Result = "..returnTable.result)
end
```

Example 3

This example displays a message box for 3 seconds and then closes itself:

Lua

```
return function ()
    -- This variable contains the table used as argument for the messagebox
    local messageTable = {
        title = "Do not worry",
        message = "This message will self destruct\nGoodbye!",
        timeout = 3000,
        timeoutResultCancel = false,
        timeoutResultID = 99,
    }

    -- This creates the messagebox pop-up and store the return table in a
variable
    local returnTable = MessageBox(messageTable)

    -- Print the content of the returned table
    Printf("Success = "..tostring(returnTable.success))
    Printf("Result = "..returnTable.result)
end
```

Example 4

This example adds state buttons to the message box. The buttons are added to a table for a better overview.

Lua

```
return function ()
    -- A table with two default buttons for the pop-up
    local defaultCommandButtons = {
        {value = 2, name = "OK"},
        {value = 1, name = "Cancel"}
    }
    -- A table with three state buttons
    -- The buttons will be displayed alphabetically in the pop-up
    local stateButtons = {
        {name = "State B", state = false},
        {name = "State A", state = false},
        {name = "New State", state = false}
    }

    -- A table with the elements needed for the pop-up
```

```
    local messageTable = {
        icon = "object_smart",
        backColor = "Window.Plugins",
        title = "This is state buttons",
        message = 'Toggle the states and click "Ok"',
        commands = defaultCommandButtons,
        states = stateButtons,
    }

    -- The creation on the actual pop-up with the result stored in a variable
    local returnTable = MessageBox(messageTable)

    -- Print the content of the returned table
    Printf("Success = "..tostring(returnTable.success))
    Printf("Result = "..returnTable.result)

    -- Print a list with the state of the stateButtons
    for name,state in pairs(returnTable.states) do
        Printf("State '%s' = '%s'",name,tostring(state))
    end
end
```

Example 5

This example shows the input fields.

Lua

```
return function ()
    -- A table with two default buttons for the pop-up
    local defaultCommandButtons = {
        {value = 2, name = "OK"},
        {value = 1, name = "Cancel"}
    }
    -- A table with three input fields
    -- The fields will be displayed alphabetically in the pop-up based on
name
    local inputFields = {
        {name = "Numbers Only", value = "1234", whiteFilter = "0123456789",
vkPlugin = "NumericInput"},
        {name = "Text Only", value = "abcdef", blackFilter = "0123456789"},
        {name = "Maximum 10 characters", value = "", maxTextLength = 10}
    }
    -- Possible vkPlugin values:
    -- - "TextInput" : same as default - standrd on-screne keyboard
    -- - "TextInputNumOnly" : text input but only with number buttons
    -- - "TextInputNumOnlyRange" : text input but only with number and
related range buttons
    -- - "TextInputTimeOnly" : text input styled for time input - includes
buttons for time values
    -- - "NumericInput" : general number input
    -- - "CueNumberInput" : number input styled for cue number
    -- - "RelCueNumberInput" : number input with the relative "delta" button
    -- - "IP4Prefix" : designed for inputting an IPv4 address allowing CIDR
notation

    -- A table with the elements needed for the pop-up
    local messageTable = {
```

```
        icon = "object_smart",
        backColor = "Window.Plugins",
        title = "This is input fields",
        message = 'Change the values in the input fields and click "Ok"',
        commands = defaultCommandButtons,
        inputs = inputFields,
        autoCloseOnInput = false
    }

    -- The creation on the actual pop-up with the result stored in a variable
    local returnTable = MessageBox(messageTable)

    -- Print the content of the returned table
    Printf("Success = "..tostring(returnTable.success))
    Printf("Result = "..returnTable.result)
    -- Print a list with the values of the input fields
    for name,value in pairs(returnTable.inputs) do
        Printf("Input '%s' = '%s'",name,tostring(value))
    end
end
```

Example 6

This example shows the different selector buttons.

Lua

```
return function ()
    -- A table with two default buttons for the pop-up
    local defaultCommandButtons = {
        {value = 2, name = "OK"},
        {value = 1, name = "Cancel"}
    }
    -- A table with selector buttons
    -- The buttons will be displayed alphabetically in the pop-up based on
name
    local selectorButtons = {
        { name="Swipe Selector", selectedValue=1, type=0,
values={["Swipe1"]=1,["Swipe2"]=2}},
        { name="Radio Selector", selectedValue=2, type=1,
values={["Radio1"]=1,["Radio2"]=2}},
        { name="Another Radio", selectedValue=3, type=1,
values={["Radio3"]=3,["Radio4"]=4}}
    }

    -- State button to show grouping with swipe Selector button
    local stateButton = {
        {name = "State Button", state = false},
    }
    -- A table with the elements needed for the pop-up
    local messageTable = {
        icon = "object_smart",
        backColor = "Window.Plugins",
        title = "This is input fields",
        message = 'Change the values in the input fields and click "Ok"',
        commands = defaultCommandButtons,
        states = stateButton,
        selectors = selectorButtons,
```

```
    }

    -- The creation on the actual pop-up with the result stored in a variable
    local returnTable = MessageBox(messageTable)

    -- Print the content of the returned table
    Printf("Success = "..tostring(returnTable.success))
    Printf("Result = "..returnTable.result)
    -- Print a list with the values of the selection buttons
    for name,value in pairs(returnTable.selectors) do
        Printf("Input '%s' = '%s'",name,tostring(value))
    end
    -- Print a list with the state of the stateButton
    for name,state in pairs(returnTable.states) do
        Printf("State '%s' = '%s'",name,tostring(state))
    end
end
```

1.44.5.97. MouseObj()

Description

The **MouseObj** function returns the handle to the first found mouse object.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns the handle to the mouse object.

Example

This example prints the information of the mouse object. Is uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

```lua
return function()
    -- Print all informatin about the MouseObj object
    Printf("=============== START OF DUMP ===============")
    MouseObj():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.98. NeedShowSave()

Description

The **NeedShowSave** Lua function returns a boolean indicating if there are unsaved changes to the showfile.

Arguments

This function does not accept any arguments.

Return

- **Boolean**:
  The boolean returns True if there are unsaved changes to the show file. False indicates that the show file has not changed since the last save. These indications do not include changes to the playback state of the show.

Example

This example prints feedback indicating if the show file should be saved or not.

Lua

```
return function ()
    -- Check if the show should be saved.
    if NeedShowSave() then
        Printf("You should save your showfile.")
    else
        Printf("You do not need to save your showfile.")
    end
end
```

1.44.5.99. ObjectList(string[, table])

Description

The **ObjectList** Lua function returns a table with handles. The table is created based on a string input that should create a selection.

Argument

- **String**:
  The string must be a command that would create a range of objects in the command line.
- **Table** (optional):
  The table can contain two possible named elements. Each element can have a boolean true or false value. See the examples below for how to use them.
  - **'reverse_order'**:
    This must have a boolean value. If this is true then the returned list is in reverse order.
  - **'selected_as_default'**:
    This must have a boolean value. If this is true then the object list will only contain the object that is selected in the pool. For instance, it only returns the currently selected filter from the filter pool.

Return

- **Table**:
  The function returns a table with handles to the objects based on the string argument.

Examples

This example returns the names and patch addresses of fixtures 1 through 10. It assumes these fixtures exist - if they do not, then it returns an error text.

Lua

```
return function()
    -- Create a list of handles based on the "Fixture 1 Thru 10" selection
and store it in a table.
    local myObjects = ObjectList("Fixture 1 Thru 10", {reverse_order=true})
    -- If the selection returned a table, then go through all elements and
print information of the object.
    if myObjects~= nil then
        for i in pairs(myObjects) do
            Printf("Fixture: " .. myObjects[i].name .. " - Patch: "
..myObjects[i].patch)
        end
    else
        ErrPrintf("An error occured. Does Fixture 1 Thru 10 exist?")
    end
end
```

This example creates an object list with the selected sequence. It then dumps all information about the sequence using the **Dump function**.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```lua
return function()
    -- Create a list of one handle to the selected sequence and store it to a table.
    local myObjects = ObjectList("Sequence", {selected_as_default=true})
    -- If the selection returned a table, then dump the first (and only) element.
    if myObjects~= nil then
        myObjects[1]:Dump()
    else
        ErrPrintf("An error occured.")
    end
end
```

1.44.5.100. Patch()

Description

The **Patch** Lua function returns a handle to the patch object.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the patch.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- This example dumps all information about the patch object
    Printf("=============== START OF DUMP ===============")
    Patch():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.101. Printf(string)

Description

The Printf Lua function prints a string in the **Command Line History** and **System Monitor**.

Command Line History

The **Command Line History** window shows feedback from the system based in the user input.

Learn more in the **Command Line History topic**.

System Monitor

The **System Monitor** window shows what is happening at the station. This includes feedback on user commands. It is a log of the different things happening in the background. It also shows warnings, errors, and changes to the system.

Learn more in the **System Monitor topic**.

Argument

- **String**:
  The string text to be printed to the Command Line History.

Return

This function does not return anything.

Example

This example prints "Hello World!" in the Command Line History:

Lua
```
return function()
    Printf("Hello World!")
end
```

1.44.5.102. Programmer()

Description

The **Programmer** Lua function references the current programmer object.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle to the Programmer object.

Example

This example uses the **Dump() function** on the programmer object:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

```lua
return function ()
    -- Dumps information about the programmer object.
    Printf("=============== START OF DUMP ===============")
    Programmer():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.103. ProgrammerPart()

Description

The **ProgrammerPart** Lua function references the current programmer part object.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle to the programmer part object.

Example

This example uses the **Dump() function** on the programmer part object:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- Dumps information about the current programmer part object.
    Printf("=============== START OF DUMP ===============")
    ProgrammerPart():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.104. Pult()

Description

The **Pult** Lua function returns a handle to the current "Pult" object at position Root/GraphicsRoot/PultCollect. The "Pult" object contains display and device information.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle to the pult object.

Example

This example prints the data connected to the handle. It uses the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- The following prints the dump for the pult object
    Printf("================ START OF DUMP ================")
    Pult():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.105. ReleaseType()

Description

The ReleaseType Lua function returns a string with the type of release for the MA software. All the software versions available from MA Lighting will return "Release". Internally and during development, there can be other release types.

Arguments

This function does not accept any arguments.

Return

- **String**:
  The returned string is the release type of the grandMA3 software.

Example

This example prints the release type in the Command Line History:

```lua
return function()
    Printf("The ReleaseType is "..ReleaseType())
end
```

1.44.5.106. Root()

Description

The **Root** Lua function returns a handle to the object at the root position.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle to the Root object.

Example

This simple example prints the information of the Root object in the Command Line History using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

```lua
return function()
    -- The following prints the dump for the root object
    Printf("================ START OF DUMP ================")
    Root():Dump()
    Printf("================= END OF DUMP =================")
end
```

1.44.5.107. SelectedFeature()

Description

The **SelectedFeature** Lua function returns the handle of the selected feature.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle for the selected feature.

Example

This example prints all information about the selected feature in the Command Line History using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- The following prints the dump for the selected feature object
    Printf("================ START OF DUMP ================")
    SelectedFeature():Dump()
    Printf("================= END OF DUMP =================")
end
```

1.44.5.108. SelectedLayout()

Description

The **SelectedLayout** Lua function returns the handle of the selected **layout**.

Layouts are two-dimensional drafts where it is possible to arrange fixtures, macros, groups, and other pool objects. Learn more in the **Layout topics**.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle for the selected layout.

Example

This example prints all information about the selected layout in the Command Line History using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- The following prints the dump for the selected layout object
    Printf("=============== START OF DUMP ===============")
    SelectedLayout():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.109. SelectedSequence()

Description

The **SelectedSequence** Lua function returns the handle of the selected sequence.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle for the selected sequence.

Example

This example prints all information about the selected sequence in the Command Line History using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- The following prints the dump for the selected sequence object
    Printf("================ START OF DUMP ================")
    SelectedSequence():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.110. SelectedTimecode()

Description

The **SelectedTimecode** Lua function returns the handle of the selected **timecode** object.

The selected timecode object is the Timecode show currently selected in the Timecodes pool. Learn more in the **Timecodes topics**.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle for the selected timecode object.

Example

This example prints all information about the selected timecode show in the Command Line History using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```lua
return function ()
    -- The following prints the dump for the selected timecode object
    local myTimecodeShow = SelectedTimecode()
    if myTimecodeShow ~= nil then
        Printf("================ START OF DUMP ================")
        myTimecodeShow:Dump()
        Printf("================ END OF DUMP ================")
    end
end
```

1.44.5.111. SelectedTimer()

Description

The **SelectedTimer** Lua function returns the handle of the selected **timer** object.

The selected timer object is the Timer currently selected in the Timers pool. Timers are stopwatch and timers that can be used to measure time. Learn more in the **Timers topics**.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The returned handle for the selected timer object.

Example

This example prints all information about the selected timer in the Command Line History using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```lua
return function ()
    -- The following prints the dump for the selected timer object
    local myTimer = SelectedTimer()
    if myTimer ~= nil then
        Printf("================ START OF DUMP ================")
        myTimer:Dump()
        Printf("================ END OF DUMP ================")
    end
end
```

1.44.5.112. Selection()

Description

The Selection Lua function returns a handle to the object holding the current selection of fixtures.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle to the Selection object.

Example

This example prints the information of the Selection object in the Command Line History using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- The following prints the dump for the object for the selection
    Printf("================ START OF DUMP ================")
    Selection():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.113. SelectionCount()

Description

The **SelectionCount** Lua function returns a number indicating the total amount of currently selected fixtures.

Arguments

This function does not accept any arguments.

Return

- **Integer**:
  The function returns an integer number depicting the total amount of fixtures in the current selection.
  If there is no selection, then it returns 0.

Example

This example prints the number of fixtures in the current selection to the Command Line History:

```lua
return function()
    Printf('Number of fixtures in the current selection: %i',
SelectionCount())
end
```

1.44.5.114. SelectionFirst()

Description

The **SelectionFirst** Lua function returns a set of integers for the selection's first fixture. It is the patch index number and the XYZ grid values in the selection grid.

It is not required to use all four returned integers, but they are returned in order.

Arguments

This function does not accept any arguments.

Return

- **Integer**:
  The returned number is the patch index of the first fixture in the current selection. It is not the FID or CID. The index is 0-based.
- **Integer**:
  The returned number is the current position on the X-axis in the selection grid. The selection grid is 0-based.
- **Integer**:
  The returned number is the current position on the Y-axis in the selection grid. The selection grid is 0-based.
- **Integer**:
  The returned number is the current position on the Z-axis in the selection grid. The selection grid is 0-based.

Example

This example prints the returned numbers of the first fixture in the selection, to the Command Line History:

Lua

```lua
return function()
    -- Store the return in a local variable
    local fixtureIndex, gridX, gridY, gridZ = SelectionFirst();

    -- Cancel the plugin if no fixture is selected
    assert(fixtureIndex,"Please select a fixture and try again.");

    -- Print the index number of the first fixture in the selection
    Printf("First selected fixture has index number: "..fixtureIndex
        .." and gridX value: "..gridX
        .." and gridY value: "..gridY
        .." and gridZ value: "..gridZ);
end
```

Related Functions

- **SelectionNext**
- **GetSubfixture**
- **GetSubFixtureCount**

1.44.5.115. SelectionNext()

Description

The SelectionNext function returns a set of integers for the next fixture in a selection based on the index number input as an argument. It is the index number in the patch and the XYZ grid values in the selection grid.

It is not required to use all four returned integers, but they are returned in order.

Arguments

- **Integer**:
  The index number is used to find the next fixture. The index number needs to be part of the current selection.

Return

- **Integer**:
  The returned number is the patch index of the first fixture in the current selection. It is not the FID or CID. The index is 0-based.
- **Integer**:
  The returned number is the current position on the X-axis in the selection grid. The selection grid is 0-based.
- **Integer**:
  The returned number is the current position on the Y-axis in the selection grid. The selection grid is 0-based.
- **Integer**:
  The returned number is the current position on the Z-axis in the selection grid. The selection grid is 0-based

Example

This example prints the patch index number and grid positions of all the fixtures in the current selection:

Lua

```
return function()
    -- Store the return in a local variable
    local fixtureIndex, gridX, gridY, gridZ = SelectionFirst()

    -- Cancel the plugin if no fixture is selected
    assert(fixtureIndex,"Please select a (range of) fixture(s) and try
again.")

    -- Loop that prints the index and gridpositions of all the fixtures in
the selection
    while fixtureIndex do
        Printf('The fixture has index number: %i and gridposition %i / %i /
%i',
          fixtureIndex, gridX, gridY, gridZ);
```

```
        -- Here is SelectionNext actually used to find the next fixture in
the selection
        fixtureIndex, gridX, gridY, gridZ = SelectionNext(fixtureIndex)
    end
end
```

Related Functions

- **SelectionFirst**
- **GetSubfixture**
- **GetSubfixtureCount**

1.44.5.116. SerialNumber()

Description

The **SerialNumber** Lua function returns the serial number of the grandMA3 hardware or grandMA3 onPC.

Arguments

This function does not accept any arguments.

Return

- **String**:
  The returned string is the serial number of the grandMA3 hardware or grandMA3 onPC.

Example

This example prints the serial number in the Command Line History:

```lua
return function()
    Printf("Serial number: " .. SerialNumber())
end
```

1.44.5.117. SetBlockInput(boolean)

Description

The **SetBlockInput** function is an internal function used during the system tests. It stops input from USB-connected keyboards and mouse. The built-in keyboard on some models is internally connected using a USB connection, which is also blocked by this function. The block affects the station where the function is executed.

	**Important:**
	Should the station be blocked and cannot be unblocked using a new Lua command, then pressing the keyboard keys A, S, D, and F simultaneously unblock the input again.

Arguments

- **Boolean**:
  The boolean indicates if the stations' input should be blocked or unblocked.
  - **true** (or 1): The station input is blocked.
  - **false** (or 0): The station input is unblocked.

Return

This function does not return anything.

Example

This example blocks mouse and keyboard input for 10 seconds:

Lua

```lua
return function()
    -- Set a variable for yield time in seconds
    yieldTime = 10
    -- Set the block to true
    SetBlockInput(true)
    -- Wait the [yieldtime]
    coroutine.yield(yieldTime)
    -- Unblock the station
    SetBlockInput(false)
end
```

1.44.5.118. SetLED(handle,table)

Description

The **SetLED** Lua function sends a table with a set of LED brightness values to an MA3Module. After around two seconds, the system automatically sets the LED values to what it believes it should be.

Below the example is a table listing all the grandMA3 hardware modules and which index number matches which LED on the hardware module.

	**Important:**
	Setting a value above 0 for a table index number not connected to an LED can cause the module to crash and reboot

Arguments

- **Handle:**
  This function does not accept any arguments.
- **Table:**
  The table should be an indexed table with a set of integer values. The value range is from 0 to 255. This range indicates a brightness level. A special value of "-1" is used to release the LED to the system. The table should contain 1024 indexes.

Return

This function does not return anything.

Example

This example sets the LEDs on encoder 1 to green on a full-size console:

Lua

```
return function()
    -- Create the LED table
    local myLedTable = {}
    -- Fill the table with default "release" value
    for index=1,256 do
        myLedTable[index] = -1;
    end
    -- Set values in the table
    -- Encoder_inside1 = green
    myLedTable[7] = 0
    myLedTable[10] = 255
    myLedTable[22] = 0
    -- Encoder_outside1 = green
    myLedTable[8] = 0
    myLedTable[11] = 255
    myLedTable[23] = 0
    -- Get the handle for the MasterModule on a console
    local usbDeviceHandle = Root().UsbNotifier.MA3Modules["UsbDeviceMA3 2"]
```

```
    -- Set the values for the LEDs
    SetLED(usbDeviceHandle, myLedTable)
end
```

Hardware Modules LED Table

This table provides a list of index numbers (indexed from 1). The numbers are matched with elements on the three different kinds of hardware modules that are relevant. Notice that the naming here comes from the internal hardware definition and might not match exactly the print on the keys or the official name.

Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
0			
1	ENCODER_INSIDE4 Red	Executor 108 Button	Executor 108 Button
2	ENCODER_OUTSIDE3 Red	Executor 110 Button	Executor 110 Button
3	ENCODER_INSIDE2 Red	Executor 211 Button	Executor 211 Button
4	EXEC_GrandKnob Red	Executor 212 Button	Executor 212 Button
5	MENU	Executor 213 Button	Executor 213 Button
6	ENCODER_OUTSIDE2 Red	Executor 214 Button	Executor 214 Button
7	ENCODER_INSIDE1 Red	Executor 215 Button	Executor 215 Button
8	ENCODER_OUTSIDE1 Red	Executor 209 Button	XFade1Btn Knob Red
9	EXEC_GrandKnob Green	Executor 210 Button	XFade2Btn Knob Red
10	ENCODER_INSIDE1 Green	Executor 208 Button	Executor 209 Button
11	ENCODER_OUTSIDE1 Green	Executor 115 Button	Executor 210 Button
12	ENCODER_INSIDE2 Green	Executor 114 Button	Executor 208 Button
13	ENCODER_OUTSIDE2 Green	Executor 113 Button	XFade1Btn Knob Green
14	ENCODER_INSIDE3 Green	Executor 112 Button	XFade2Btn Knob Green
15	ENCODER_OUTSIDE3 Green	Executor 207 Button	DEF_GO
16	ENCODER_INSIDE4 Green	Executor 206 Button	Executor 115 Button
17	ENCODER_OUTSIDE4 Green	Executor 205 Button	Executor 114 Button
18	ENCODER_INSIDE5 Green	Executor 105 Button	Executor 113 Button
19	ENCODER_OUTSIDE5 Green	Executor 106 Button	Executor 112 Button
20	ENCODER_OUTSIDE4 Red	Executor 107 Button	Executor 207 Button
21	EXEC_GrandKnob Blue	Executor 109 Button	Executor 206 Button
22	ENCODER_INSIDE1 Blue	Executor 204 Button	Executor 205 Button
23	ENCODER_OUTSIDE1 Blue	Executor 203 Button	XFade2 Fader Red
24	ENCODER_INSIDE2 Blue	Executor 202 Button	XFade2 Fader Green
25	ENCODER_OUTSIDE2 Blue	Executor 201 Button	XFade2 Fader Blue
26	ENCODER_INSIDE3 Blue	Executor 111 Button	Executor 105 Button
27	ENCODER_OUTSIDE3 Blue	Executor 101 Button	Executor 106 Button
28	ENCODER_INSIDE4 Blue	Executor 102 Button	Executor 107 Button
29	ENCODER_OUTSIDE4 Blue	Executor 103 Button	XFade1Btn Knob Blue
30	ENCODER_INSIDE5 Blue	Executor 104 Button	XFade2Btn Knob Blue
31	ENCODER_OUTSIDE5 Blue	Executor 312 Fader Red	Executor 109 Button
32	ENCODER_OUTSIDE5 Red	Executor 311 Fader Red	Executor 204 Button
33	ENCODER_INSIDE5 Red	Executor 413 Fader Red	Executor 203 Button
34	ENCODER_INSIDE3 Red	Executor 411 Button	Executor 202 Button
35	ESC	Executor 412 Button	Executor 201 Button
36	CLEAR	Executor 414 Fader Red	XFade2Btn Button
37	HELP	Executor 411 Fader Red	DEF_PAUSE
38	GOTO	Executor 412 Fader Red	XFade1Btn Button
39	ALIGN	Executor 415 Button	Executor 111 Button
40	COPY	Executor 414 Button	Executor 101 Button
41	OFF	Executor 413 Button	Executor 102 Button

Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
42	FULL	Executor 411 Fader Green	Executor 103 Button
43	PLEASE	Executor 412 Fader Green	Executor 104 Button
44	NUM4	Executor 413 Fader Green	DEF_GOBACK
45	NUM5	Executor 414 Fader Green	XFade1 Fader Red
46	UNDO	Executor 415 Fader Green	XFade1 Fader Green
47	GROUP	Executor 311 Fader Green	XFade1 Fader Blue
48	THRU	Executor 312 Fader Green	Executor 312 Fader Red
49	NUM6	Executor 313 Fader Green	Executor 311 Fader Red
50	NUM2	Executor 314 Fader Green	Executor 413 Fader Red
51	STORE	Executor 315 Fader Green	Executor 411 Button
52	ASSIGN	Executor 312 Button	Executor 412 Button
53	AT	Executor 313 Fader Red	Executor 414 Fader Red
54	MA1	Executor 311 Button	Executor 411 Fader Red
55	SLASH	Executor 411 Fader Blue	Executor 412 Fader Red
56	NUM1	Executor 412 Fader Blue	Executor 415 Button
57	CUE	Executor 413 Fader Blue	Executor 414 Button
58	TIME	Executor 414 Fader Blue	Executor 413 Button
59	SEQUENCE	Executor 415 Fader Blue	Executor 411 Fader Green
60	CHANNEL	Executor 311 Fader Blue	Executor 412 Fader Green
61	NUM7	Executor 312 Fader Blue	Executor 413 Fader Green
62	NUM8	Executor 313 Fader Blue	Executor 414 Fader Green
63	NUM9	Executor 314 Fader Blue	Executor 415 Fader Green
64	NUM3	Executor 315 Fader Blue	Executor 311 Fader Green
65	MINUS	Executor 313 Button	Executor 312 Fader Green
66	NUM0	Executor 315 Fader Red	Executor 313 Fader Green
67	DOT	Executor 314 Fader Red	Executor 314 Fader Green
68	IF	Executor 314 Button	Executor 315 Fader Green
69	PLUS	Executor 315 Button	Executor 312 Button
70	LEARN	Executor 415 Fader Red	Executor 313 Fader Red
71	Executor 297 Knob Red	Executor 307 Fader Red	Executor 311 Button
72	Executor 295 "X5 | Step"	Executor 306 Fader Red	Executor 411 Fader Blue
73	Executor 296 "X6 | TC"	Executor 408 Fader Red	Executor 412 Fader Blue
74	Executor 298 Knob Red	Executor 406 Button	Executor 413 Fader Blue
75	Executor 295 Knob Red	Executor 407 Button	Executor 414 Fader Blue
76	Executor 296 Knob Red	Executor 409 Fader Red	Executor 415 Fader Blue
77	GOFAST	Executor 406 Fader Red	Executor 311 Fader Blue
78	Executor 195 "X13 | Phaser"	Executor 407 Fader Red	Executor 312 Fader Blue
79	Executor 196 "X14 | Macro"	Executor 410 Button	Executor 313 Fader Blue
80	Executor 295 Knob Green	Executor 409 Button	Executor 314 Fader Blue
81	Executor 296 Knob Green	Executor 408 Button	Executor 315 Fader Blue
82	Executor 297 Knob Green	Executor 406 Fader Green	Executor 313 Button
83	Executor 298 Knob Green	Executor 407 Fader Green	Executor 315 Fader Red
84	DELETE	Executor 408 Fader Green	Executor 314 Fader Red
85	Executor 197 "X15 | Page"	Executor 409 Fader Green	Executor 314 Button
86	GOBACKFAST	Executor 410 Fader Green	Executor 315 Button
87	Executor 298 "X8 | DMX"	Executor 306 Fader Green	Executor 415 Fader Red
88	STOMP	Executor 307 Fader Green	Executor 307 Fader
89	SELECT	Executor 308 Fader Green	Executor 306 Fader Red
90	Executor 295 Knob Blue	Executor 309 Fader Green	Executor 408 Fader Red
91	Executor 296 Knob Blue	Executor 310 Fader Green	Executor 406 Button
92	Executor 297 Knob Blue	Executor 307 Button	Executor 407 Button
93	Executor 298 Knob Blue	Executor 308 Fader Red	Executor 409 Fader Red

Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
94	Executor 198 "X16 | Exec"	Executor 306 Button	Executor 406 Fader Red
95	Executor 297 "X7 | View"	Executor 406 Fader Blue	Executor 407 Fader Red
96	ON	Executor 407 Fader Blue	Executor 410 Button
97	MOVE	Executor 408 Fader Blue	Executor 409 Button
98	FIXTURE	Executor 409 Fader Blue	Executor 408 Button
99	PRESET	Executor 410 Fader Blue	Executor 406 Fader Green
100	EDIT	Executor 306 Fader Blue	Executor 407 Fader Green
101	UPDATE	Executor 307 Fader Blue	Executor 408 Fader Green
102	PAUSE	Executor 308 Fader Blue	Executor 409 Fader Green
103	GOBACK	Executor 309 Fader Blue	Executor 410 Fader Green
104	Executor 293 Knob Red	Executor 310 Fader Blue	Executor 306 Fader Green
105	SOLO	Executor 308 Button	Executor 307 Fader Green
106	HIGHLIGHT	Executor 310 Fader Red	Executor 308 Fader Green
107	Executor 294 Knob Red	Executor 309 Fader Red	Executor 309 Fader Green
108	Executor 291 Knob Red	Executor 309 Button	Executor 310 Fader Green
109	Executor 292 Knob Red	Executor 310 Button	Executor 307 Button
110	GO	Executor 410 Fader Red	Executor 308 Fader Red
111	LIST	Executor 302 Fader Red	Executor 306 Button
112	PAGE_DOWN	Executor 301 Fader Red	Executor 406 Fader Blue
113	Executor 291 Knob Green	Executor 403 Fader Red	Executor 407 Fader Blue
114	Executor 292 Knob Green	Executor 401 Button	Executor 408 Fader Blue
115	Executor 293 Knob Green	Executor 402 Button	Executor 409 Fader Blue
116	Executor 294 Knob Green	Executor 404 Fader Red	Executor 410 Fader Blue
117	SELFIX	Executor 401 Fader Red	Executor 306 Fader Blue
118	MA2	Executor 402 Fader Red	Executor 307 Fader Blue
119	PAGE_UP	Executor 405 Button	Executor 308 Fader Blue
120	XKEYS	Executor 404 Button	Executor 309 Fader Blue
121	BLIND	Executor 403 Button	Executor 310 Fader Blue
122	Executor 192 "X10"	Executor 401 Fader Green	Executor 308 Button
123	Executor 193 "X11"	Executor 402 Fader Green	Executor 310 Fader Red
124	Executor 194 "X12"	Executor 403 Fader Green	Executor 309 Fader Red
125	Executor 291 Knob Blue	Executor 404 Fader Green	Executor 309 Button
126	Executor 292 Knob Blue	Executor 405 Fader Green	Executor 310 Button
127	Executor 293 Knob Blue	Executor 301 Fader Green	Executor 410 Fader Red
128	Executor 294 Knob Blue	Executor 302 Fader Green	Executor 302 Fader Red
129	PREVIEW	Executor 303 Fader Green	Executor 301 Fader Red
130	FREEZE	Executor 304 Fader Green	Executor 403 Fader Red
131	DOWN	Executor 305 Fader Green	Executor 401 Button
132	PREV	Executor 302 Button	Executor 402 Button
133	RESET	Executor 303 Fader Red	Executor 404 Fader Red
134	UP	Executor 301 Button	Executor 401 Fader Red
135	Executor 291 "X1 | Clone"	Executor 401 Fader Blue	Executor 402 Fader Red
136	Executor 292 "X2 | Link"	Executor 402 Fader Blue	Executor 405 Button
137	Executor 293 "X3 | Grid"	Executor 403 Fader Blue	Executor 404 Button
138	Executor 294 "X4 | Layout"	Executor 404 Fader Blue	Executor 403 Button
139	Executor 191 "X9"	Executor 405 Fader Blue	Executor 401 Fader Green
140	NEXT	Executor 301 Fader Blue	Executor 402 Fader Green
141	All LEDs on the Keyboard	Executor 302 Fader Blue	Executor 403 Fader Green
142	Small Screen Backlight	Executor 303 Fader Blue	Executor 404 Fader Green
143	Letterbox Screen Backlight	Executor 304 Fader Blue	Executor 405 Fader Green
144		Executor 305 Fader Blue	Executor 301 Fader Green
145		Executor 303 Button	Executor 302 Fader Green

Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
146		Executor 305 Fader Red	Executor 303 Fader Green
147		Executor 304 Fader Red	Executor 304 Fader Green
148		Executor 304 Button	Executor 305 Fader Green
149		Executor 305 Button	Executor 302 Button
150		Executor 405 Fader Red	Executor 303 Fader Red
151		RateBtn2	Executor 301 Button
152		ExecBtn1	Executor 401 Fader Blue
153		SpeedBtn1	Executor 402 Fader Blue
154		RateBtn1	Executor 403 Fader Blue
155		SpeedBtn2	Executor 404 Fader Blue
156		ProgBtn1	Executor 405 Fader Blue
157		ProgBtn2	Executor 301 Fader Blue
158		ProgBtn3	Executor 302 Fader Blue
159		ExecBtn3	Executor 303 Fader Blue
160		ExecBtn2	Executor 304 Fader Blue
161		Executor 201 Fader Red	Executor 305 Fader Blue
162		Executor 201 Fader Green	Executor 303 Button
163		Executor 201 Fader Blue	Executor 305 Fader Red
164		Executor 202 Fader Red	Executor 304 Fader Red
165		Executor 202 Fader Green	Executor 304 Button
166		Executor 202 Fader Blue	Executor 305 Button
167		Executor 203 Fader Red	Executor 405 Fader Red
168		Executor 203 Fader Green	Executor 201 Fader Red
169		Executor 203 Fader Blue	Executor 201 Fader Green
170		Executor 204 Fader Red	Executor 201 Fader Blue
171		Executor 204 Fader Green	Executor 202 Fader Red
172		Executor 204 Fader Blue	Executor 202 Fader Green
173		Executor 205 Fader Red	Executor 202 Fader Blue
174		Executor 205 Fader Green	Executor 203 Fader Red
175		Executor 205 Fader Blue	Executor 203 Fader Green
176		Executor 206 Fader Red	Executor 203 Fader Blue
177		Executor 206 Fader Green	Executor 204 Fader Red
178		Executor 206 Fader Blue	Executor 204 Fader Green
179		Executor 207 Fader Red	Executor 204 Fader Blue
180		Executor 207 Fader Green	Executor 205 Fader Red
181		Executor 207 Fader Blue	Executor 205 Fader Green
182		Executor 208 Fader Red	Executor 205 Fader Blue
183		Executor 208 Fader Green	Executor 206 Fader Red
184		Executor 208 Fader Blue	Executor 206 Fader Green
185		Executor 209 Fader Red	Executor 206 Fader Blue
186		Executor 209 Fader Green	Executor 207 Fader Red
187		Executor 209 Fader Blue	Executor 207 Fader Green
188		Executor 210 Fader Red	Executor 207 Fader Blue
189		Executor 210 Fader Green	Executor 208 Fader Red
190		Executor 210 Fader Blue	Executor 208 Fader Green
191		Executor 211 Fader Red	Executor 208 Fader Blue
192		Executor 211 Fader Green	Executor 209 Fader Red
193		Executor 211 Fader Blue	Executor 209 Fader Green
194		Executor 212 Fader Red	Executor 209 Fader Blue
195		Executor 212 Fader Green	Executor 210 Fader Red
196		Executor 212 Fader Blue	Executor 210 Fader Green
197		Executor 213 Fader Red	Executor 210 Fader Blue

Index	grandMA3 Master Module(MM)	grandMA3 Fader Module Encoder(MFE)	grandMA3 Fader Module Crossfader(MFX)
198		Executor 213 Fader Green	Executor 211 Fader Red
199		Executor 213 Fader Blue	Executor 211 Fader Green
200		Executor 214 Fader Red	Executor 211 Fader Blue
201		Executor 214 Fader Green	Executor 212 Fader Red
202		Executor 214 Fader Blue	Executor 212 Fader Green
203		Executor 215 Fader Red	Executor 212 Fader Blue
204		Executor 215 Fader Green	Executor 213 Fader Red
205		Executor 215 Fader Blue	Executor 213 Fader Green
206		Desklights	Executor 213 Fader Blue
207		Letterbox Screen Backlight	Executor 214 Fader Red
208		Small Screen Backlight	Executor 214 Fader Green
209			Executor 214 Fader Blue
210			Executor 215 Fader Red
211			Executor 215 Fader Green
212			Executor 215 Fader Blue
213			Desklights
214			Letterbox Screen Backlight
215			Small Screen Backlight

1.44.5.119. SetProgress(handle, integer)

Description

The SetProgress Lua function defines a value on the range for a progress bar. A handle input argument defines the progress bar. The progress bar needs have been created using the **StartProgress** function.

See the **ProgressBar topic** for more info regarding progress bars and links to other related functions.

Arguments

- **Handle**:
  The handle for the progress bar.
- **Integer**:
  The desired value indicating the current status or position of the progress bar.

Return

This function does not return anything.

Example

This example sets a range value for the progress bar created using the example in the StartProgress topic (link above):

Lua
```
return function()
    -- Sets the current value to 5 for a progress bar with the matching
handle
    SetProgress(progressHandle, 5)
end
```

1.44.5.120. SetProgressRange(handle, integer, integer)

Description

The SetProgressRange Lua function defines a range for a progress bar.

A handle input argument defines which progress bar it defines a range for. The progress bar must exist to have a handle. Progress bars can be created using the **StartProgress** function.

See the **ProgressBar topic** for more info regarding progress bars and links to other related functions.

Arguments

- **Handle**:
  The handle for the progress bar.
- **Integer**:
  The start value for the range.
- **Integer**:
  The end value for the range.

Return

This function does not return anything.

Example

This example sets a range for the progress bar created using the example in the StartProgress topic (link above):

Lua

```
return function()
    -- Sets the range of a progress bar with the matching handle
    SetProgressRange(progressHandle, 1, 10)
end
```

1.44.5.121. SetProgressText(handle, string)

Description

The SetProgressText Lua function defines a text string to be displayed in a progress bar next to the progress bar title text. The title cannot be changed after creation, but this text can be changed. It could be used to describe the current step in the progress.

A handle input argument defines which progress bar it defines a text for. The progress bar must exist for it to have a handle. Progress bars can be created using the **StartProgress** function.

See the **ProgressBar topic** for more info regarding progress bars and links to other related functions.

Arguments

- **Handle**:
  The handle for the progress bar.
- **String**:
  The text string to be displayed.

Return

This function does not return anything.

Example

This example sets a text string for the progress bar created using the example in the StartProgress topic (link above):

Lua

```
return function()
    -- Sets the text next to progress title
    SetProgressText(progressHandle, "- This is text next to the progress
title")
end
```

1.44.5.122. SetVar(handle, string, value)

Description

The SetVar Lua function sets a value to a specific variable in a set of variables. To learn more about the variables in plugins, look at the **Variable Functions** topic.

If the variable exists, then the value is overwritten. If it does not exist, then it is created with value.

Arguments

- **Handle**:
  The handle of variable set.
- **String**:
  The name of the variable. It needs to be in quotation marks.
- **Value**:
  The value can be a string, integer, or double.

Return

- **Boolean**:
  - True / 1: The variable was set.
  - False / 0: The variable was not set.

Example

This example sets a value to the variable called "myUserVar" in the set of user variables if it exists.

Lua

```
return function()
    -- Sets the value of user variable "myUserVar" to "Hello World" and store
the returned boolean in a Lua variable
    local success = SetVar(UserVars(), "myUserVar", "Hello World")
    -- Prints the ressult
    if success then
        Printf("Variable is stored.")
    else
        Printf("Variable is NOT stored!")
    end
end
```

1.44.5.123. ShowData()

Description

ShowData is an object-free function that returns a handle to the object at position Root/ShowData.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle to the ShowData object.

Example

This simple example prints the information of the ShowData object in the Command Line History using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- The following prints the dump for the object for the show data
    Printf("=============== START OF DUMP ===============")
    ShowData():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.124. ShowSettings()

Description

ShowSettings is an object-free function that returns a handle to the object at Root/ShowData/ShowSettings.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle to the ShowSettings object.

Example

This simple example prints the information of the ShowSettings object using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function ()
    -- The following prints the dump for the object for the show settings
    Printf("================ START OF DUMP ================")
    ShowSettings():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.125. StartProgress(string)

Description

The StartProgress Lua function creates and displays a progress bar on all screens. A string input argument creates a title for the progress bar. The function returns a handle that is used to further interact with the progress bar.

Executing the function displays the progress bar on the screens. It only disappears using the **StopProgress** function - which needs the handle. So it is highly recommended to store the returned handle from the start function.

See the **ProgressBar topic** for more info regarding progress bars and links to other related functions.

Arguments

- **String**:
  The string is used as the title for the progress bar.

Return

- **Handle**:
  The returned handle is the identifier for the progress bar.

Example

This creates and displays a progress bar on all screens. The progress bar does not disappear using this example - see the example in the StopProgress (link above) function to remove:

Lua

```lua
return function()
    -- Create and display a progress bar with a title
    -- IMPORTANT: The Lua variable 'progressHandle' is needed to remove the progressbar again - StopProgress()
    progressHandle = StartProgress("ProgressBar Title")
end
```

	**Important:**
	Running this example multiple times creates new progress bars that cannot be removed. So only run this once, and then remove it again using the StopProgress function (link above).

1.44.5.126. StopProgress(handle)

Description

The StopProgress Lua function removes a progress bar. A handle input argument defines which progress bar it removes. The progress bar must exist before it can be removed. Progress bars are created using the **StartProgress** function.

See the **ProgressBar topic** for more info regarding progress bars and links to other related functions.

Arguments

- **Handle**:
  The handle for the progress bar to be stopped.

Return

This function does not return anything.

Example

This example stops the progress bar created using the example in the StartProgress topic (link above):

```lua
return function()
    -- Stops and closes the progress bar with the matching handle
    StopProgress(progressHandle)
end
```

1.44.5.127. StrToHandle(string)

Description

The object-free StrToHandle Lua function converts a string with a hexadecimal number format into a handle. The string needs to correlate with an actual handle.

See the **Handle topic** for more info regarding handles and links to other related functions.

Arguments

- **String**:
  The string with a handle number in a hexadecimal format.

Return

- **Handle**:
  The returned handle based on the string with a hexadecimal number.

Example

This example prints the handle hex number for the selected sequence. It also converts the string back to a handle and uses this to print the name of the sequence:

Lua

```
return function()
    -- Store a variable with the string of the handle converted to hex
    local mySeqStr = HandleToStr(SelectedSequence())
    -- Print some feedback with the handle in a string version
    Printf("The handle for the selected sequence (string version): %s",
mySeqStr)
    -- Print some feedback where the string is converted back to a handle
    Printf("The name of the selected sequence is: %s",
StrToHandle(mySeqStr).name)
end
```

1.44.5.128. TextInput([string[, string[, integer[, integer]]]])

Description

The **TextInput** Lua function opens a text input pop-up and returns the typed input as a string. It is part of the user interface functions.

Arguments

- **String** (optional):
  This string is the title for the pop-up. The title bar has a default "Edit" text at the beginning of the title that cannot be removed.
- **String** (optional):
  This string is the text already in the input field - can be used to provide user guidance.
- **Integer** (optional):
  This integer defines a position on the x-axis where the pop-up should appear (on all screens). "0" is on the left side of the screen. Nil or undefined is centered.
- **Integer** (optional):
  This integer defines a position on the y-axis where the pop-up should appear (on all screens). "0" is at the top of the screen. Nil or undefined is centered.

Return

- **String**:
  The returned user input.

Example

To open a text input and print the entered value in the Command Line History, create a plugin with this code:

```Lua
return function()
    -- Create a pop-up with the title and an input field containing some default text
    -- The returned text is store in a Lua variable
    local input = TextInput("This is the title","Please provide your input here")
    -- Print the returned text value
    Printf("You entered this message: %s",tostring(input))
end
```

1.44.5.129. Time()

Description

The **Time** function returns the time (in seconds) the station has been on, as a number (float). It is basically a stopwatch that starts when the grandMA3 application starts. It is not the current time of day or the session online time.

Arguments

This function does not accept any arguments.

Return

- **Number**:
  The returned number (float) is the on-time for the station.

Example

This example finds, formats, and prints the time.

Lua

```lua
return function()
    -- Get the current time
    local time = Time()

    --Calculate the different elements
    local days = math.floor(time/86400)
    local hours = math.floor((time % 86400)/3600)
    local minutes = math.floor((time % 3600)/60)
    local seconds = math.floor(time % 60)

    --Print the result
    Printf("The time is %d:%02d:%02d:%02d", days, hours, minutes, seconds)
end
```

1.44.5.130. Timer(string, integer, integer[, string[, handle]])

Description

The **Timer** Lua function call a different function using a timer. The other function can be called multiple times using the timer interval.

Arguments

- **Function**:
  This is the name of the function that is called multiple times using the timer.
- **Integer**:
  This is the wait time between the calls. The value is in seconds.
- **Integer**:
  This is the number of times the function is called.
- **Function** | **nil** (optional):
  This is an optional argument that is the name of a function that is called when the Timer function is finished.
- **Handle** (optional):
  This is an optional argument for a handle to an object that is passed to the called function.

Return

This function does not return anything.

Example

This example prints a greeting three times and then calls a clean up function:

Lua

```lua
-- Function that will be called several times.
function TimedFunction()
    -- Check the value of RunAmount and print something.
    if RunAmount < 1 then
        Printf("Hello")
    else
        Printf("Hello again")
    end
    -- Add 1 to the RunAmount variable.
    RunAmount = RunAmount + 1
end

-- Cleanup function.
function TimerCleanup()
    Printf("Goodbye")
    -- Delete the RunAmount variable.
    RunAmount = nil
end

-- Function with the Timer call.
```

```
function Main()
    -- Set a wait variable.
    local waitSeconds = 1
    -- Set a variable with the number of iterations.
    local iterations = 3
    -- Create a counter variable and set it to 0.
    RunAmount = 0
    -- Call the timer function.
    Timer(TimedFunction, waitSeconds, iterations, TimerCleanup);
end

-- call the main function.
return Main
```

1.44.5.131. ToAddr(handle[, boolean])

Description

The **ToAddr** Lua object-free function converts a handle to an address string that can be used in commands.

See the Handle topic for more info regarding handles, addresses, and links to other related functions.

Arguments

- **Handle**:
  The function takes a handle of an object as an argument.
- **Boolean** (optional):
  This returns the address using the names instead of numbers. The default is False, which returns the number version of the address.

Return

- **String**:
  String with the address value.

Example

This example prints the address of the selected sequence in both the numbered and named versions.

Lua
```
return function ()
    local mySequence = SelectedSequence()
    -- Print the address to the selected sequence in number and name format.
    Printf(ToAddr(mySequence))
    Printf(ToAddr(mySequence, true))
end
```

1.44.5.132. TouchObj()

Description

The **TouchObj** function returns the handle to the first found touch object.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns the handle to the touch object.

Example

This example prints information about the touch object using the **Dump() function**:

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- Print all informatin about the TouchObj object
    Printf("=============== START OF DUMP ===============")
    TouchObj():Dump()
    Printf("================ END OF DUMP ================")
end
```

1.44.5.133. Unhook(integer)

Description

The **Unhook** Lua function removes a hook.

Hooks are an automatically triggered function that activates when a grandMA3 object changes. A hook can be created using the **HookObjectChange** function.

Arguments

- **Integer**:
  This must be the integer matching the hook that should be unhooked.

	**Hint:**
	All hooks can be listed using the **DumpAllHooks** function, but this does not reveal the corresponding hook integer ID. Use the **UnhookMultiple** function if the integer is unknown.

Return

This function does not return anything.

	**Hint:**
	See also these related functions: **DumpAllHooks**, **HookObjectChange**, **UnhookMultiple**.

Example

This example unhooks the hook created using the example in the HookObjectChange - please run that example before this one.

Lua

```
return function()
    -- Unhooks the specific Hook integer ID.
    Unhook(SequenceHookId)
end
```

1.44.5.134. UnhookMultiple(function, handle, handle)

Description

The **UnhookMultiple** Lua function unhooks multiple hooks based on an input. This input acts like a filter to identify all the hooks that should be unhooked.

The **DumpAllHooks** function can be used to list all the existing hooks in the system.

Arguments

- **Function** or **nil**:
  This must be the name of a triggered function or nil.
- **Handle** or **nil**:
  This must be the handle for the target object or nil.
- **Handle** or **nil**:
  The must be the handle for the context object or nil.

The target and context names can be seen using the **DumpAllHooks** function.

Return

- **Integer**:
  The function returns an integer indicating how many hooks were unhooked.

	**Hint:**
	See also these related functions: **DumpAllHooks**, **HookObjectChange**, **Unhook**.

Example

This example unhooks all hooked related to the function created in the example for the **HookObjectChange** - please run the example from that topic before running this one.

Lua

```
return function ()
    -- Unhooks all hooks related to the "MySequencePoolCallback" function.
    local amount = UnhookMultiple(MySequencePoolCallback)
    -- Print how many hooks that were unhooked.
    Printf(amount .. " hook(s) were unhooked.")
end
```

1.44.5.135. UserVars()

Description

The UserVars function returns a handle to the set of user variables. Read more about these in the **Variables** topic in the Macro section.

Arguments

This function does not accept any arguments.

Return

- **Handle**:
  The function returns a handle of the set of user variables.

Example

This example sets, gets, and deletes a user variable:

```lua
return function()
    -- Stores a local Lua variable with the handle for the user variables.
    local variableSection = UserVars()

    -- Sets a user variable with an integer value using the SetVar function.
    SetVar(variableSection, "myUserVar", 42)

    -- Prints the user variable using the GetVar function.
    Printf("The value  of myUserVar is: " .. GetVar(variableSection,
"myUserVar"))

    -- Deletes the user variable using the DelVar function.
    DelVar(variableSection, "myUserVar")
end
```

1.44.5.136. Version()

Description

The Version Lua function returns the software version.

Arguments

This function does not accept any arguments.

Return

- **String**:
  The returned string is the version of the grandMA3 software.

Example

This example prints the software version in the Command Line History:

```lua
return function()
    Printf("Software version: %s", Version())
end
```

1.44.6. Lua Functions - Object API

Object API means Lua functions that are functions/methods of an object.

All of the object functions take a handle as an argument. This can often be omitted if the function is used with the colon notation. This is the most common use with object functions.

Syntax with the handle:

object.function(object-handle)

Syntax with colon operator:

object:function()

Most examples in the object subtopics use the colon operator notation.

Subtopics

- **Addr(handle[, handle[, boolean]])**
- **AddrNative(handle[, handle[, boolean]])**
- **Children(handle)**
- **Count(handle)**
- **Dump (handle)**
- **Export(handle, string, string)**
- **Get(handle, string[, integer])**
- **GetChildClass(handle)**
- **GetClass(handle)**
- **GetDependencies(handle)**
- **HasActivePlayback(handle)**
- **GetFader(handle, {[string],[integer]})**
- **GetFaderText(handle, {[string], [integer]})**
- **Import(handle, string, string)**
- **Ptr(handle, integer)**
- **GetReferences(handle)**
- **GetUIEditor(handle)**
- **GetUISettings(handle)**
- **ToAddr(handle)**
- **SetFader(handle, {[number], [boolean], [string]})**

1.44.6.1. Addr(handle[, handle[, boolean]])

Description

The **Addr** Lua object function converts a handle to an address string that can be used in commands.

See the **Handle topic** for more info regarding handles and links to other related functions.

Arguments

- **Handle**:
  The function takes a handle of the type "light_userdata" as an argument. This is the handle to the object where the address is requested.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

- **Handle** (optional):
  The returned address is from the root as a default. This optional handle can specify a different base location. It must still be a base location in the address path from the root to the object.
- **Boolean | nil** (optional):
  This can be useful if there is a difference between the **ToAddr()** and Addr(). Setting this to "true" uses the index number from the ToAddr() instead of the Addr() index number. See the example below.

  The ToAddr() object function returns the address as a text string using names. Learn more in the **ToAddr() topic**.

- **Boolean** (optional):
  In some edge cases, the cue address is not resolved correctly. Setting this boolean to true will fix this.

Return

- **String**:
  Text string with the address in a parent-child number format separated by dots.

Example

This example prints different versions of the address to a cue in a sequence:

Lua

```
return function()
    -- Creates a cue in sequence 1
    Cmd("Store Sequence 1 Cue 100 /Merge /NoConfirmation")
    --Store a handle to the created cue
    local cueObject = ObjectList("Sequence 1 Cue 100")[1]
    --Print different version of the handle address
    Printf("ToAddr:                " .. cueObject:ToAddr())
    Printf("Addr:                  " .. cueObject:Addr())
    Printf("Addr(Parent, false, false): " ..
cueObject:Addr(cueObject:Parent(), false, false))
    Printf("Addr(Parent, true, false):  " ..
cueObject:Addr(cueObject:Parent(), true, false))
    Printf("Addr(Parent, false, true): " ..
cueObject:Addr(cueObject:Parent(), false, true))
    Printf("Addr(Parent, true, true):  " ..
cueObject:Addr(cueObject:Parent(), true, true))
end
```

1.44.6.2. AddrNative(handle[, handle[, boolean]])

Description

The AddrNative Lua object function converts a handle to an address string that can be used in commands.

See the **Handle topic** for more info regarding handles and links to other related functions.

Arguments

- **Handle**:
  The function takes a handle as an argument. This is the handle to the object where the address is requested.
  It can be omitted when using the **colon notation** on an object.

  > The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
  >
  > This is the general syntax with the colon notation: **object:function()**
  >
  > This is the general syntax with standard handle notation: **object.function(object)**
  >
  > Learn more in the **Lua Functions - Object API topic**.

- **Handle** (optional):
  The returned address is from the root as a default. This optional handle can specify a different base location. It still needs to be a base location in the address path from the root to the object.
- **Boolean** (optional):
  Set this to "true" to get the returned names in quotation marks.

Return

- **String**:
  Text string with the address in a parent-child name format separated by dots.

Example

This example prints the address of the first sequence:

```lua
return function()
    -- Stores the handle to the first sequence.
    local mySequence = DataPool().Sequences[1]
    -- Print the native address.
    Printf("The full address is: " .. mySequence:AddrNative())
    -- Stores a handle to the default DataPool.
    local myDataPool = DataPool()
    -- Print the native address to the datapool using the default datapool as
```

```
a base.
    Printf("The address in the datapool is: " ..
mySequence:AddrNative(myDataPool))
    -- Print the native address to the datapool, using the default datapool
as a base, with names as strings.
    Printf("The address in the datapool with quotes around the names is: " ..
mySequence:AddrNative(myDataPool, true))
end
```

1.44.6.3. Children(handle)

Description

The **Children** Lua function creates a table of handles for the children of an object.

Arguments

- **Handle**:
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object

> The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
>
> This is the general syntax with the colon notation: **object:function()**
>
> This is the general syntax with standard handle notation: **object.function(object)**
>
> Learn more in the **Lua Functions - Object API topic**.

Return

- **Table**:
  The function returns a table with handles for the child objects. If there are no children, then it returns an empty table.

Example

This example returns the name of the cues in the first sequence of the selected data pool:

Lua

```
return function()
    -- Stores the handle for sequence 1 in a variable.
    local mySequence = DataPool().Sequences[1]
    if mySequence ~= nil then
            -- Use the "Children()" funciton to store a table with all the
children in a new variable.
        local cues = mySequence:Children()
        -- For loop that uses the length operator on the cue variable.
        for i = 1, #cues do
            -- Text is printed for each child.
            Printf("Sequence 1 Child " .. i .. " = " .. cues[i].name)
        end
    else
        ErrPrintf("Sequence could not be found.")
    end
end
```

1.44.6.4. Count(handle)

Description

The **Count** function returns an integer number indicating the number of child objects.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

Return

- **Integer**:
  The function returns an integer indicating the number of children of the object.

Examples

This example prints the selected sequence's number of children (cues).

```lua
return function()
    local numberChildren = SelectedSequence():Count()
    Printf("The selected Sequence has " .. numberChildren .. " cues.")
end
```

1.44.6.5. Dump (handle)

Description

The **Dump** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** for object-oriented calls.

> The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
>
> This is the general syntax with the colon notation: **object:function()**
>
> This is the general syntax with standard handle notation: **object.function(object)**
>
> Learn more in the **Lua Functions - Object API topic**.

Return

The function returns nothing but outputs information about the object in the **Command Line History window**.

Examples

These examples all print information about the selected sequence in the Command Line History.

The first example using the colon operator:

```lua
return function ()
    -- Dump() is called on a function
    Printf("=============== START OF DUMP ===============")
    SelectedSequence():Dump()
    Printf("================ END OF DUMP ================")
end
```

The second example uses a variable with the same result:

```lua
return function ()
    --Stores the handle for the selected sequence in a local variable.
    local mySeqHandle = SelectedSequence()
    -- Dump() is called on the variable.
    Printf("=============== START OF DUMP ===============")
    mySeqHandle:Dump()
    Printf("================ END OF DUMP ================")
```

```
end
```

1.44.6.6. Export(handle, string, string)

Description

The **Export** object Lua function exports an object into an XML file.

Arguments

- **Handle**:
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  > The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
  >
  > This is the general syntax with the colon notation: **object:function()**
  >
  > This is the general syntax with standard handle notation: **object.function(object)**
  >
  > Learn more in the **Lua Functions - Object API topic**.

- **String**:
  This is a string with the file path for the exported file.
- **String**:
  This is a string containing the file name of the exported file.

Return

- **Boolean**:
  The function returns a boolean indicating if the export was a success.

Examples

This example exports the selected sequence into an XML file:

Lua

```
return function()
    --SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    if selectedSequence == nil then
        ErrPrintf("The selected sequence could not be found.")
        return
    end
    --The path is stored in a variable.
    local exportPath = GetPath(Enums.PathType.UserSequences)
    --The actual export function.
    local success = selectedSequence:Export(exportPath,
"mySelectedSequence.xml")
    --Print some feedback.
    if success then
        Printf("The sequence is exported to: " .. exportPath)
```

```
    else
        ErrPrintf("The sequence could not be exported.")
    end
end
```

Related Object Functions

Import - object function used to import an XML table.

1.44.6.7. Get(handle, string[, integer])

Description

The **Get** function returns a string with information about a specified property of the object, for instance, the object's name, class, or path.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  > The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
  >
  > This is the general syntax with the colon notation: **object:function()**
  >
  > This is the general syntax with standard handle notation: **object.function(object)**
  >
  > Learn more in the **Lua Functions - Object API topic**.

- **String**:
  The string must be the name of a valid property for the object.
- **Integer** (optional):
  A valid role integer can be supplied. This will make the returned value a text string.

Return

- **String**:
  The function returns the value of the property. If the property is a boolean, then the return is "0" or "1" unless a role is defined (see optional integer argument above). When the role is supplied, a boolean is returned as "No" or "Yes".

Examples

This example prints information about the "Tracking" property of the selected sequence.

Lua

```
return function ()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Check if there is a selected sequence. If not, then exit the function.
    if selectedSequence == nil then
        ErrPrintf("The selected sequence could not be found.")
        return
    end
    -- Set a variable with the property name.
    local propertyName = "Tracking"
    -- Get the value of the property.
    local propertyValue = selectedSequence:Get(propertyName)
    local propertyValueString = selectedSequence:Get(propertyName,
```

```
Enums.Roles.Edit)
    -- Return some feedback.
    if propertyValue ~= nil then
        Printf("The selected sequence's property " .. propertyName.. " has
the value '" .. propertyValue .. "' and a string value of '" ..
propertyValueString .. "'.")
    else
        ErrPrintf("The property could not be found.")
    end
end
```

1.44.6.8. GetChildClass(handle)

Description

The **GetChildClass** function returns a string with the name of the class of the object's children.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object

> The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
>
> This is the general syntax with the colon notation: **object:function()**
>
> This is the general syntax with standard handle notation: **object.function(object)**
>
> Learn more in the **Lua Functions - Object API topic**.

Return

- **String**:
  The function returns a text string with the name of the class of the object's children.

Examples

This example prints the class name of the selected sequences' children.

Lua

```
return function()
    -- Gets the class name of children of the selected sequence.
    Printf("The class name is " .. SelectedSequence():GetChildClass())
end
```

1.44.6.9. GetClass(handle)

Description

The **GetClass** function returns a string with information about the class for the object.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

Return

- **String**:
  The function returns the text string with the name of the object's class.

Examples

This example prints the class name of the selected sequence.

Lua

```
return function()
    -- Gets the class name of the selected sequence.
    Printf("The class name is " .. SelectedSequence():GetClass())
end
```

1.44.6.10. GetDependencies(handle)

Description

The **GetDependencies** function returns a table with the objects' dependencies.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

> The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
>
> This is the general syntax with the colon notation: **object:function()**
>
> This is the general syntax with standard handle notation: **object.function(object)**
>
> Learn more in the **Lua Functions - Object API topic**.

Return

- **Table**:
  The function returns a table with the handles for the different dependency objects.

Examples

This example prints a **dump** of the selected sequence's first object in the returned table.

> **Dump()**
>
> The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.
>
> Learn more in the **Dump() topic**.

Lua

```
return function()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Get the dependcies for the sequence.
    local seqDependencies = selectedSequence:GetDependencies()
    -- Check if there are any dependicies and output a relevant feedback.
    if seqDependencies ~= nil then
        -- There is a dependency table returned. Print a dump of the first
table element.
        Printf("=============== START OF DUMP ===============")
        seqDependencies[1]:Dump()
```

```
        Printf("================ END OF DUMP ================")
    else
        Printf("No dependencies found")
    end
end
```

1.44.6.11. GetFader(handle, {[string],[integer]})

Description

The **GetFader** function returns a float number indicating a fader position for the object.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument. It can be omitted when using the colon notation on an object. See the examples below.
- **Table**:
  The table can contain two different elements: Token and Index. The important element is the token.
  - **Token**: This is used to specify which fader the value is requested for. These are the valid values:
    - FaderMaster
    - FaderX
    - FaderXA
    - FaderXB
    - FaderTemp
    - FaderRate
    - FaderSpeed
    - FaderHighlight
    - FaderLowlight
    - FaderSolo
    - FaderTime

Return

- **Number**:
  The function returns a float number indicating the fader position.

Examples

This example prints the fader positions of the Master and Rate faders for the selected sequence.

Lua

```
return function()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Get the value for the Master fader.
    local faderMasterValue = selectedSequence:GetFader({})
    -- Get the value for the Rate fader.
    local faderRateValue = selectedSequence:GetFader({token="FaderRate"})
    -- Print feedback with the values.
    Printf("The selected sequence Master fader value is: "..
tostring(faderMasterValue))
    Printf("The selected sequence Rate fader value is: "..
tostring(faderRateValue))
```

```
end
```

1.44.6.12. HasActivePlayback(handle)

Description

The **HasActivePlayback** Lua function returns a boolean indicating if an object has a currently active playback, for instance, if a sequence has an active cue.

Arguments

- **Handle**:
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

Return

- **Boolean**:
  The function returns a boolean indicating the playback status:
  - **True**: There is active playback.
  - **False**: There is no active playback.

Example

To return the information if the selected sequence has an active playback, create a plugin with this code:

Lua

```lua
return function()
    -- Stores the handle of the selected sequence.
    local selectedSequence = SelectedSequence()

    -- The following 'if' gives different feedback based on the playback
status.
    if selectedSequence:HasActivePlayback() then
        Printf("Sequence '" ..selectedSequence.name.. "' has active
playback.")
    else
        Printf("Sequence '" ..selectedSequence.name.. "' has NO active
playback.")
    end
end
```

1.44.6.13. GetFaderText(handle, {[string], [integer]})

Description

The **GetFaderText** function returns a text string indicating a fader value for the object.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument. It can be omitted when using the colon notation on an object. See the examples below.
- **Table**:
  The table can contain two different elements: Token and Index. The important element is the token.
  - **Token**: This is used to specify which fader the value is requested for. These are the valid values:
    - FaderMaster
    - FaderX
    - FaderXA
    - FaderXB
    - FaderTemp
    - FaderRate
    - FaderSpeed
    - FaderHighlight
    - FaderLowlight
    - FaderSolo
    - FaderTime

Return

- **String**:
  The function returns a text string indicating the fader value.

Examples

This example prints the fader value text of the Master and Rate faders for the selected sequence.

Lua

```
return function()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Get the value for the Master fader. Since it is the default, no token
needs to be defined.
    local faderMasterText = selectedSequence:GetFaderText({})
    -- Get the value for the Rate fader.
    local faderRateText = selectedSequence:GetFaderText({token="FaderRate"})
    -- Print feedback with the values.
    Printf("The selected sequence Master fader value text is: "..
tostring(faderMasterText))
    Printf("The selected sequence Rate fader value text is: "..
```

```
tostring(faderRateText))
end
```

1.44.6.14. Import(handle, string, string)

Description

The Import object Lua function imports an object written in XML format.

	Restriction:
	The imported files need to exist already to be imported.
	Important:
	The Lua import will merge the content of the XML file into the object without any confirmation pop-up.

Arguments

- **Handle**:
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

- **String**:
  This is a string with the path to the file location.
- **String**:
  This is a string containing the file name of the desired file.

Return

- **Boolean**:
  The function returns a boolean indicating if the import was a success.

Example

This example imports the content of an XML file into the selected sequence. The file is called "MySelectedSequence", and it is located at "../gma3_library/datapools/sequences". The file can be created using the example in the **Export object function**.

Lua

```
return function()
    --SelectedSequence() creates a handle to the selected sequence.
    -- The imported object will be merged into this sequence.
    local selectedSequence = SelectedSequence()
    -- Check if there is a selected sequence - if not then exit the function.
    if selectedSequence == nil then
```

```
        ErrPrintf("The selected sequence could not be found.")
        return
    end
    --The path is stored in a variable.
    local path = GetPath(Enums.PathType.UserSequences)
    --The actual import function.
    local success = selectedSequence:Import(path, "mySelectedSequence.xml")
    --Print some feedback.
    if success then
        Printf("The sequence is imported from: " .. path ..
GetPathSeparator() .. "mySelectedSequence.xml")
    else
        ErrPrintf("The object could not be imported.")
    end
end
```

Related Object Function

Export - object function used to export an XML table.

1.44.6.15. Ptr(handle, integer)

Description

The **Ptr** Lua function returns the handle to a child object.

Arguments

- **Handle**:
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

- **Integer**:
  This is the index number for the desired child object. This index is 1-based.

Return

- **Handle** | **nil**:
  The function returns a handle to the child object. If the child object does not exist, then it returns nil.

Example

This example prints the data connected to the first child of the selected sequence. It uses the **Dump() function**.

Dump()

The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.

Learn more in the **Dump() topic**.

Lua

```
return function()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Check that a handle was returned - if not then exit function.
```

```
    if selectedSequence == nil then
        ErrPrintf("There is no selected sequence.")
        return
    end
    -- Get a handle to the first child object.
    local firstChild = selectedSequence:Ptr(1)
    -- Print some feedback.
    if firstChild ~= nil then
        Printf("=============== START OF DUMP ===============")
        firstChild:Dump()
        Printf("================ END OF DUMP ================")
    else
        ErrPrintf("The object do not have a child object.")
    end
end
```

1.44.6.16. GetReferences(handle)

Description

The **GetReferences** function returns a table with handles for the objects referencing this object.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

> The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
>
> This is the general syntax with the colon notation: **object:function()**
>
> This is the general syntax with standard handle notation: **object.function(object)**
>
> Learn more in the **Lua Functions - Object API topic**.

Return

- **Table**:
  The function returns a table with the handles for the different objects referencing this object.

Examples

This example prints a **dump** of the selected sequence's first object in the returned table.

> **Dump()**
>
> The **Dump()** function returns a string with information about the object, for instance, the name, class, path of the object, its properties, and children.
>
> Learn more in the **Dump() topic**.

Lua

```
return function()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Get the references for the sequence.
    local seqReferences = selectedSequence:GetReferences()
    -- Check if there are any references and output a relevant feedback.
    if seqReferences ~= nil then
        -- There is a reference table returned. Print a dump of the first
table element.
        Printf("=============== START OF DUMP ===============")
        seqReferences[1]:Dump()
```

```
        Printf("================ END OF DUMP ================")
    else
        Printf("No references found")
    end
end
```

1.44.6.17. GetUIEditor(handle)

Description

The **GetUIEditor** function returns a text string with the name of the UI editor for the object.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

Return

- **String**:
  The function returns a text string with the name of the object's UI editor.

Examples

This example prints the name of the selected sequence's editor.

Lua

```lua
return function()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Get the name of the editor for the sequence object.
    local seqEditor = selectedSequence:GetUIEditor()
    -- Print some feedback.
    if seqEditor ~= nil then
        Printf("The name of the editor is: " .. seqEditor)
    else
        Printf("The object doesn not appear to have an editor.")
    end
end
```

1.44.6.18. GetUISettings(handle)

Description

The **GetUISettings** function returns a text string with the name of the UI settings for the object.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

Return

- **String**:
  The function returns a text string with the name of the object's UI settings.

Examples

This example prints the name of the selected sequence's settings.

Lua

```lua
return function()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Get the name of the editor for the sequence object.
    local seqSettings = selectedSequence:GetUISettings()
    -- Print some feedback.
    if seqSettings ~= nil then
        Printf("The name of the settings is: " .. seqSettings)
    else
        Printf("The object doesn not appear to have an editor.")
    end
end
```

1.44.6.19. ToAddr(handle)

Description

The **ToAddr** Lua object function converts a handle to an address string that can be used in commands.

See the **Handle topic** for more info regarding handles and links to other related functions.

Arguments

- **Handle**:
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  The Colon Notation is a way to omit the handle as the first argument when using the Object functions.

  This is the general syntax with the colon notation: **object:function()**

  This is the general syntax with standard handle notation: **object.function(object)**

  Learn more in the **Lua Functions - Object API topic**.

- **Boolean**:
  Set this to "true" to get the returned name. "False" will return the object type and index number.

Return

- **String**:
  Text string with the address.

Example

This example returns the address of the first sequence of the selected data pool, prints the address in the **Command Line History**, and creates a grandMA3 command with a "Go" keyword in front of the address. This command is sent to the grandMA3 command line.

The command line history shows the commands entered and how the system interprets the command and feedback. Learn more in the **Command Line topic**.

Lua

```
return function()
    -- Stores the handle in a variable.
    local mySequence = DataPool().Sequences[1]
    if mySequence ~= nil then
        -- Converts the handle to the address and store in variable.
        local mySequenceAddressName = mySequence:ToAddr(true)
        local mySequenceAddress = mySequence:ToAddr(false)
        -- Print the address to the Command Line History.
        Printf("The named address of the sequence is: " ..
mySequenceAddressName)
```

```
        Printf("The system address of the sequence is: " ..
mySequenceAddress)
        -- Send a 'Go' command with the address appended.
        Cmd("Go %s", mySequenceAddress)
    else
        ErrPrintf("The sequence could not be found")
    end
end
```

1.44.6.20. SetFader(handle, {[number], [boolean], [string]})

Description

The **SetFader** function sets a fader to a specified level. It must be used on an object that has faders.

Arguments

- **Handle:**
  The function takes a handle of the type "light_userdata" as an argument.
  It can be omitted when using the **colon notation** on an object.

  > The Colon Notation is a way to omit the handle as the first argument when using the Object functions.
  >
  > This is the general syntax with the colon notation: **object:function()**
  >
  > This is the general syntax with standard handle notation: **object.function(object)**
  >
  > Learn more in the **Lua Functions - Object API topic**.

- **Table**:
  The table can contain up to three named elements using the key/value methods.
  - **"value"**:
    This is a float number indicating the fader position on a scale from 0 to 100. This should always be part of the table.
  - **"token"**:
    This is a string indicating the fader. The string must start with "Fader". It can be omitted, and then the value will be assigned to the Master fader. The fader name must be valid for the object being used. Possible tokens include:
    - "FaderMaster"
    - "FaderX"
    - "FaderXA"
    - "FaderXB"
    - "FaderTemp"
    - "FaderRate"
    - "FaderSpeed"
    - "FaderHighlight"
    - "FaderLowlight"
    - "FaderTime"
    - "FaderSolo"
  - **"faderEnabled"**:
    If the fader can be toggled, then this boolean can be used to enable or disable the fader. A true value sets the fader to enabled.

Return

This function does not return anything.

Examples

This example changes the selected sequences' Master fader to 100% and the Time fader to 5 seconds and enables the time fader.

Lua

```
return function()
    -- SelectedSequence() creates a handle to the selected sequence.
    local selectedSequence = SelectedSequence()
    -- Set the master fader to 100. The FaderMaster is the default token, so
it can be omitted.
    selectedSequence:SetFader({value=100.0})
    -- Set the time fader to 5 seconds and enable the fader.
    selectedSequence:SetFader({value=50.0, faderEnabled=1,
token="FaderTime"})
end
```

1.45. Quickeys

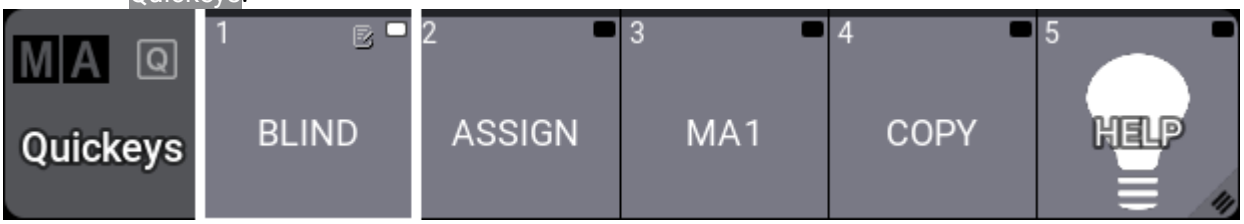
Quickeys can be used to set soft versions of various hardkeys and functions.

Quickeys can be addressed by tapping on the particular pool object or by using the **Quickey keyword**.

Quickeys can be assigned to XKeys, layouts and executors or can be used directly in the Quickeys pool.

Quickeys are stored in the Quickeys pool.

- To open the Quickeys pool, open the Add Window dialog, tap Data Pools and then tap Quickeys.



Quickeys pool

Quickeys Pool Settings

To open the Pool Settings, tap MA in the top left corner of the window.

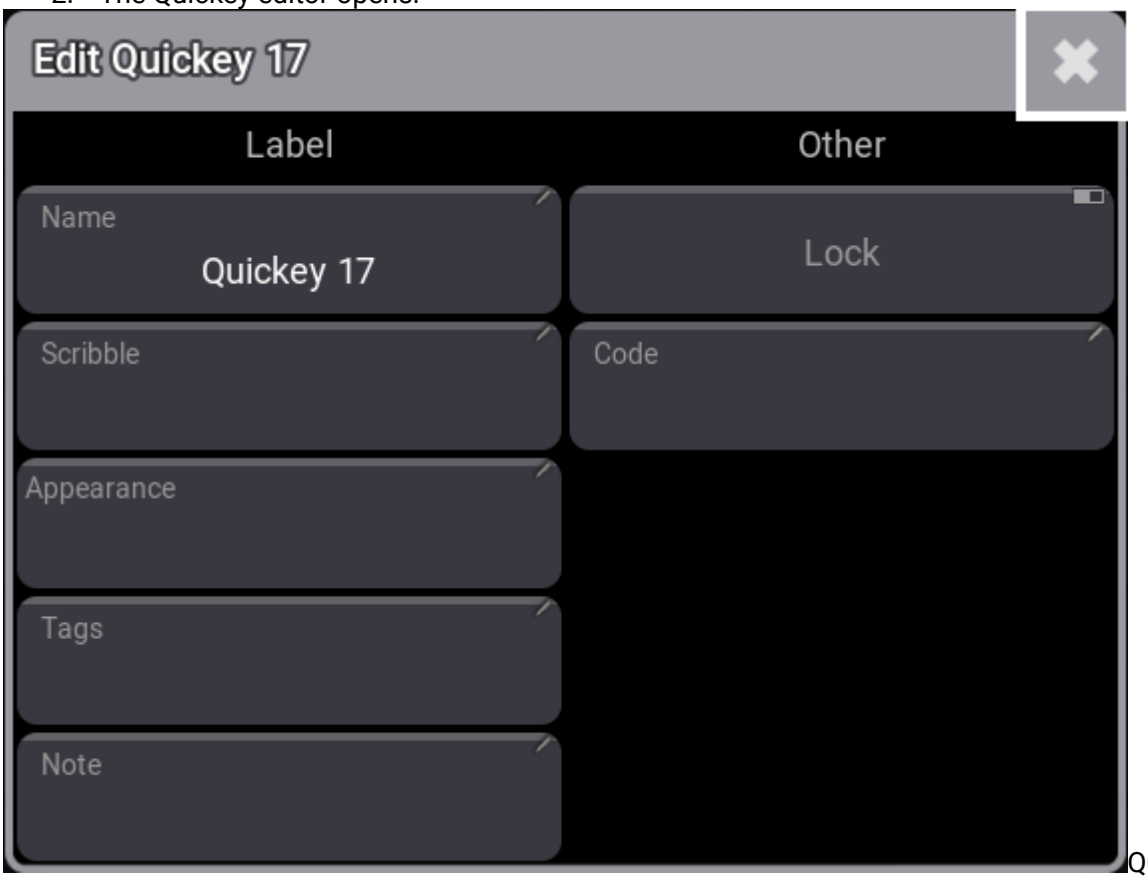
- Show Empty: This toggle button can hide or show empty pool objects.
- Appearance: The appearance is applied behind the pool objects.
- Pool Columns: This defines the width for the pool objects. It does not change the size of the window. It defines how many columns of pool objects are in the window. If the window is wider than the number of columns, then the extra space is displayed as black (default color). If the window is smaller than the number of columns, the pool window can be scrolled horizontally. If the pool has a set width, then there is an icon (⏭) in the upper right corner of the title field. The **Not Defined** value dynamically sets the width to match the window size even when the window is resized.
  The **Take Current Width** sets the width to match the current size of the window. It does not dynamically change if the window is resized.
- Font Size: There are some different font size properties from 10 to 32. There is also a default property. This is the same as size 18. This simply changes the font size on the pool objects.
- DataPool: This defines what data pool the pool window shows data from. This makes it possible to have pools showing objects from different data pools. For instance, a group pool window from the default data pool can be shown next to a different group pool window showing groups from a different data pool.
- Pool Color: This is the color for the title button in the pool.
- Empty Color: This color is applied to empty pool objects.
- Reset Colors: This resets the colors to the colors in the default color theme.
- CLI: Command Line Interaction.

Subtopics

- **Quickey Editor**
- **Use Quickey Pool Objects**
- **Example**

1.45.1. Quickey Editor

1. To open the Quickey editor, use Swipey on an empty pool object and then tap Edit.
2. The Quickey editor opens:



Quickey editor

	Hint:
	To edit an existing Quickey, enable CLI in the Quickey Pool Settings or use the command line, for example **Edit Quickey 1**.

The following six settings are available in the Quickey editor:

- **Name:**
  This is the name of the pool object.
  Selecting Code first takes over the name of the function.

- **Scribble:**
  Assigns a scribble to the pool object. See **Scribble**.

- **Appearance:**
  Assigns an appearance to the pool object. See **Appearances**.

- **Tags**:
  Opens the Tags pop-up. See **Tags**.

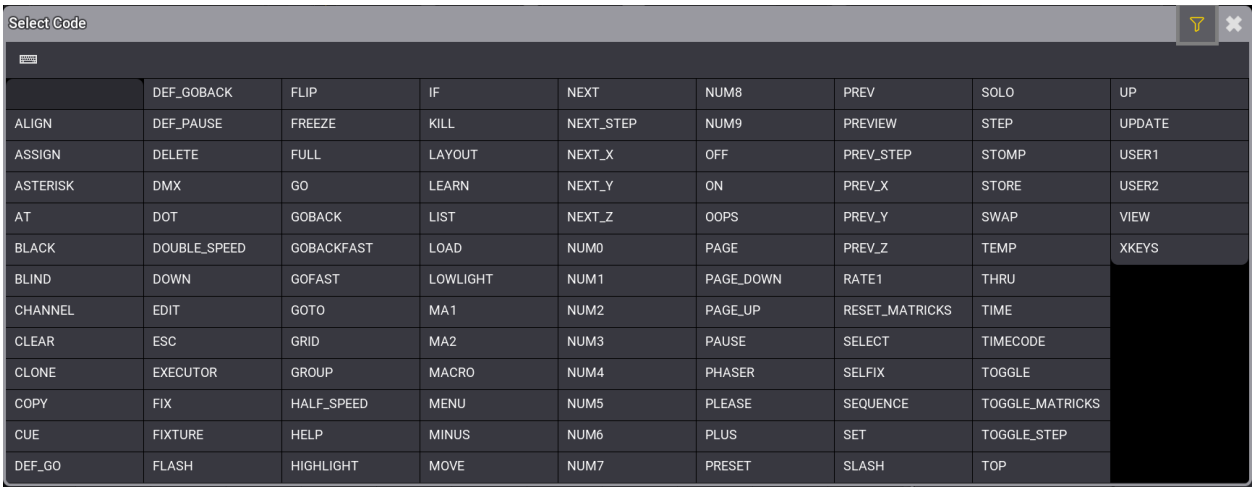
- **Note:**
  Edits a note to the pool object. See **Notes**.

- **Lock:**
  When enabled the pool object is looked for further changes.

- **Code:**
  Opens the code list.

Tapping Code in the pool editor opens the Quickey code list. To filter codes, tap and type the function into the text field.

For more information about the individual functions, see **Keys**.



	DEF_GOBACK	FLIP	IF	NEXT	NUM8	PREV	SOLO	UP
ALIGN	DEF_PAUSE	FREEZE	KILL	NEXT_STEP	NUM9	PREVIEW	STEP	UPDATE
ASSIGN	DELETE	FULL	LAYOUT	NEXT_X	OFF	PREV_STEP	STOMP	USER1
ASTERISK	DMX	GO	LEARN	NEXT_Y	ON	PREV_X	STORE	USER2
AT	DOT	GOBACK	LIST	NEXT_Z	OOPS	PREV_Y	SWAP	VIEW
BLACK	DOUBLE_SPEED	GOBACKFAST	LOAD	NUM0	PAGE	PREV_Z	TEMP	XKEYS
BLIND	DOWN	GOFAST	LOWLIGHT	NUM1	PAGE_DOWN	RATE1	THRU	
CHANNEL	EDIT	GOTO	MA1	NUM2	PAGE_UP	RESET_MATRICKS	TIME	
CLEAR	ESC	GRID	MA2	NUM3	PAUSE	SELECT	TIMECODE	
CLONE	EXECUTOR	GROUP	MACRO	NUM4	PHASER	SELFIX	TOGGLE	
COPY	FIX	HALF_SPEED	MENU	NUM5	PLEASE	SEQUENCE	TOGGLE_MATRICKS	
CUE	FIXTURE	HELP	MINUS	NUM6	PLUS	SET	TOGGLE_STEP	
DEF_GO	FLASH	HIGHLIGHT	MOVE	NUM7	PRESET	SLASH	TOP	

Select Code pop-up.

1.45.2. Use Quickey Pool Objects

For general information about pool objects, see **Pool Windows**.

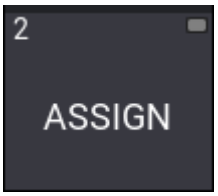
To activate or deactivate a Quickey, tap on the particular pool object.

A virtual LED in the top right corner in each pool object indicates the current status of the Quickey, depending on the LED color:

- **Black**: Quickey is off. Some Quickeys are triggered without a LED color, for example, MENU or NUM1.

- **White**: Quickey is active.

- **Grey**: Certain Quickeys trigger keywords. The appropriate keyword is entered into the command line, when the virtual LED is grey.



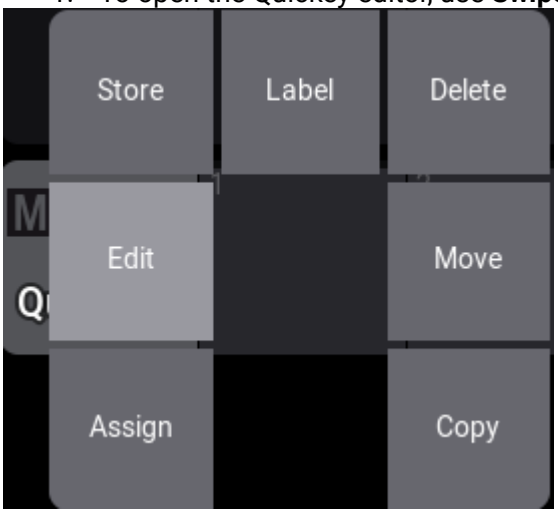
	Hint:
	The Quickeys MA1 and MA2 can be latched. For more information about latch, see **Command area**.

1.45.3. Example

Requirements:

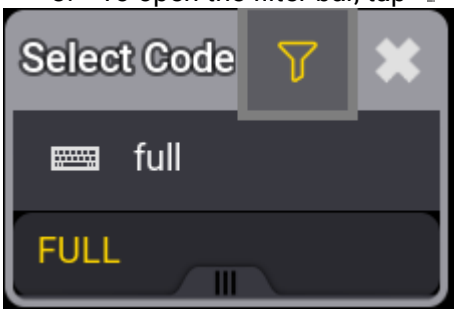
- Demo show file is loaded.
- Fixture Sheet is open.
- Quickeys pool is open.
- Clear the Programmer and turn the running playback off first. For more information, see **Clear Key** and **Running Playbacks**.
- Type **Fixture 9 Thru 13** into the command line and then press Please, to select fixtures.

1. To open the Quickey editor, use **Swipey** on an empty pool object and then tap Edit.



Swipey

2. To open the code list, tap Code.
3. To open the filter bar, tap and type "full":



Filter

4. Tap FULL, displayed in yellow font. The code list closes.
5. Tap to close the editor. A new Quickey pool object is created.

6. Create a second Quickey with the BLIND function, using steps 1 to 5. The pool should look like this:

Quickey pool

7. To set the dimmer of the selected fixtures to 100%, tap FULL in the Quickey pool.
8. To toggle blind, tap BLIND repeatedly.

For more information about the used keywords, see **Full** and **Blind**.

1.46. Data Pools

Much of the different data belonging to the show file are stored in pools. Many of these pools exist as children inside a **Data Pool** parent object.

A new show file creates a default **Data Pool**. The Data Pool object contains the other pools.

New **Data Pool** objects can be created, giving an entirely new set of pools.

This can be very useful if several shows or acts use the same patch; for instance, each song in an extensive band catalog can be in its own data pool.

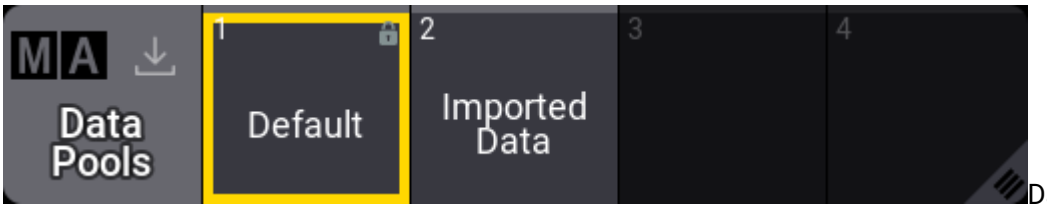
These are the pools inside the Data Pool objects:

- **Bitmaps**
- **Executor Configurations**
- **Filters**
- **Generators - Random**
- **Groups**
- **Layouts**
- **Macros**
- **MAtricks**
- **Pages**
- **Plugins**
- **PresetPools**
- **Quickeys**
- **Sequences**
- **Timecodes**
- **Timers**
- **Worlds**

Each window of pools can be linked to the selected data pool or a specific data pool. Learn more in the **Window Settings topic**.

Data Pools Window

The best way to see the different data pools is in the **Data Pools window**. This can be created like any other window - learn how in the **Add Windows topic**.



Data pools with some data pool objects

The pool can be used to select the desired data pool by tapping it. Any of the different data pool operations mentioned below can be done using the appropriate keyword in combination with the pool window. But it can also be done using the command line and the **DataPool keyword**.

The data pools can also be found in Menu / Settings / User Configuration / Pools. This is a list form of the data pools.

Create a New Data Pool Object

Data pools need to be stored to be created. These are different ways to create a new Data Pool.

- Press Store and then tap an empty pool object.
- Open the swipey commands on an empty pool object and choose Store.
- Using the command line: **Store DataPool [DataPool_number]**

Performing an "edit" command on an empty pool object also creates the data pool object.

Copy a Data Pool Object

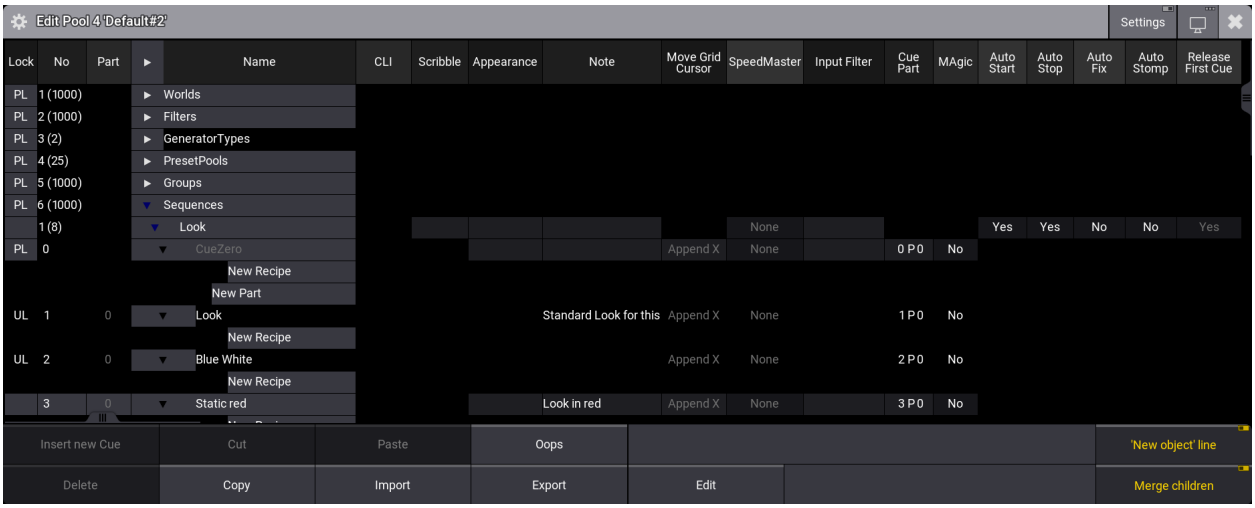
A data pool can be copied. This makes a copy of all the elements in the data pool. The copies are not linked, but objects inside the copy might be referencing objects in the source data pool. For instance, a copy of a sequence might contain links to presets in the source data pool - the presets are not copied automatically, and the references are not changed in the copy.
Changing one of the copies after the copy action does not change the other.

- Press Copy, tap the source pool object, and then an empty pool object.
- Open the swipey commands on the source pool object, choose Copy, then tap an empty pool object.
- Using the command line: **Copy DataPool ["DataPool_Name" or DataPool_Number] At ["DataPool_Name" or DataPool_Number]**

	**Hint:**
	Objects inside a data pool can be copied from one pool to another using the normal **Copy** and **Paste** commands.

Edit a Data Pool Object

Editing a data pool object opens an editor with a structured list of the elements in the data pool. Unfolding the elements in the list shows the settings for the elements.



Data pool in edit mode - Sequence unfolded

This is a deep look into the structure of the data.

Changing a setting here changes the setting for the object.

Edit a pool object using one of these methods:

- Press Edit and then tap the desired pool object.
- Open the swipey commands on the pool object and choose Edit.
- Using the command line: **Edit DataPool ["DataPool_Name" or DataPool_Number]**.

Reference Data Pool Objects from a Different Data Pool

Other data pools can use a pool element inside a different data pool.

For instance, imagine a repertoire theater. All default elements are stored in the data pool one, including a sequence that controls the house lights. Tonight's show is programmed in data pool two. The sequence can be copied from data pool one, but if changes are made to it later, it would need to be copied again into the second data pool. Instead, the original sequence can be assigned to an executor in data pool two.

The image of the edit menu above shows that the structure in the data pool is numbered. So sequence 2 in this data pool is data pool object 1.6.2. The first number is the data pool number. The second number specifies that it is a sequence. The third number specifies that it is sequence number 2. If this element needs to be assigned somewhere, the data pool object must be included in the assign command. This can be done using the data pool object number or simply by adding the data pool information before the object - see the example below.

Example

Data pool 1 has sequence 2. This needs to be used in data pool 2 on (page 1) executor 201.

Make sure data pool 2 is the selected pool.

The example can be achieved using the command line: **Assign DataPool 1 Sequence 2 At Page 1.201**.

Now the sequence in data pool 1 can be controlled by the executor in data pool 2.

Some menus give access to select the data pool. In the title bar of the menus, there is a button called DataPool that can toggle between the different data pools.

Delete a Data Pool Object

	**Important:**
	Deleting a data pool object also deletes everything inside the data pool!

Unlocked data pools can be deleted using any of the following methods:

- Press Delete and then tap the pool object.
- Open the swipey commands on the relevant pool object and choose Delete.
- Using the command line: **Delete DataPool ["DataPool_Name" or DataPool_Number]**.

1.47. System

This section holds system windows that provide information on the system, operational views, and overall system settings.

Subtopics

- **Date and Time**
- **Clock**
- **Desk Lights**
- **System Information**
- **System Monitor**
- **Info Window**

1.47.1. Date and Time

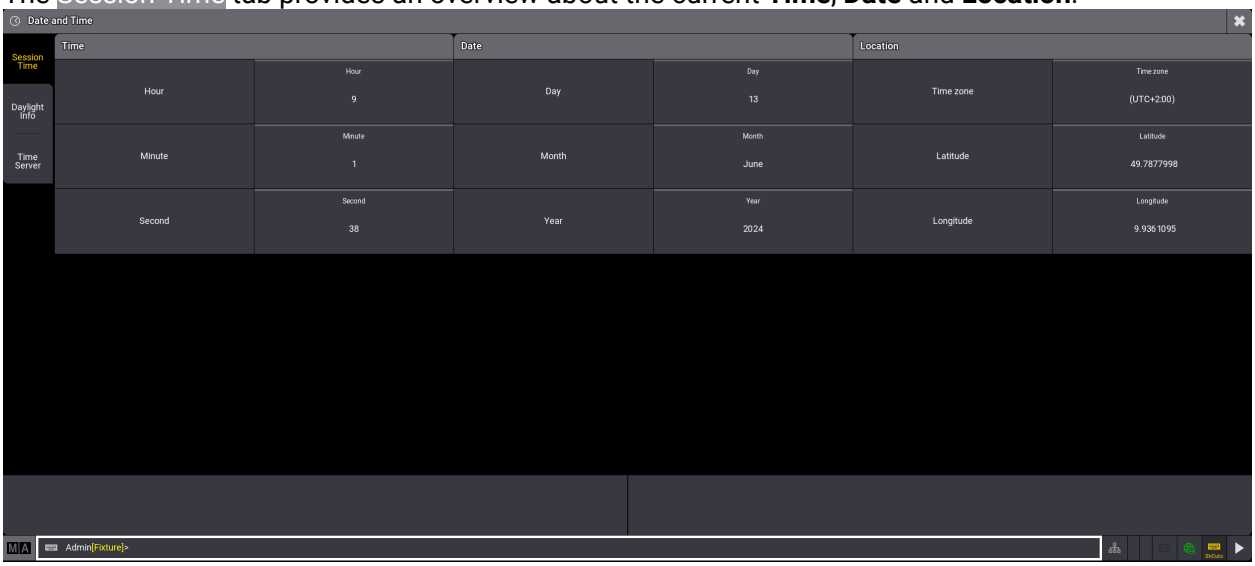
Sets date and time.

To access **Date and Time**:

1. Press Menu.
2. Tap Settings. A drop-down menu opens.
3. Tap Date and Time. The Date and Time window opens.

Session Time

The Session Time tab provides an overview about the current **Time**, **Date** and **Location**.



Date and Time window with Session Time settings

Date and Time:

To set date and time manually:

1. Go to Session Time.
2. Tap the buttons with a gray title bar. The calculator opens.
3. Enter values and tap Please.

Date and time are set.

	Important:
	If Date and Time is NTP synchronized, it is not possible to adjust the values manually. Go to the Time Server tab first. Set Extern TimeSync Mode to None. Now go back to Session Time and adjust manually.
	Restriction:

	When using grandMA3 onPC with **macOS**, avoid changing the system time in the general system settings of macOS as it can cause critical time conflicts in different parts of the onPC software. Adjusting the session time in the onPC software with macOS is safe.

Location:

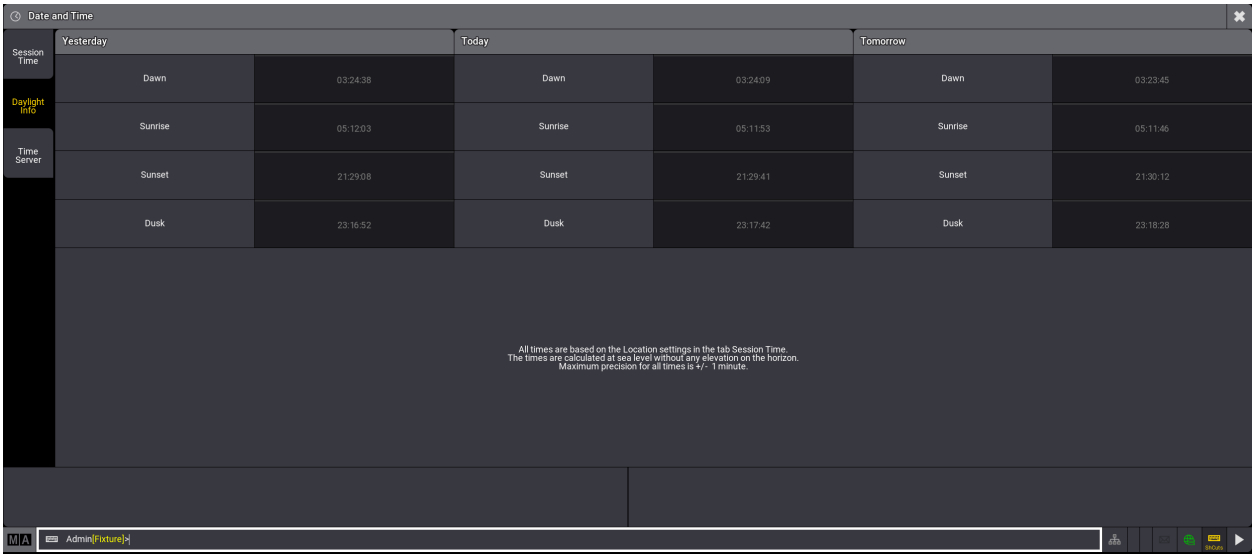
The time zone refers to the international time standard UTC.

1. To enter the current location, tap each of the buttons –Timezone, Longitude, Latitude – on the right of the column Location.
   The calculator opens.
2. Enter values and tap Please.

Location is set.

Daylight Info

The Daylight Info tab, displays yesterday's, today's, and tomorrow's dawn, sunrise, sunset, and dusk.

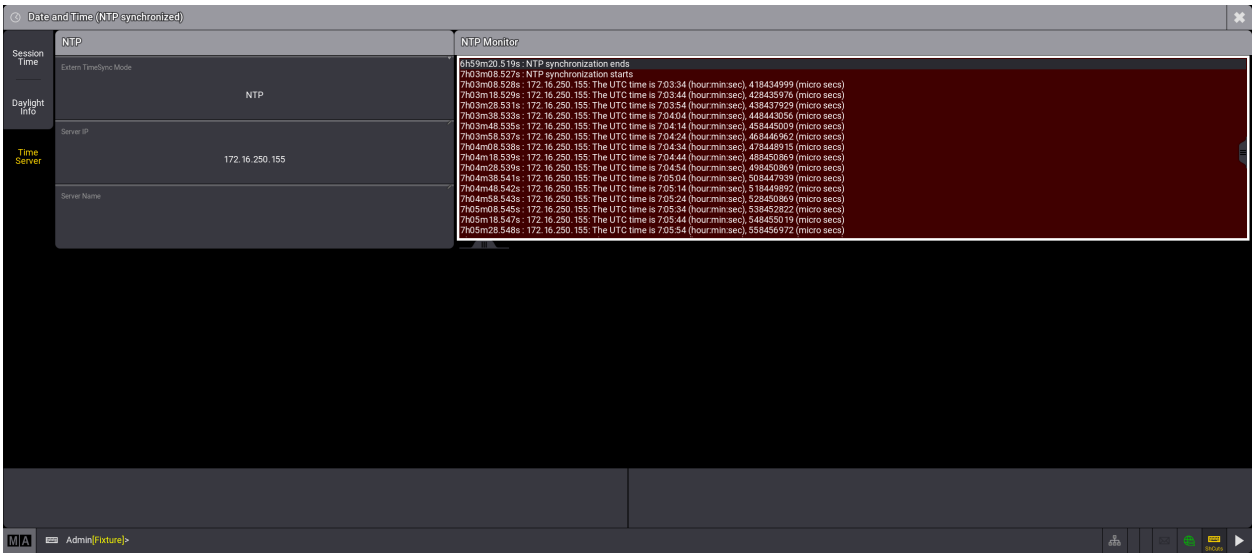


Date and Time window with Daylight Info settings

All times are based on the Location settings in the tab Session Time. The times are calculated at sea level without any elevation on the horizon. Maximum precision for all times is +/- 1 minute. Dawn and Dusk are calculated using nautical twilight.

Time Server

The Time Server tab allows you to synchronize the Session Time with an external time source through Network Time Protocol (NTP).



Date and Time window with Time Server settings and NTP Monitor

The Time Server tab offers several options:

- **Extern TimeSync Mode**: When set to NTP, Time and Date are synchronized with the NTP settings referring to Server IP and Server Name. Setting it to None, Date and Time can be set manually in the **Session Time**.
- **Server IP**: Tap on Server IP. A calculator opens. Type in any IP address from an NTP server. The NTP server now provides Date and Time.
- **Server Name**: Type in any Server Name of an NTP server. This displays the Server IP automatically. The NTP server now provides Date and Time.

The **NTP Monitor** on the right side displays details about the server's connection.

	Important:
	The NTP server delivers always Coordinated Universal Time (UTC +0:00). To match it to your local time, adjust the Time zone in the **Session Time** tab.
	Important:
	The **IdleMaster** or **GlobalMaster** station determines the time of the session, even if other stations in the session have different NTP servers set up.

1.47.2. Clock

Open the Clock Viewer window with the **Add Window** pop-up in the Tools tab.

Clock window

- The clock is displayed in yellow digits by default.
- Tap and hold Clock Source in the title bar to change the clock source.
  The values are:
  - Session Time
  - Time Zone
  - Timecode
  - Timer

  The different clock sources are explained in detail below.

	Hint:
	The name of the Clock Viewer in the title bar of the window changes according to the selected clock source and style. This also means that for some settings the title changes while the clock source and style are not actively changed, for example from AM to PM.

Session Time

Session Time is the default setting for the clock viewer.

Session time and dawn, sunrise, sunset, and dusk depend on the time, date, and location set in the Date and Time pop-up. For more information about those settings and the Date and Time pop-up, see **Date and Time**.

To change the clock viewer to Session Time manually:

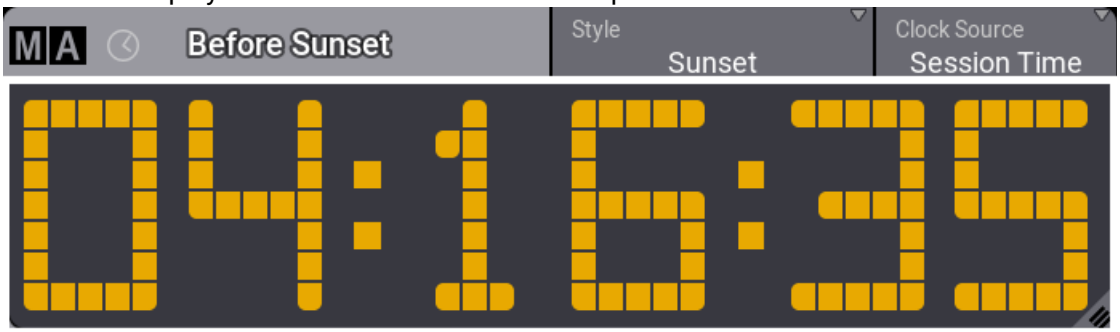
1. Tap and hold Clock Source.
2. Select **Session Time**.

To change the session time style:

1. Tap and hold Style.
2. Select one of the values.

There are eight different ways to display the session time.
They are:

- **Digital**:
  Displays the time in in 24-hour format.
- **Digital AM/PM**:
  Displays the time in AM/PM format.
- **Date DD-MM-YYYY**:
  Displays the date starting with the day.
- **Date MM-DD-YYYY**:
  Displays the date starting with the month.
- **Dawn**:
  Displays the time to dawn or the time elapsed after dawn.
- **Sunrise**:
  Displays the time left to sunrise or the elapsed time after sunrise.
- **Sunset**:
  Displays the time left to sunset or the elapsed time after sunset.
- **Dusk**:
  Displays the time left to dusk or the elapsed time after dusk.



Session Time with Style set to Sunset

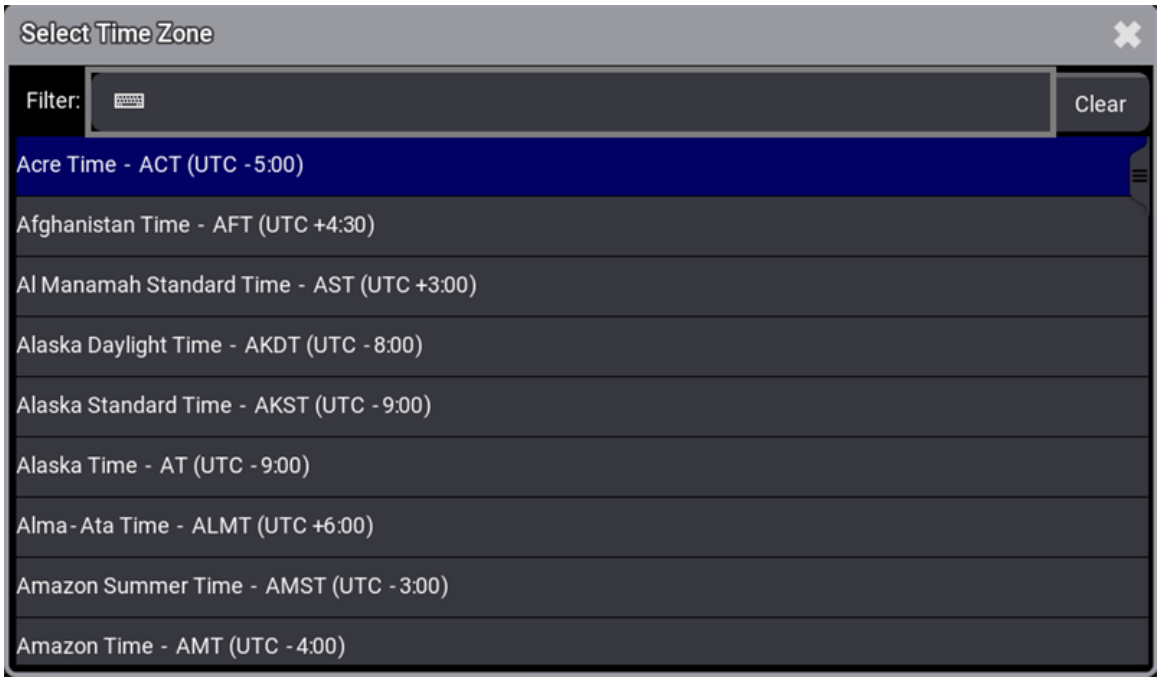
Time Zone Clock

1. Set the Clock Source to **Time Zone Clock**.

Time Zone Clock window

2. Tap Time Zone in the title bar of the clock viewer.
   The Select Time Zone pop-up opens.

3. Select the time zone.



Select Time Zone pop-up

4. Tap and hold Style.
5. Select **Digital** to display the time in 24-hour format.
   Select **Digital AM/PM** to display the time in AM/PM format.

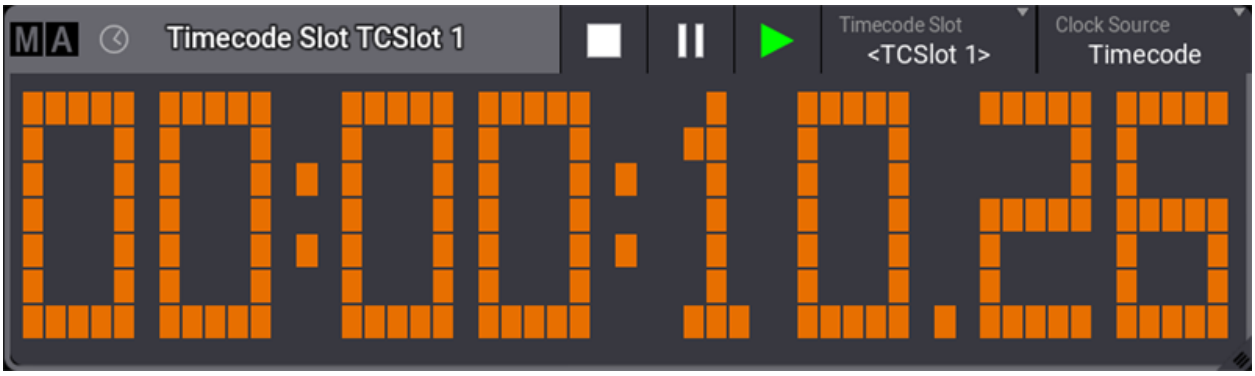
	Hint:
	For more information on how to set the **Session Time**, see **Date and Time**.

Timecode Slot

To set the clock to a timecode slot:

1. Tap and hold Clock Source in the title bar, then tap Timecode.
2. Tap and hold Timecode Slot in the title bar, then tap the desired timecode slot.
   To use the timecode slot selected in the **Timecode Slots** pool, tap <Selected>.

The clock displays the selected timecode slot.



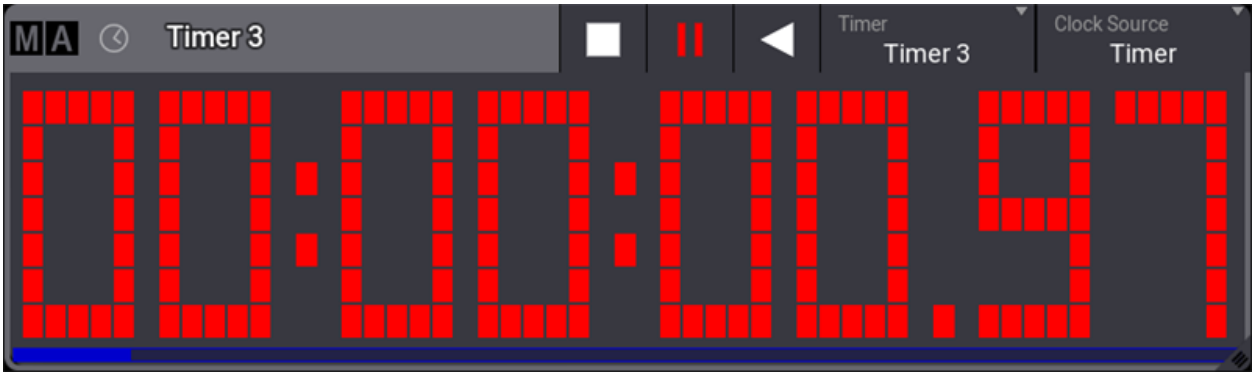
Timecode Slot clock window

Timer

To set the clock to display a timer:

1. Tap and hold Clock Source in the title bar, then tap Timer.
2. Tap and hold Timer in the title bar, then tap the desired timer.
   To use the timer selected in the **Timers** pool, tap <Link Selected>. For more information, see **Timers**.

The clock displays the selected timer.



Timer clock window

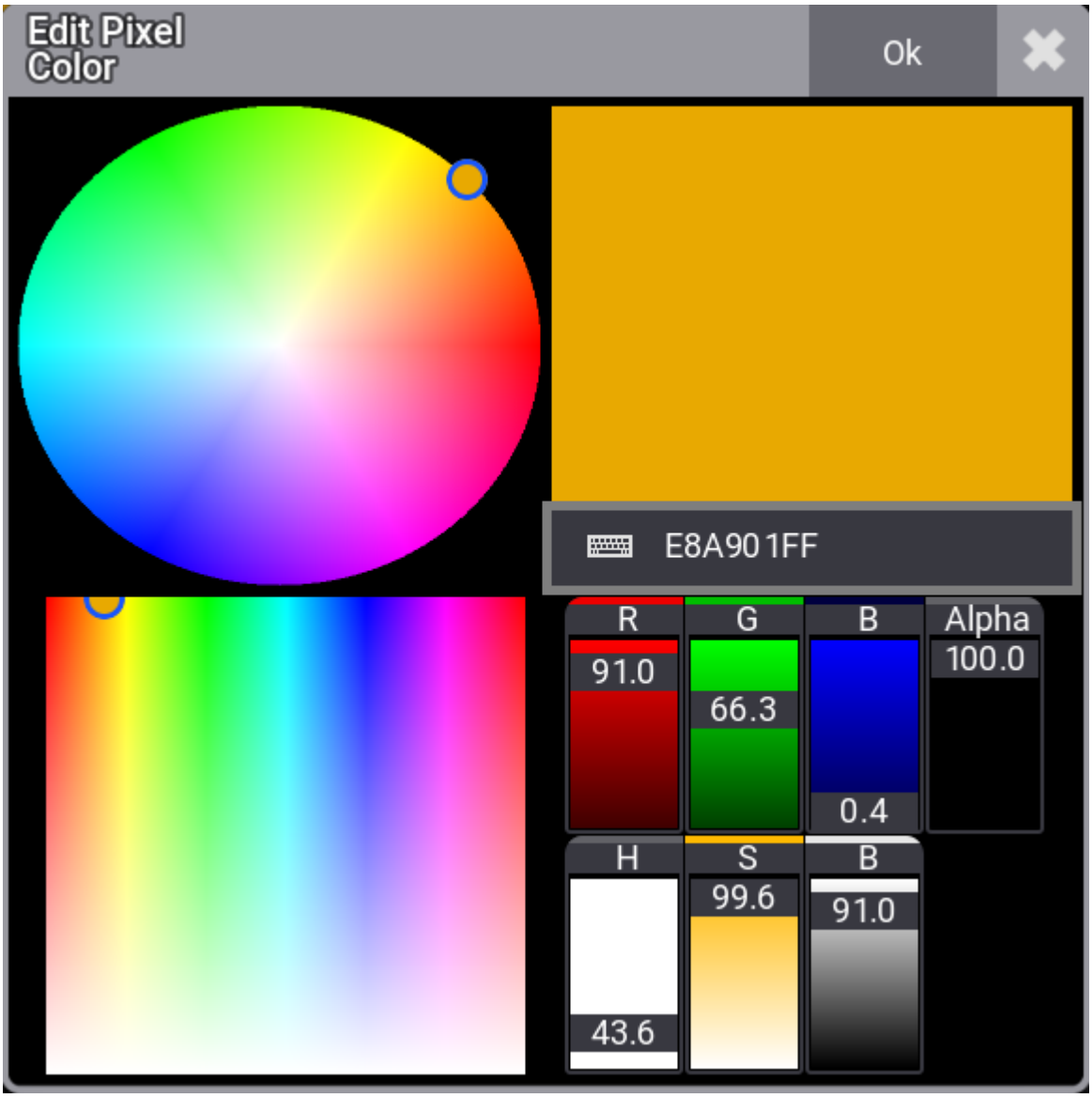
Tap to start the countdown for the displayed timer. A descending progress bar is displayed at the bottom of the window.

Tap to pause the countdown.

Tap to stop the clock, this will also reset the **Countdown Time** set for this timer. For more information, see **Timers**.

Edit Color of Digits

1. To change the color of the digits, tap MA in the top left corner, then tap Pixel Color.
2. The pop-up **Edit Pixel Color** opens:



Edit Pixel Color pop-up

1. Tap to select a color in the color picker.
2. Tap Ok.

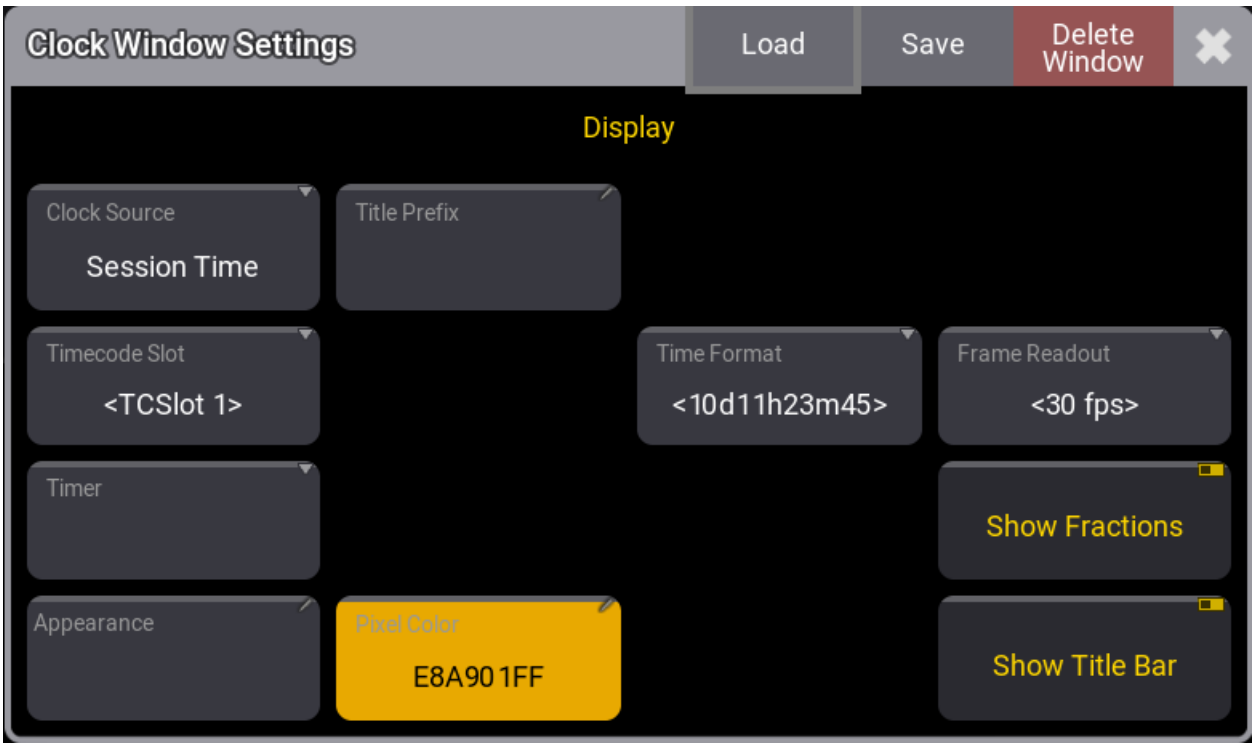
The color of the digits is modified.

Appearance as Background Color

1. Create an appearance in the appearance pool.
   For more information, see **Create Appearances**.
2. Press Assign, tap an appearance in the pool and tap the clock title bar.
   For more information, see **Use Appearances**.

Clock Window Settings

Tap MA in the left corner of the clock viewer title bar to open the clock window settings.



Clock window settings pop-up

- Clock Source: Tap and hold to open the dropdown menu, then select the source.
- Timecode Slot: For more information, see **What are timecode slots**.
- Timer: For more information, see **Timers**.
- Pixel Color: Opens the Color Editor.
- Appearances: Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.
- Title Prefix: Tap to open the virtual keyboard, then enter a prefix name for the time zone clock.
- Title Format: To learn more about **Time Format** and **Frame Readout**, see **Single User and Multi User Systems - User Settings**.
- Frame Readout: This defines the frame readout for this window. It can be used to overwrite the default set in the **user profile**.
- Show Fractions: Tap to toggle on or off the fractions of a second when the **Clock Source** is set to **Timer** or frames when the **Clock Source** is set to **Timecode**.
- Show Title Bar: This shows or hides the window's title bar. It is On by default. If it is Off, then the title bar can be shown temporarily by pressing both MA keys in the control area. In grandMA3 onPC, the title bar can be temporarily shown by pressing Ctrl + Alt on Windows and Ctrl + Option on Mac.

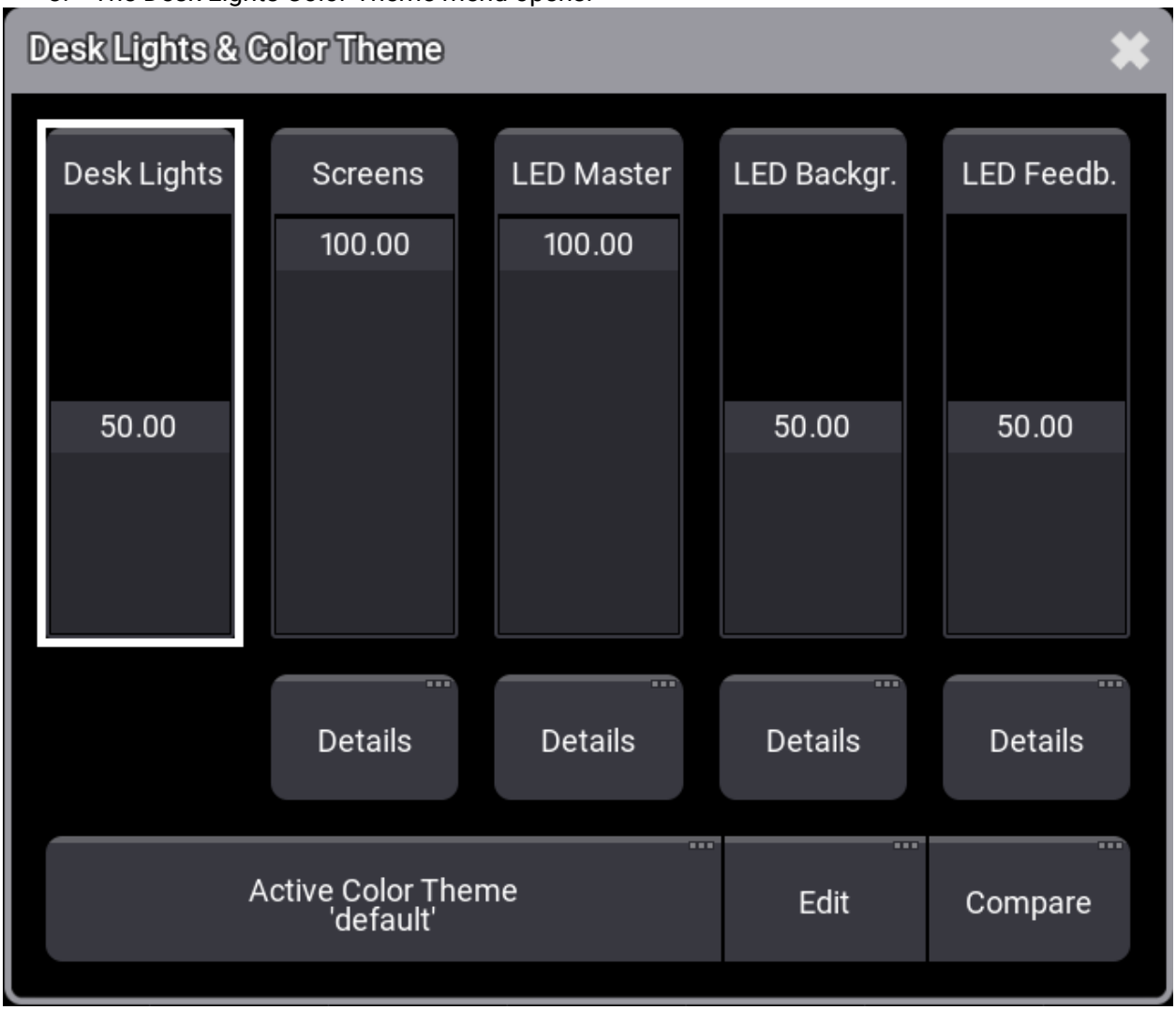
1.47.3. Desk Lights

This topic describes the setting of desk lights.

To learn more about color themes, see **Color Themes**.

To access the Desk Lights Color Theme menu:

1. Press Menu.
2. Tap Desk Lights & Color Theme.
3. The Desk Lights Color Theme menu opens.



Desk Lights & Color Theme menu

- To adjust the desk light intensity, move the Desk Lights fader up or down.
- To adjust the general brightness of all screens, move the Screens fader up or down.
- To adjust the brightness of individual screens, tap Details below the Screens fader. To adjust the brightness of the desired screen, move the respective Screen fader up or down.

	Restriction:

	The Screen External fader currently has no function.

- To adjust the general intensity of all LEDs, move the LED Master fader up or down.
- To adjust the LED intensity of the individual elements, tap Details below the LED Master fader. To adjust the intensity of the desired element, move the respective LED fader up or down.
- To adjust the general LED Background intensity, move the LED Background fader up or down.
- To adjust the LED background intensity of the individual elements, tap Details below the LED Master fader.
- To adjust the general LED Feedback intensity, move the LED Feedback fader up or down.
- To adjust the LED Feedback intensity of the individual elements, tap Details below the LED Feedback fader.

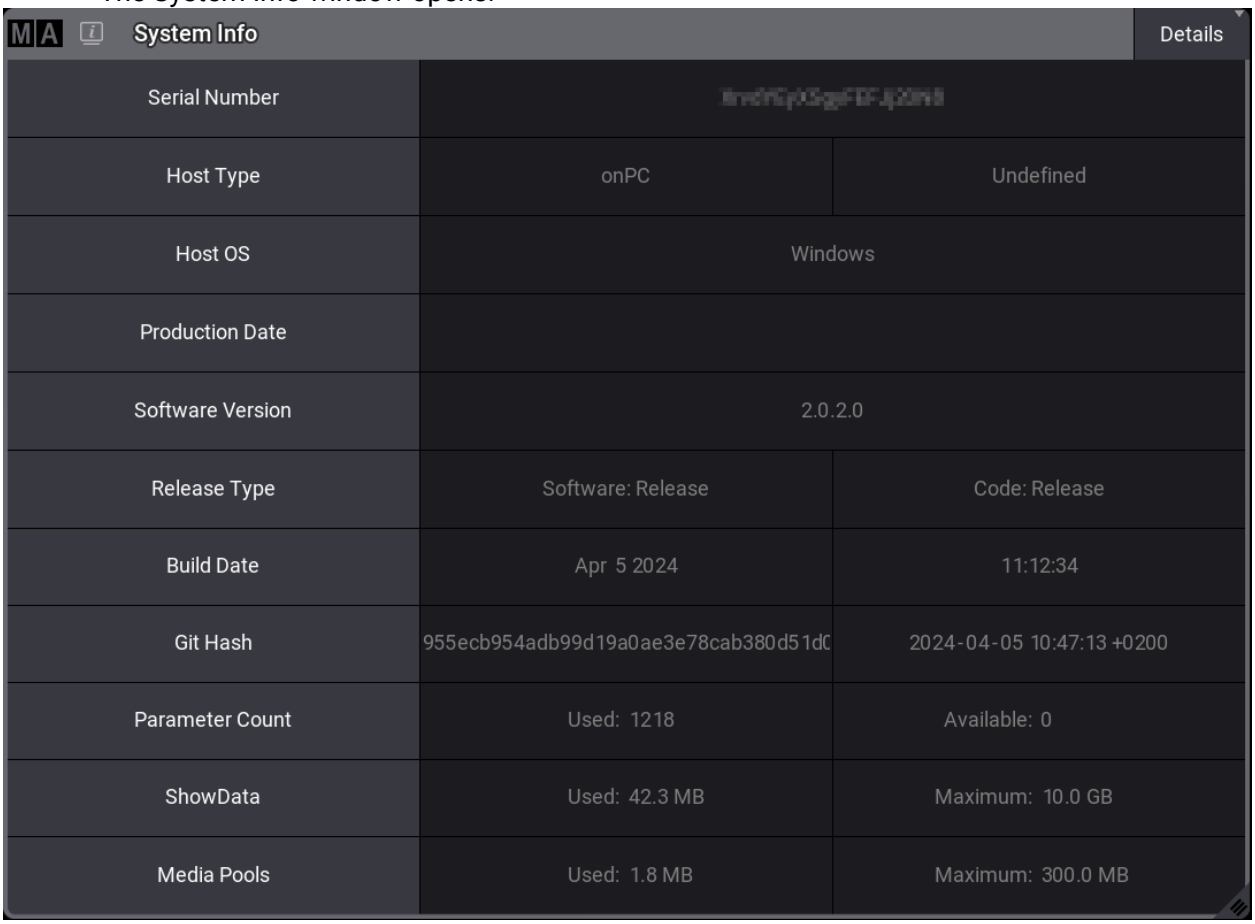
1.47.4. System Information

System Info is a mixture of system and performance monitoring of the console.

To add a System Info window, follow the instructions under **Add windows**.

In the Add Window pop-up:

1. Tap More.
2. Tap System Info.
   The System Info window opens.



System Info – details on onPC

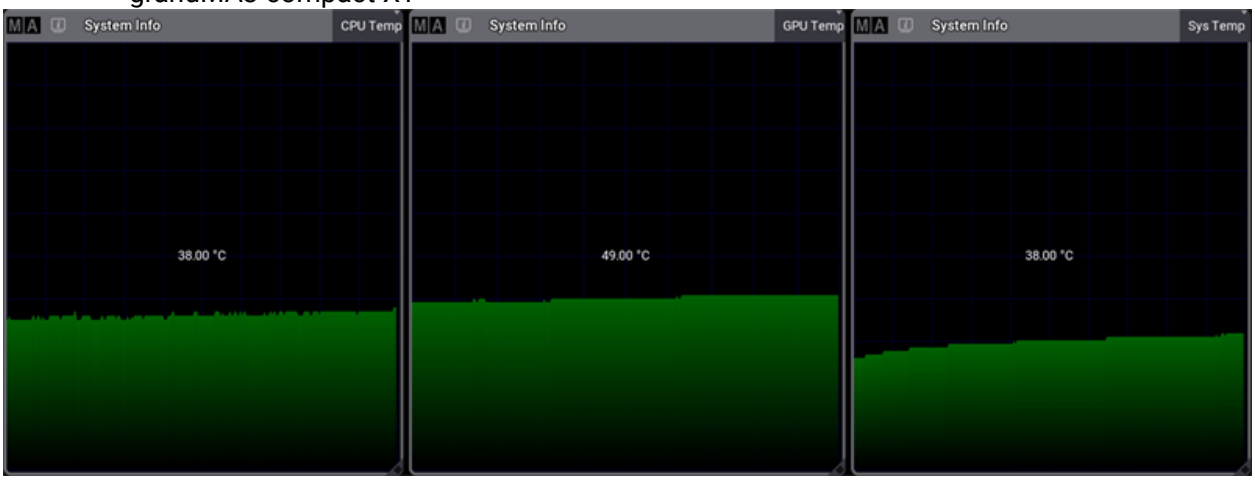
To adjust the System Info window settings, follow the instructions under **Window Settings**.

Temperature Check

CPU, GPU, and system temperatures can be displayed on:

- grandMA3 full-size

- grandMA3 light
- grandMA3 replay unit
- grandMA3 compact XT



Example of temperature windows

The grandMA3 compact can only display CPU and system temperatures.

These products are not able to display temperature information:

- grandMA3 extension
- grandMA3 processing unit
- grandMA3 nodes
- grandMA3 viz-key
- All grandMA3 onPC solution devices

Fan Check

The fan view in the System Info window displays the system/CPU cooling fan speed in correlation to its maximum speed.

Fan speeds can be displayed on:

- grandMA3 full-size
- grandMA3 light
- grandMA3 replay unit
- grandMA3 compact (XT)

These products are not able to display fan speed information:

- grandMA3 extension
- grandMA3 processing unit
- grandMA3 nodes
- grandMA3 viz-key
- All grandMA3 onPC solution devices

System Info is divided into different views, which can be selected using the button in the title bar on the right-hand side.

Tap and hold the button to toggle between:

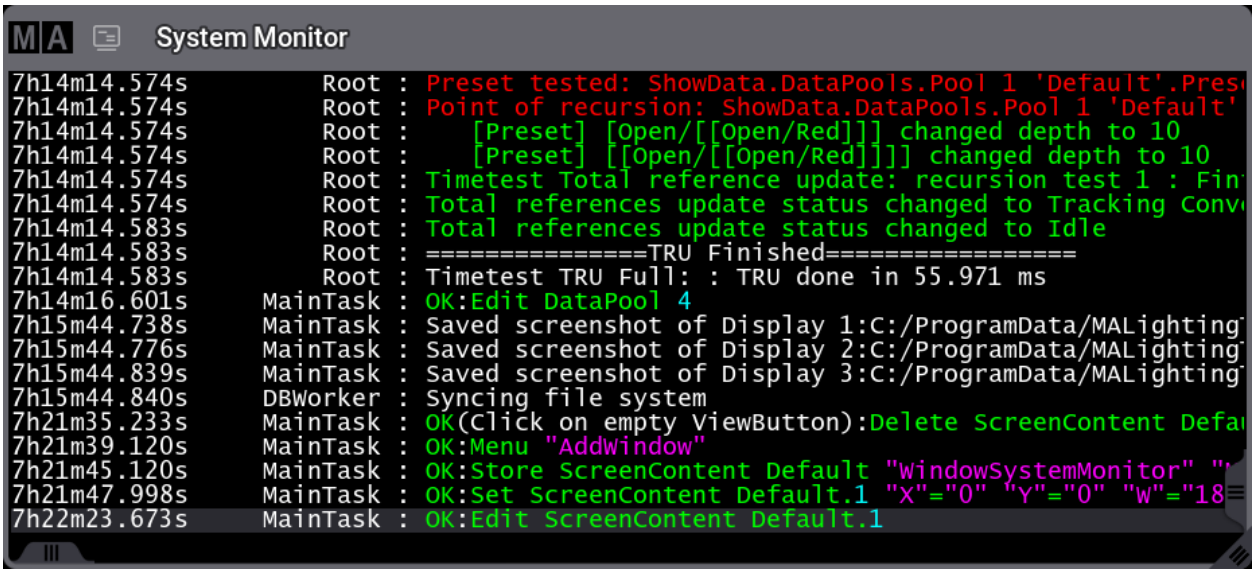
- Realtime: Realtime displays the workload of the system in milliseconds per DMX frame.
- Timing: Timing displays the time that is required to render screens on the console.MainLoop, Swap, and End show the internal processing and in which displays it takes place.
- CPU: CPU displays the workload of the main processor.
- Memory: Memory displays the RAM of the system in GB.
- CPU Temp: CPU Temp displays the current temperature of the console.
- GPU Temp: GPU Temp displays the current temperature of the graphics processing unit of the console.
- Sys Temp: Sys Temp displays the current temperature of the central computer board.
- Fan: Fan displays the rotational speed of the fan of the cooling system in RPM.
- Details: Details displays other relevant system info such as the serial number or the build date.
- HDD: HDD lists relevant information about the available hard drives.
- Network: Lists the number of connected and expected stations in the session. Green numbers shows the expected numbers of devices. Red shows when a device is disconnected.

	**Hint:**
	Graphs such as the Memory graph show the current used value on the left and the total value on the right.

The title bar can be enabled or disabled by tapping Show Title Bar in the System Info Window Settings.

1.47.5. System Monitor

The system monitor is a window that can be created like any other window using the **Add Window pop-up**.



System Monitor

It shows what is happening at the station. This includes feedback on user commands. It is a log of the different things happening in the background. It also shows warnings, errors, and changes to the system.

It is a debugging window that can provide useful information if there are any problems with the system.

There are only two settings for the window.

- **Appearance**:
  Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.
- **Font Size**:
  This selects the font size in the window. It is a swipe button that opens a list of sizes from 10 to 32. There is also a **Default** property. The default is the same as size 18.

The default background is black. The bottom line shows the latest thing that happened.

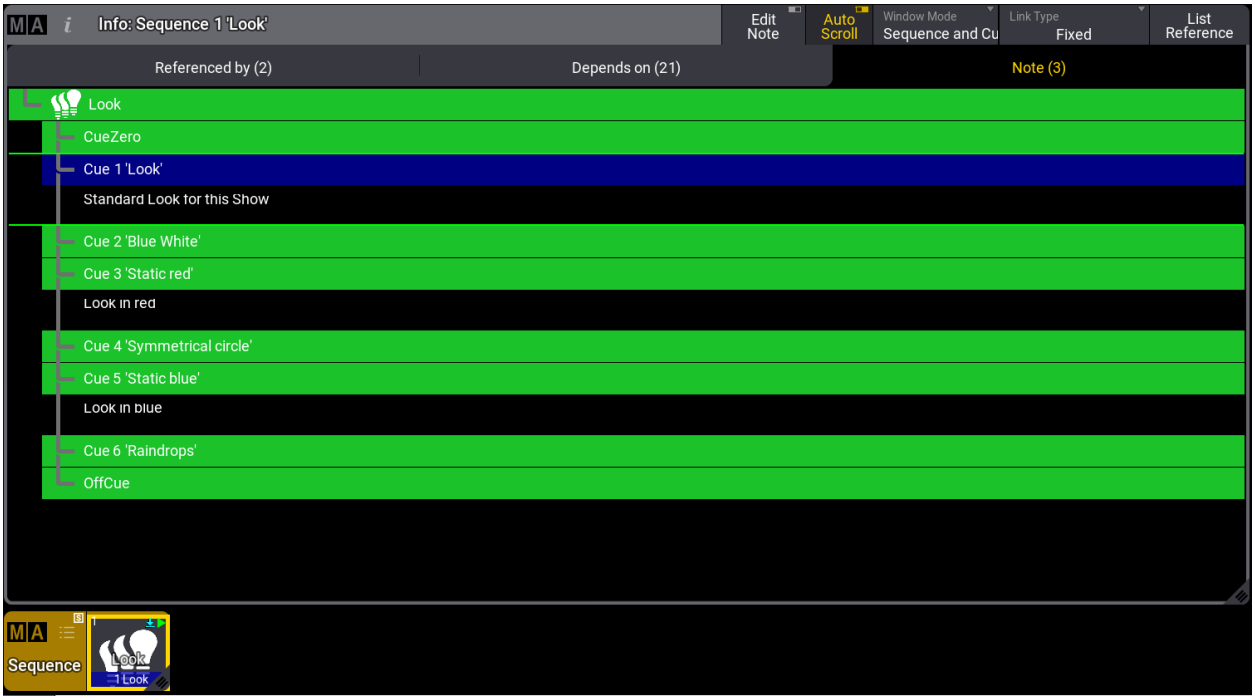
If the system monitor is scrolled up, then the background changes to a red color to indicate that is not the latest data, and the auto-scrolling function that automatically shows the latest info is not active. Scroll back down to the bottom to reactivate the auto-scroll function.

1.47.6. Info Window

The Info window is a helpful tool to show the operator references, dependencies, and notes for objects in the show file.

To learn more about displaying dependencies and references in the Command Line History, see the **ListReference Keyword**.

To open the Info window, tap More and then tap Info in the Add window dialog.



Info window

The Info window is seperated into three tabs: **Referenced by (x)**, **Depends on (x)**, and **Note (x)**.

Referenced by (x) -tab displays the following information:

- **(x)**, displays the number of references for this object.
- **Type,** displays the targeted object reference type.
- **No,** displays the targeted object reference number.
- **Name,** displays the targeted object reference name.

Depends on (x)-tab displays the following information:

- **Type,** displays the type of the targeted object dependencies.
- **No,** displays the number of the targeted object dependencies.
- **Name,** displays the name of the targeted object dependencies.

The Note (x)-tab displays the notes of an object. Depending on the size of the Info window, the notes are resized to fit the window.

For more information about Notes, see **Notes** topic.

List Reference

To list the references of an object, for example, a sequence pool object or macros:

Requirements:

- grandMA3 demo show file is loaded.
- Open Sequence pool and Info window.

1. Tap List Reference in the title bar of the Info window or type **ListReference** in the command line.
2. Tap Look in the Sequence pool. The related information about the selected sequence pool object is shown in the Info window.

Info window displaying information of Sequence 1.

	Important:
	The results of **List Reference** for a targeted object will stay visible until you perform a new search or change the view. It can not be cleared or deleted.

Link Type

The Link Type button defines if the information of a specific object is automatically displayed or not. The following parameters can be set in the dropdown menu:

- **Fixed**: References are not listed automatically.
- **Selected Sequence**: The references are automatically displayed for the selected sequence.

	Hint:
	When Link Mode is set to Selected Sequence, the List Reference button will be grayed out. Executing the ListReference command will open a separate pop-up. For more information, see **ListReference** keyword.

Window Mode

To specify the displayed information in the Info window, Window Mode can be set to the following values:

- **Fixed:** Is set for all pool objects that are not sequences or macros.
- **Sequence / Macro**: Displays the note of the selected sequence/macro via the List Reference button.

- **Current Cue / Current MacroLine**: Displays the note of the currently active cue/macro line.
- **Next Cue / Next MacroLine**: Displays the note of the cue that will be played back next when executing Go+ again on the sequence. When using **Load**, the note of the loaded cue will be displayed.
- **All Cues / All MacroLines**: Displays the notes of all cues/macro lines.
- **Sequence and Cues / Macro and MacroLines**: Displays the notes of the parent object and all cues/macro lines.

	Hint: • The current cue is marked with a green frame and a blue background in the notes tab. • Notes of Cue Parts are also displayed in the Info window

Values are named by the children after selecting an object via the List Reference button. For example, to set different Link Types in the Note tab: Store two cues into Sequence 1. For more information to store cues, see **Store cues**.

1. Add notes to cues. For more information to store cues, see **Notes** topic.
2. Tap List Reference and then tap Sequence 1 in the sequence pool.
3. Use Go+ to go through the cues.

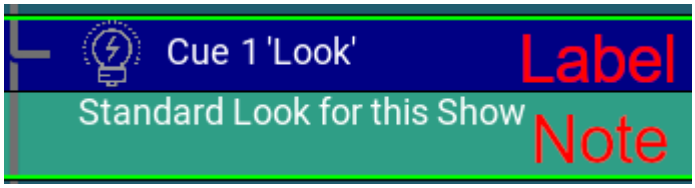
Info Window Settings

To open the Info Window settings, tap MA in the title bar of the window.

Window Preferences can be loaded or saved. For more information, see **Window Settings**.

- **Tabs**: Enable Tabs to show the tab bar.
- **Use Target Appearance**: Shows the selected appearance in the **Referenced by** and **Depends on** tabs when enabled. For more information about appearances, see **Appearances**.
- **Appearance**: Tapping this button opens a **Select Appearance** pop-up that lists all the defined appearances and the possibility of creating a new appearance. Selecting one will apply that appearance to the window.
- **Selected Tab**: Sets the currently displayed tab. Referenced by, Depends on, or Note.
- **Notes Label Color**: Sets the color for the note label.
- **Font Size**: There are some different font size properties from 10 to 32. There is also a default property. This is the same as size 18. This simply changes the font size on the pool objects.
- **Edit Note**: When enabled, all cues will be displayed. Existing notes can be edited and new ones can be added.
- **Auto Scroll**: This On/Off button activates the auto-scrolling function. This will keep the active object visible in the window by scrolling the sheet or grid.
- **Notes Appearance**: Arranges the appearances in the notes tab.
    - Off: No appearance is displayed.
    - Note: The cue´s background appearance is shown next to the corresponding note.
    - Label + Note: The cue´s image is shown in the cue label and the cue's background appearance is shown in the note and cue label. This setting will also override a currently set Notes Label Color.
- **Show Empty**: Defines whether cues without notes are displayed or not.

The differences between label and note are shown in the following image:



Info window - Note tab

Example

To get an overview of this example, watch the following video

Requirements:

- grandMA3 demo show file is open
- Info window and the **Note (x)**-tab is open

To display the note of the next cue in the selected sequence in the Info window:

1. In the title bar of Info window, tap Link Type and select **Selected Sequence**.
2. Select a Sequence Pool object, for example, **Sequence 1**.
3. Tap Window Mode in the title bar of the Info window and select Next Cue.
4. Press Go+. The note of the next cue is shown in the Info window

1.48. Remote In and Out

Remote Inputs and Outputs are handled through the In & Out menu. The In & Out menu can be used to define parameters, for example, to trigger a command via a remote controller.

Press Menu and then tap In & Out. The In & Out menu opens.

The menu is separated into six tabs:

DC Remotes, **MIDI Remotes**, **DMX Remotes**, **OSC**, **PSN**, and **MVR**.

Common Procedures

The **DC Remotes**-, **MIDI Remotes**-, and **DMX Remotes**- tabs have a number of options in common. These are explained below. For the options specific to each Remote, see the corresponding topics.

- Enabled: Toggle between **Yes** and **No** to enable the Input for the corresponding configuration line.
- Target: Set a target in the Assignment Editor.
- Fader: Sets the Fader to the corresponding target.
- Key: Sets the Key to the corresponding target.
- Trigger On / Trigger Off: These values define the range of the key reaction. If the Trigger On, for example, is set to 75 %, the Remote reacts as soon as the incoming signal exceeds this value. If the Trigger Off, for example, is set to 25 %, the Remote stops reacting as soon as the incoming signal has fallen below this value.
- In From / In To: These values define the range of the input signal reaction for the defined fader.
- Out From / Out To: These values recalculate the input signal range to match up the output signal range. The range of a fader is defined by the output signal range. If a fader, for example, should not be moved to 100 %, restrict the Out To value to 90 %.In / Out:These columns at the end of the table display the signal value of the incoming signal (In) and the resulting value for the selected fader function (Out).
- In / Out:These columns at the end of the table display the signal value of the incoming signal (In) and the resulting value for the selected fader function (Out).
- Note: Opens the note editor of the corresponding configuration line.

Output Configuration Settings for Remotes

The Connector Configuration menu offers multiple options for DMX, Timecode, and Remotes. The Output Configuration options for remotes are explained here. For all other configurations, see **Connector Configuration**.

- MIDI Data Mode: Select the Mode for MIDI. For more information, see **MIDI Remotes**.
  - **In**: Receiving MIDI data.
  - **Out**: Sending MIDI data.

- **In & Out & Thru**: Outputs and transmits received MIDI data.
- **In & Out**: MIDI data can be received and different MIDI data can be output without sending the incoming data.

- MIDI Offset: Changes the MIDI Index. For more information, see **MIDI Remotes**.
- DC Start: Set the trigger for the DC Remote. For more information, see **DC Remotes**.
- OSC Interface: These settings mirror those in the In & Out menu. For more information, see **Interfaces and IP**.
- PSN Interface: These settings mirror those in the In & Out menu.

Examples

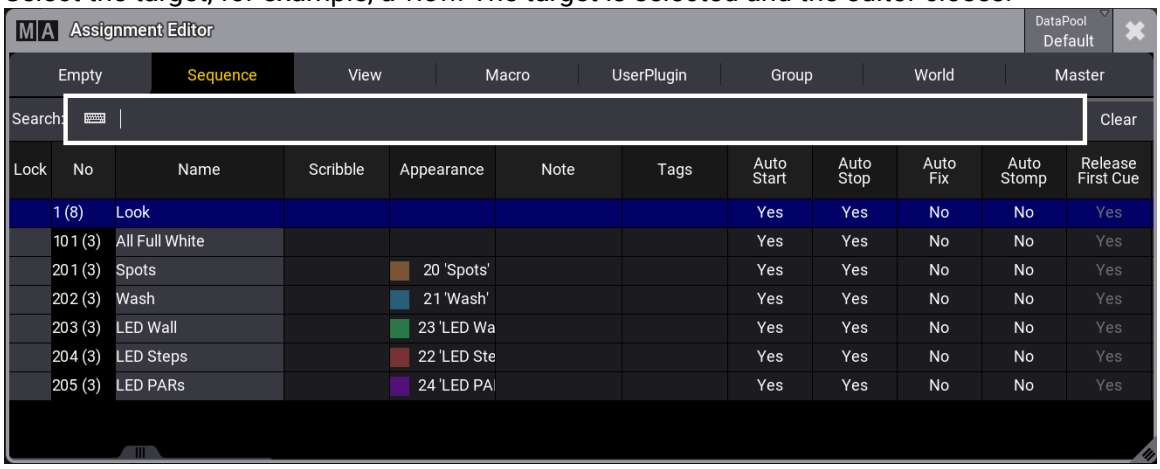
Example 1

To add a new element, such as, DC Remote:

	Hint:
	Make sure that the corresponding hardware is connected properly. For more information, see **First Steps**.

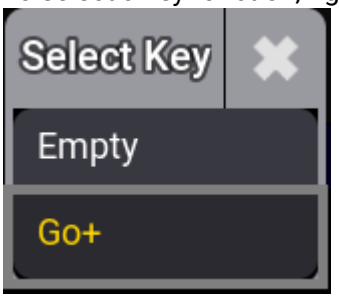
Each step of the example is shown in the video below or in the text below.

1. Open the In & Out menu.
2. Tap DC Remotes.
3. Tap Insert new DCRemote. A new DC remote configuration line is added.
4. To enable the Remote Input, tap Enable Input. The button turns yellow. For more information about session and station communication, see Station Control.
5. Set the Signal of the remote, for example, 1.
6. Two finger-tap the corresponding configuration line in the Target column. The Assignment Editor opens.
7. Select the target, for example, a view. The target is selected and the editor closes.



Assignment Editor

8. To select a key function, right-click or tap and hold Key. The Select Key pop-up opens.



Select Key pop-up

9. Trigger the DC Remote signal. The In and Out values go to 100%.

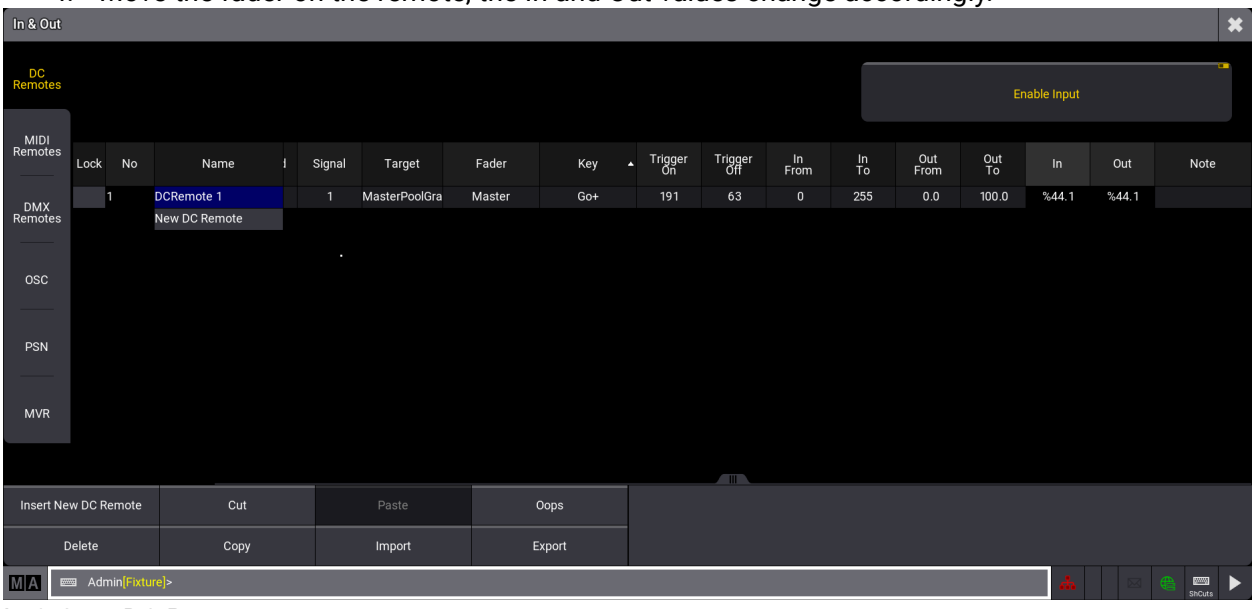
	Hint:
	DC Remotes and DMX Remotes thresholds are defined in percent, while MIDI Remotes thresholds are defined between 1 and 127 MIDI velocity.
	Hint:
	The different readout options take effect on the Trigger On/Off, In From/To, Out From/To values.

Example 2

	Restriction:
	grandMA3 onPC products, such as the grandMA3 command wing, can switch on and off but do not fade. For more information, see **Connect DC Remote In**

To add a fader remote:

1. Follow steps 1-6 from above.
2. Select a Target, e.g. **Master**.
3. Set Fader to **Master**.
4. Move the fader on the remote, the In and Out values change accordingly.



In & Out - DC Remotes

Example 3

In some cases, scaling of the input and output values is needed. The range of the incoming and outgoing values can be modified.

To set In From to 20%:

1. Right-click or tap and hold In From. The calculator opens:

Calculator In From

1. Tap and hold Dec8. A dropdown opens.
2. Tap Percent. The entity has changed.
3. Press 2 0 and press Please. The In From value is entered.
4. Move the fader on the remote, the In and Out values change accordingly.

In & Out - DC Remotes

	Hint:
	For detailed information about the values, read the **Remote keyword** topic.

Subtopics

- **DC Remotes**
- **MIDI Remotes**
- **DMX Remotes**
- **OSC (Open Sound Control)**
- **PSN (PosiStageNet)**
- **MVR-xchange**

1.48.1. DC Remotes

The DC Remotes tab is used to configure the DC Remote Control input on the rear panel of the console, command wing or I/O node.

To learn more about the hardware part of the input, read the **Connect DC Remote In topic**.

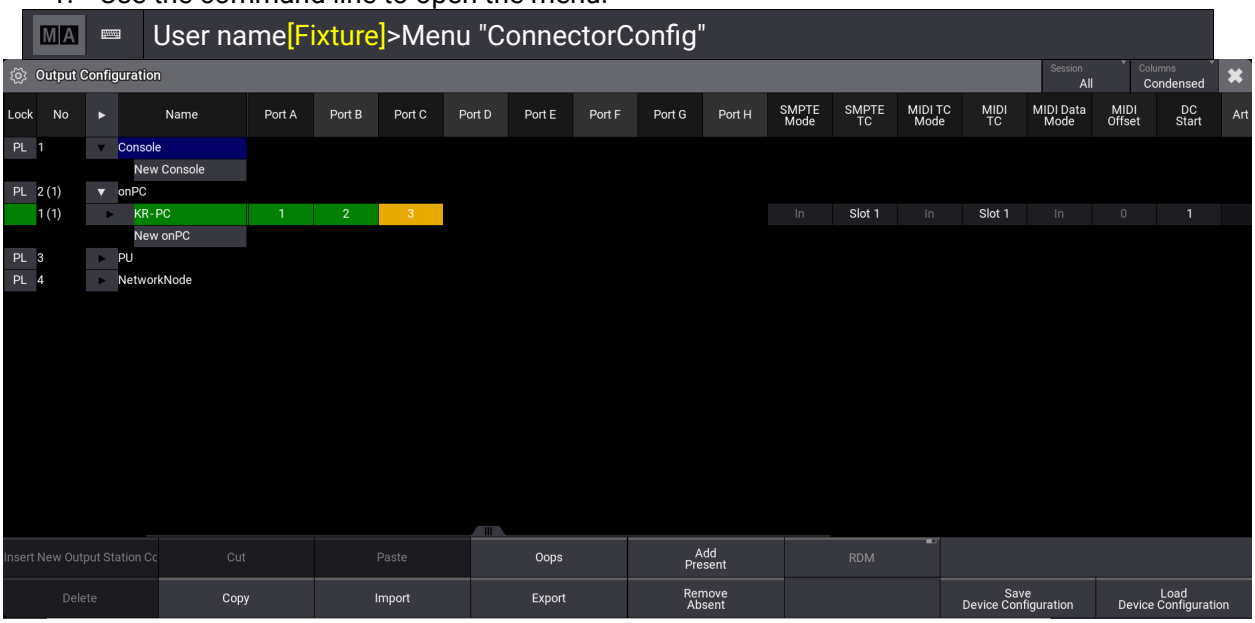
Example

To set the DC start signal:

1. Press Menu and tap Connector Configuration. The Connector Configuration menu opens:

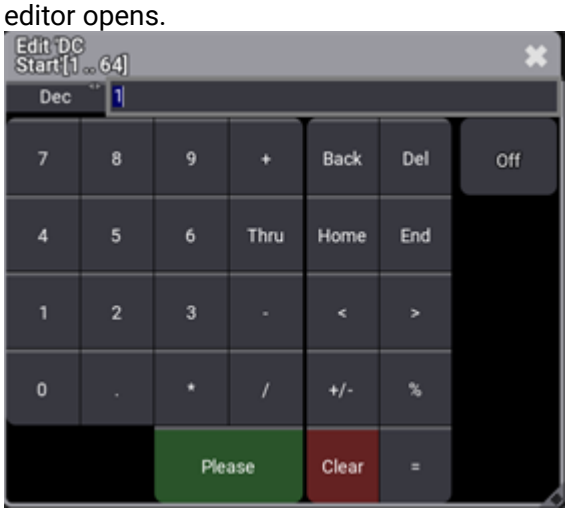
-Or-

1. Use the command line to open the menu:



Output Configuration menu

2. Right-click or tab and hold the corresponding configuration line below DC Start. The DC Start editor opens.



3. Set the DC Start value and then tap Please. The start signal is set.

	Hint:
	The Signal value in the In & Out menu corresponds with the DC Start value in Connector Configuration.
	Hint:
	• As the DC Remotes connector offers 7 different signals, the start value has to be set. If set to 1, the first pin on the Connector has the DC Start value 1 and the 2nd pin has the value 2. • If, for example, two devices in a session are used for DC remote connections, one device can be set to DC Start 1 and the 2nd to DC Start 8 to gain in total 14 DC Remote options.

To learn more about the general DC Remotes settings, read the **Remote In and Out topic**.

1.48.2. MIDI Remotes

MIDI stands for **M**usical **I**nstrument **D**igital **I**nterface. MIDI is used as a world-wide standard protocol that allows communication between different digital devices, for example:

- a MIDI keyboard and a grandMA3 command wing.
- a MIDI pad controller and a grandMA3 console.
- a grandMA3 console and an audio mixing console.

The MIDI Remotes tab is used to define actions for incoming MIDI notes or MIDI Control Changes (CC).

	Restriction:
	MIDI does not transmit an audio signal.

To learn more about the hardware part of the input, read the **Connect MIDI topic**.

This topic is divided into several chapters:

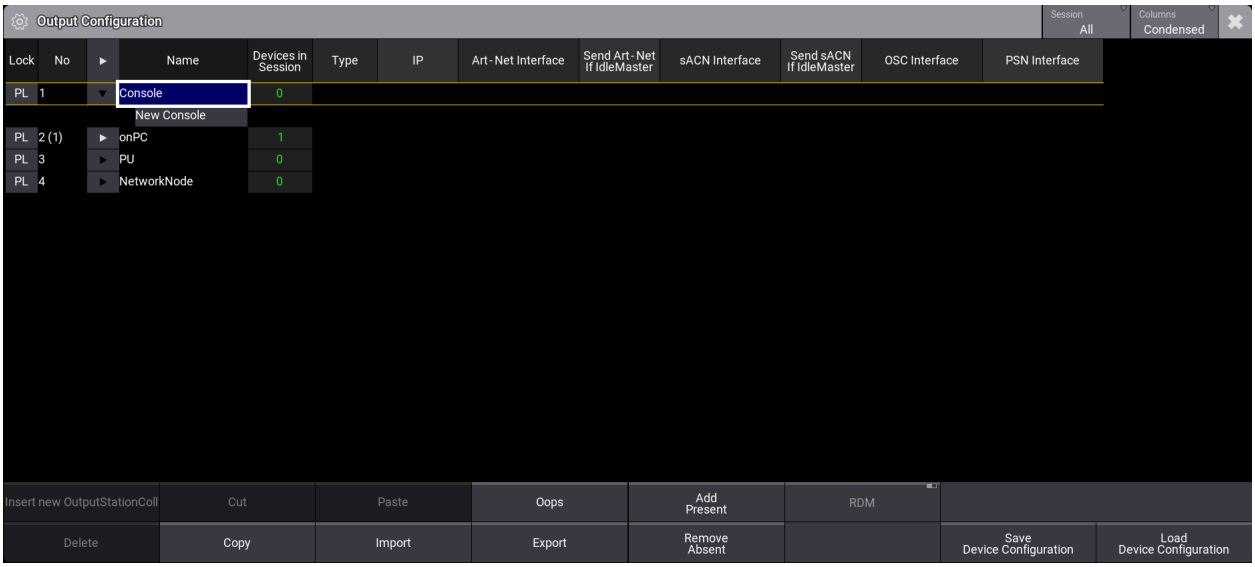
- **Output Configuration Window**
- **In & Out Menu**
- **Examples**
- **Control an External MIDI Device**
- **Receive MIDI**
- **Send MIDI**
- **MIDI Connection via USB**

Output Configuration Window

Adjusting the MIDI Offset will shift the MIDI Index in general.

The MIDI Offset can be adjusted in the Output Configuration window as followed:

1. To set the MIDI Offset, open the **Output Configuration** menu.
2. Open the product in the Output Configuration category tree using the arrows ( ).



Output Configuration menu

3. To set the MIDI Offset to a new value, for example -2, right-click or tap and hold MIDI Offset in the product row. A pop-up opens:

MIDI Offset pop-up

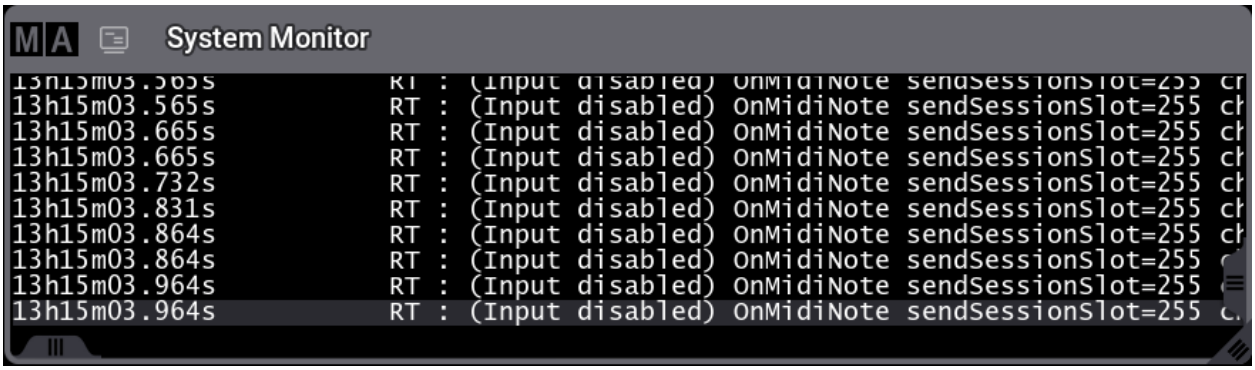
4. The MIDI Offset is adjusted. For example, MIDI Index 50 is changed to 48, after MIDI Offset is set to -2 as shown below:

MIDI Offset changes the MIDI Index (note) shown in the System Monitor.

	Hint:
	If the data input is disabled, a text in the System Monitor next to the actual MIDI Note will inform the user about it.

For example:

- Data Mode input disabled: MIDI Data Mode in the Output Configuration is set to **Out**.
- Input disabled: Enable Input in the In & Out menu is disabled.



System Monitor with hint text about disabled input

In & Out Menu

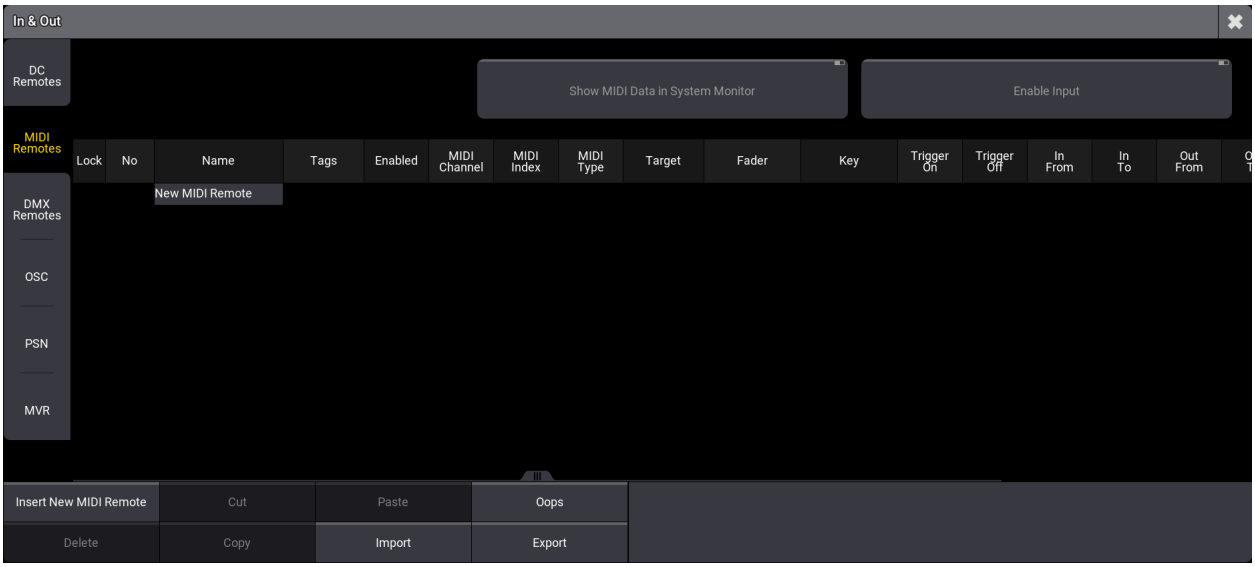
To learn more about the general Remotes settings, see **Remote In and Out**.

The specific parameters for MIDI Remotes are described below:

- MIDI Channel: The MIDI channel value features 16 channels that can be controlled individually. The MIDI receiver must use the same MIDI channel as the MIDI sender to understand each other.
- MIDI Index: The MIDI Index value is the MIDI note or the MIDI control change number (depending on the MIDI Type). The MIDI receiver must use the same MIDI Index value as the MIDI sender to understand each other.
- MIDI Type: Four MIDI Types are selectable:
    - **Note:** Only MIDI note is analyzed.
    - **NoteAttack:** In addition to the MIDI note, the attack of the note will be analyzed. The higher the attack value is, the more a fader moves.
    - **NoteAttackDecay:** Together with the MIDI note, the Attack and Decay of the MIDI Note will be used.
    - **Control:** MIDI Control Change will be used.

Examples

1. To adjust the settings of the MIDI Remotes, switch to the In & Out window.
2. To open the In & Out window, press Menu and then tap In & Out.
3. To open MIDI Remotes, tap MIDI Remotes.



MIDI Remotes menu

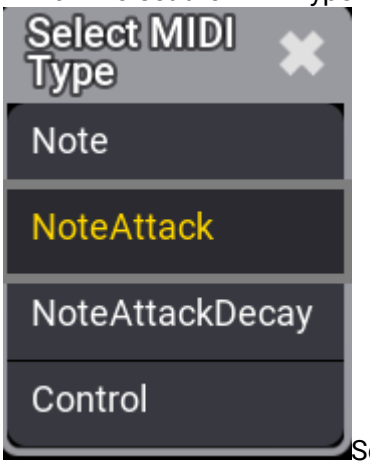
4. To set the MIDI Channel value to a new value, right-click or tap and hold MIDI Channel. The Edit MIDI Channel pop-up opens:

MIDI Channel Editor

5. To set the MIDI Index value to a new value, right-click or tap and hold MIDI Index. The Edit MIDI Index pop-up opens:

MIDI Index Editor

6. To set the MIDI Type value to a new value, right-click or tap and hold MIDI Type. A pop-up opens:



Select MIDI Type

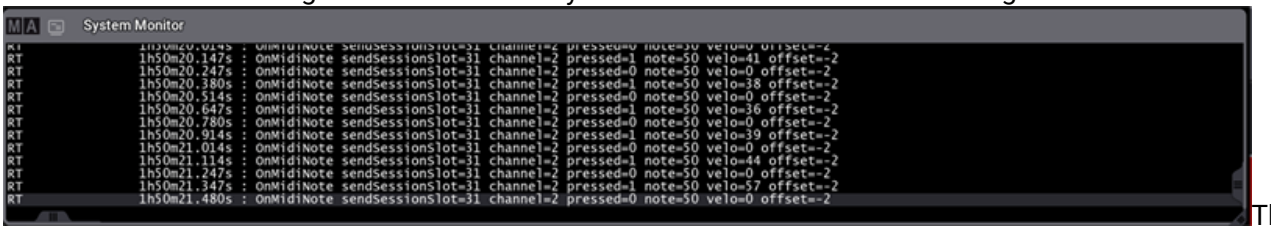
Control an External MIDI Device

It is possible to connect an external MIDI device to certain grandMA3 products to receive messages. Therefore, a standard MIDI cable can be used.

Requirements:

- 1x **grandMA3 console** or **grandMA3 onPC** station either with a **command wing**, an **I/O node** or a third-party USB device connected.
- 1x MIDI capable device.
- 1x Standard MIDI cable.

1. Connect the MIDI cable to the MIDI data output connector (MIDI Out) from the external MIDI device to the MIDI data input connector (MIDI In) on the console. Make sure the external MIDI device is powered, too!
2. Open Menu - In & Out - MIDI Remotes.
3. To make the messages of the external MIDI device visible in grandMA3, enable Show MIDI Data in System Monitor.
4. To allow MIDI inputs on a grandMA3 console from an external MIDI device, make sure to enable Enable Input.
5. To see the incoming MIDI messages of the external MIDI device on the console, open the System Monitor window and press different hardkeys (for example the keys) on the external MIDI device. The MIDI messages are listed in the System Monitor as shown in the image below:



The MIDI messages are shown in the System Monitor.

A MIDI message can contain several parameters:

- **channel:** Displays the external devices MIDI channel.
- **pressed:** Shows if a MIDI key is pressed (1) or released (0).
- **note**: Shows the MIDI note which is oblique to a key on the external MIDI device.
- **velo**: Shows the velocity which displays the intensity of a pressed key from 0 to 127.
- **control**: Shows the control change address and its value.
- **offset:** Shows the changes of the MIDI Index.

Receive MIDI

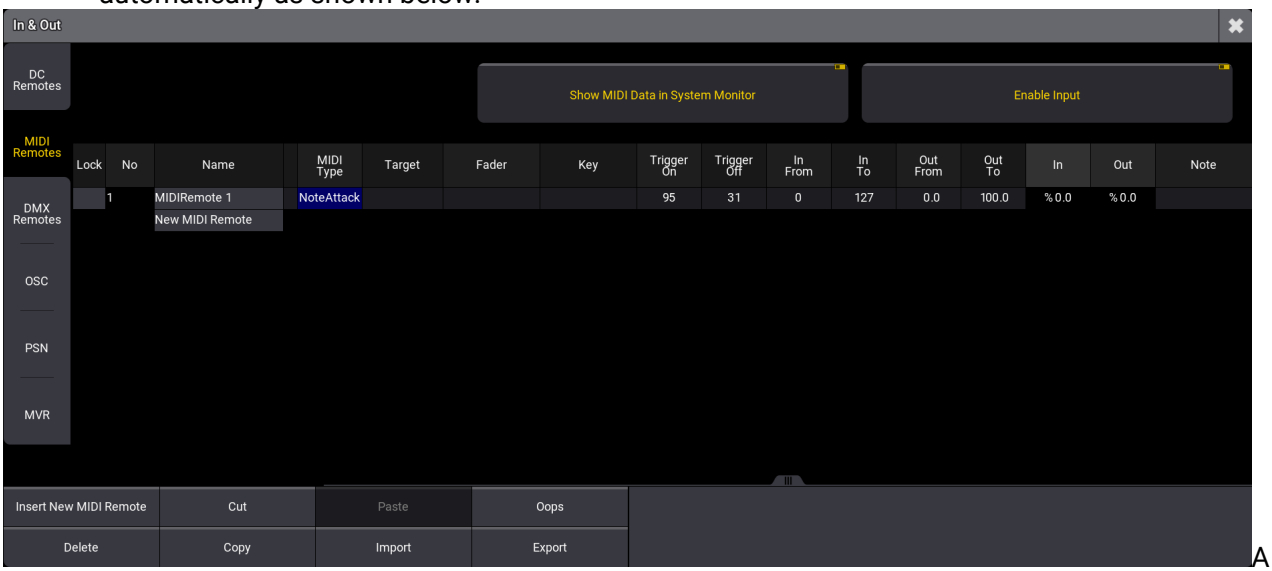
Requirement:

- 1x grandMA3 demo show file
- To receive MIDI messages, make sure the MIDI Data Mode is set to In in the Output Configuration menu:

After connecting an external MIDI device to the grandMA3 console, it is time to insert and customize a MIDI remote.

In this example, the sequence is controlled by pressing a key on a external MIDI keyboard:

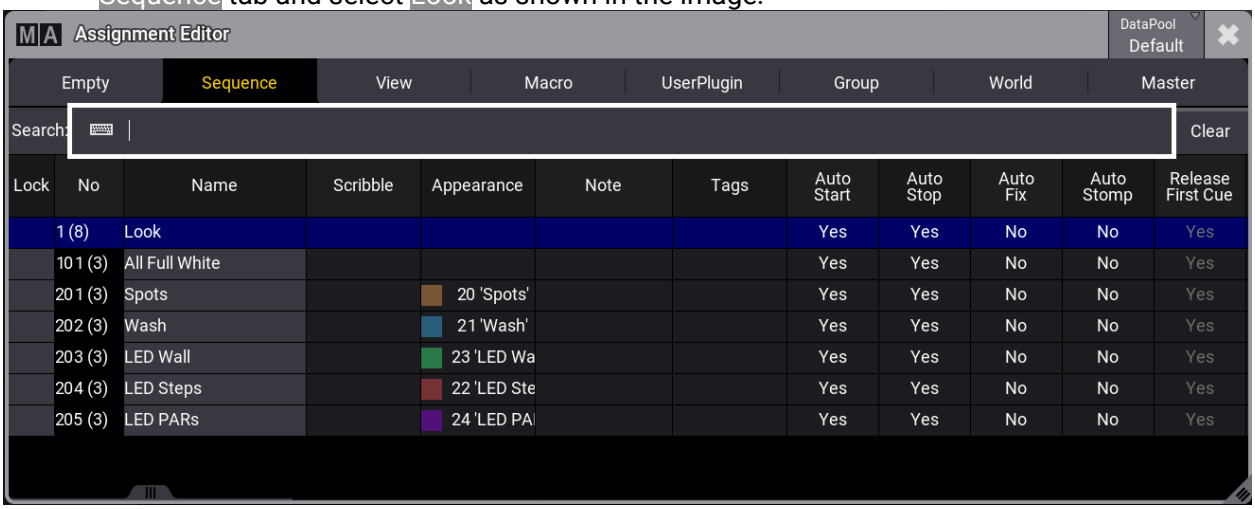
1. To insert a new MIDI remote, tap Insert new MIDIRemote. The row MIDIRemote 1 is added automatically as shown below:



A new MIDIRemote is set in the MIDI Remotes-tab.

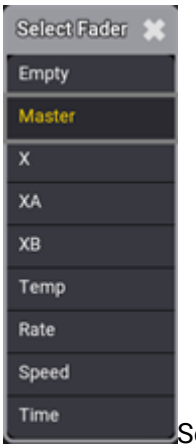
2. To rename, tap MIDIRemote 1 in the Name column. Start to type "Key 1", tap Enter on the virtual keyboard, or press Please.
3. To allocate a specific hardkey on the MIDI keyboard, press a key on the keyboard and use the information **channel** and **note** from the System Monitor window to edit the MIDI Channel and MIDI Index columns. For this example MIDI Channel is set to 2 and MIDI Index is set to 48.
4. Set the MIDI Type to NoteAttack, to make use of the velocity.

5. To open the Assignment Editor, tap and edit the empty cell in the Target column. Tap the Sequence tab and select Look as shown in the image:



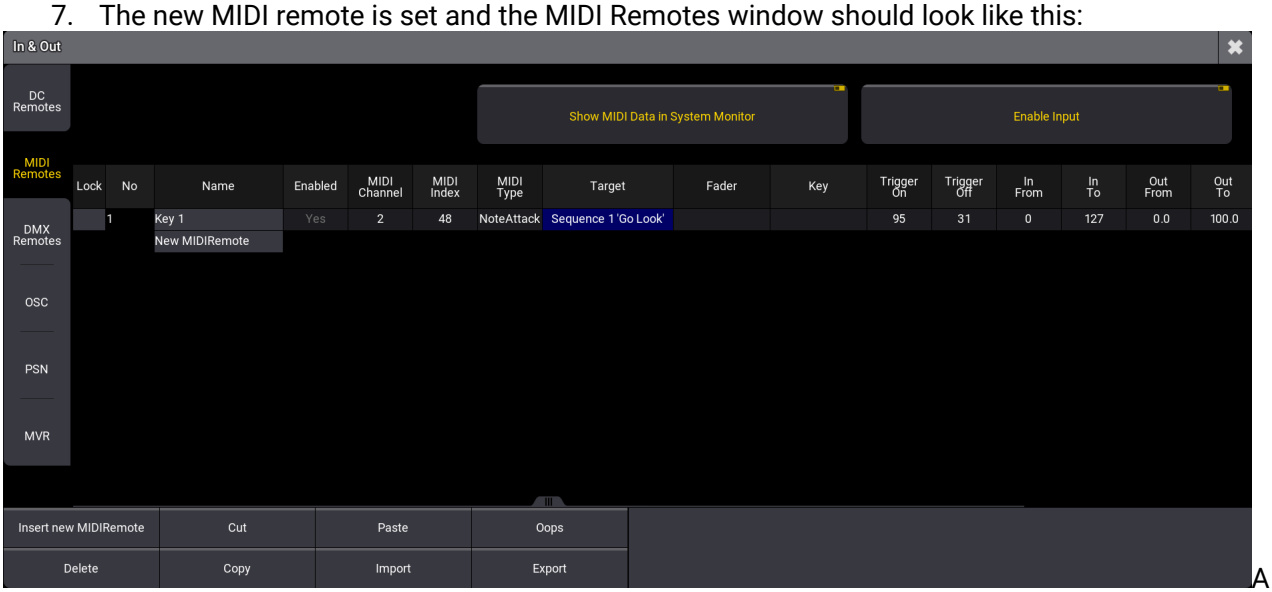
Choose a target in the Assignment Editor

6. To open the Select Fader pop-up, edit the empty Fader column. Set the Fader to Master as shown in the image below:



Select Fader pop-up.

7. The new MIDI remote is set and the MIDI Remotes window should look like this:



A new MIDI remote is set.

	Hint:

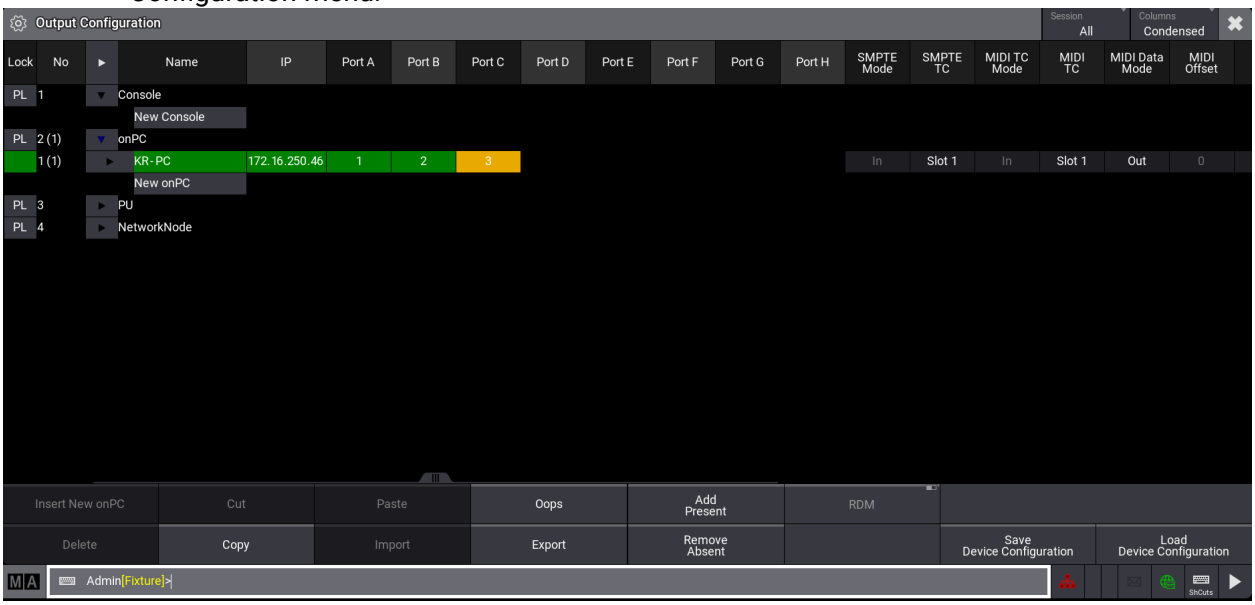
	The MIDI output channel can be changed on the external MIDI device.
	**Hint:**
	To reproduce this example using syntax, see **Remote** keyword.

Send MIDI

It is also possible to send MIDI messages from a grandMA3 console to an external MIDI device.

To send MIDI messages from a grandMA3 console:

- Connect the MIDI cable to the MIDI data input connector (MIDI In) from the external MIDI device to the MIDI data output connector (MIDI Out) on the console.
- MIDI messages can be sent using the **SendMIDI** keyword.
- To send MIDI messages, make sure the MIDI Data Mode is set to Out in the Output Configuration menu:



MIDI Data Mode is set to Out in the Output Configuration menu.

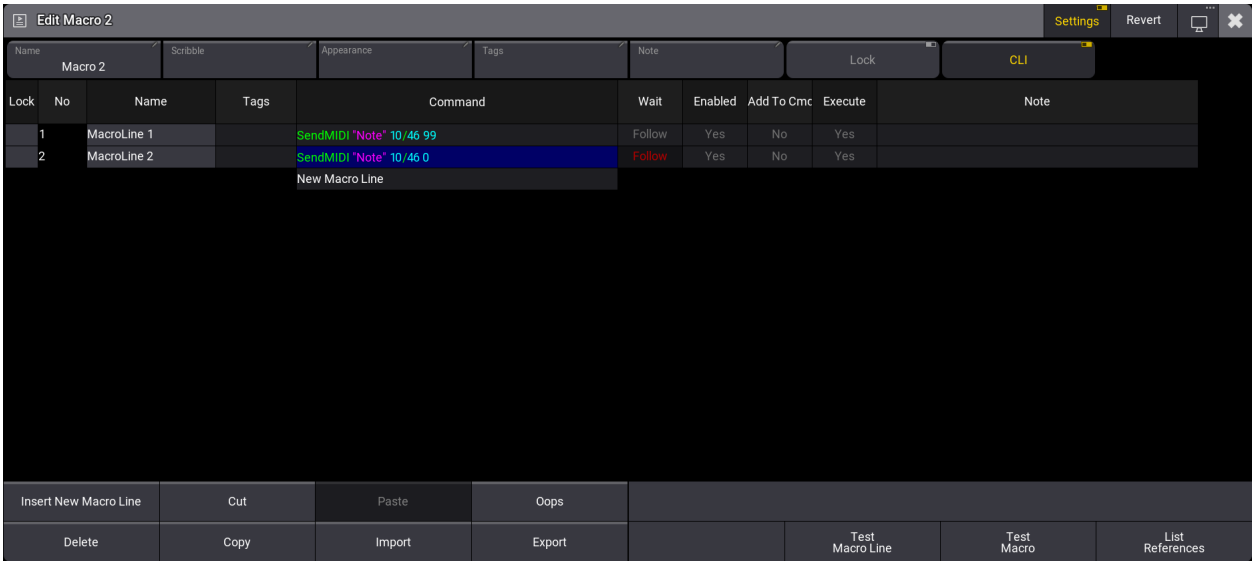
The given MIDI channel (1 to 16) in the command line will be used to send out a MIDI message. For the example below, MIDI channel 10, Index (note) 46 and velocity 99/0 is used:

For the example, MIDI channel 10 is used.

The MIDISend command applies the following rules:

- If a MIDI channel is entered in the command, the entered MIDI channel will be used. If no MIDI channel is entered, MIDI channel 1 will be used.
- If a velocity is entered in the command, the entered velocity will be used. If no velocity is entered, a velocity full (127) will be used.
- If no status (On or Off) is entered in the command, On will be used.

Depending on the external MIDI device, it is necessary to send a MIDI message to the external MIDI device that mimics the release of a hardkey, by sending an **Note Off** (Value 0) message.This is shown in the image below:



Macro Editor

MIDI Connection via USB

An external device can also be connected to the grandMA3 onPC Software using a third-party USB MIDI device.

	Restriction:
	Connecting an external MIDI device via USB is only possible using the grandMA3 onPC Software.

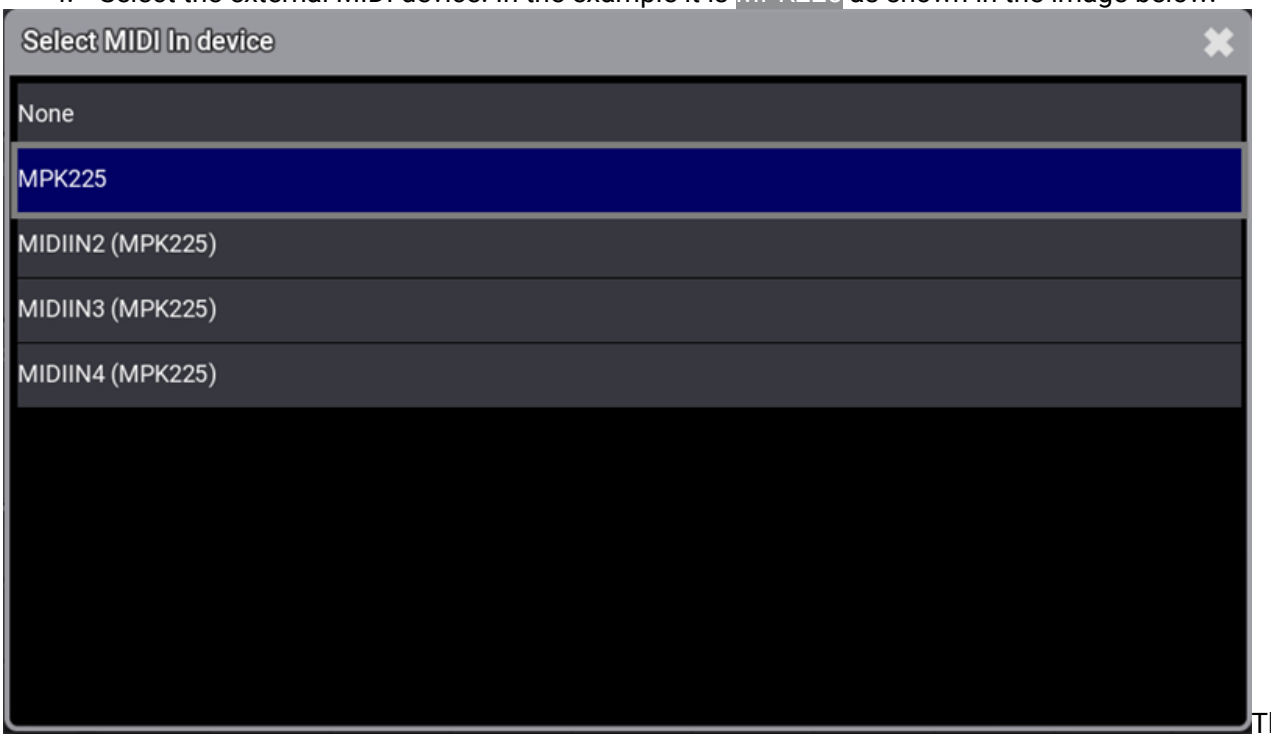
Requirements:

- 1x grandMA3 onPC Software
- 1x MIDI capable device
- 1x USB cable

To connect an external MIDI device via USB:

1. Connect both, the external MIDI device and computer, using an USB cable.
2. Press Menu, tap Settings and then tap onPC Settings. The onPC Settings window opens.
3. To select the external MIDI device, tap MIDI In Device to receive MIDI or MIDI Out Device to output MIDI. The Select MIDI In device pop-up opens:

4. Select the external MIDI device. In the example it is MPK225 as shown in the image below:



The Select MIDI IN device pop-up shows the connected keyboard.

5. The external MIDI device is connected to the grandMA3 onPC Software. To set up a MIDI remote, go back to **Receive MIDI**.

	Hint:
	For more information about the grandMA3 onPC settings, see **onPC Local Settings**.

1.48.3. DMX Remotes

The **DMX Remotes** tab uses DMX channels as remote triggers. The DMX source can also be the console itself.

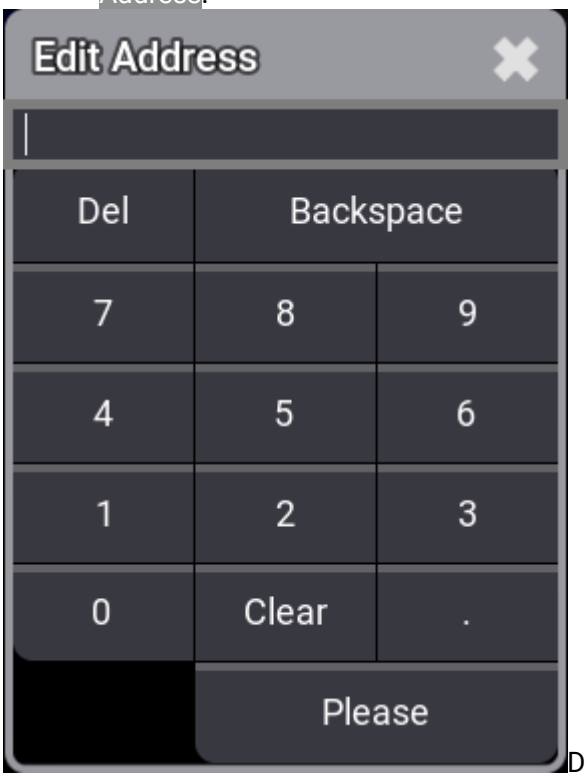
- To learn more about the hardware part of the input, read the **Connect DMX topic**.
- To learn more about the general Remotes settings, see **Remote In and Out**.

The following parameters are used in the DMX Remotes configuration:

- Address: Sets the DMX Address.
- Resolution: Sets the DMX Resolution.

Example

1. To adjust the settings of the DMX Remotes, tap DMX Remotes. The DMX Remotes menu opens:
2. To set the DMX address the DMX Remote input should listen to, right-click or tap and hold Address.



DMX address pop-up

3. To adjust the DMX resolution, open the Resolution cell of the DMX Remote entry you want to edit.

DMX resolution pop-up

	Hint:
	The DMX Remote can be controlled by 16 bit or 24 bit DMX channels. This allows for more precise control.
	Hint:
	On the controlling DMX device, the 16 bit and 24 bit channels must be patched directly after the 8 bit channel.

1.48.4. OSC (Open Sound Control)

Open Sound Control, or OSC, is a client and server system that defines a message address pattern used to address elements in the receiving server. Open Sound Control allows devices of different types to control other devices via a peer-to-peer messaging protocol. OSC messages are human-readable, so they are more than just numbers and strings (unlike, for example, MIDI Show Control, or MSC).

The grandMA3 software supports OSC 1.1. For more general information about OSC, e.g. OSC Packets, see **https://ccrma.stanford.edu/groups/osc/spec-1_0.html.**

OSC Structure

OSC messages follow a specific pattern:

"(/prefix)/[OSC Address],[OSC Type],[Value]"

Prefix: This is optional, depending on your system setup. It can be used in a more complex OSC network to distinguish messages intended for one set of devices (e.g., lighting consoles) from others (e.g., sound consoles). If a prefix is specified, only OSC messages beginning with the specified prefix are processed, and the prefix is prepended to outgoing OSC messages.

	**Hint:**
	The prefix must not contain any slashes ("/").

OSC Address: This is the target you are controlling on the receiving device(s), for example /Fader201 would be the address to move the fader for executor 201 in grandMA3. Sometimes the address will be more complex, for example /Page1/Fader201 would be the address to move the fader for executor 201 on page 1 in grandMA3.

	**Restriction:**
	Only OSC messages are supported when receiving or sending an OSC packet. OSC Bundle messages are currently not supported.

OSC Type: This is the type of value you're sending, for example:

i = integer

f = float

s = string

T = true

F = false

Value: This is the value you send to the target.

An example OSC command to set the fader for executor 201 to 100 might be:

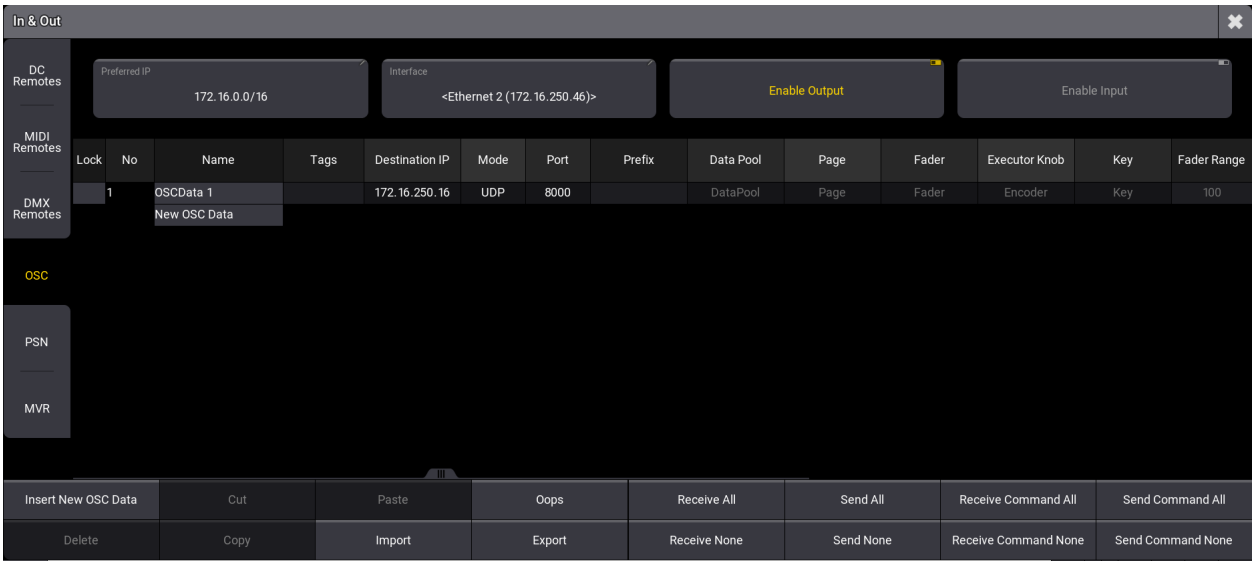
- "/Page1/Fader201,i,100"
- or with a prefix to specify only, e.g. grandMA3 devices: "/gma3/Page1/Fader201,i,100"

OSC Menu

To open the OSC Menu:

- Menu - In & Out - OSC.

The OSC Menu opens:



OSC Menu

At the top of the menu, the following four options can be set:

- Preferred IP: This is the preferred IP address or address range used by the OSC protocol.
- Interface: Tap this button to open the **Select Interface pop-up** to select the desired network interface. For more information, see **Interfaces and IP**.
- Enable Output: This toggle button must be enabled to transmit OSC. See **Station Control** in Network preferences for more information.
- Enable Input: This toggle button must be enabled to receive OSC. See **Station Control** for more information.

	Hint:
	When receiving OSC messages, Enable Input will highlight its title bar in yellow. When sending OSC messages, the title bar of Enable Output will be highlighted.

The following are the specific parameters that can be set in the OSC menu:

- Name: Sets the name for this configuration.
- Destination IP: Sets the IP address for sending OSC data. A specific IP address or a broadcast IP can be set.
- Tags: Opens the Tags editor.

- Mode: OSC packets can be sent via UDP or TCP.
- Port: Specifies the network port of the incoming and/or outgoing OSC packets.
- Prefix: A prefix can be set by the user if he needs to. A prefix can be used for example as a criterion for limiting the range of possible receivers, e.g. /lighting would only take packets with /lighting into account, and discard OSC packets with the /sound-prefix.
- Page: Specifies which OSC Address of incoming OSC messages is routed to pages.
- Fader: Specifies which OSC Address of incoming OSC messages is routed to faders.
- ExecutorKnob: Specifies which OSC Address of incoming OSC messages is routed to the mini encoders.
- Key: Specifies which OSC Address of incoming OSC messages is routed to keys.
- FaderRange: Specifies which OSC value range is used for the fader, e.g. FaderRange 255 sets OSC 0-255 to 100%.
- Receive: Specifies if OSC data (but no commands) shall be received.
- Send: Specifies if this OSC configuration sends OSC data (but no commands).
- Receive Command: Specifies if commands for the command line will be received via OSC. This setting is independent of the general receive setting.
- Send Command: Specifies if commands of the command line will be sent via OSC. This setting is independent of the general send setting.
- EchoInput: Specifies if the input data shall be displayed in the system monitor.
- EchoOutput: Specifies if the output data shall be displayed in the system monitor.
- Note: Sets a note for this configuration.

With the buttons Receive All, Receive None, Send All, Send None, Receive Command All, Receive Command None,Send Command All, and Send Command None all OSC configuration lines can be modified together for the properties Receive, Send, Receive Command and Send Command.

Receive OSC

Requirement:

• A network connection is established and the Interface is on both devices. For more information, see **Enabling or Disabling the Network Connection**.
• Make sure that the network protocol (UDP, TCP), and the port are set correctly. Please note that the port configuration is used for sending and receiving OSC data.

	**Hint:**
	If you want to use different ports for sending and receiving, you can create multiple configuration lines.

Example

Example to receive OSC packets on a grandMA3 light console:

1. Open the In & Out menu / OSC.
2. Tap Enable Input. The input is enabled.
3. Two-finger tap the first row below Receive in the grid. The parameter is set to **Yes**.

	**Hint:**
	When receiving OSC messages, Enable Input will highlight its title bar in yellow.

	Hint:
	Please note that in the example above, no prefix is defined.

For more examples, see **Advanced Examples**.

Command Line Control

The entirety of the grandMA3 command line can be accessed via OSC by using the "/cmd" OSC Address and the string 's' OSC Type.

	Hint:
	To receive OSC messages for the command line, set Receive Command to **Yes** in the corresponding OSC configuration line.

Examples that can be sent from another console to the receiving console:

- SendOSC 1 "/cmd,s,FaderMaster Page 1.201 At 50": Brings fader 201 on page 1 to 50% (same as the examples above but using gMA3 command line syntax instead).
- SendOSC 1 "/cmd,s,FaderMaster Page 1.201 At 50 Fade 5": Brings fader 201 on page 1 to 50% and additionally adds a fade time of five seconds
- SendOSC 1 "/cmd,s,Fixture 1 At 75": grandMA3 command line syntax is used to execute the command "Fixture 1 At 75" in the command line.
- SendOSC 1 "/cmd,s,Go+ Exec 402": Triggers executor 402.
- SendOSC 1 "/cmd,s,Patch Fixture 1 3.42": Patches fixture 1 to address 42 in Universe 3.

Object Playback Control

The Playback of the following pool objects are controlled through OSC:

- Sequences
- Masters
- Groups
- Presets
- Sounds
- Worlds
- Plugins
- Screen Configuration
- Timers

Pool objects are addressed by their enumerated address in the grandMA3 directory structure. For more information about the grandMA3 structure, see **List keyword**.

Target	Addresses	Type Tags	Arguments	Results	Example
Sequence X	/13.13.1.6.X	si	<Key function>,1	Press the Key of the given Key function.	/13.13.1.6.1,si,Flash,1
-	-	si	<Key function>,0	Release the Key of the given Key function	
-	-	sif	<Fader function>,3,0 … 100	Set the Fader of the given Fader function to the	

Target	Addresses	Type Tags	Arguments	Results	Example
				given value in percent.	
-	-	sii	<Fader function>,0 … 3, -100 … 100	Incrementally move the Fader of the Fader function by the given value in percent.	

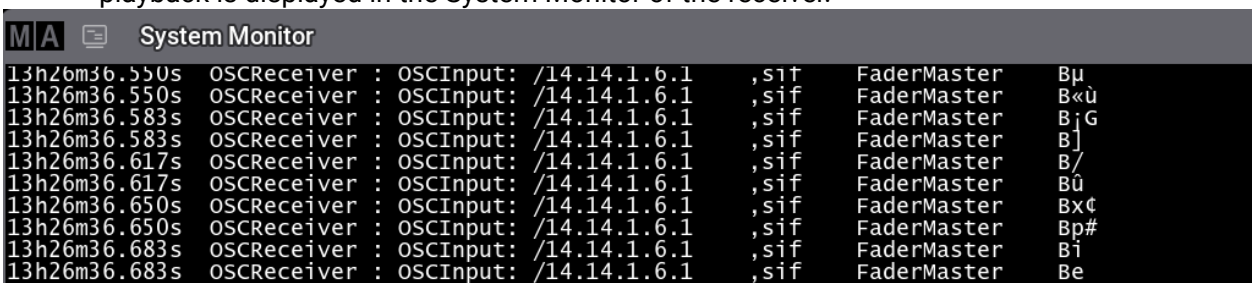
	Important:
	The address in the grandMA3 directory structure may change between software versions. Be sure to check the addresses when performing a software update.

All playback functions that can be used when assigning the object type to an Executor can also be controlled through OSC. For more information, see **Assign Object to an Executor**.

Example

To display the argument of a playback control function in the System Monitor:

1. On the reciving device, press Menu - tap In & Out - tap OSC menu.
2. Tap Enable Input.
3. Set Receive to **Yes** in the corresponding OSC configuration Line.
4. Set Echo Input to **Yes**.
5. Tap to close the In & Out menu.
6. Open Add Window - More - System Monitor.
7. Execute a playback command, e.g. move Fader201 on the sending device. The argument of the playback is displayed in the System Monitor of the receiver.



To retrieve the enumerated address of an object and display it in the Command Line History:

1. Tap the command line.
2. Type Lua "Printf(ObjectList('Master 1')[1]:Addr())" and press Please (Replace the Master 1 in the example with another object). The enumerated address is shown in the command line history.

Command Line History

For more information, see the **Printf(string)** Lua function.

Send OSC

	Hint:
	For more information about sending OSC, see **SendOSC keyword**.

Example 1

To send OSC packets from a grandMA3 onPC station:

1. Open the In & Out Menu / OSC.
2. Tap Enable Output. The output is enabled.
3. Two-finger tap Destination IP in the first row of the grid. The editor opens.
4. Set the Destination IP of the receiving device and then press Please.
5. Two-finger tab the first row below Send Command. The parameter is set to **Yes.**
6. Send data to the receiving device, e.g by using the command: SendOSC 1 "/Page1/Fader201,i,50".
7. The title bar of Enable Output turns yellow and OSC packets are sent from grandMA3 onPC to the grandMA3 light console.

Title Bar of the Enable Output button is yellow when sending data

	Hint:
	When sending OSC messages, the title bar of Enable Output will be highlighted.

Object Playback Feedback

	Hint:
	To send OSC messages for playback action, set Send to **Yes** in the corresponding OSC configuration line.

Playback actions generate OSC output on the following objects, which can be used in the System Monitor when Echo Output is enabled:

- Sequences
- Masters
- Groups
- Presets
- Sounds
- Worlds
- Plugin Components
- Screen Configuration
- Timers

For more information on Plugin Components, see **Plugins**.

Example 2

To send a command, e.g., a playback control function, to a grandMA3 console:

1. On the sending device, press Menu - tap In & Out - tap OSC menu.
2. Tap Enable Output.
3. Set Send to **Yes** in the corresponding OSC configuration line.
4. Set Echo Output to **Yes**.
5. Tap X to close the In & Out menu.
6. Open Add Window - More - System Monitor.
7. Execute a playback command, e.g. move Fader201. The command is send to the receiver.

Example 3

To send OSC data from a grandMA3 onPC workstation and receive it on a console:

Requirement:

- The IP addresses are set up correctly and the network is enabled on both devices .
- Some Fixtures are patched on the receiving console.

Sender:

1. Open the In & Out Menu / OSC.
2. Tap Enable Output. The output is enabled.
3. Set Send Command to Yes.
4. Tap the command line in grandMA3 onPC:
5. Type SendOSC 1 "/cmd,s,Fixture 1 At 75" and press Please.

Receiver:

1. Open the In & Out Menu / OSC.
2. Tap Enable Input. The input is enabled.
3. Set Receive Command to **Yes**.
4. If the command is received, Fixture 1 on the console is dimmed down to 75%.

Example 4

To send OSC data from a grandMA3 onPC workstation to a third party device, e.g., a videoserver:

Requirement:

- The IP addresses are set up correctly on both devices

Sender:

1. Open the In & Out Menu / OSC.
2. Tap Enable Output. The output is enabled.
3. Set Send Command to **Yes**.
4. Tap the command line in grandMA3 onPC:

5. Type SendOSC 1 "/Videoserver/Master,i,100" and press Please.

For more examples, see **Advanced Examples**.

Subtopics

- **Advanced Examples**
- **Use Cases**

1.48.4.1. Advanced Examples

The examples below are in addition to those in the **OSC topic**.

Executor Control

For more information, see **Executor Control** in OSC.

	**Hint:**
	If no EncoderLeft or EncoderRight function is assigned, the Encoder function is used. If an EncoderLeft or EncoderRight function is assigned, it overrides the Encoder function and the Encoder function is ignored.
	**Hint:**
	Each executor can have functions assigned to all handles (keys, faders, encoders), even if no such physical handle exists on that executor. OSC controls these functions regardless of whether the physical handle exists or not.
	**Hint:**
	Address names for DataPool, Page, Fader, Encoder, and Key can be edited in the OSC Configuration menu. Fader ranges can also be edited.

Targets	Addresses	Type Tags	Arguments	Results	Examples
Fader of Executor X on the selected Page in the selected DataPool	/FaderX	i f,	0 ... 100	Set the Fader position in percent.	/Fader201,i,100 /DataPool2/Page10/Fader302,f, 75.5
Fader of Executor X on Page Y in the selected DataPool	/PageY/FaderX	T	-	True is interpreted as 1.	-
Fader of Executo	/DataPoolZ/FaderX	F	-	False is interpreted as 0.	

Targets	Addresses	Type Tags	Arguments	Results	Examples
r X on the selected Page in DataPool Z					
Fader of Executor X on Page Y in DataPool Z	/DataPoolZ/PageY/FaderX	-	-	-	-
Encoder of Executor X on the selected Page in the selected DataPool	/EncoderX	if	-100 … 100	Positive values move the Fader up by the given percentage. Negative values move the Fader down by the given percentage.	/Encoder201,i,-50
Encoder of Executor X on Page Y in the selected DataPool	/PageY/EncoderX	if	-1 -1	Executes the assigned EncoderLeft function.	/Encoder201,i,-1
Encoder of Executor X on the selected Page in DataPool Z	/DataPoolZ/EncoderX	if	1 1	Execute the assigned EncoderRight function.	/Encoder201,i,1
Encoder of Executo	/DataPoolZ/PageY/EncoderX	-	-	-	-

Targets	Addresses	Type Tags	Arguments	Results	Examples
r X on Page Y in DataPool Z					
Key of Executor X on the selected Page in the selected DataPool	/KeyX	if	1 1	Press the Key.	/Key101,i,1
Key of Executor X on Page Y in the selected DataPool	/PageY/KeyX	T	-	Press the Key.	/Key101,T,
Key of Executor X on the selected Page in DataPool Z	/DataPoolZ/KeyX	if	0 0	Release the Key.	/Key201,f,0
Key of Executor X on Page Y in DataPool Z	/DataPoolZ/PageY/KeyX	F	-	Release the Key.	/Key201,F,

Object Playback Feedback

For more information, see **Object Playback Feedback** in OSC.

Sources	Addresses	Type Tags	Arguments	Action	Examples
Sequence X	/13.13.1.6.X	sis	<Key function>,0 … 1,<SequenceName CueNumber CueName>	The Key of the given Key function was pressed (Argument 2 = 1) or released (Argument 2 = 0). The given Cue got triggered when the Key got pressed or released. Argument 3 is not sent if the Key function is "Off".	/13.13.1.6.1,sis,Flash,1,Strobe 1 Cue 1
Sequence X	/13.13.1.6.X	sif	<Fader function>,0 … 3,0.0 … 100.0	The Fader of the given Fader function was set to the given value in percent. Argument 2 represents the type of playback handle that was used: Onscreen Fader/Command (1), Physical Fader (3).	/13.13.1.6.1,sif,FaderMaster,3,63.5
Sequence X	/13.13.1.6.X	sii	<Fader function>,0 … 3,-100 … 100	The Fader of the given Fader function was moved by the given value in percent using a Rotary Knob. Argument 2 represents the type of playback handle that was used: Physical Rotary Knob (0), Onscreen Rotary Knob/Command (1).	/13.13.1.6.1,sii,FaderMaster,0,-1
Master X.Y	/13.12.X.Y	si	<Key function>,0 … 1	The Key of the given Key function got	/13.12.1.1,si,Black,1

Sources	Addresses	Type Tags	Arguments	Action	Examples
				pressed (Argument 2 = 1) or released (Argument 2 = 0).	
Master X.Y	/13.12.X.Y	sif	<Fader function>,0 … 3,0.0 … 100.0	The Fader of the given Fader function was set to the given value in percent. Argument 2 represents the type of playback handle that was used: Onscreen Fader/Command (1), Physical Fader (3).	/13.12.3.1,sif,FaderMaster,3,63.5
Master X.Y	/13.12.X.Y	sii	<Fader function>,0 … 3,-100 … 100	The Fader of the given Fader function was moved by the given value in percent using a Rotary Knob. Argument 2 represents the type of playback handle that was used: Physical Rotary Knob (0), Onscreen Rotary Knob/Command (1).	/13.13.12.3.1,sii,FaderMaster,0,-1

1.48.4.2. Use Cases

The use of OSC in different situations is described in the following topics.

Subtopics

- **TouchOSC**
- **QLab**
- **Open Stage Control**
- **zactrack**
- **Protocol Viewer**

1.48.4.2.1. TouchOSC

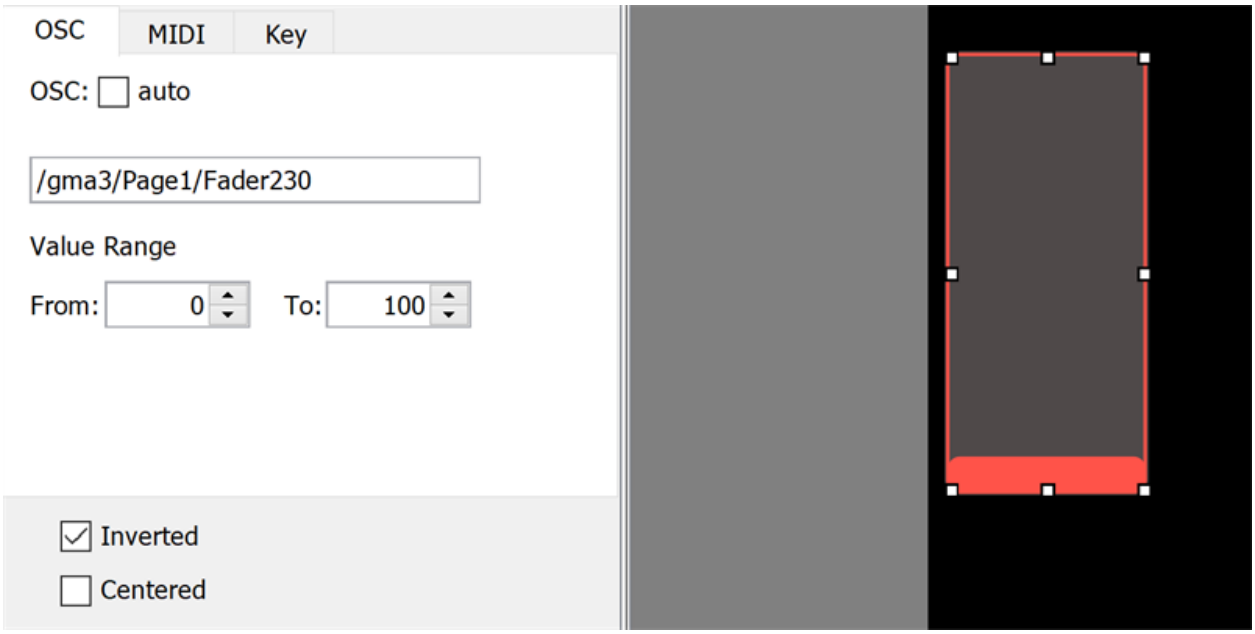
TouchOSC

TouchOSC is a modular OSC and MIDI control surface for Windows, macOS, and Android by **hexler.net**.

It supports sending and receiving Open Sound Control and MIDI messages over Wi-Fi and CoreMIDI inter-app communication and compatible hardware.

Fader

This example will control the fader of Executor 230 of Page 1:



Notes:

- Assumes the OSCData line on the console has a prefix of "gma3" configured. If the prefix is empty, this would just be /Page1/Fader230.
- Assumes the "Page" and "Fader" cells in the OSCData line on the console are set to "Page" and "Fader" respectively (this is the default).

Executor Button

This example will press the button for Executor 230 of Page 1:

Notes:

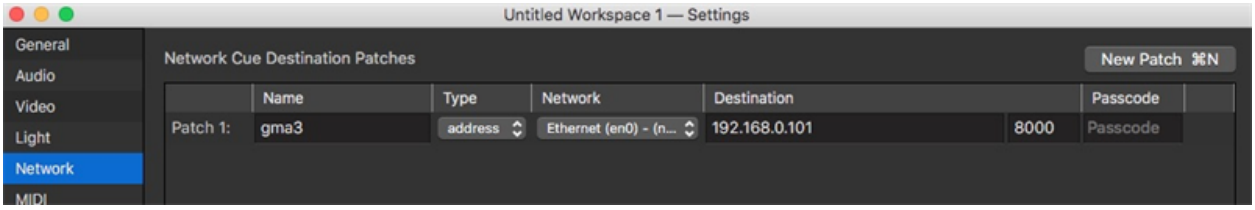
- Assumes the OSCData line on the console has a prefix of "gma3" configured. If the prefix is empty, this would just be /Page1/Key230.
- Assumes the "Page" and "Key" cells in the OSCData line on the console are set to "Page" and "Key" respectively (this is the default).
- The {Send on Press} and {Send on Release} settings (not pictured above) should both be enabled/checked.

1.48.4.2.2. QLab

QLab

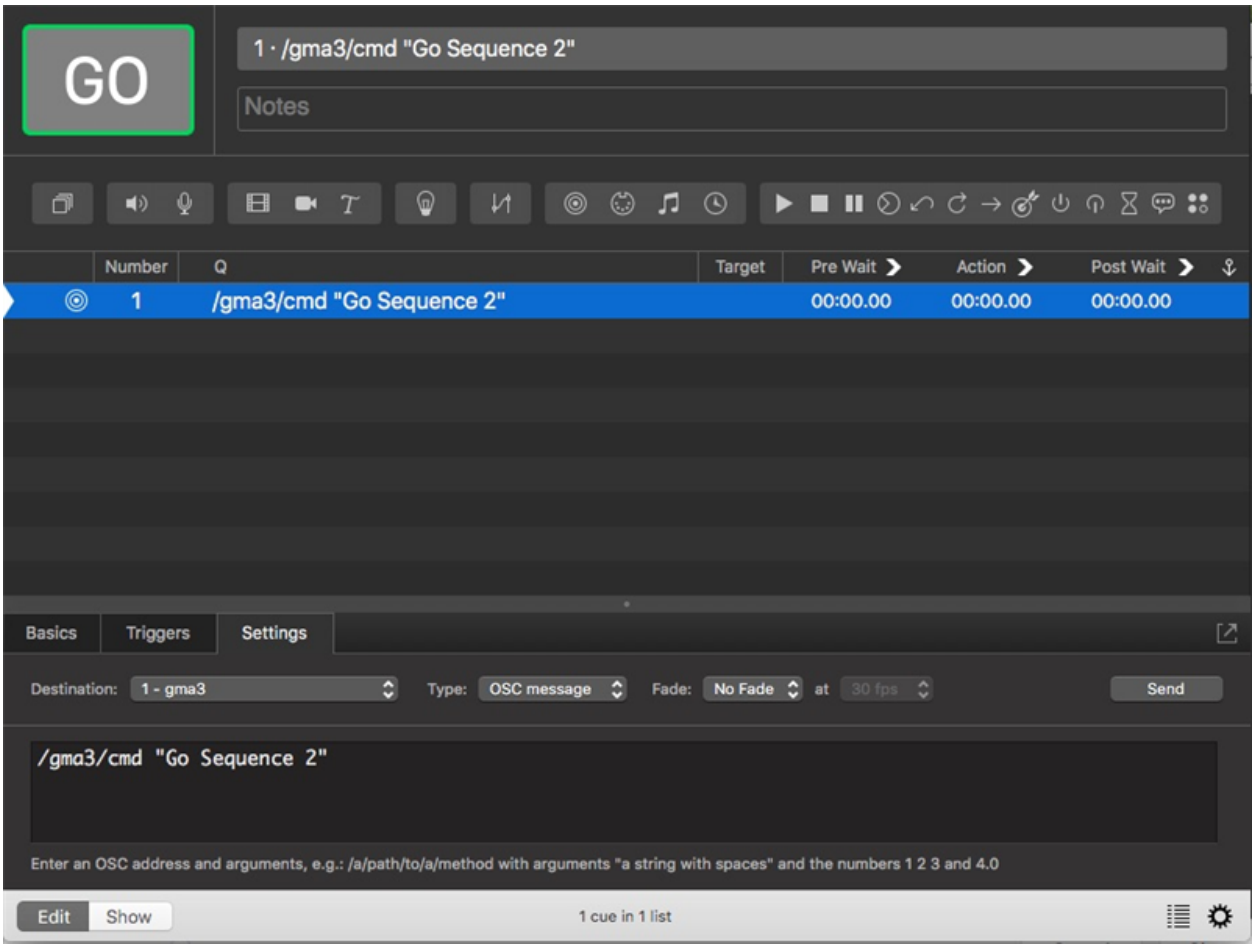
QLab is sound, video, and lighting control for macOS by **qlab.app**.

QLab is fairly simple to use with OSC. The QLab network setting are shown in the example below:



- Name = something to identify this particular configuration
- Network = the network interface on your computer connected to the grandMA3 system
- Destination
    - IP address of the grandMA3 console
    - Port set in the OSCData line configuration in the console (8000 is default)

Cue setup in QLab is shown below:

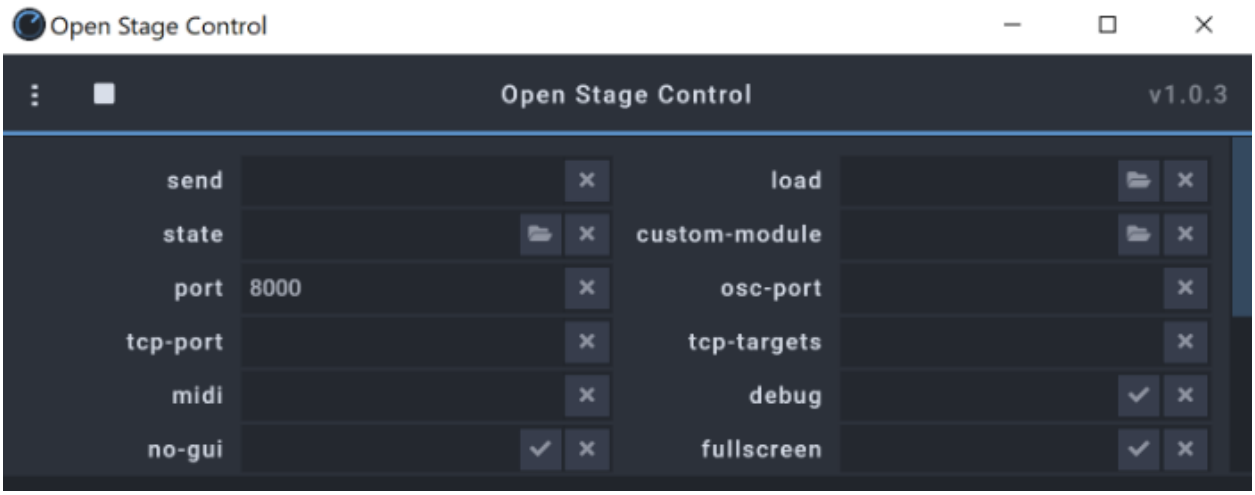


- Destination = configuration as set in the "Network Cue Destination Patches" above
- Type = OSC message
- Enter the desired OSC message
    - OSCAddress - in this example we are sending a command syntax string directly, so the address is: /gma3/cmd
        - This assumes the OSCData line on the console has a prefix of "gma3" configured! If the prefix is empty, this would just be /cmd
        - *Requires "Receive Command" to be enabled for that OSCData line on the console!*
    - Argument - enter the command syntax in quotes, for example, "Go Sequence 2" (advancing to the next cue in Sequence 2)

1.48.4.2.3. Open Stage Control

pen Stage Control

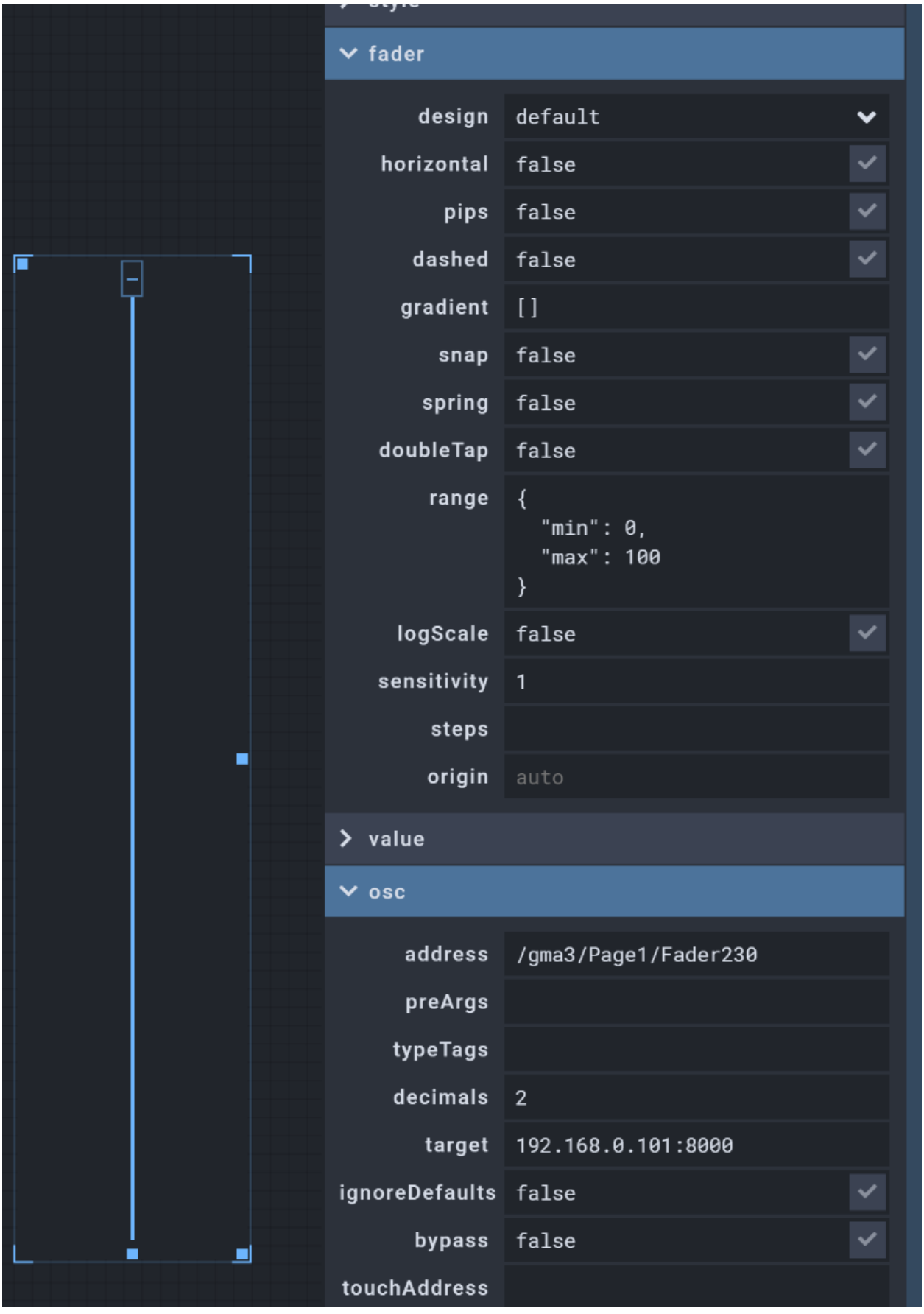
Open Stage Control is a free program used to build a simple OSC interface. When first opening the program, the network configuration settings pop up. Besides telling it which network interface to use on the computer, the only thing to fill in here is the port:



This needs to match the port that has been set in the corresponding OSCData line in the console. Port 8000 is the default. Afterwards, the Open Stage Control session can be started.

Fader

This example will control the fader for Executor 230 on Page 1:

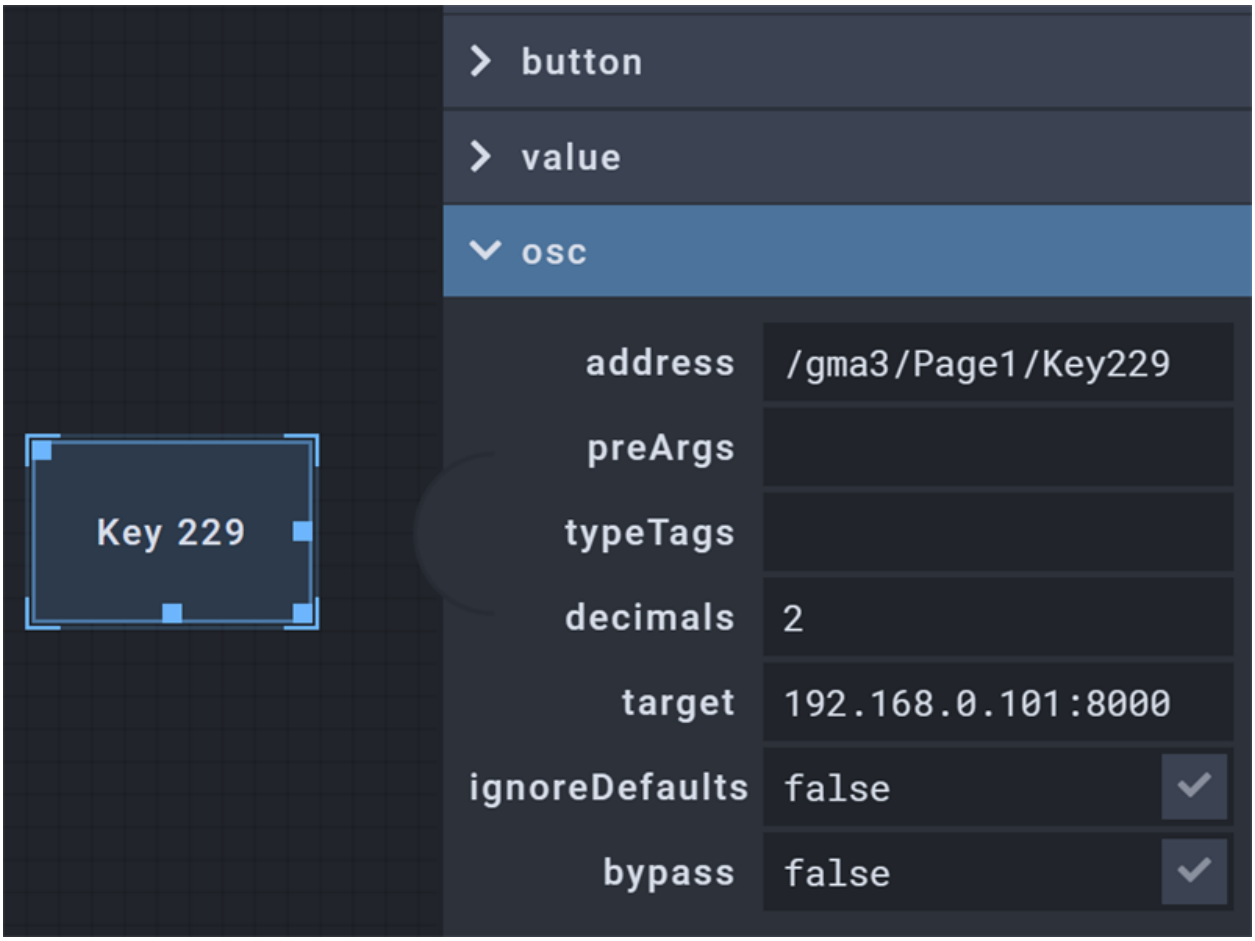


Notes:

- Assumes the OSCData line on the console has a prefix of "gma3" configured. If the prefix is empty, this would just be /Page1/Fader230.
- Assumes the "Page" and "Fader" cells on the OSCData line in the console are set to "Page" and "Fader" respectively (this is the default).
- All of the settings in the picture above are at their defaults except for:
    - fader settings: range: change the 'max' to 100 instead of 1
    - osc settings: address

Executor button

This example will press the button for Executor 229 on Page 1:



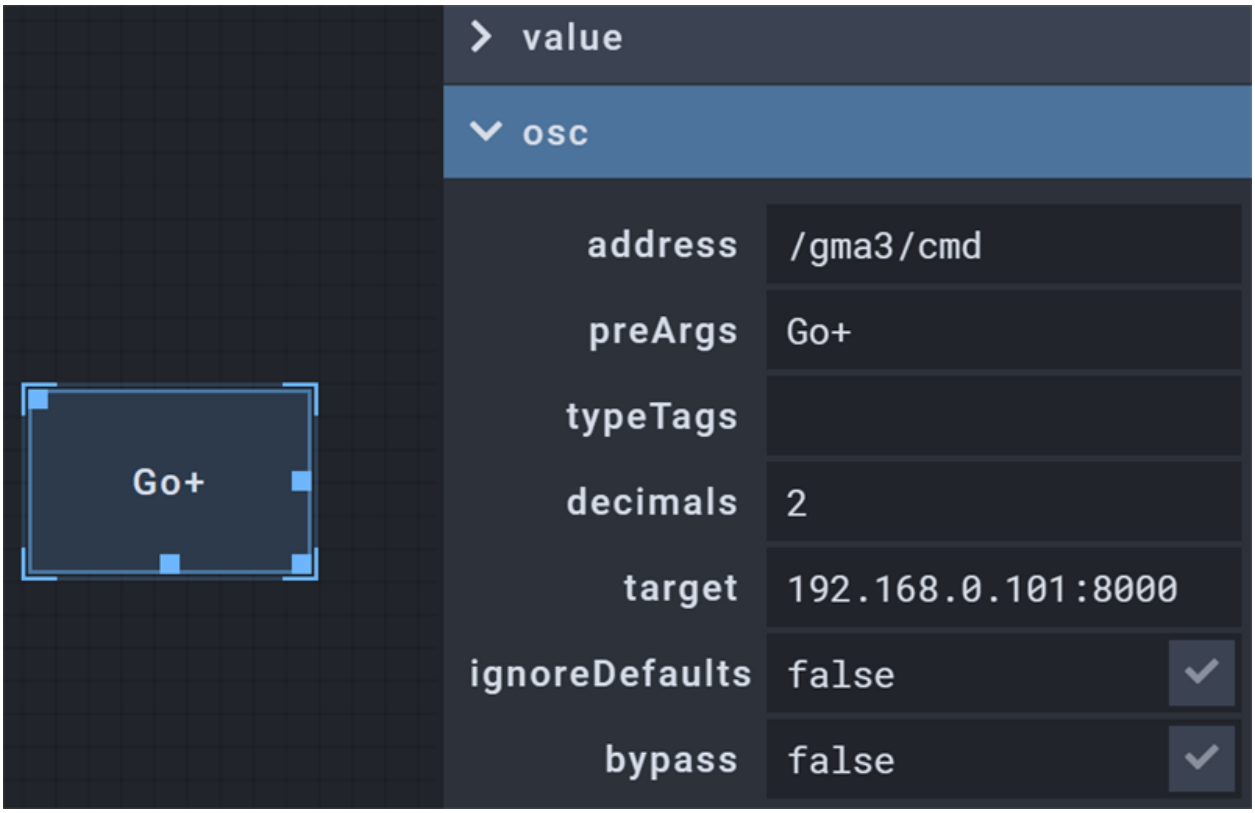
Notes:

- Assumes the OSCData line on the console has a prefix of "gma3" configured. If the prefix is empty, this would just be /Page1/Key229.
- Assumes the "Page" and "Key" cells in the OSCData line on the console are set to "Page" and "Key" respectively (this is the default).

- All of the settings in the picture above are at their defaults except for the address in the osc settings.
- Open Stage Control buttons default to functioning as 'toggle' - you may wish to change this to 'tap'.

Command Line Syntax

Here we have a button that will execute command line syntax on the console, in this case triggering the Selected Sequence:



Notes:

- Assumes the OSCData line on the console has a prefix of "gma3" configured. If the prefix is empty, this would just be /cmd.
- Requires Receive Command to be enabled for that OSCData line on the console.
- All of the settings in the picture above are at their defaults except for:
    - button mode set to "momentary"
    - address
    - preArgs - this is where you enter the syntax string you wish to execute

1.48.4.2.4. zactrack

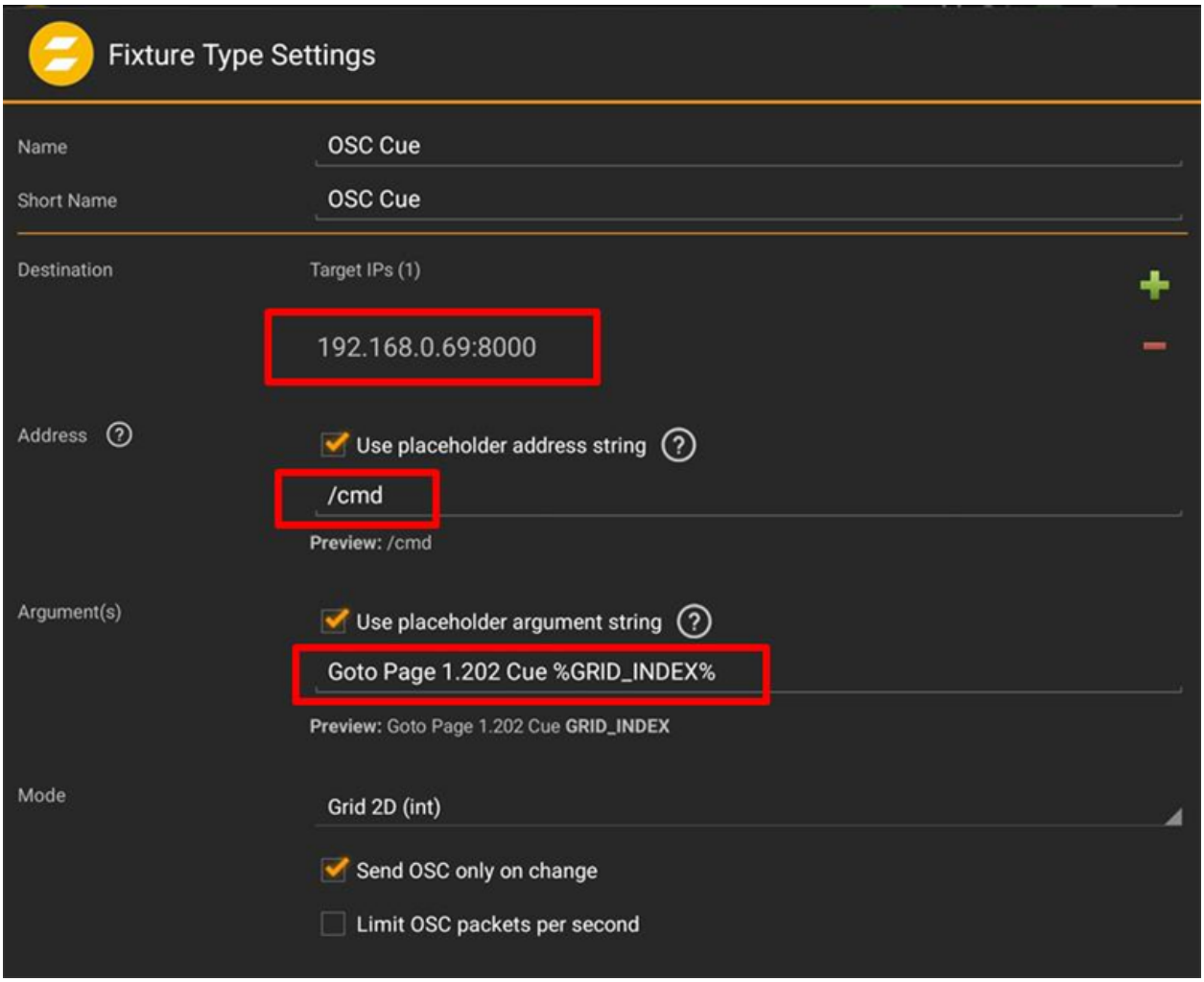
zactrack

zactrack is an automated full-tracking follow system designed for open-air events, theater stages, and studios. Any number of performers can be placed in a 2D or 3D space. zactrack precisely aligns output devices and special effects equipment and sends XYZ coordinates to grandMA3.

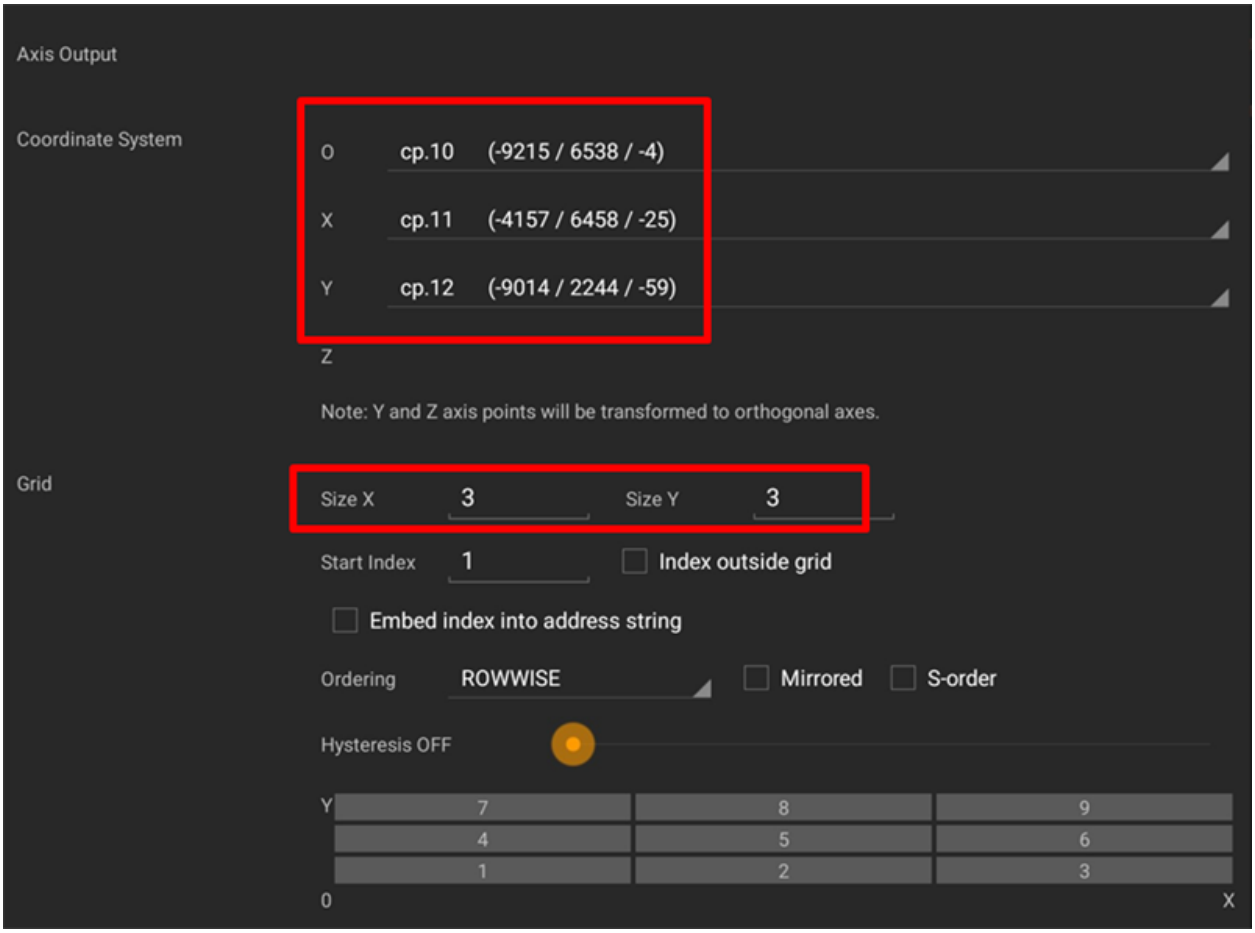
Example 1:

Trigger cues on a grandMA3 console with a performance area defined by a 3x3 2D grid. The position will trigger cues in that specific zone number, when walking with a tracker.

1. Enter the console IP and Port.
2. Enter the exact string address and argument the console expects. In the example below on the console, it is executor 202 on page1.



3. Define the Coordinate System using calibration points (via Pucks / Trackers / Disto).
4. Set the grid size and order.

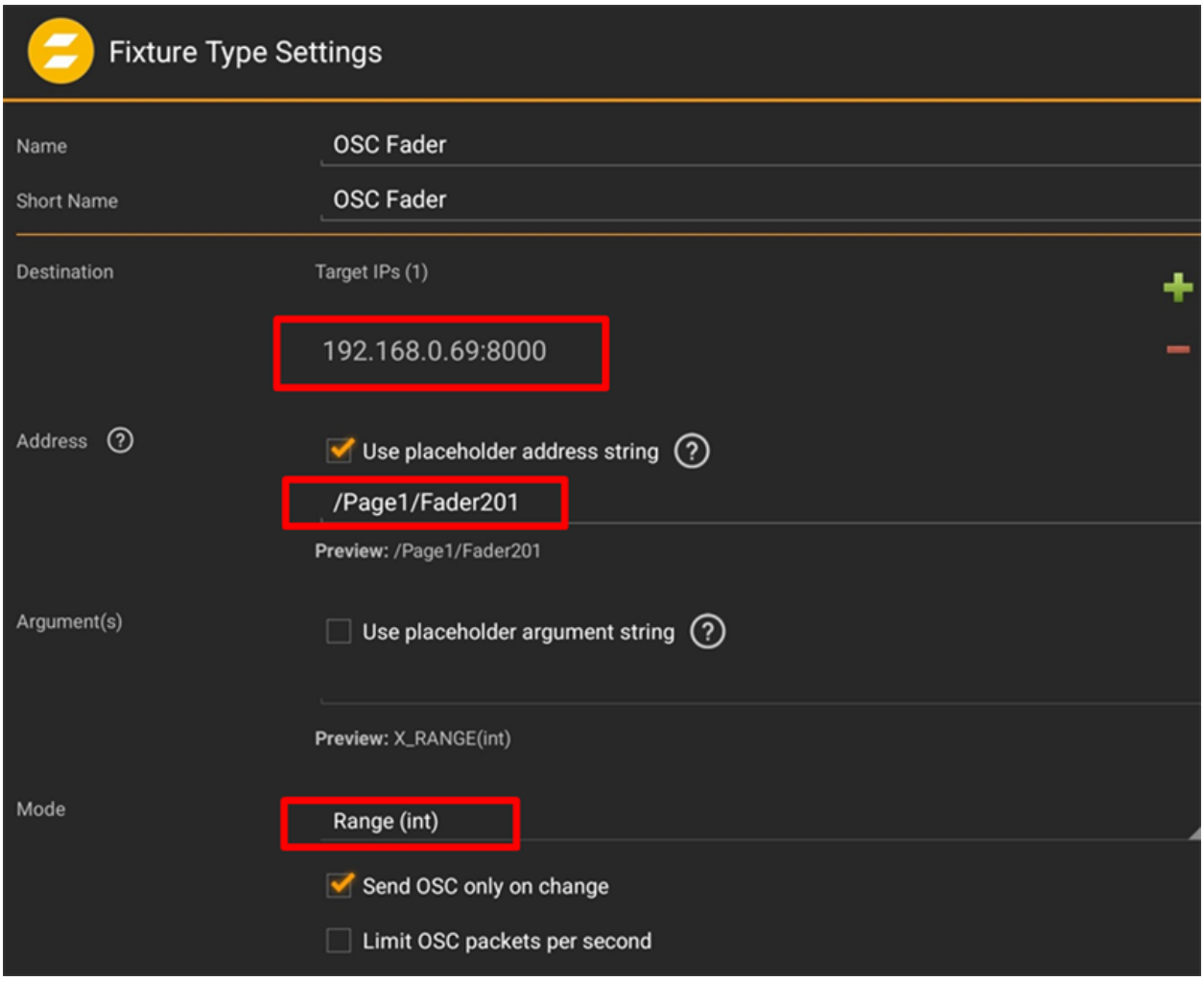


The 2D grid is created.

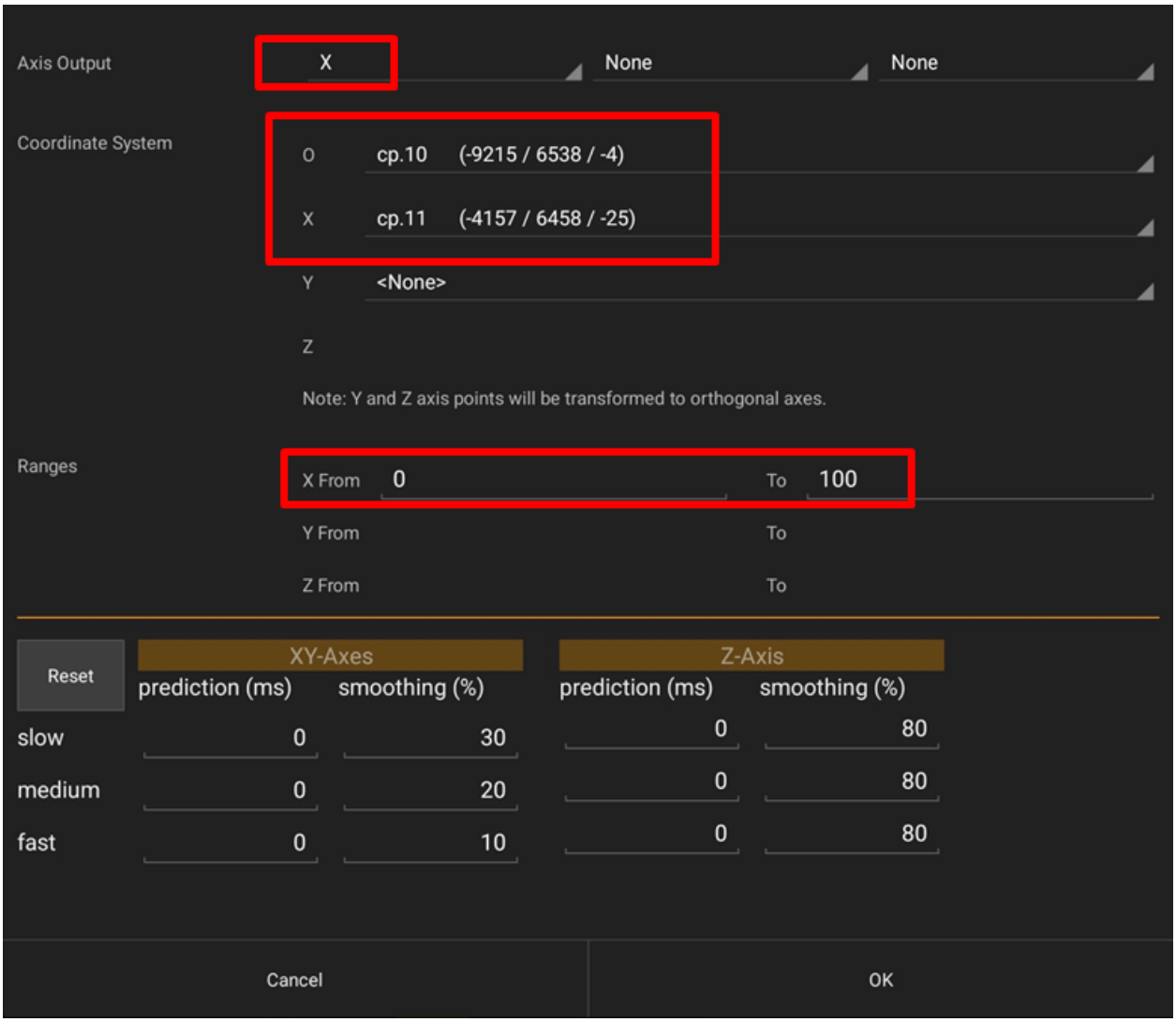
Example 2:

Trigger a fader on a grandMA3 console. When walking in defined axis with a Tracker the position will trigger for example, linear intensity, for that specific fader.

1. Enter the console IP and Port.
2. Enter the exact string address the console expects. In the example, it is the fader of executor 201 on page 1 of the grandMA3 software.
3. Set Range (int) as Mode.



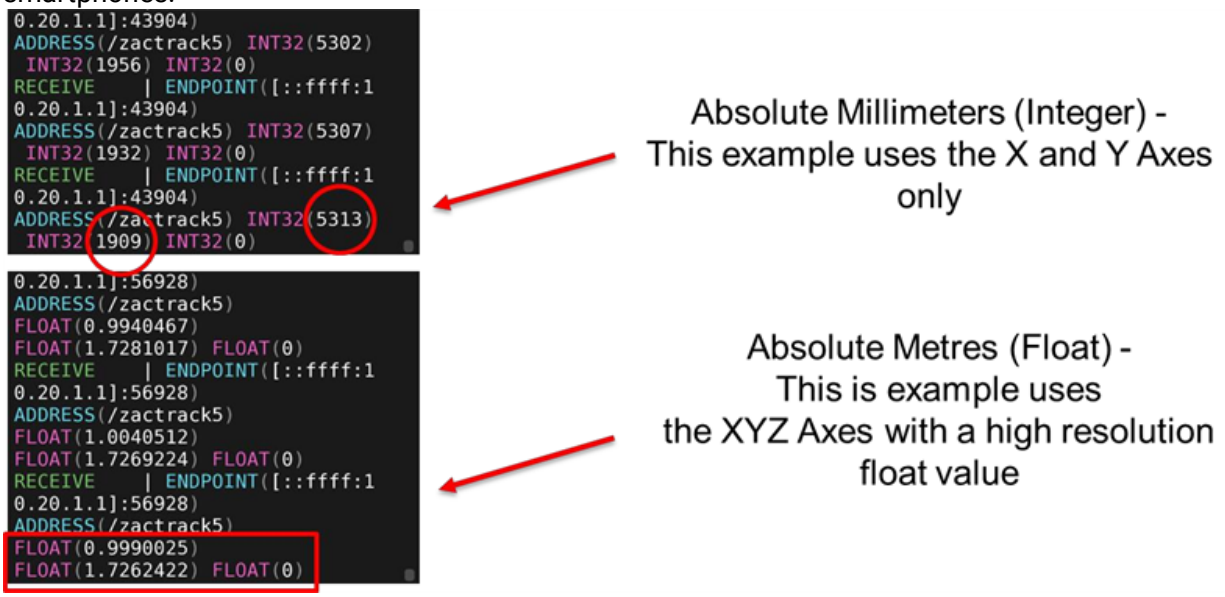
4. Define the Axis Output, X only.
5. Define the Coordinate System.
6. Set the X Range for the fader. These values must match the corresponding setting in the grandMA3 software. The default value is 0 to 100.



1.48.4.2.5. Protocol Viewer

Protocol Viewer

The **Protokol app** by Hexler is a test / monitoring tool to check OSC. It works on any device including smartphones.



1.48.5. PSN (PosiStageNet)

grandMA3 stations can receive PosiStageNet (PSN) data. PSN is a protocol designed to communicate the position of identified points in a 3D space; it is an open protocol that can be used to transmit tracking information between servers and hardware systems.

For a detailed description of PosiStageNet, see the external link to the **PosiStageNet Protocol description v2.0 PDF**.

To open the PSN menu, Menu - In & Out - PSN.

The following is a description of the specific parameters that can be set in the PSN menu:

- Sender IP: Adds the IP of the PSN Source.
- Requested: Set to **Yes** to request the PSN Source.
- Port: Sets the Port for the PSN Source.
- Mulitcast IP: In case of sending the PSN data to a multicast address, the user has to manually add the multicast IP of the PSN system
- MapX, MapY, MapZ: Maps each axis to a different axis. A drop-down opens to choose the target axis.
- InvX, InvY, InvZ: Inverts the incoming data per axis. Toggles the cell between **No** (=empty cell) and **Yes** (= data will be inverted).
- Merge Mode: The merge mode can be selected in a dropdown. For more information, see **DMX Port Configuration**.
- DMX Priority: The priority is used for merging DMX inputs. For more information, see **DMX Port Configuration**.
- Tracker ID: Sets the ID of the Tracker.
- MArker ID: Sets the ID of the Marker Fixture.
- Position XYZ: Shows the position of the tracker.
- Speed XYZ: Shows the speed of the tracker.
- Rot XYZ: Shows the speed of the tracker.

	**Hint:**
	As soon as an axis is mapped or inverted, the individual trackers display their values according to the settings made by the user.
	**Hint:**
	If the sender is transmitting the data via Unicast to the selected Interface in the PSN menu, the PSN system automatically detects the data.
	**Hint:**
	The basic Port configuration is **56565** and the standard Multicast IP is **236.10.10.10**.

Example

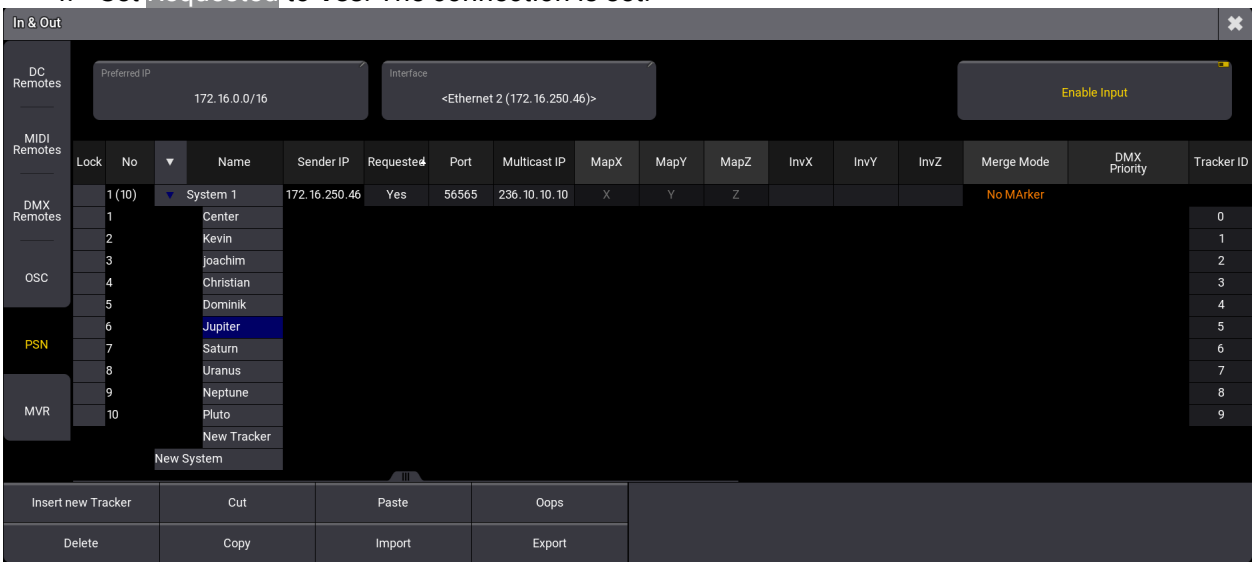
Requirements:

- Your external hardware system, such as a tracking system, is set up correctly.
- The grandMA3 console and the external hardware system share the same network.

- The Simple Show demo showfile is loaded.

To connect a tracking system with a grandMA3 device:

1. Open the PSN menu.
2. Enable Enable Input.
3. Tap Insert new System. A new configuration line is created.
4. Set Requested to **Yes**. The connection is set.



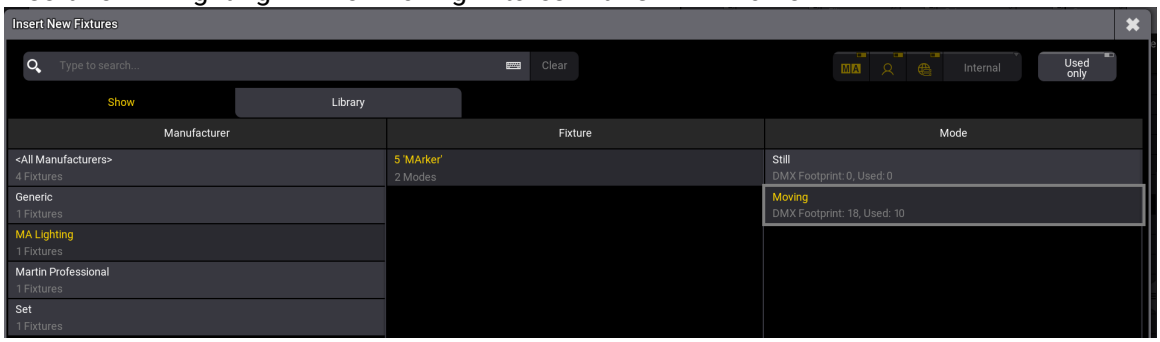
In & Out Menu - PSN

To enable XYZ on the fixtures for the follow spot:

1. Open Menu - Patch - Fixture Types.
2. Tap MAC Encore Performance CLD and then tap Edit. The Fixture Type Editor opens.
3. Set the corresponding XYZ configuration line to **Yes**.
4. Close the editor. XYZ is enabled.

To patch MArkers:

1. Open the Patch menu.
2. Tap Insert new Fixture. The fixture library opens.
3. Insert 10 MA Lighting MArker Moving fixtures with CID 1 Thru 10.



Patch Menu - MArker Fixtures

4. Save and Exit the Patch Menu. The MArkers are patched.

	Hint:
	For more information on MArkers, see the **MArker Fixture topic**.

To allow the fixtures to follow specific markers:

1. Open the PSN menu.
2. Set the configuration lines for the MArker ID. For example:

Tracker ID	MArker ID	Position X	Position Y	Position Z	Speed X	Speed Y	Speed Z	Rot X	Rot Y	Rot Z	Note
0	1	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	
1	2	0.5230646	0.0000000	0.2506059	0.0028477	0.0000000	-0.005943	0.0000000	575.135986	0.0000000	
2	3	0.7935011	0.0000000	-0.732636	-0.003260	0.0000000	-0.003531	0.0000000	575.135986	0.0000000	
3	4	-1.188048	0.0000000	-0.915718	-0.002507	0.0000000	0.0032531	0.0000000	575.135986	0.0000000	
4	5	2.2753186	0.0000000	0.1460321	0.0002125	0.0000000	-0.003311	0.0000000	575.135986	0.0000000	
5	6	5.6758270	0.0000000	5.3210330	0.0012283	0.0000000	-0.001310	0.0000000	575.135986	0.0000000	
6	7	-0.631790	0.0000000	-14.276026	-0.001326	0.0000000	0.0000587	0.0000000	575.135986	0.0000000	
7	8	-3.308186	0.0000000	28.518764	0.0009289	0.0000000	0.0001077	0.0000000	575.135986	0.0000000	
8	9	-5.792982	0.0000000	-44.665904	-0.000741	0.0000000	0.0000962	0.0000000	575.135986	0.0000000	
9	10	3.8345115	0.0000000	59.005538	0.0006512	0.0000000	-0.000042	0.0000000	575.135986	0.0000000	

PSN menu - MArker ID

3. Press Fixture 1 Thru 10 Please. The fixtures are selected.
4. Tap Position in the Encoder Bar.
5. Tap and hold the Encoder Page.
6. Select XYZ 3 of 3.
7. Rotate the first Encoder 1 click. The fixtures are following the first Marker 1 1'MArker 1'.

Encoder Bar - Encoder Page

1.48.6. MVR-xchange

The MVR menu offers a platform to exchange and manage MVR files via a network connection instead of using a USB drive. For more information about MVR files in general, see **My Virtual Rig (MVR)**.

The MVR-xchange is specified in the DIN SPEC 15801:2023-12. For more information, see the external link **https://www.beuth.de/en/technical-rule/din-spec-15801/373968511**.

The idea behind the MVR menu is to send MVR files from a station into a defined network and share the files with other stations. For more information about the appropriate keyword, see **SendMVR Keyword**.

To open the MVR menu:

1. Press Menu. The menu opens.
2. Tap In & Out.
3. Tap MVR. The MVR menu opens.

The three elements on top of the menu are described as follows:

- Group: The standard group name is **Default**. The group name can be edited. Stations with identical group names share the same exchange group.
- Interface: Defines the network interface.

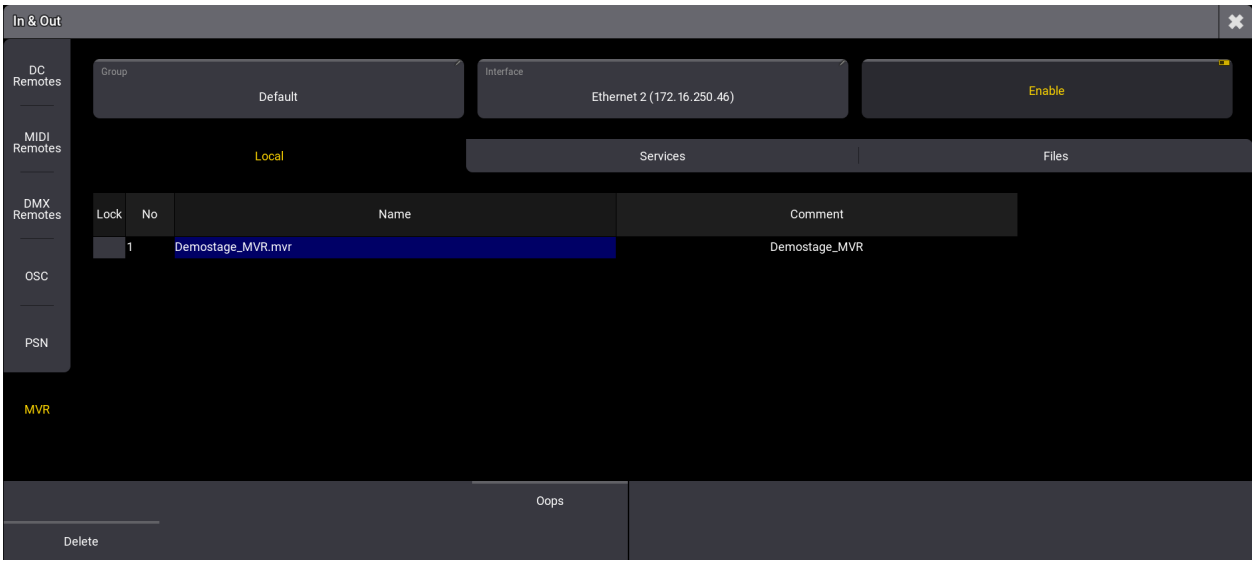
	**Hint:**
	To share data between different devices, the devices need to be in the same network.
	**Hint:**
	To share data between different devices, a network connection has to be established. For more information, see **Enabling or Disabling the Network Connection**.

- Enable: To be able to exchange MVR files, make sure to activate Enable. For more information about station and session communication, see **Station Control**.

The main area of the MVR is separated into three tabs: Local, Services, and Files. Selecting a tab changes the grid below it accordingly and the font color of the selected tab changes to yellow.

Local

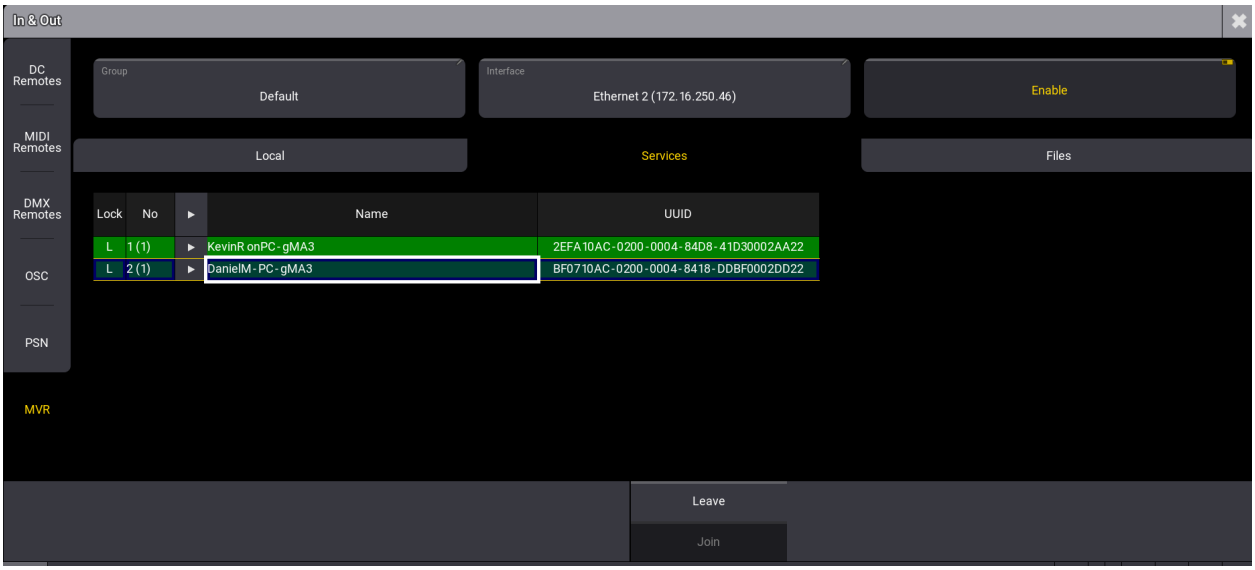
- Local: Shows all MVR files that have been locally committed into an exchange group. Files can be deleted or Oopsed in the menu, too.



MVR - Local

Services

- Services: Lists all stations in the exchange group. If a station joins the exchange group, it gets listed in Services.

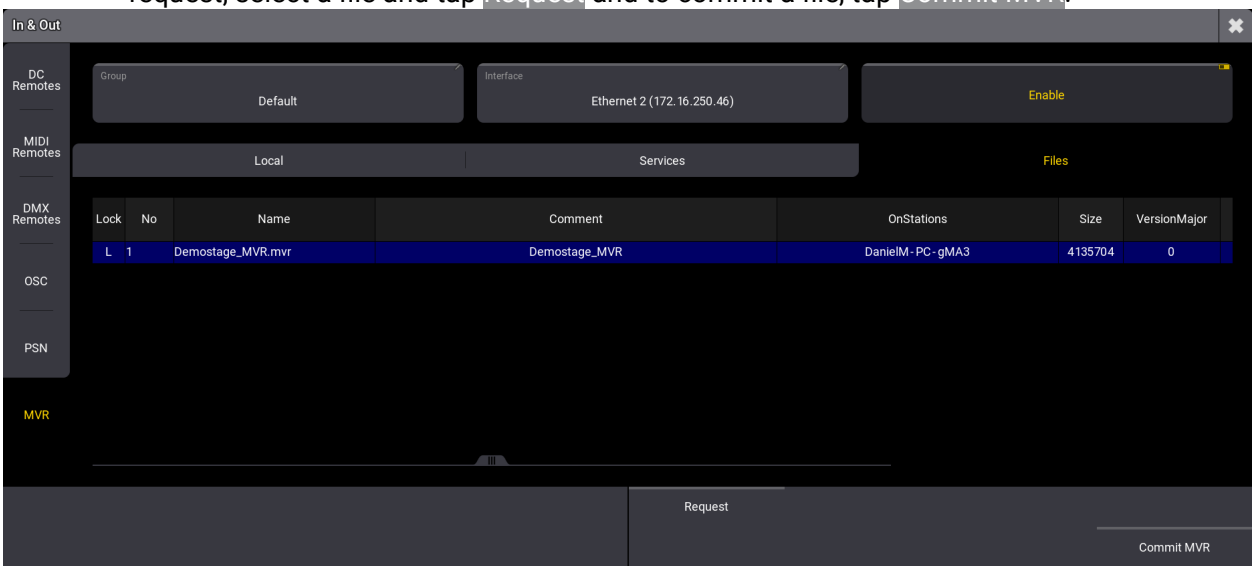


MVR - Services

- Green Background: Shows an existing connection. To reestablish a connection, select the station and tap Join.
- Light Green: Displays your connected station.
- Red Background: Shows an interrupted connection. To interrupt a connection to a station, select the station and tap Leave.
- UUID: Universally Unique Identifier.

Files

- Files: Allows to request MVR files that are sent by other stations or commit MVR files. To request, select a file and tap Request and to commit a file, tap Commit MVR.

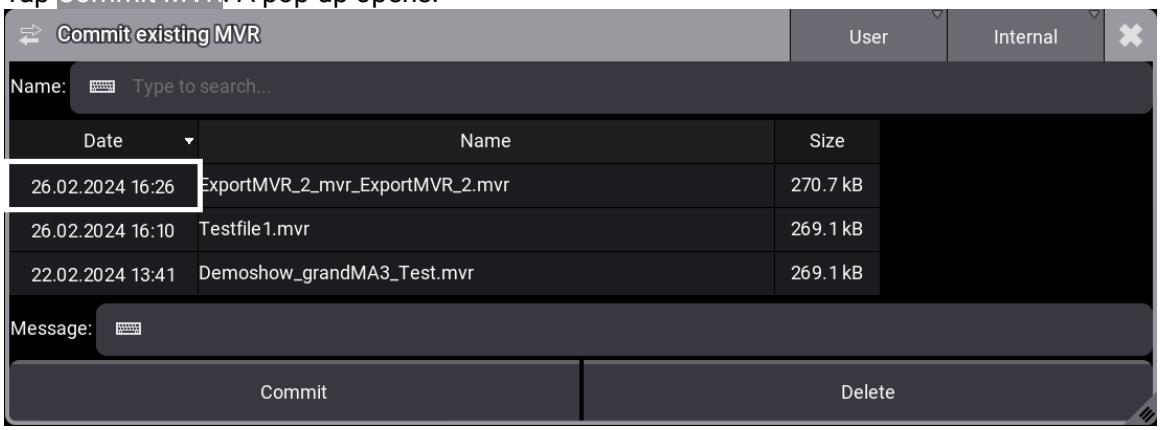


MVR - Files

Committing an MVR File

To commit an MVR file:

1. Tap Files.
2. Tap Commit MVR. A pop-up opens.



3. Select an MVR and tap Commit.

	Hint:
	A comment can be set in the Message area. This comment is visible for other stations.

Requesting an MVR File

To request a file from another station:

1. Tap Files.
2. Tap a file in the list. The file gets selected.

3. Tap Request. A status bar shows the status of the request.
4. A pop-up opens with a confirmation.
5. Tap Ok. The file has been requested.

For more information on where the file is stored, see **Folder Structure**.

1.49. Sound

The grandMA3 system can receive and transmit sound. See how to connect audio devices in the **Connect Audio In** and **Connect Sound Out** topics.

The **Sound Viewer** can be used to show the received sound.

Sound Viewer is a window that can be created using the **Add Window pop-up**.

Received sounds can be used in phasers to dynamically change attribute values or in sequences where incoming sounds can be used to trigger cues. Learn more about received sound in the Sound Viewer topic (link below).

Sound files can be imported into the **Sounds** Pool. Imported sounds can be played back from the pool, and they can be added to timecode tracks. Learn more about importing sound in the Sounds Pool topic (link below).

Subtopics

- **Sound Viewer**
- **Sounds Pool**

1.49.1. Sound Viewer

The grandMA3 system can receive a sound signal. This signal can be used in Phasers, and different **Sound Channels** can be used to give value to attributes or trigger cues.

The **Sound Viewer** displays an incoming audio signal as a raw waveform. This window also breaks down the strength of that signal into several bands of a few different widths. The **Sound Viewer** also displays calculations performed by the console to find steady beats within the signal.

Requirement:
Active sound input is required in order for the console to display useful information in the **Sound Viewer**. For information about connecting an audio input to a console, see the **Connect Audio In** topic. The grandMA3 onPC software uses any sound input hardware designated by the operating system.

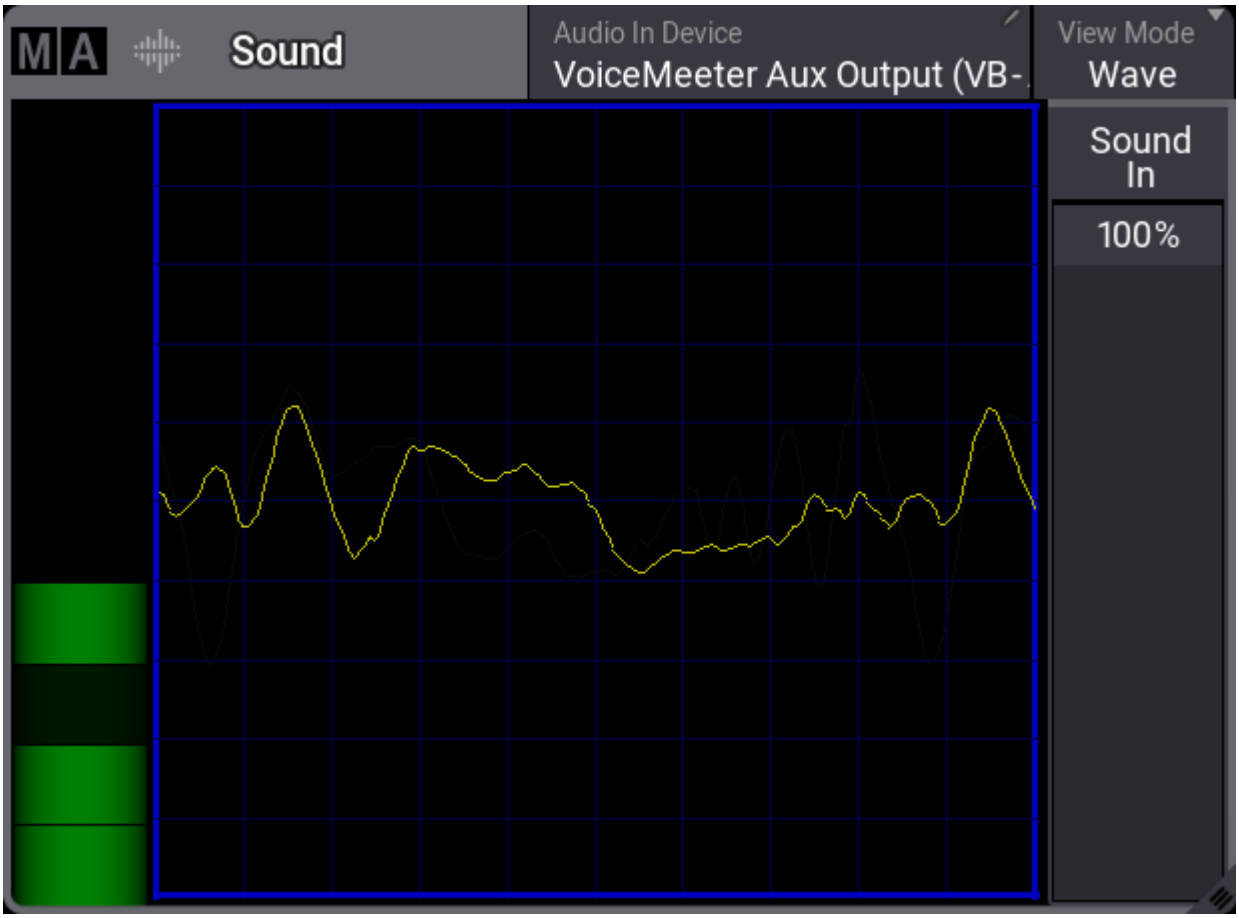
The **Add Window** pop-up includes the **Sound Viewer** under the Tools tab. See the **Add Windows topic** for more information about the Add Window pop-up.

The right corner of the **Sound Viewer** title bar includes a View Mode button. Tap this button to cycle through available display mode options, or tap and swipe to open a pop-up menu containing all of the available display mode options for the **Sound Viewer**.

On the grandMA3 onPC, there is another button in the title bar called Audio In Device. This is a list of available sound devices on the computer that can be used as sound input devices. The used input can be selected here or in the **onPC settings**.

These settings define different settings specifically for the onPC. Learn more in the **onPC Settings topic**.

Wave View Mode



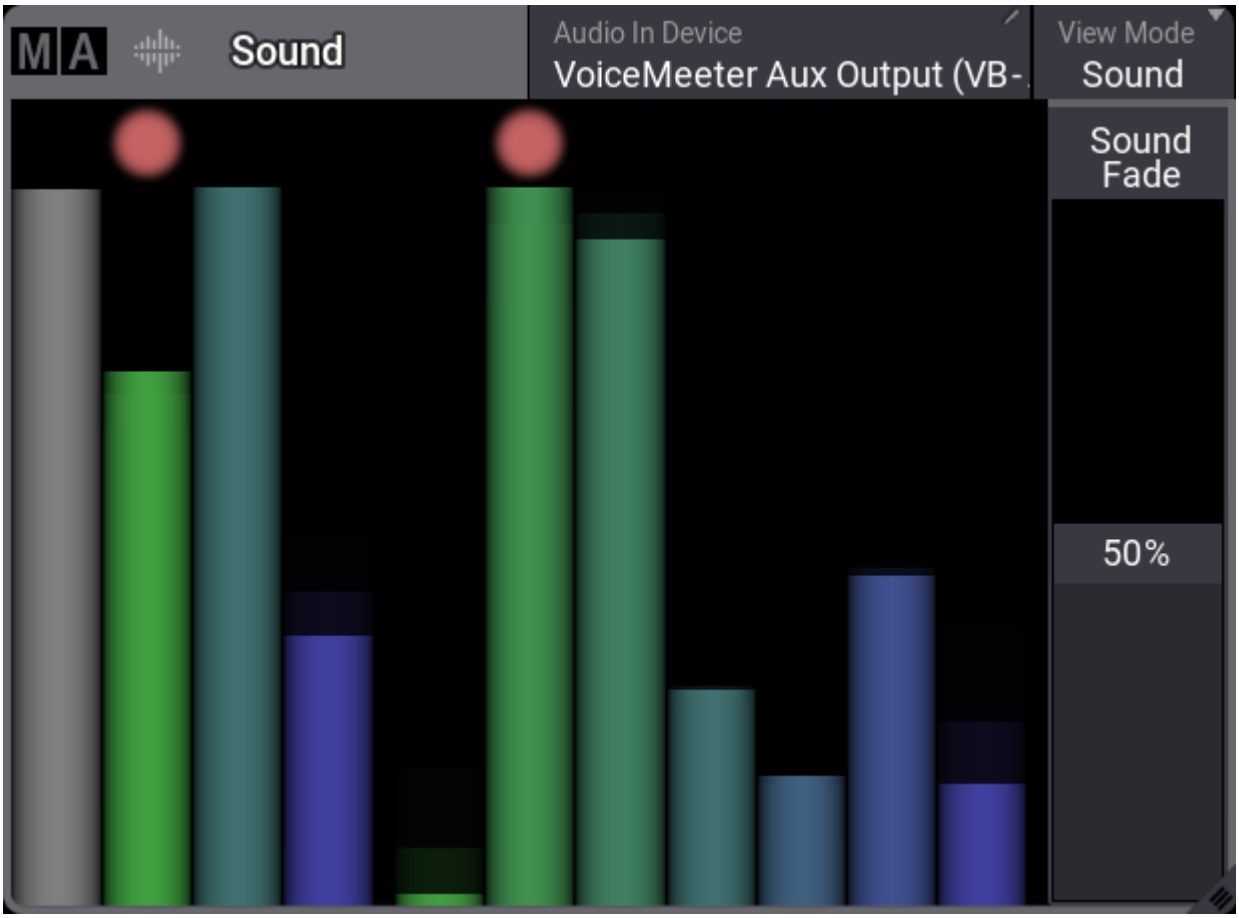
Sound window in Wave view mode

The left side of the window includes a simple VU meter, which is shown as a vertical bar.

The main area of the window shows the raw waveform of the incoming audio signal.

The right side of the window includes quick access to the **Sound In** master. Adjusting the **Sound In** master changes the volume of the input signal. This change is immediately visible in both the VU meter and the waveform. For more information about the **Sound In** master, see the **Grand Masters** topic.

Sound View Mode



Sound window in Sound view mode

The **Sound** view mode of the **Sound Viewer** displays the incoming audio signal as a series of bars representing the volume of different bands of frequencies. When a given bar reaches a peak, a dot appears above the bar. This dot represents a possible sound trigger based on that frequency band.

The eleven bars shown in the window break down the frequencies of the audio signal in three different ways, displaying all three breakdowns simultaneously. The first bar represents the volume of the whole signal. The next three bars divide the signal into **Bass**, **Mid**, and **High** frequencies. And the remaining seven bars divide the frequencies equally into narrower bands.

These eleven bars relate directly to the first eleven **Sound Channels** and inversely to the remaining **Sound Channels**. Available **Sound Channels** include:

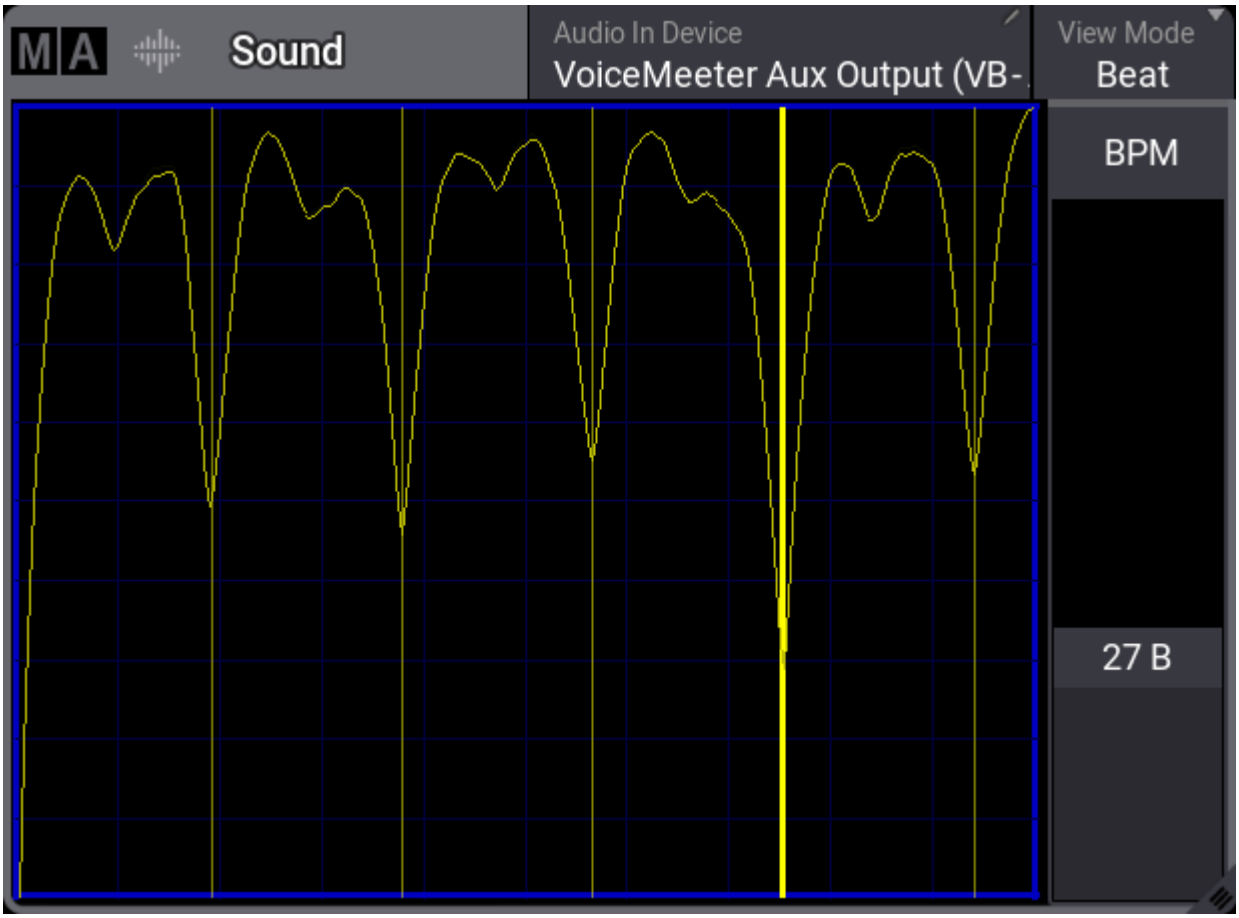
1. **All**
2. **Bass**
3. **Mid**
4. **High**
5. **Band1**
6. **Band2**
7. **Band3**
8. **Band4**
9. **Band5**
10. **Band6**

11. **Band7**
12. **InvAll**
13. **InvBass**
14. **InvMid**
15. **InvHigh**
16. **InvBand1**
17. **InvBand2**
18. **InvBand3**
19. **InvBand4**
20. **InvBand5**
21. **InvBand6**
22. **InvBand7**

Set the value of any parameter to follow one of the **Sound Channels** in the command line using the **SoundChannel** keyword or the Sound Codes tab of the **Calculator.**

The right side of the window includes quick access to the **Sound Fade** master. Adjusting the **Sound Fade** master changes the fall-off speed of all of the channels. While a lower number results in more accurate tracking of quickly changing dynamics, the response may appear undesirably erratic. A higher value results in a smoother response. For more information about the **Sound Fade** master, see the **Grand Masters** topic.

Beat View Mode



Sound window in Beat view mode

The **Beat** view mode of the **Sound Viewer** displays the incoming audio signal after some additional processing. This processing looks for repeating beats. The processed signal appears as a series of hills and valleys. Deeper valleys form where louder pulses in the incoming signal line up more consistently over time.

Dim, yellow, vertical lines appear in the valleys with the most consistent repeated pulses. These lines represent beat candidates, which the **BPM** speed master can follow.

A bold, yellow, vertical line appears when the **BPM** master locks onto a beat. The **BPM** master adjusts automatically to follow any changes detected in the incoming signal.

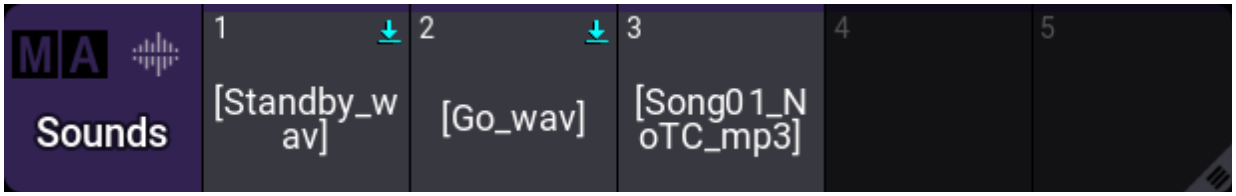
The right side of the window includes quick access to the **BPM** speed master fader. For more information about the **BPM** speed master, see the **Speed Masters** topic.

1.49.2. Sounds Pool

The **Sounds** pool can contain sound files. The pool is part of the media pools.

	Important:
	The Sounds pool has a limit of 100 MB for all objects. Files exceeding the limit will be highlighted with a red color in the import pop-up. The overall size of the media pools is a maximum of 200 MB.

Sound pool objects can be played back in the pool or added as a track in a timecode object.



Sounds Pool with imported sound files

Import a Sound

Sounds can be imported using the **Show Creators import function**.

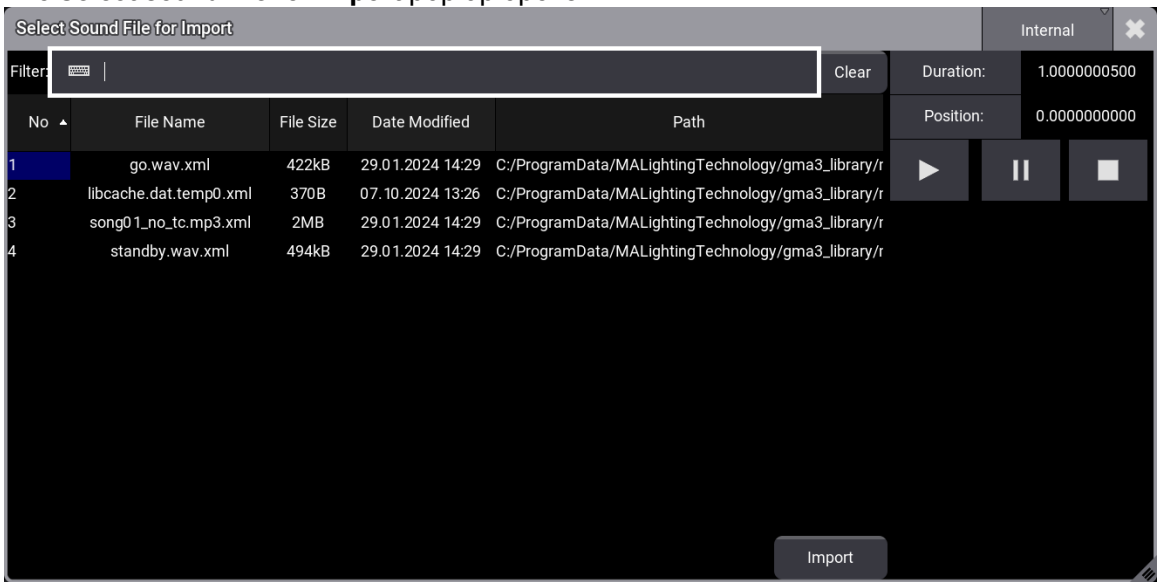
The **Show Creator** has an import function that can be used to import objects into pools. Learn more in the **Show Creator Import/Export topic**.

They can also be imported using the Sounds pool.

1. Edit a **Sounds** pool object.
   The **Edit Sound** pop-up opens.

Edit Sound pop-up

2. Tap Import.
   The **Select sound file for import** pop-up opens.



Select sound file for import

3. Select the desired sound file and tap Import.
4. Edit the values in the Edit Sound pop-up to the desired settings and close it by tapping the ✖.

Playing Sounds from the Pool

Sound files in the sounds pool can be played using a Go+ command on the pool object. Other relevant commands are:

- Off

- Toggle
- Pause
- On

Sounds triggered by a sequence, macro, or executor are played by the **Master** in a session. The only exception to this is playing it in the import pop-up.

The master is a session outputs the sound. Masters are the stations with one of the following stastus': GlobalMaster, IdleMaster, and Standalone.
Learn more about session masters in the **Session topic**.

Some requirements must be met before the sound can come out of the system.

- The master needs a physical audio output connected. Learn how in the **Connect Sound Out topic**.
- The **Sound Out** master fader must be turned up. This can be done in multiple locations, such as the **Master Controls**.
- There is a master for the pool object. It can be a little hard to find. The easy way is to **assign the sound to an executor** with a master fader or knob. Here, the master level can be seen and turned up.

  Learn more about assigning sounds to executors in the **Assign Object to an Executor topic**.

Using Sounds in Timecode Show

Sounds in the pool can be used as a target in a timecode track. Learn more about timecode tracks in the **Track Groups topic**.

Running the timecode show plays the sound from the timecode object.

Sounds Pool Object Settings

There are some settings for a sound object. The Settings are in the **Edit Sound pop-up** (see above). Besides the standard settings like Appearance, Auto-Start, Auto-Stop, etc., there are the following uncommon settings:

- **dB**:
  The sound file level can be adjusted using the dB setting. The valid range is from -6 db to +6 db.
- **Duration** (information only):
  This shows the length of time for the sound file.
- **FilePath**:
  This is a path (sub-folders) for the sound file inside the folder (gma3_library/media/sounds).
- **Installed**:
  If this is set to Yes, the sound file will be updated from the file archive they were imported from. This is useful when sound files are changed and copied into the folder using an external editor.

1.50. RDM (Remote Device Management)

Remote Device Management is a protocol that allows bi-directional communication between a grandMA3 device and RDM-ready devices attached to it (= RDM-ready fixtures) over a standard DMX line. RDM protocol allows grandMA3 devices to send commands and receive messages from specific moving lights for device configuration and status monitoring. For example adjusting the DMX starting address. This is especially useful for devices installed in a remote area. The **parameters** of each fixture determine which commands can be send and which messages can be received between the fixture and the grandMA3 device.

ANSI E1.20 - 2010 by PLASA specifies the RDM standard as an extension of the DMX 512-A protocol (ANSI E1.11).

RDM is integrated in DMX without influencing the connections. The RDM data is transmitted via the standard XLR-poles – new DMX cables are not required. RDM-ready and conventional DMX devices can be operated in one DMX line. The RDM protocol sends its own data packages in the DMX512 data feed and does not influence conventional devices.

Enable RDM

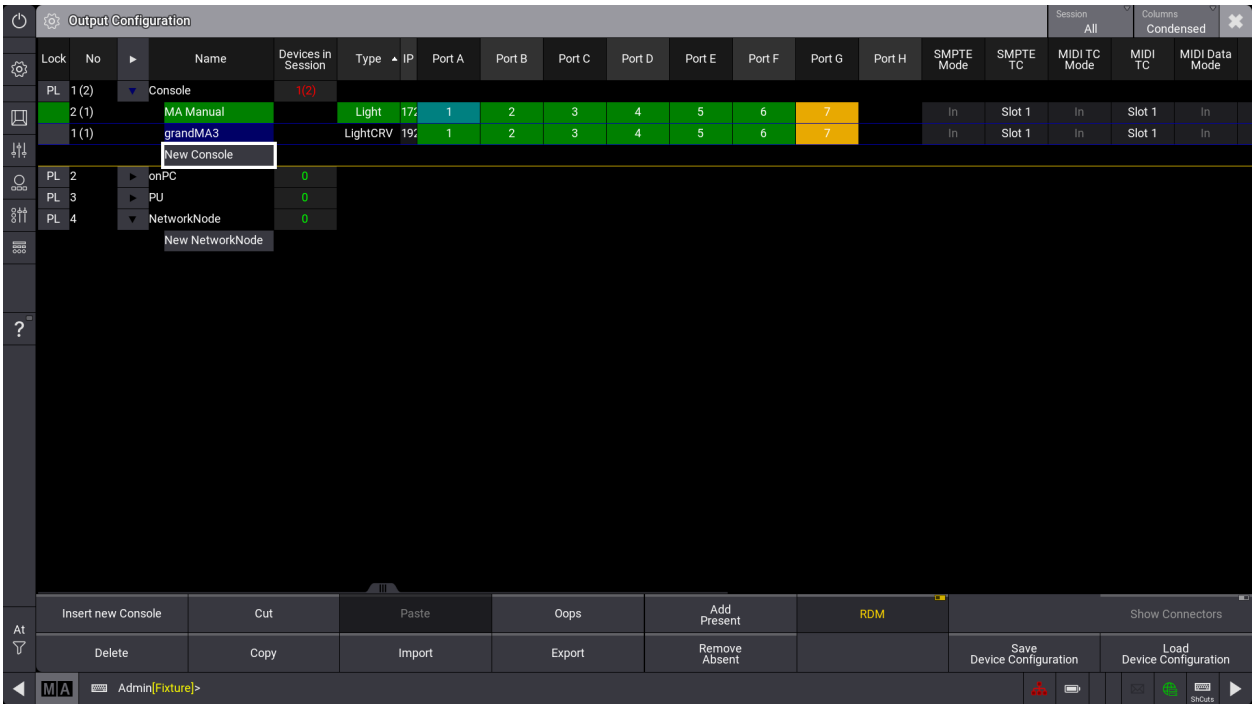
To use RDM, it has to be enabled in two different spots:

1. Globally within the show file:
   - Press Menu - Network: Enable the RDM button in the **Stations** tab.
   - Or Press Menu - Network: Enable the RDM button in the **Station Control** tab.
   - Or press Menu - Live Patch: Enable the RDM button in the **RDM** tab.
   - Or press Menu - Connector Configuration: Enable the RDM button on the bottom right.

All four RDM buttons are linked to each other and RDM is enabled globally.

2. Per XLR port that shall use RDM. Therefore, the mode of an XLR Port in the Connector Configuration needs to be set from Out to RDM. Tap and hold in the cell of the desired Port. An editor opens:

Tap Mode A and change it to RDM. Confirm the changes by tapping Apply. The background color of the port turns to turquoise:



Connector Configuration options with Port A in RDM mode and RDM activated globally in the bottom right.

In RDM mode, DMX data is only sent when there are changes of DMX values. In addition, every 500ms a refreshing packet will be sent so that DMX fixtures will not switch into DMX fail mode.

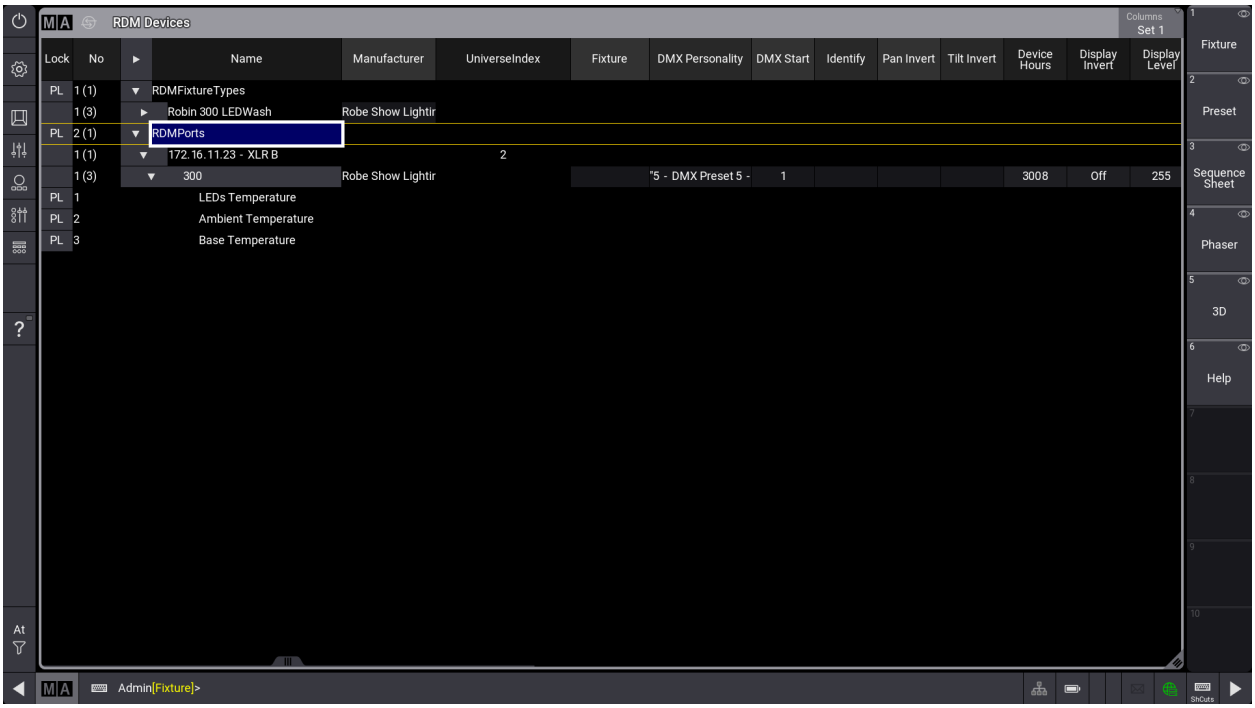
This RDM output mode allows more time on the DMX line for RDM configuration.

RDM Devices Window

Open the RDM devices window via the **Add Window** dialog in the Tools tab.
The RDM Devices window lists all devices that are discovered through RDM. The same list is also displayed in Menu - Live Patch - RDM.

When an RDM device is detected on an XLR port, a section called RDMPorts will be included in the list of RDM devices. Within each RDMPort, all fixtures that are detected via RDM on this physical XLR port are listed. An RDM port is labeled with the IP of the device and the XLR port of the device, e.g., 192.168.0.4 - XLR D. If the device is not detected anymore, the font color turns red.

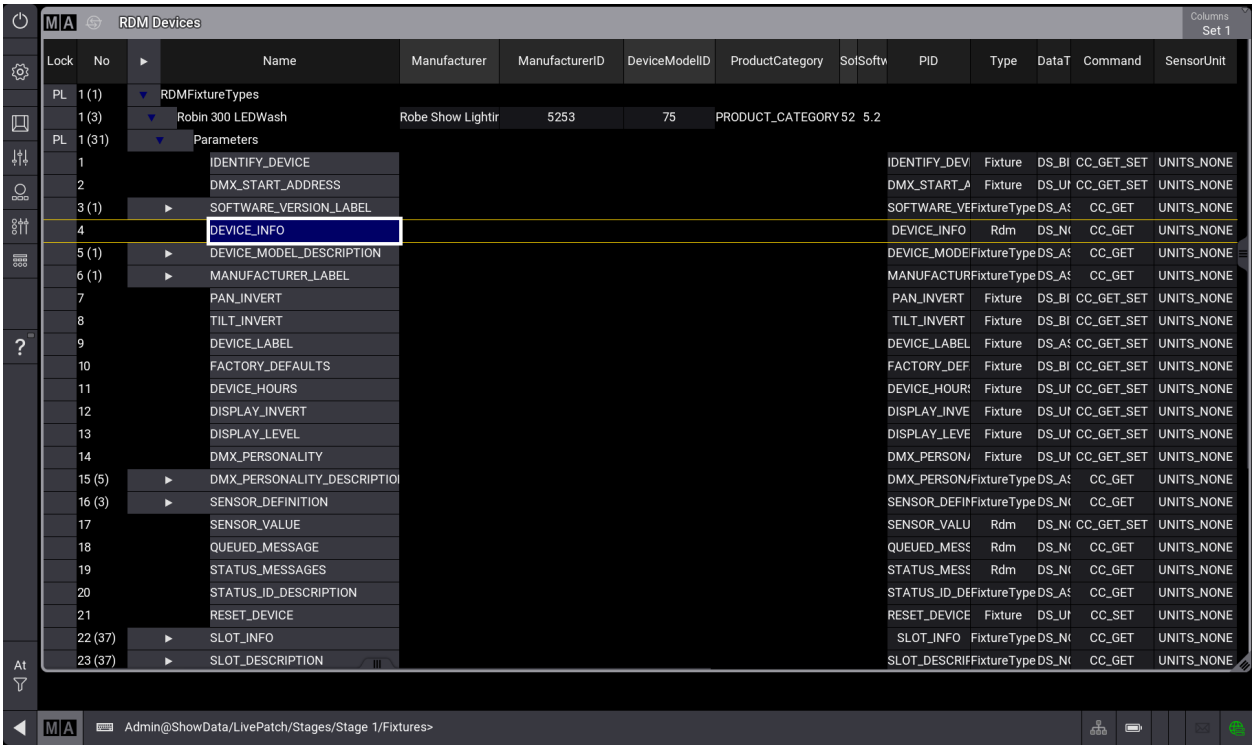
The RDM devices window with **RDMFixtureTypes** and **RDMPorts** detected could look like this:



RDM devices window with RDMPorts in expanded mode.

For every fixture type identified by RDM, a new child will be added to the **RDMFixtureTypes** in the RDM Devices window. Each RDMFixtureType contains general information of the RDM fixtures, that is similar to all fixtures of the same product, e.g., the parameter description, or the available DMX personalities.

Expanding with ▶ next to the RDM fixture shows all the different parameters of the fixture:



RDM devices window with one active RDM fixture in a expanded view.

RDM Parameters

The grandMA3 creates RDMFixtureTypes by itself depending on these three parameters: ManufacturerID, DeviceModelID, and SoftwareID.
As soon as the same physical type of lighting fixtures have different software IDs due to different firmware versions, different RDMFixtureTypes are created.

Only the cells of properties that an RDM fixture provides as set-able can be edited in the RDM devices window. Not all RDM parameters can be set by the user.

There are three different kinds of commands for each Parameter:

- Set: Parameters that are adjustable on the console.
- Get: Parameters that are only receivable.
- Set and Get: Parameters that can be sent and also received by a device.

	**Hint:**
	The RDM devices window also displays all Parameters individual commands under a specific column called **Command**.

The following RDM parameters are supported by grandMA3:

Parameter	Parameter ID	GET	SET	SET/ GET
PROXIED_DEVICES	0x0010	X		
PROXIED_DEVICE_COUNT	0x0011	X		
LANGUAGE_CAPABILITIES	0x00A0	X		
SOFTWARE_VERSION_LABEL	0x00C0	X		
BOOT_SOFTWARE_VERSION_ID	0x00C1	X		
BOOT_SOFTWARE_VERSION_LABEL	0x00C2	X		
SLOT_INFO	0x0120	X		
SLOT_DESCRIPTION	0x0121	X		
DEFAULT_SLOT_VALUE	0x0122	X		
SENSOR_DEFINITION	0x0200	X		
SENSOR_VALUE	0x0201			X
RECORD_SENSORS	0x0202		X	
DEVICE_HOURS	0x0400			X
LAMP_HOURS	0x0401	X		
LAMP_STATE	0x0403	X		
LAMP_STRIKES	0x0402	X		
LAMP_ON_MODE	0x0404			X
DEVICE_POWER_CYCLES	0x0405	X		
DEVICE_MODEL_DESCRIPTION	0x0080	X		
DEVICE_LABEL	0x0082			X
POWER_STATE	0x1010			X
IDENTIFY_DEVICE	0x1000			X
DMX_PERSONALITY	0x00E0			X
DMX_START_ADDRESS	0x00F0			X
PAN_INVERT	0x0600			X
TILT_INVERT	0x0601			X
PAN_TILT_SWAP	0x0602			X
DISPLAY_INVERT	0x0500			X

Parameter	Parameter ID	GET	SET	SET/ GET
DISPLAY_LEVEL	0x0501			X
RESET_DEVICE	0x1001		X	
FACTORY_DEFAULTS	0x0090			X

Proxy Devices

A proxy device is a device that receives DMX from the console and sends it to several devices that are connected to the proxy device. For example, a Radio DMX Link containing of Transmitter and Receiver devices.

The column of Proxied Devices displays the number of RDM devices handled by the specific proxy module. This number also includes the proxy device itself.

Important:

Connect RDM devices directly to a corresponding radio DMX receiver device. Fixtures with built-in radio DMX modules that use a wireless connection to a radio DMX transmitter are not detected by the controlling device.

Sensors

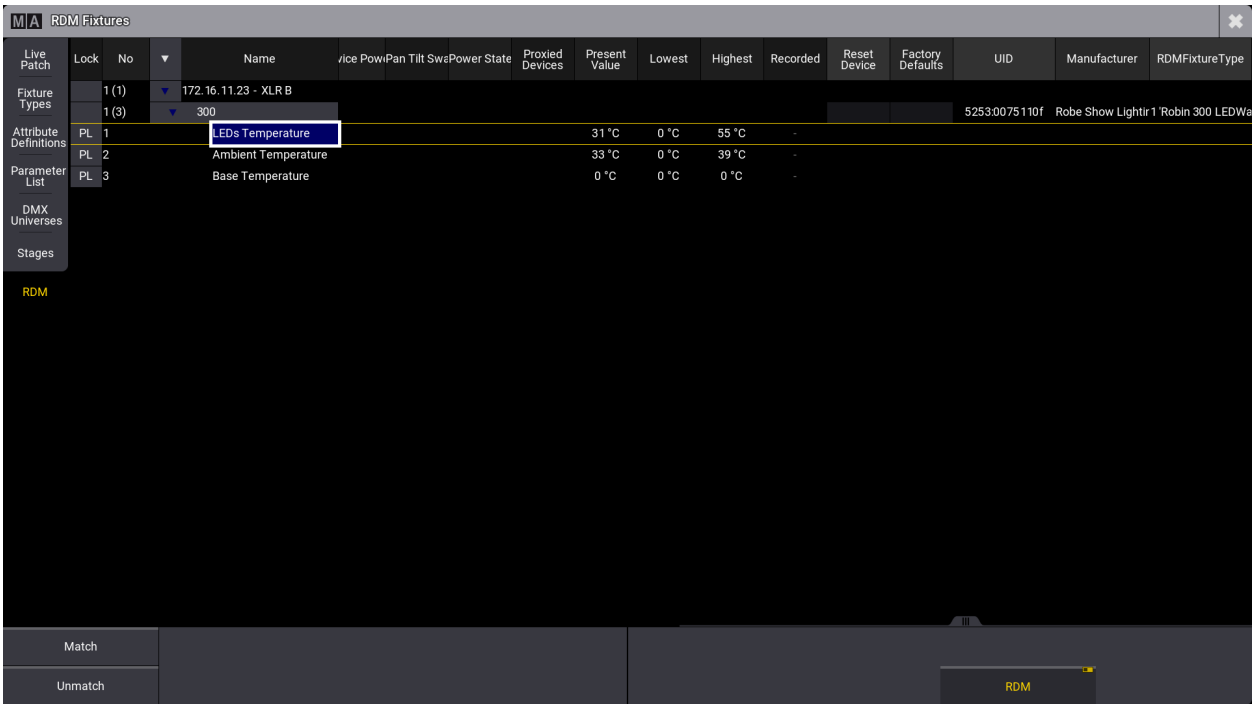
Sensors and their values are also displayed in the RDM devices window. Tap ▶ next to the desired RDM fixture. Each sensor is shown in a separate row. Four different columns are displayed on the far right side for each sensor.

To see the actual values of the sensor, go to Menu - Live Patch - RDM. Tap ▶ next to the fixture to see the four sensor columns:

- Present Value: The current value of the sensor.
- Lowest: The lowest value the sensor can reach.
- Highest: The sensor's highest value.
- Recorded: The stored value of the fixture, when RECORD_SENSORS, 0x0202 was executed.

Hint:

RECORD_SENSORS needs to be executed by a different RDM controller in order to display a value.



RDM tab in Live Patch displaying values for three different sensors.

RDM Communication Process

In grandMA3 the RDM communication follows this process:

1. Discovery for new RDM fixtures.
    a. Check if detected fixtures are still available.
    b. Check for new RDM fixtures.
2. Get parameter and sensor data.
3. 1s Pause
4. Start again at 1.

Parameters that are not changing during the runtime of a fixture, e.g., Device Info, are only pulled via RDM once when creating the corresponding RDMFixtureType. All other parameters and sensors are pulled every time in step 2.
As soon as an RDM fixture is not available for 3 discoveries in a row, it will be displayed in red in the list of RDM fixtures.

It is possible to match an RDM fixture with a fixture of the grandMA3 patch. To do so, edit the fixture cell of the desired RDM fixture in the RDM devices window. A pop-up opens and offers all fixtures of the current show file.
In addition, it is also possible to match fixtures within the RDM window in the live patch. In the live patch, it is possible to open the fixture list in the same way as described above, and by selecting any cell of a fixture then tapping Match at the bottom of the window.
To remove a match between an RDM fixture and a grandMA3 fixture, it is possible to tap Unmatch in the RDM window in the live patch or to tap Clear in the match pop-up.

	Hint:

	It is recommended that RDM be turned off for universes that are not used, or for universes including fixtures without RDM functionality. To do so, follow **step 1 and 2** vice versa.

Example

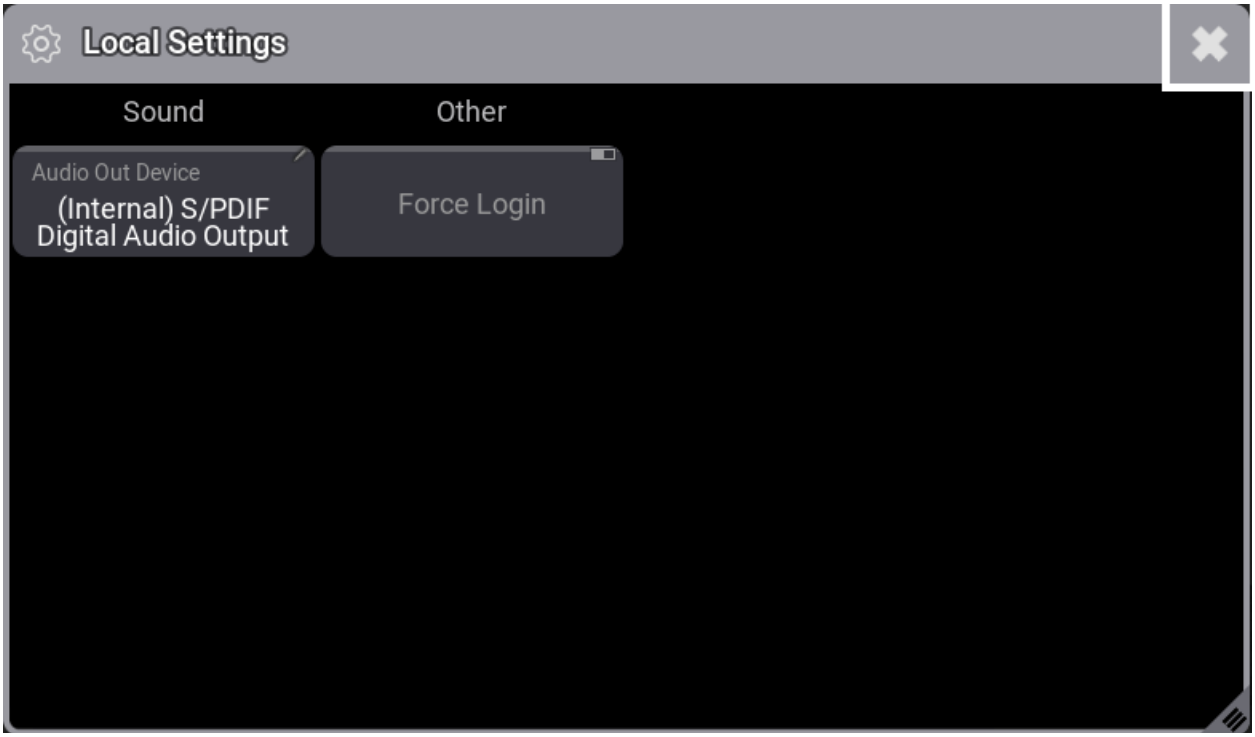
Identifying (IDENTIFY_DEVICE) a fixture with RDM:

1.51. Local Settings

	Hint:
	The local settings menu is not available for the grandMA3 onPC software. Instead, there are **onPC Local Settings**.

To open the local settings menu on a grandMA3 console:

- Press Menu - Settings - Local Settings. The local settings menu opens.



Local Settings

Sound

- **Audio Out Device:** Use the **1 finger swipe gesture** to change the sound interface that should output sound.

Other

- **Force Login:** If enabled, a login pop-up is displayed when the console is started. For more information on how to generate passwords for users, see **Create User**.

1.52. Update the Software

Every grandMA3 device is delivered with the latest version of the grandMA3 software.

All devices in a network with software versions higher than 1.0 can see and update each other.

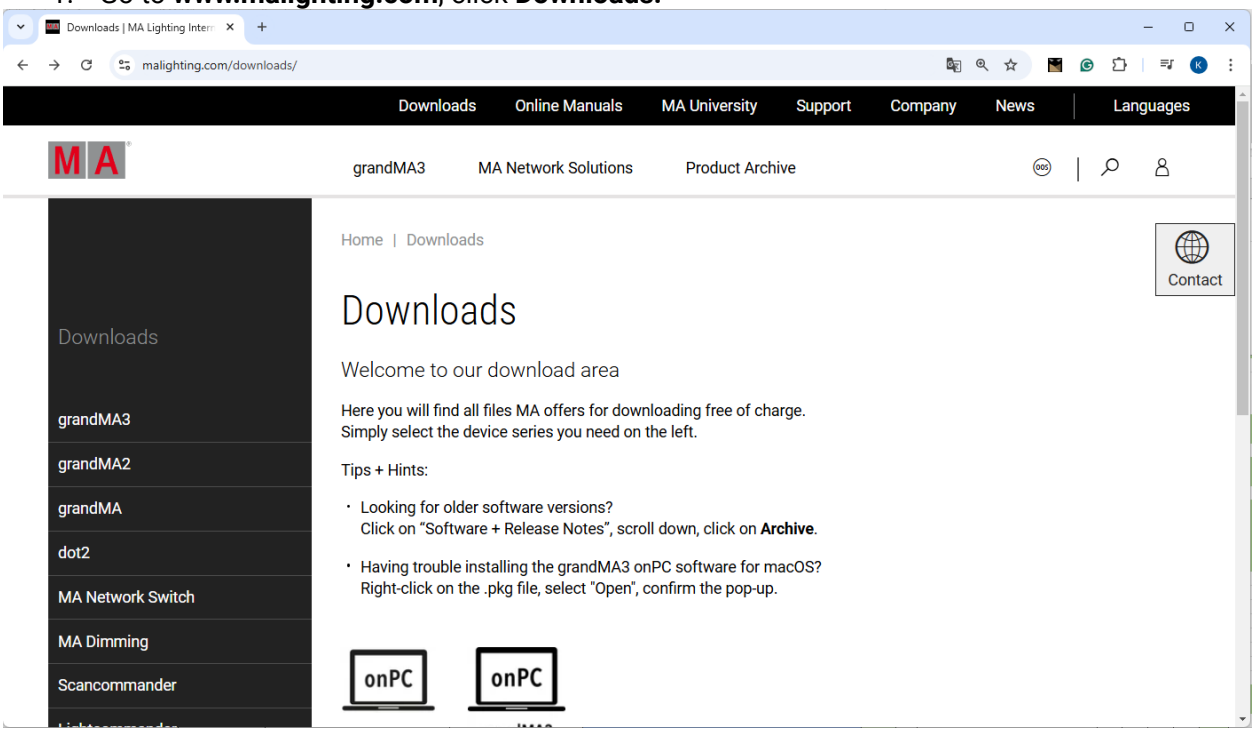
This is to give you information on how to update or downdate the grandMA3 software if needed. For more information, see **Network Update**.

To check which version your device is currently running, use the **Version keyword** in the command line (does not work with PUs and Nodes).

Check for Updates and Download the Latest Version

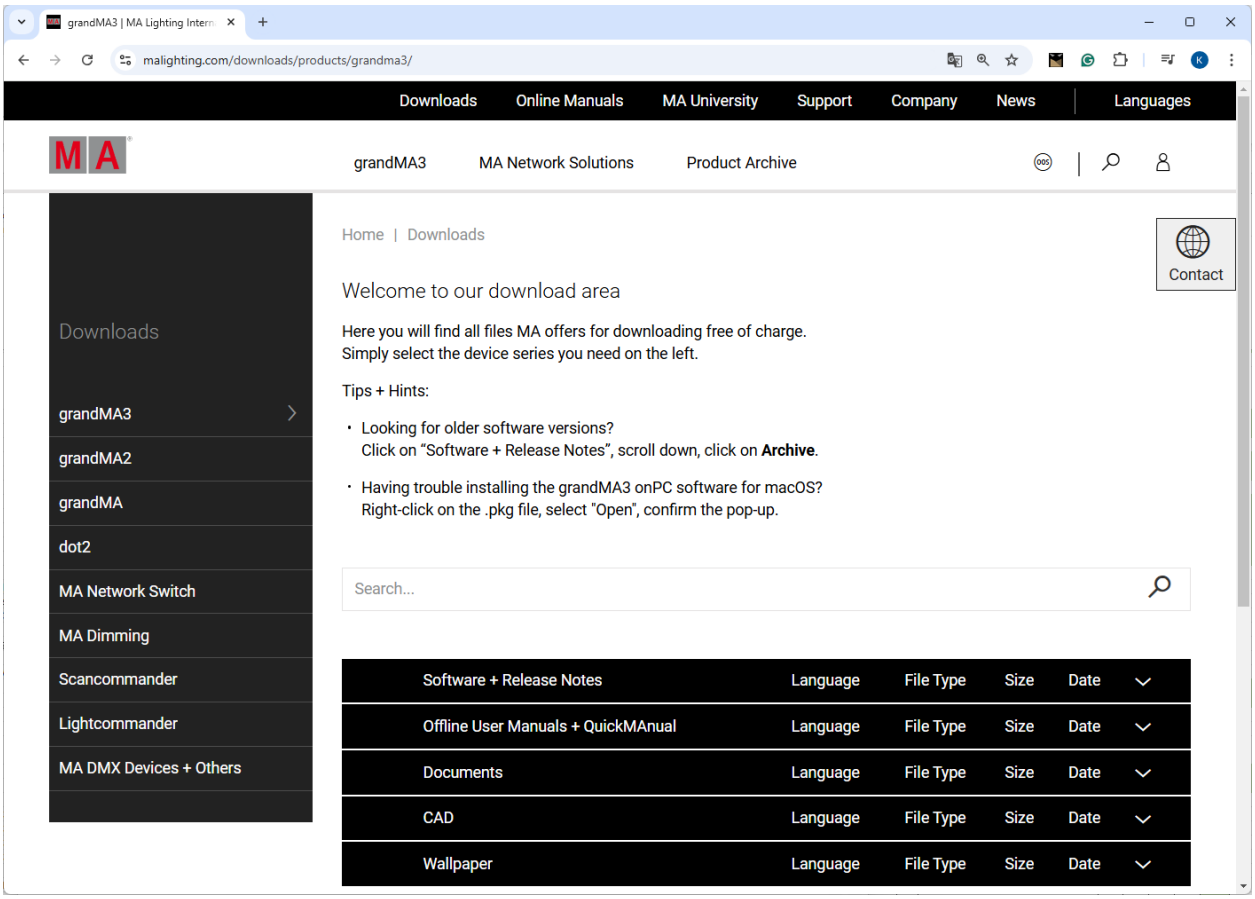
To check if there is a new grandMA3 software update:

1. Go to **www.malighting.com**, click **Downloads.**



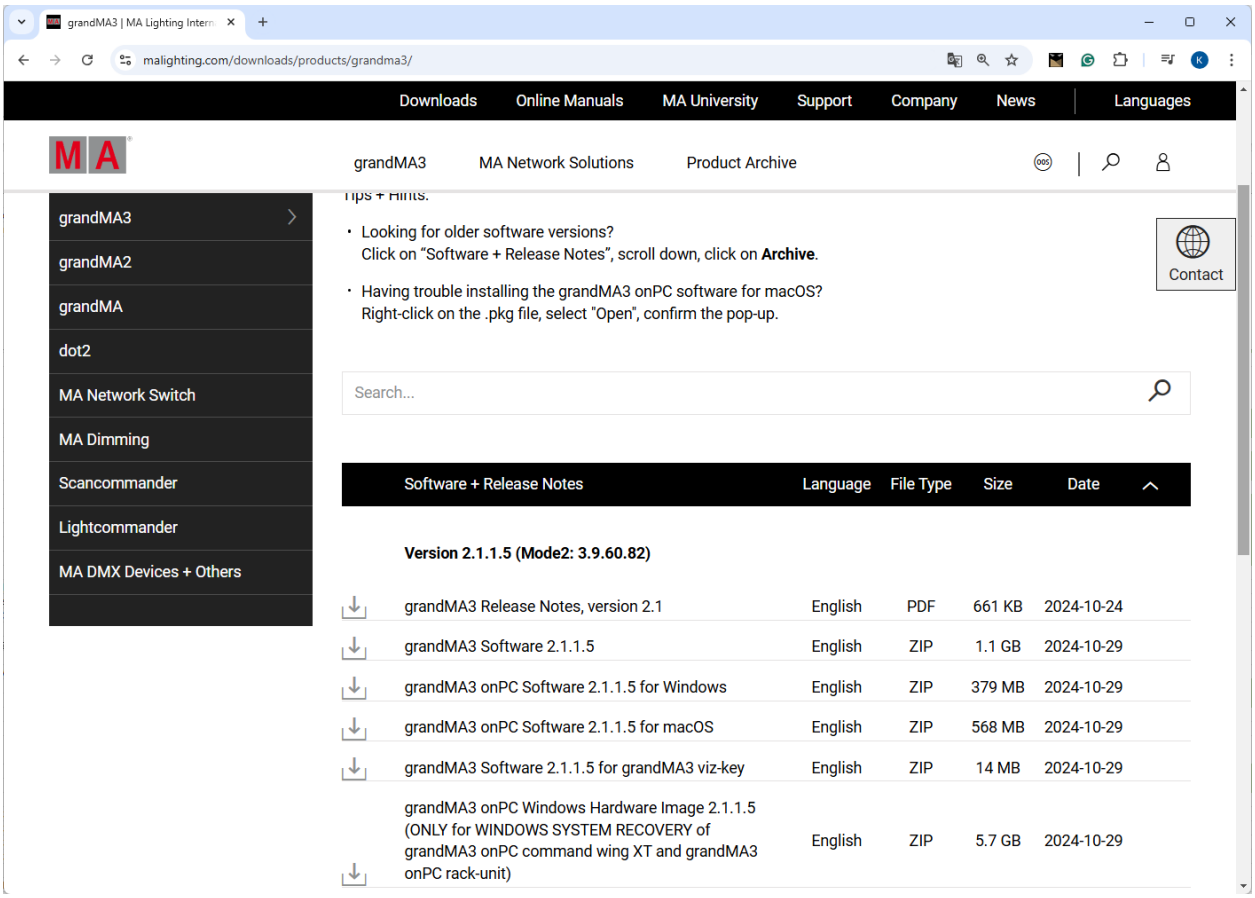
Website malighting.com Downloads

2. Click **grandMA3** in the bar on the left.



grandMA3 section

3. Click **Software + Release Notes** to find the latest software version.



grandMA3 Software section

4. Click the current version to download the desired installation package. The download process starts.

Available Installation Packages

There are five installation packages available, as shown in the table below:

Software / Package	Device
grandMA3 Software x.x.x.x / grandMA3_stick_vx.x.x.x.zip	Console, RPU, processing unit, xPort Node, and I/O Node
grandMA3 onPC Software x.x.x.x for windows / grandMA3_onPC_win_vx.x.x.x.zip	Windows operating systems, e.g., PC or laptop, command wing XT, rack-unit, viz-key
grandMA3 onPC Software x.x.x.x for macOS / grandMA3_onPC_mac_vx.x.x.x.zip	macOS
grandMA3 Software x.x.x.x for grandMA3 viz-key / grandMA3_viz_key_v1.9.7.0.zip	viz-key only
grandMA3 onPC Windows Hardware Image x.x.x.x / grandMA3_Windows_Hardware_Image_vx.x.x.x.zip	Windows system recovery of command wing XT and rack-unit, including grandMA3 onPC

	Important:
	For network updates, the target software installation package must also be downloaded. For example, when updating a console via grandMA3 onPC.

Store the Installation Packages

All files for each device type are included in the corresponding zip-file.

To update the software via network from a onPC station:

1. Open the installation package zip-file.
2. Open the ma folder:
    - For Windows systems, copy the files from the ma folder into the directory C:\ProgramData\MALightingTechnology\installation_packages.
    - For macOS systems, copy the files from the ma folder into the directory ~/MALightingTechnology/installation_packages.

To create a USB flash drive containing all the installation files, see **Update grandMA3 Consoles**.

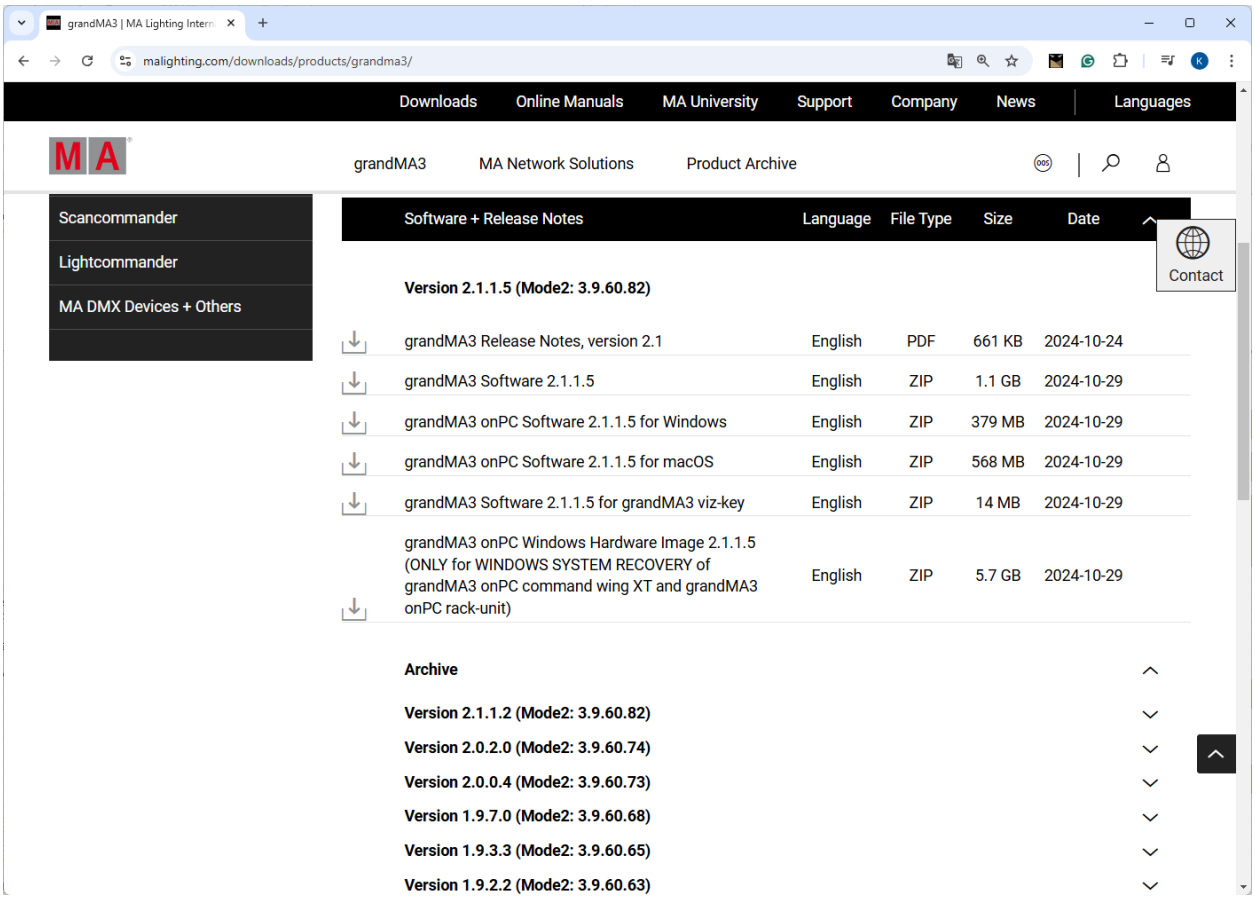
- For information about windows grandMA3 onPC installation, see **Windows™® Installation**.
- For information about macOS grandMA3onPC installation see **macOS® Installation**.

Follow all the onscreen instructions that appear during the update.

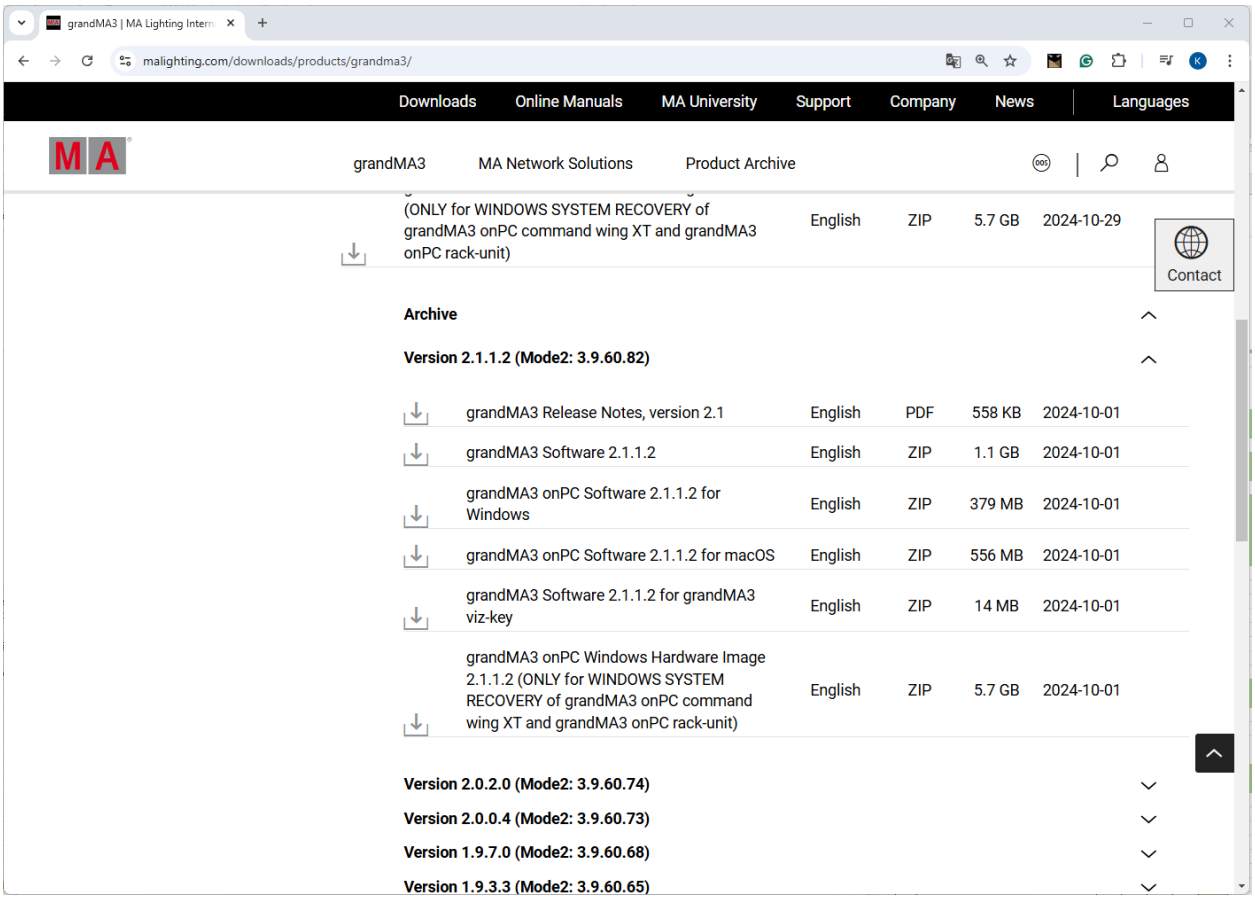
Download an Older Version

To downdate a grandMA3 device to an older software version, e.g. in a permanent installation, follow these steps:

1. Follow steps 1 through 3 above.
2. Scroll down and click **Archive** to find older software versions.



grandMA3 Software archive To open the folder, click the desired version, e.g. Version 2.1.1.2.



grandMA3 Software archive versions The download process starts.

Subtopics

- **Update grandMA3 Consoles**
- **Update grandMA3 Nodes**
- **Update grandMA3 onPC Windows Hardware**
- **Update grandMA3 viz-key**
- **Network Update**
- **Delete Update Files**
- **Troubleshooting Update Process**

1.52.1. Update grandMA3 Consoles

This topic describes the software update with additional options such as factory reset, etc.

For a simple software update via the user interface, follow the instructions in the **Network Update topic**.

For a processing unit, the following update process is also valid.

	Important:
	- The folders EFI, ma, and the update.scr file have to be directly accessible on the flash drive and must not be located in an extra folder. - The USB flash drive's data system has to be FAT32.

Download the Software Package

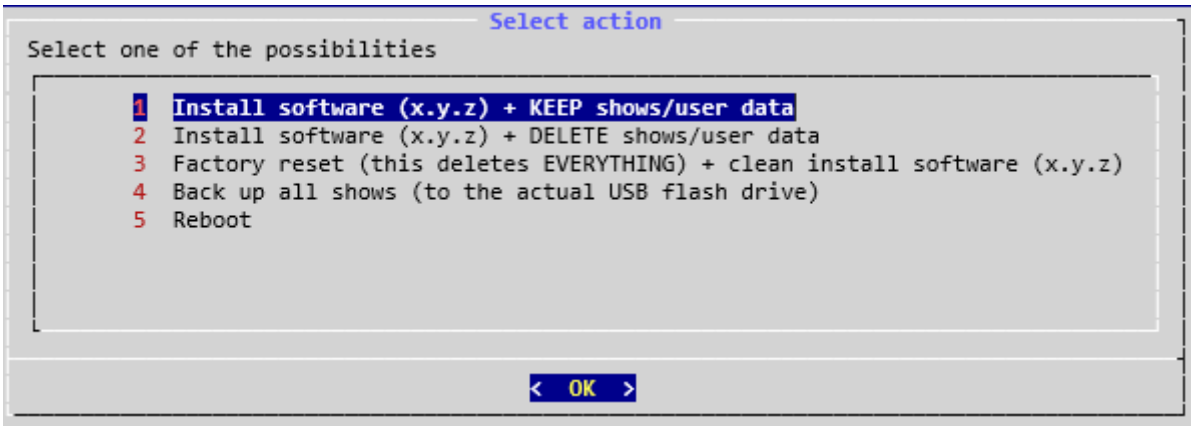
1. To update a grandMA3 device, download the latest software version from **www.malighting.com**.
   The required installer is called grandMA3 Software x.x.x.x.
2. Extract the zip file "grandMA3_stick_v.x.x.x.x.zip" and copy the folders EFI, ma, and the update.scr file into the root directory of your USB flash drive.

Install the Software Package on the Console

1. Turn off the grandMA3 device.
2. Insert the USB flash drive in a USB port.
3. For devices without an integrated keyboard (e.g. grandMA3 compact console), connect an external keyboard with a USB port.
4. Turn on the grandMA3 device.
5. Press the key 8/F8 on the internal or the external keyboard several times.
   The **Boot Manager** opens.
6. Scroll down to EFI USB Device using the arrow keys.
7. Press Enter on the (external) keyboard.
   The console starts to boot. The EULA screen opens.
8. Accept the EULA.

	Important:
	If the USB flash drive contains more than 1 version, select the version you would like to install first.

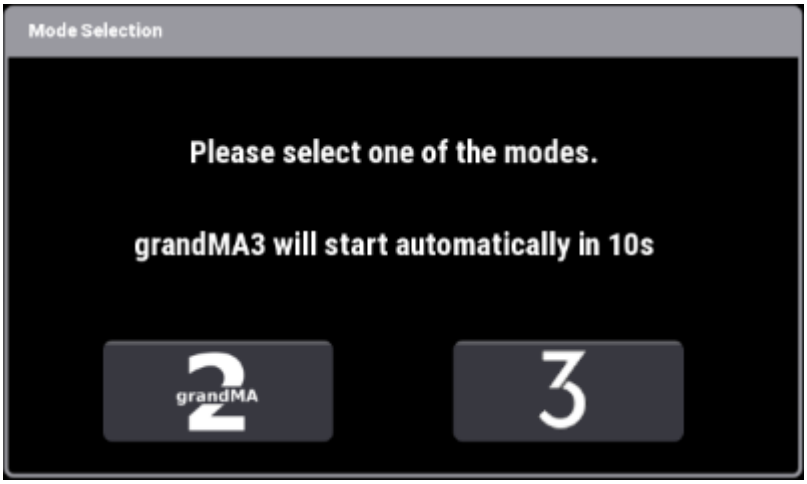
The Install Selector dialog appears:



- Select option 1 to keep the shows and user data.
- Select option 2 to delete the shows and user data.
- Select option 3 for a complete factory reset to clean the system before installing the software.
- Select option 4 to save the shows on the USB flash drive.
- Select option 5 to reboot the grandMA3 device.

1. Press Enter on the (external) keyboard.
   Wait for completion. The grandMA3 device reboots.
2. Remove the USB flash drive.

The Mode Selection dialog appears.



Select mode

3. Select one of the modes. For more information about grandMA3 mode2, read the **Mode2 topic** in the section grandMA3 Mode2 of the **grandMA2 User Manual**.
   If no selection is made, the device automatically starts in grandMA3 mode.
4. Screens 1, 2, 3 are initializing.
   The letterbox screens go into self-test mode (red, green, blue, white, and black color changer).
   The command screens stay black.
   It can take several seconds for them to start initializing.
5. The installation is complete.

1.52.2. Update grandMA3 Nodes

	Important:
	We recommend formatting with every update. Whenever formatting, the IP address will be reset back to default (DHCP).
	Important:
	-The folders EFI, ma, and the update.scr file have to be directly accessible on the flash drive and must not be located in an extra folder. -The USB flash drive's data system has to be FAT32.

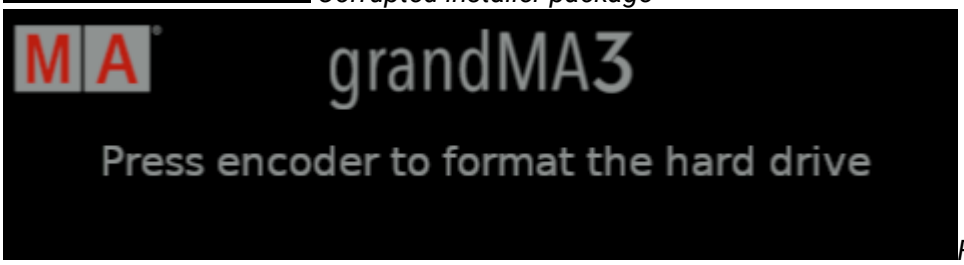
Requirement:

1. To update a grandMA3 device, download the latest software version from www.malighting.com. The required installer is called grandMA3 Software x.x.x.x.
2. Extract the zip file "grandMA3_stick_v.x.x.x.x.zip" and copy the folders EFI, ma, and the update.scr file into the root directory of your USB flash drive.
3. Insert the USB flash drive in the device's USB port.
4. Turn off the grandMA3 device.
5. Turn on the grandMA3 device.
6. Follow the onscreen instructions during the update process.

Update notification

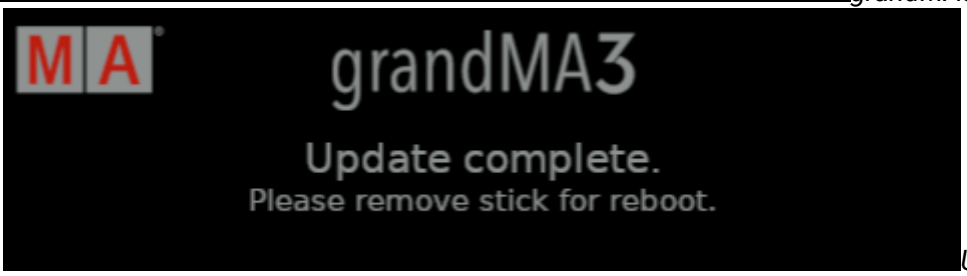
- Press the rotary knob.

Corrupted installer package



Format the hard drive

grandMA3 xPort Node is updating



Update completed

1.52.3. Update grandMA3 onPC Windows Hardware

Updating the software for the grandMA3 onPC windows hardware, such as grandMA3 onPC command wing XT or grandMA3 onPC rack-unit, is similar to the update process of the windows version of grandMA3 onPC.

For more information on how to update a grandMA3 on PC for windows, see the **Windows installation** topic.

	**Important:**
	Select the grandMA3 onPC installer for Windows from the download section!

Reset to Factory Defaults

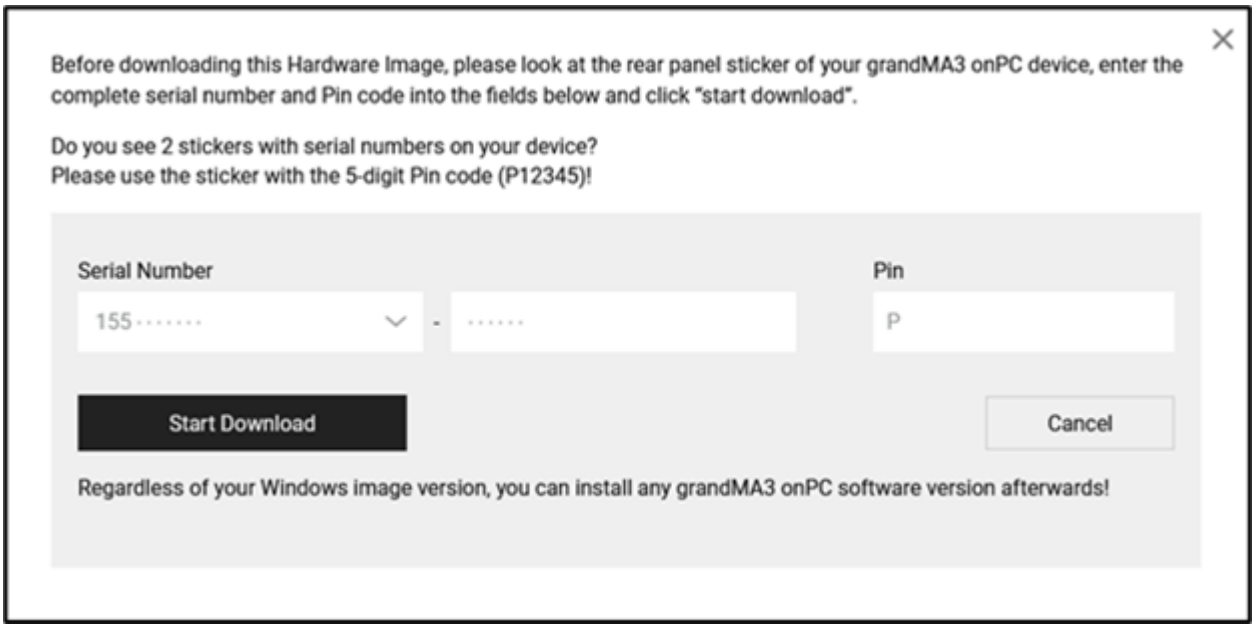
Use the following instruction to fully clean the drive or to remove your files from the drive.

Install the complete windows recovery installation package on the hard drive. This includes the operating system and the grandMA3 onPC software information.

The windows recovery installation package always installs the latest version of the grandMA3 onPC software and the Windows updates!

To download the windows recovery installation package:

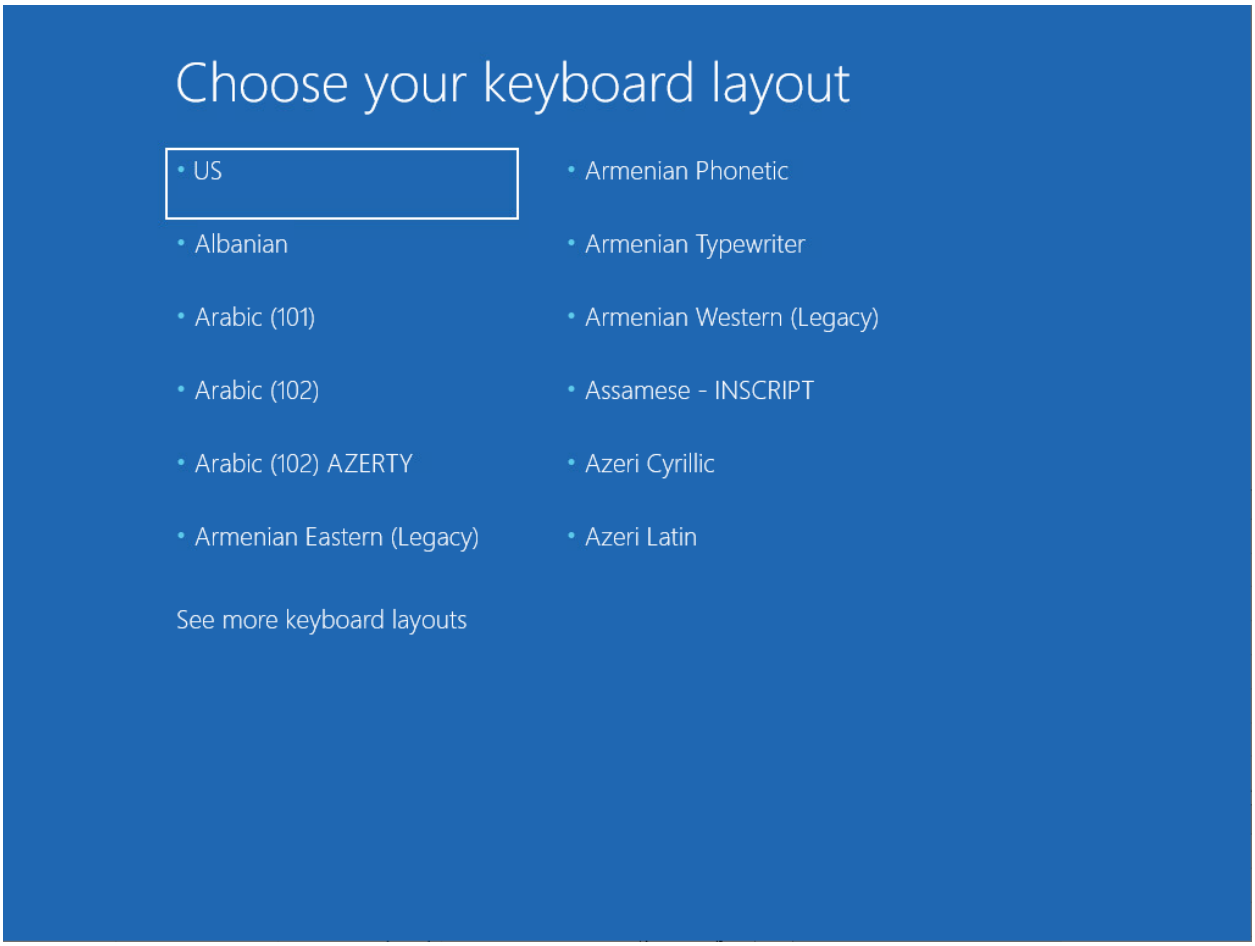
1. Go to the download area of grandMA3.
2. Click on grandMA3 onPC Windows Hardware Image x.x.x.x. The EULA and Third-Party Acknowledgements pop-up opens.
3. Click Accept and download. A pop-up opens.
4. Enter the serial number and pin as described in the pop-up:



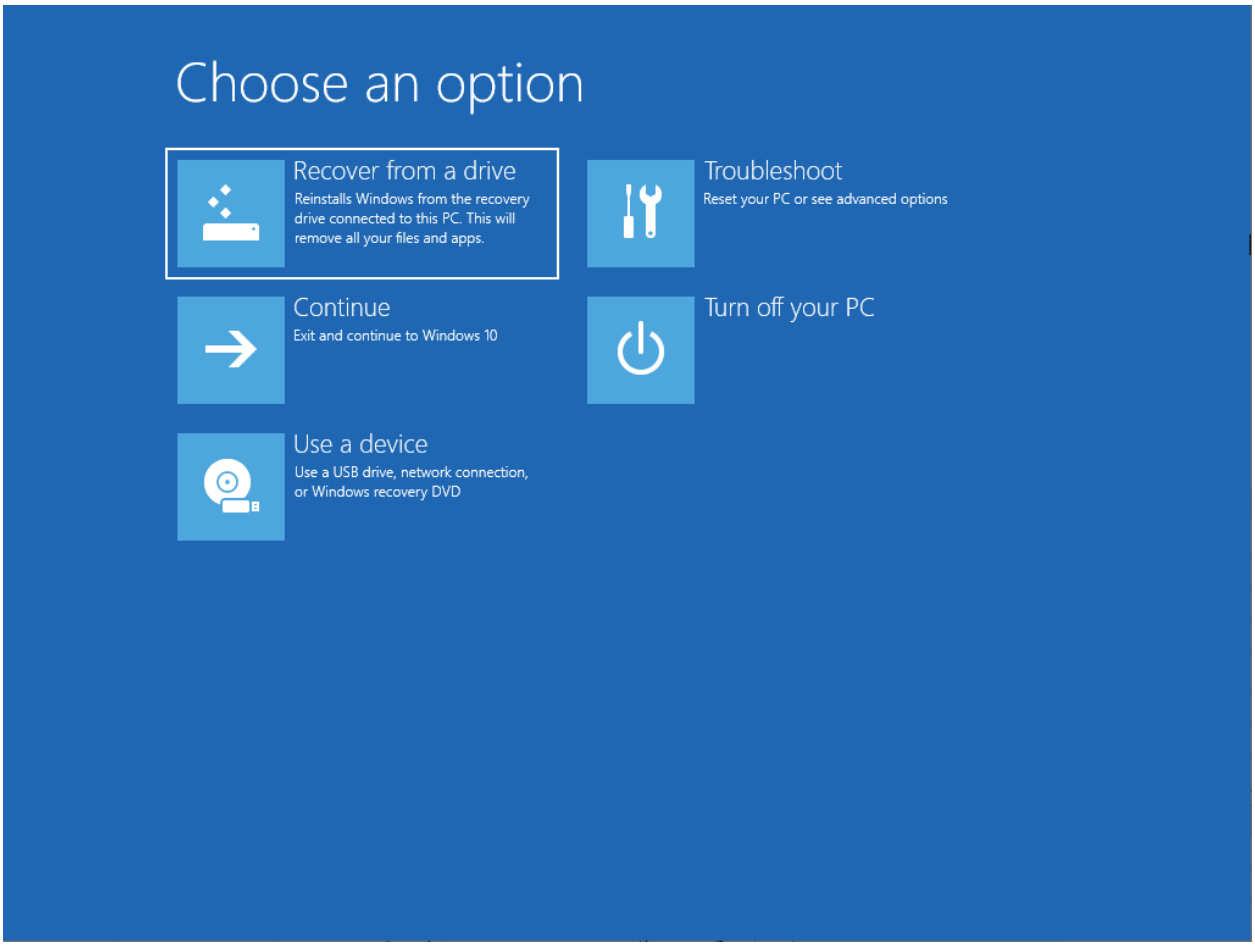
5. Click Start Download to download the file.

To install the windows recovery installation package:

1. Extract the entire data from the zip file and copy it onto a USB flash drive (FAT32 formatted, minimum 8 GB).
2. Insert the USB flash drive.
3. Power up the grandMA3 onPC command wing XT or grandMA3 onPC rack-unit.
4. If necessary, tap F8 to choose Boot device and select boot from USB.
5. Choose a keyboard layout, for example, US.

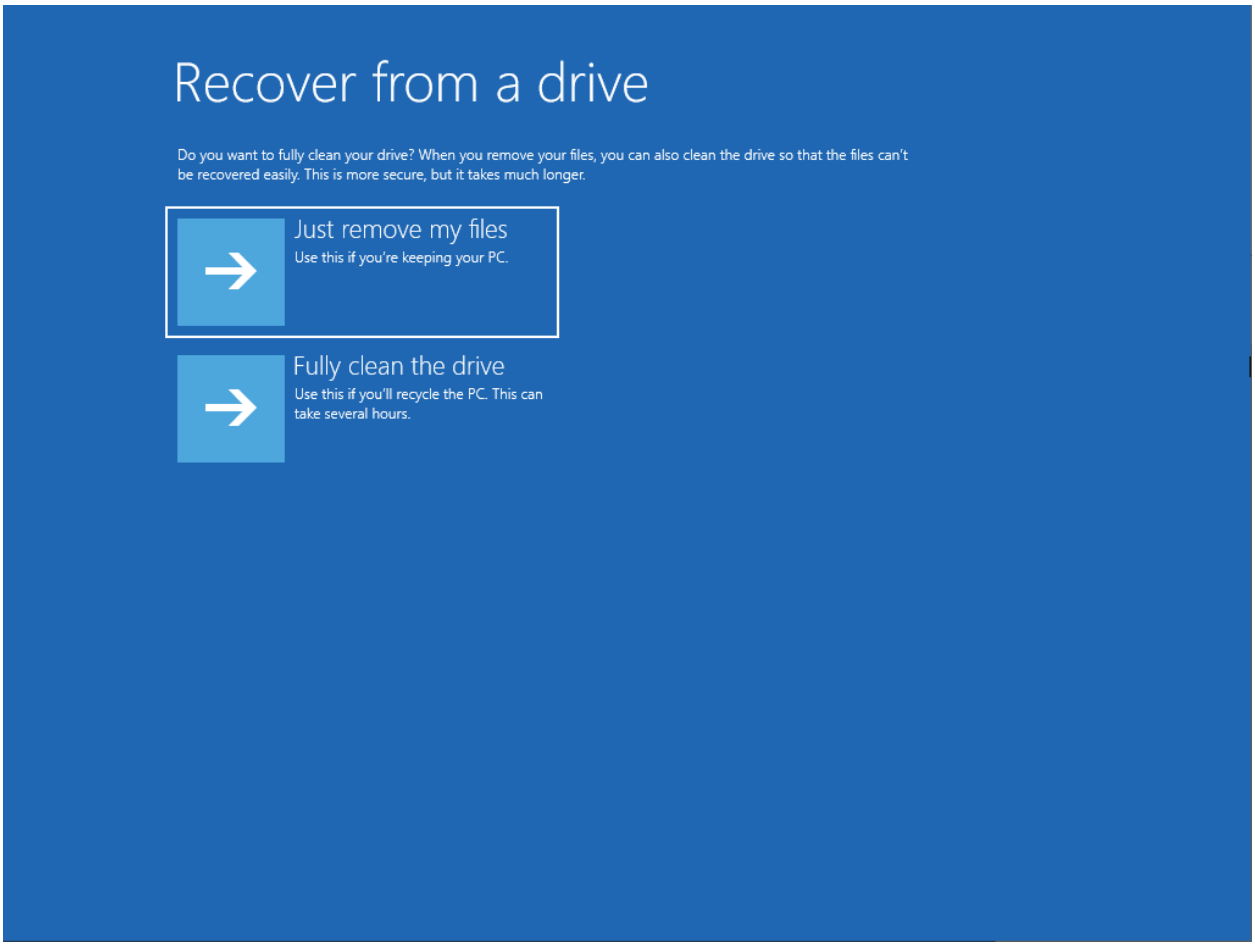


6. Click Recover from a drive.

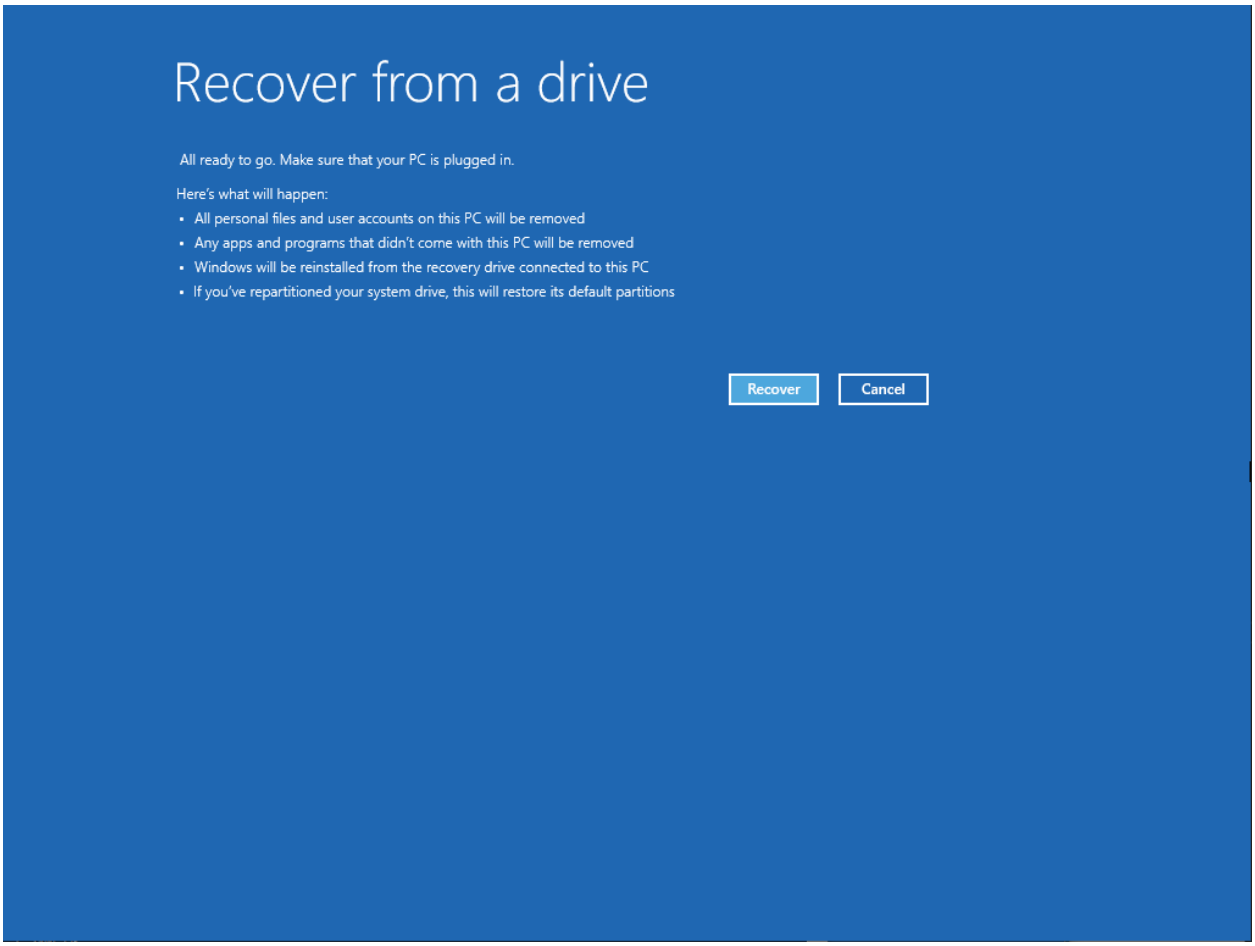


	Hint:
	If the "Recover from a drive" option does not appear, the USB flash drive may be faulty. Try another USB flash drive and make sure that the files are properly unzipped.

7. Click Just remove my files. If you want to do a complete clean install without keeping any user files, continue with Fully clean the drive.

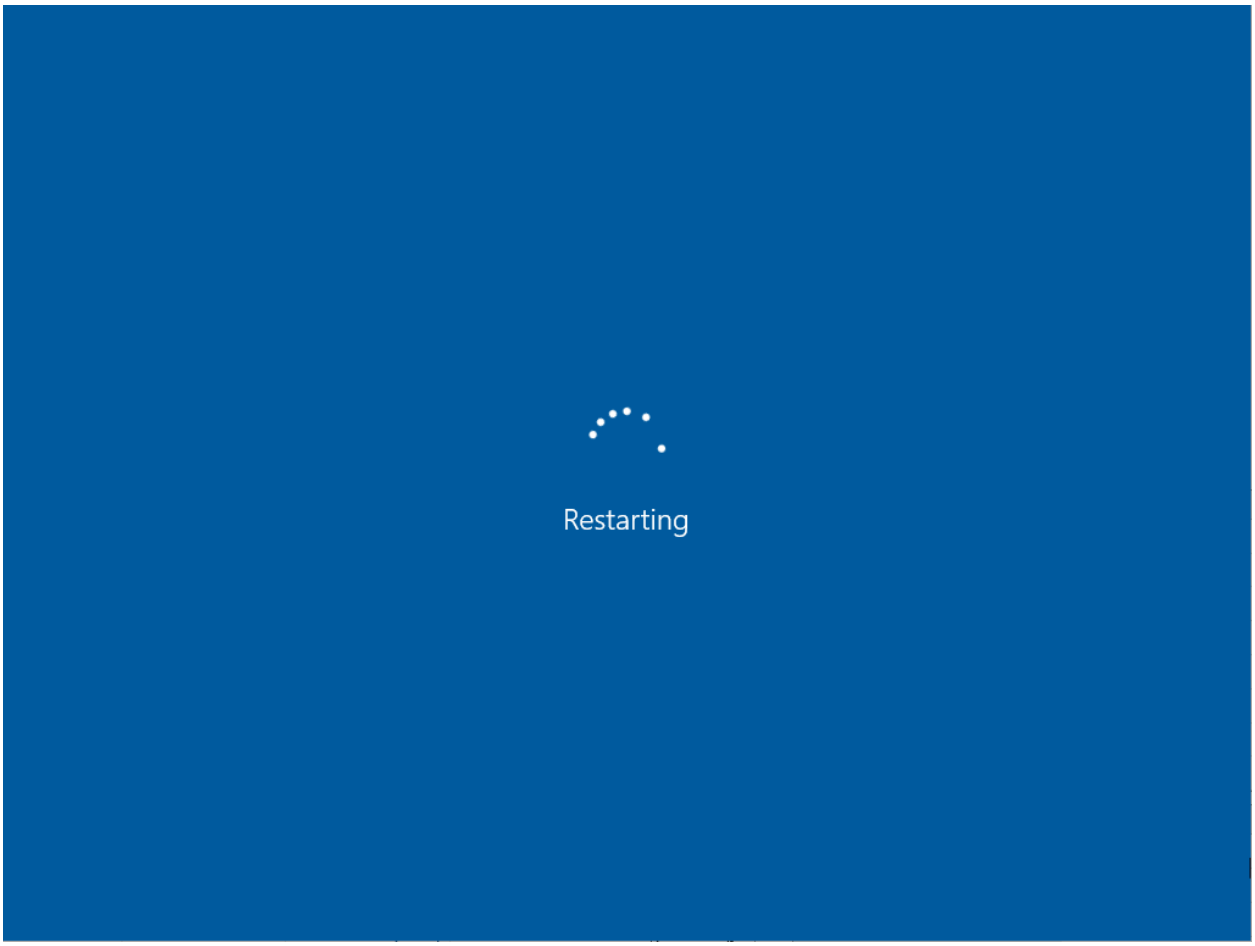


8. Click Recover. The License Agreement opens.

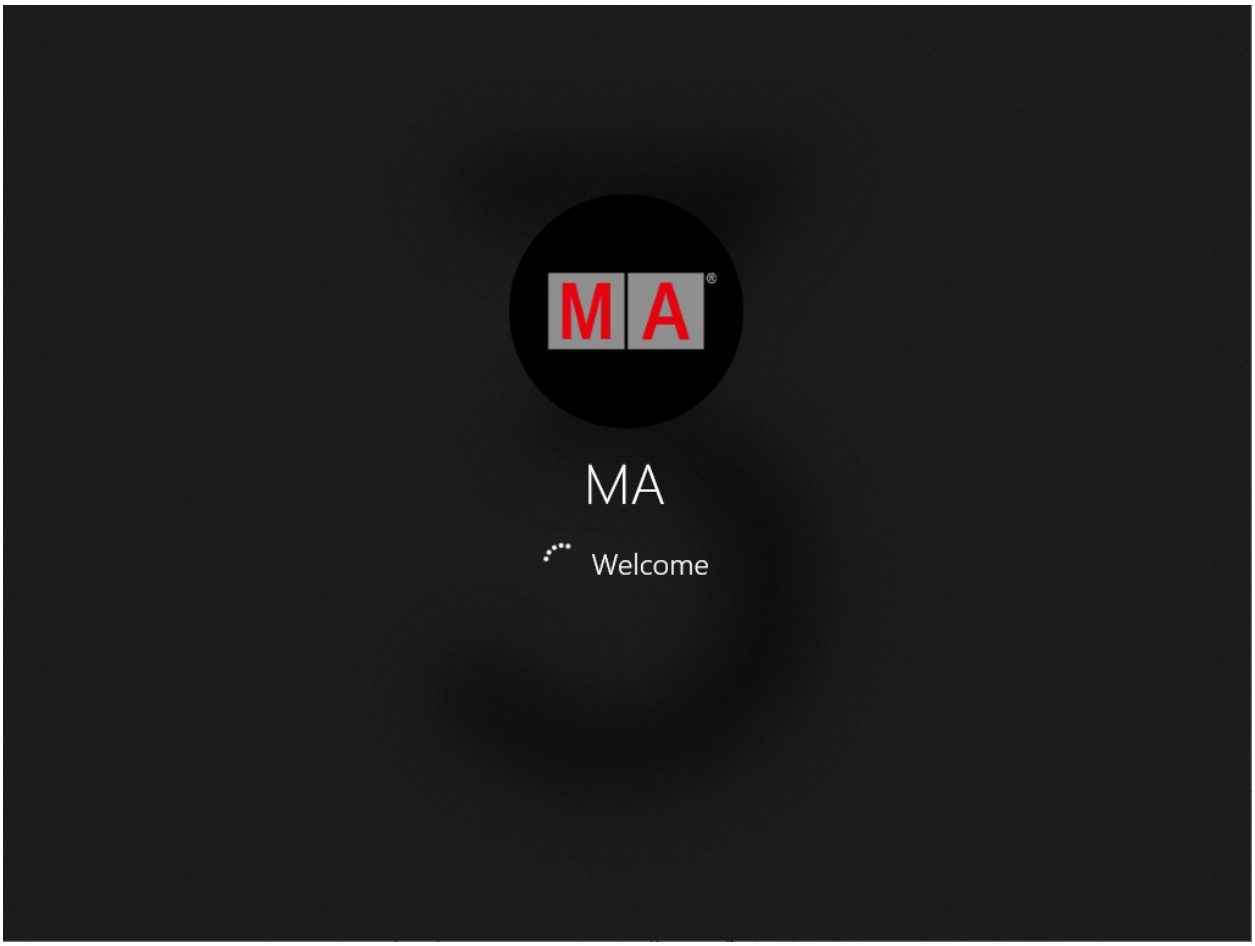


9. Click Accept. The system restarts several times.

The system restarts several times.

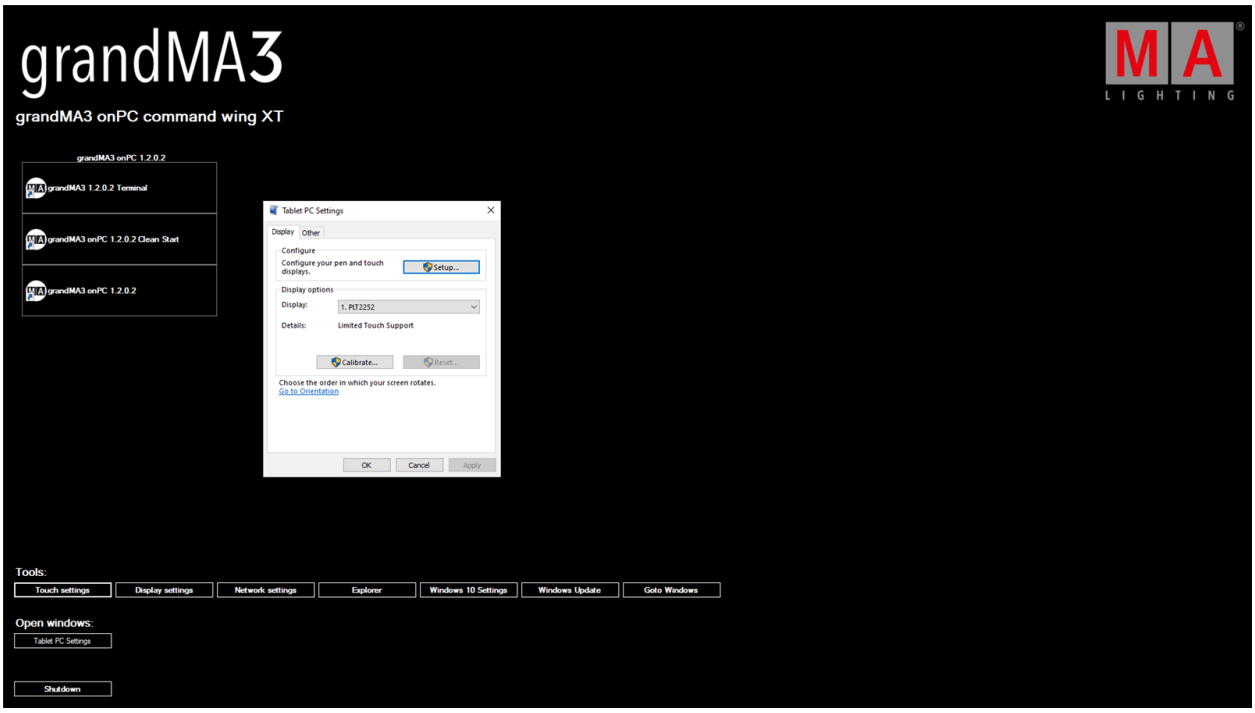


10. The MA Shell Launcher starts for the first time.

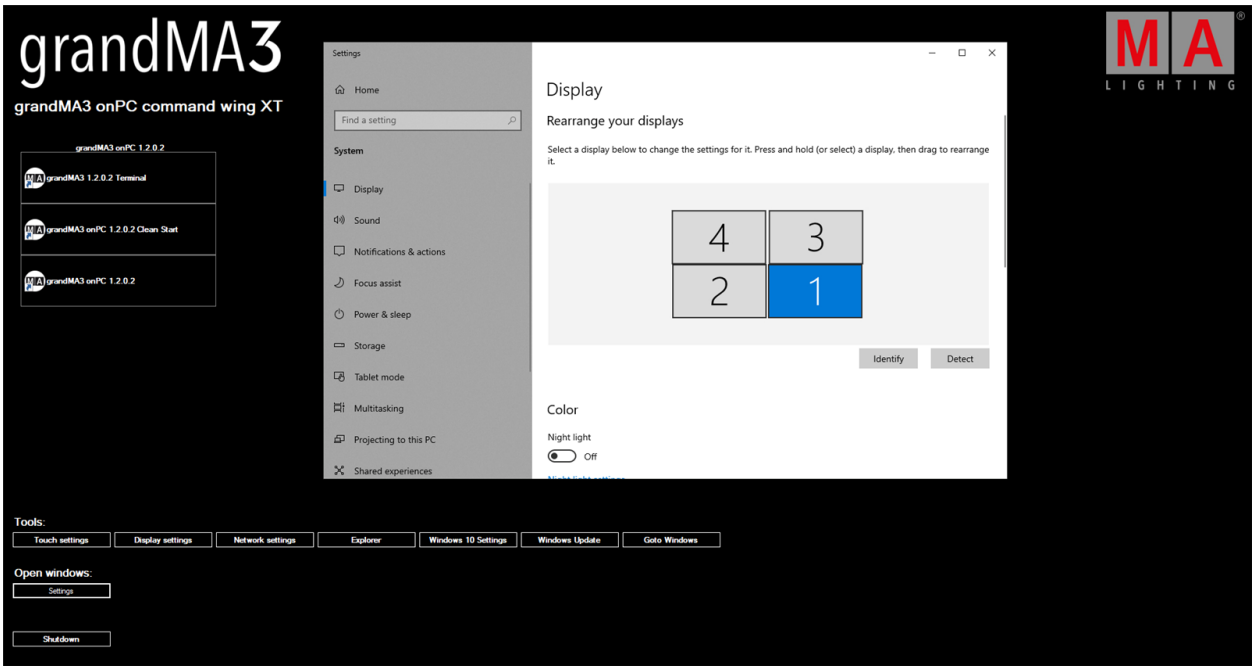


Configure the Settings

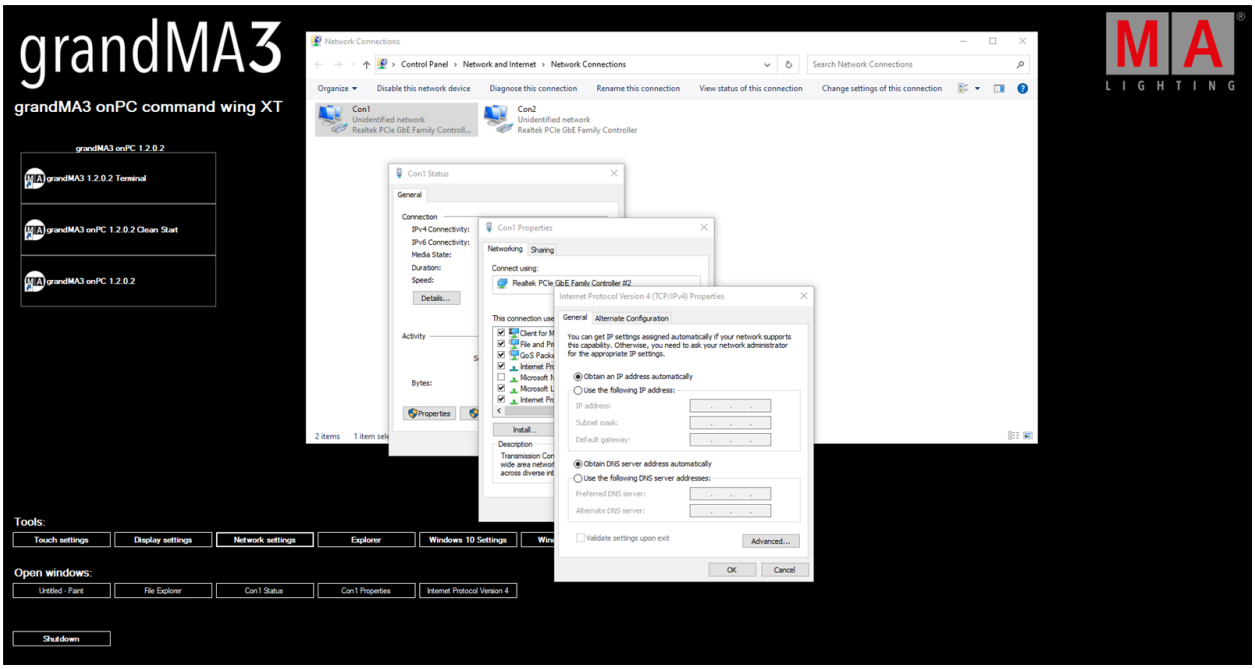
1. To configure the touch screens, click Touch Settings.



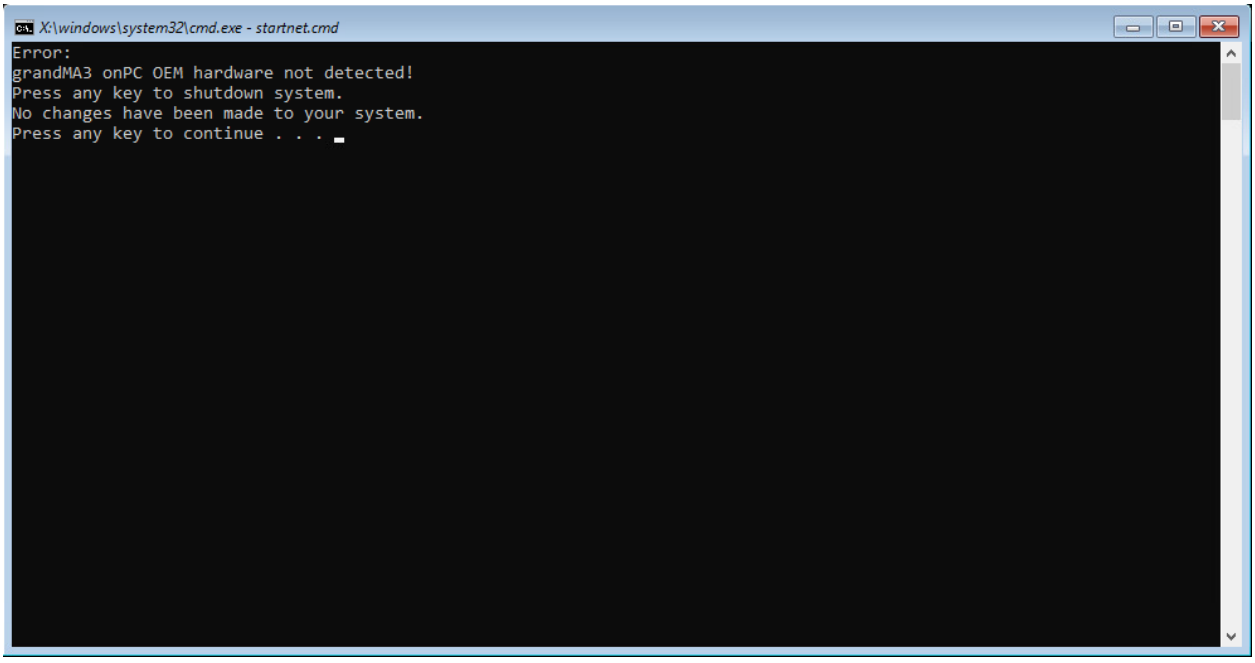
2. To set the desired screen configuration, click Display Settings.



3. To set the desired screen configuration, click Display Settings.



	Hint:
	If an incorrect motherboard is installed or if the windows recovery installer package is installed on a different device, you receive the following error message.



1.52.4. Update grandMA3 viz-key

	Important:
	The grandMA3 onPC software includes the grandMA3 viz-key software. There is no additional installation required to use the grandMA3 viz-key. In case that the grandMA3 onPC software is not installed on the computer that is running the visualizer, the separate grandMA3 viz-key software needs to be installed.
	Hint:
	If the grandMA3 viz-key hardware is connected to a grandMA3 onPC station or visualizer, the running software version is sent to the grandMA3 viz-key hardware.

- Download the latest grandMA3 viz-key software version from www.malighting.com. For more information, see **update the software**.
  The required installer is called grandMA3 Software x.x.x.x for grandMA3 viz-key.

Update Standalone Third-Party Visualizer Using a USB Stick

Requirement: The USB flash drive's data system has to be FAT32.

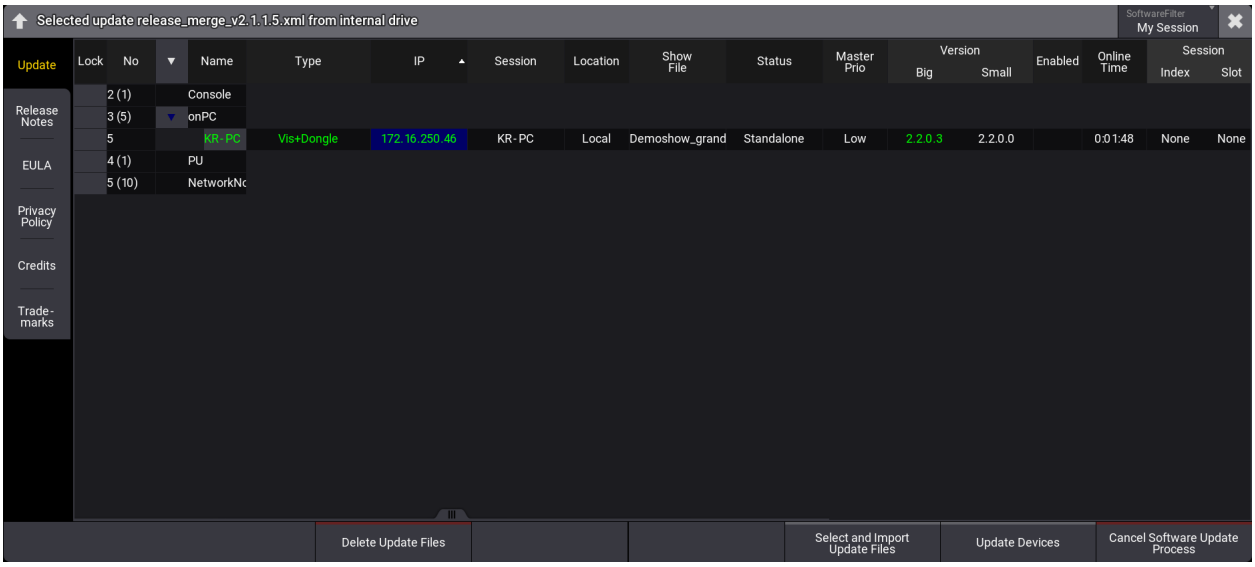
1. Extract the zip file grandMA3_viz_key_vx.x.x.x.zip and copy the ma folder into the root directory of the USB flash drive.
2. Insert the USB flash drive in the grandMA3 onPC or console USB port.

Follow the next steps from point 3.

Update Standalone Third-Party Visualizer Using the Local Hard Drive

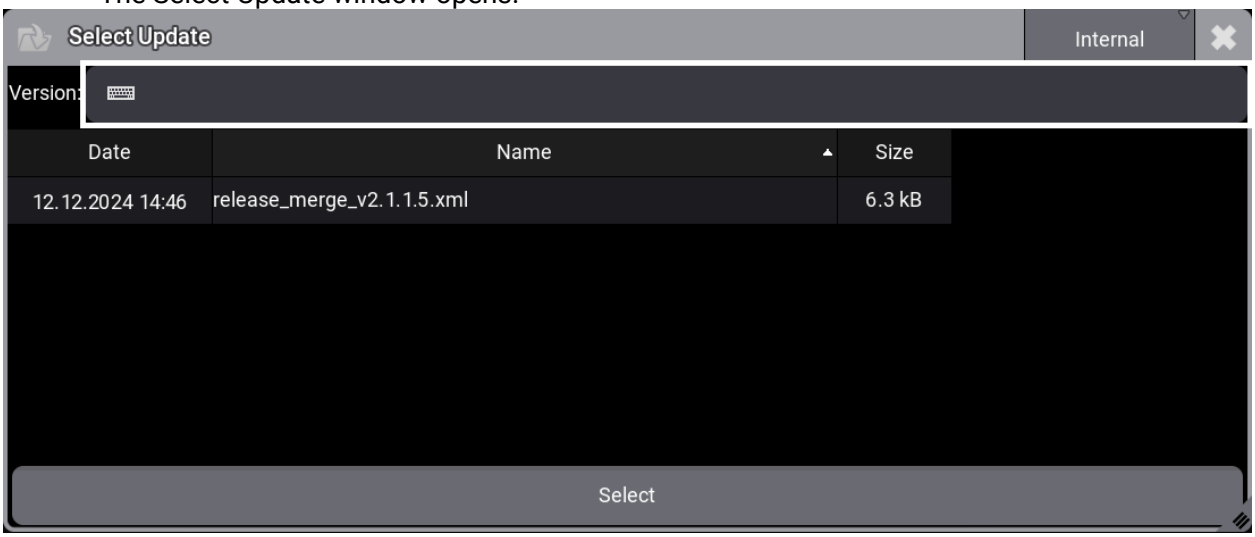
1. Extract the zip file grandMA3_viz_key_vx.x.x.x.zip.
2. For Windows systems, copy the files from the ma folder into the directory C:\ProgramData\MALightingTechnology\installation_packages.
   For macOS systems, copy the files from the ma folder into the directory ~/MALightingTechnology/installation_packages. For more information, see **Folder Structure**.
3. To access Software Update, tap .
4. Tap Settings.
5. Tap Software Update. The Software Update window opens:

	Important:
	The third-party visualizer must be running in the update mode. For more information, see the manual of the third-party software.



Software Update Window with selected update file

6. Tap Select and Import Update Files.
   The Select Update window opens.



Select Update pop-up

8. Select the location that contains the update files (internal or any plugged-in external device).
   Select the release_viz_key_vx.x.x.x.xml file.
9. Tap Select.
10. The pop-up closes and the End User License Agreement (EULA) opens.
    Confirm the End User License Agreement (EULA).
    The selected update file is displayed at the title bar of the software update window.
11. Select the third-party visualizer with viz-key support. The selected devices turns into bright blue.
12. Tap Update Devices.
13. The software update starts copying files.
14. Once the file is transferred, restart the third-party visualizer with the viz-key support software.

	Hint:
	To learn more on how to connect, visit **https://www.malighting.com/viz-key.**

1.52.5. Network Update

Use Software Update to update one or more stations on a network.

	Important:
	Make sure that all devices are in the same network (check network adapter). To learn more about network settings, read the Networking, **Interfaces and IP** topic.

Preliminary Procedures

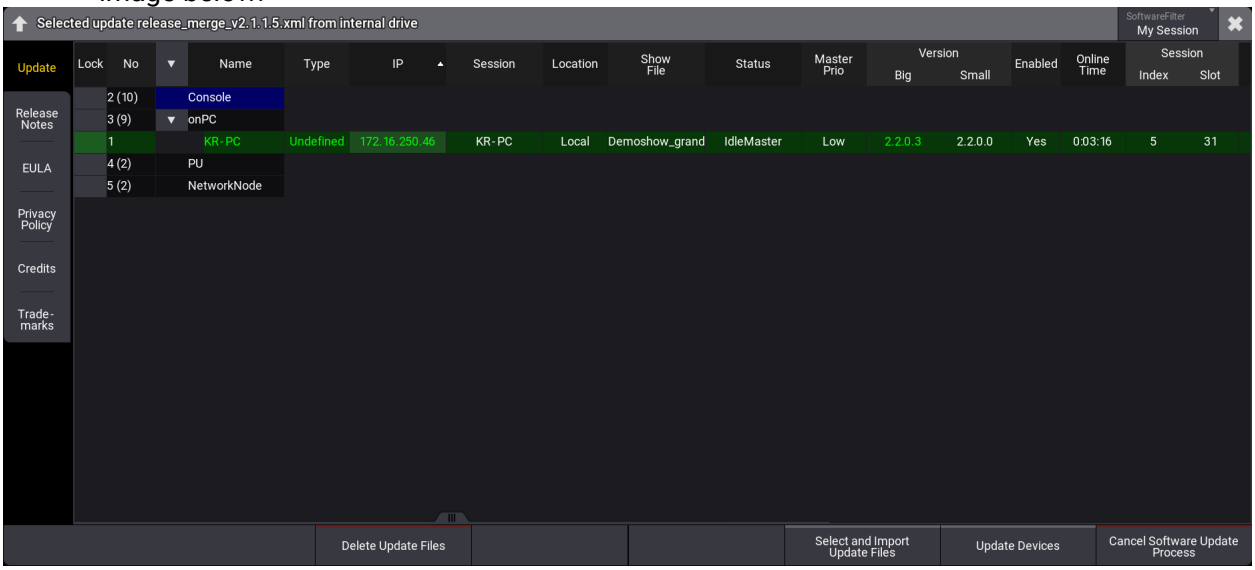
- Connect all grandMA3 devices to a network with an etherCON/RJ45 connector.
- Download the required installation packages from www.malighting.com, **Downloads.**
- Copy all installation packages for grandMA3 software into the root directory of your USB flash drive. For more information, see **Update the Software**.
- Insert the USB flash drive in the device's USB port.
- For Windows systems, copy the zipped files from the ma folder into the directory C:\ProgramData\MALightingTechnology\installation_packages.
- For macOS systems, copy the files from the ma folder into the directory ~/MALightingTechnology/installation_packages.

Procedure

To update the grandMA3 device, follow the onscreen instructions that appear during the update.

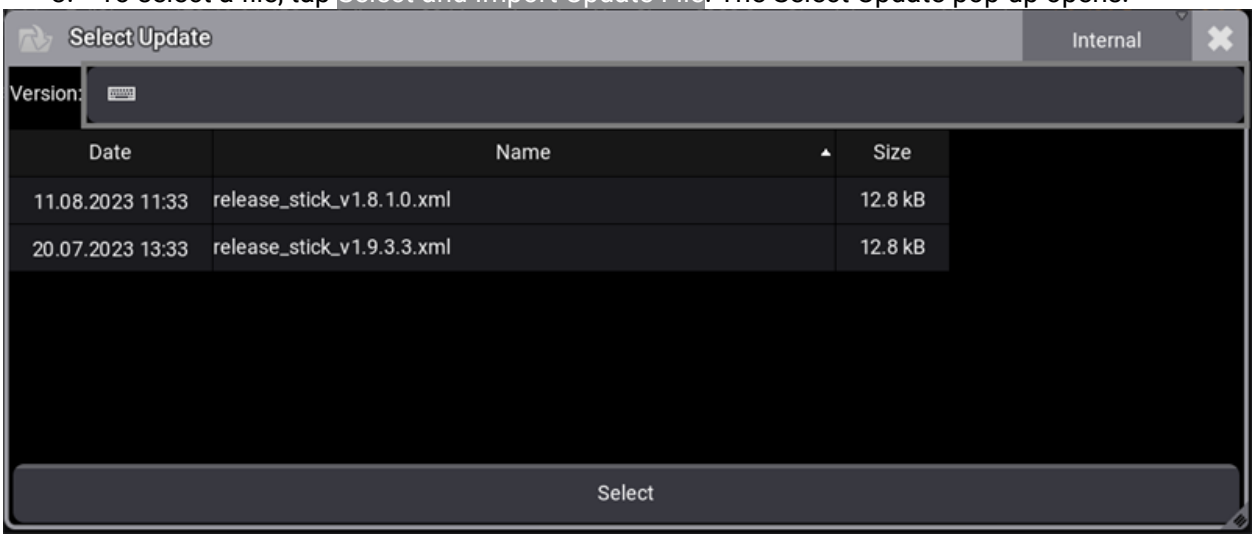
To access Software Update:

1. Press Menu or tap .
2. Tap Settings and then tap Software Update. The Software Update window opens, as shown in the image below:



Software Update window

3. To select a file, tap Select and Import Update File. The Select Update pop-up opens:



Select Update pop-up

4. Make sure the correct update file location is selected (Interal or external device). Select the desired update file.
5. Tap Select. The pop-up closes and the End User License Agreement (EULA) opens.
6. Scroll down to read the complete EULA. The button I agree in the upper right corner of the pop-up turns white. To close the EULA, tap I agree.
7. Select the desired device(s). The selected devices are marked in blue.
8. Tap Update Devices. The software update starts copying files.

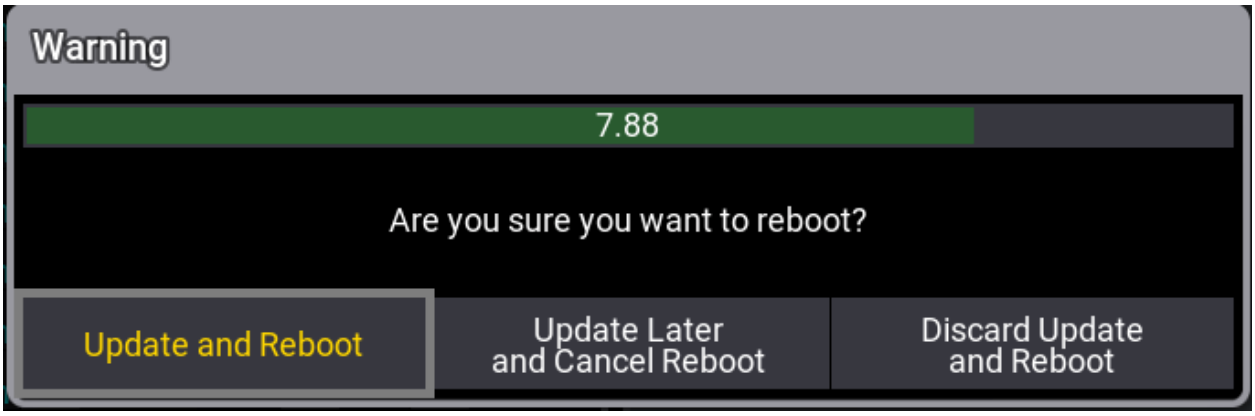
	Hint:
	The selected update file will be displayed in the title bar of the software update menu.

- Cancel Software Update Process: Tap to cancel the current update process. For more information, see **CancelSoftwareUpdate keyword**.

	Hint:
	It is also possible to update the grandMA3 **onPC**, **consoles**, **xPort Nodes**, **onPC command wing XT**, **processing unit** and **onPC rack-unit** directly.

Update Confirmation

After copying the files to the grandMA3 device, a warning pop-up will appear:

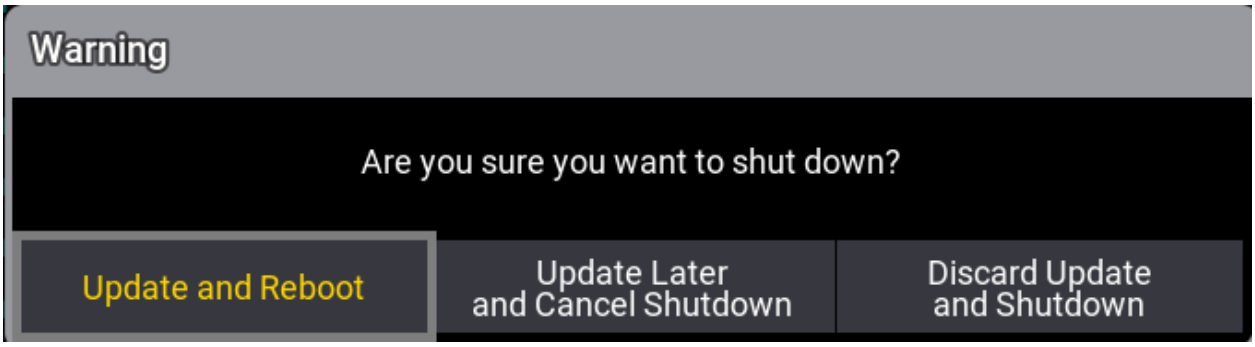


Pop-up in the update process

- **Update and Reboot:** Updates and reboots the device.
- **Update Later and Cancel Reboot:** Delays the update and leaves a red indicator in the control bar .
- **Discard Update and Reboot:** Aborts the update and reboots the device.

	Hint:
	The update will start automatically when the countdown in the pop-up window expires.

Tap in the Shutdown Menu to install delayed updates. This will open a second warning pop-up.



Shutdown pop-up

- **Update and Reboot:** Updates and reboots the device.
- **Update Later and Cancel Shutdown:** Delays the update and leaves a red indicator in the control bar .
- **Discard Update and Shutdown:** Aborts the update and shuts down the device.

	Hint:
	Tapping (Restart) in the Shutdown Menu opens an Update Confirmation pop-up as well.

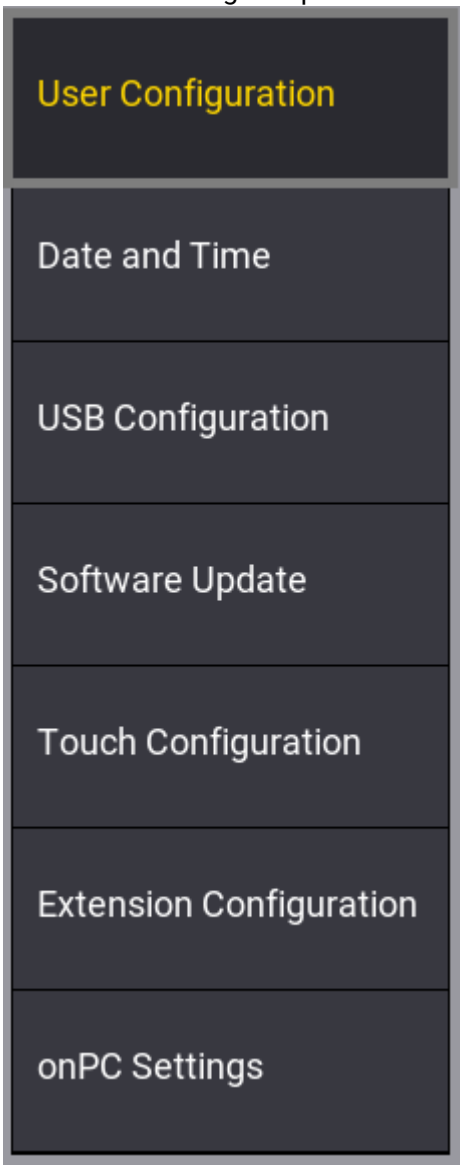
For more information about shutting down, see **Shut Down the System**.

1.52.6. Delete Update Files

To save storage on the hard drive, we recommend to delete not used update files from the hard drive.

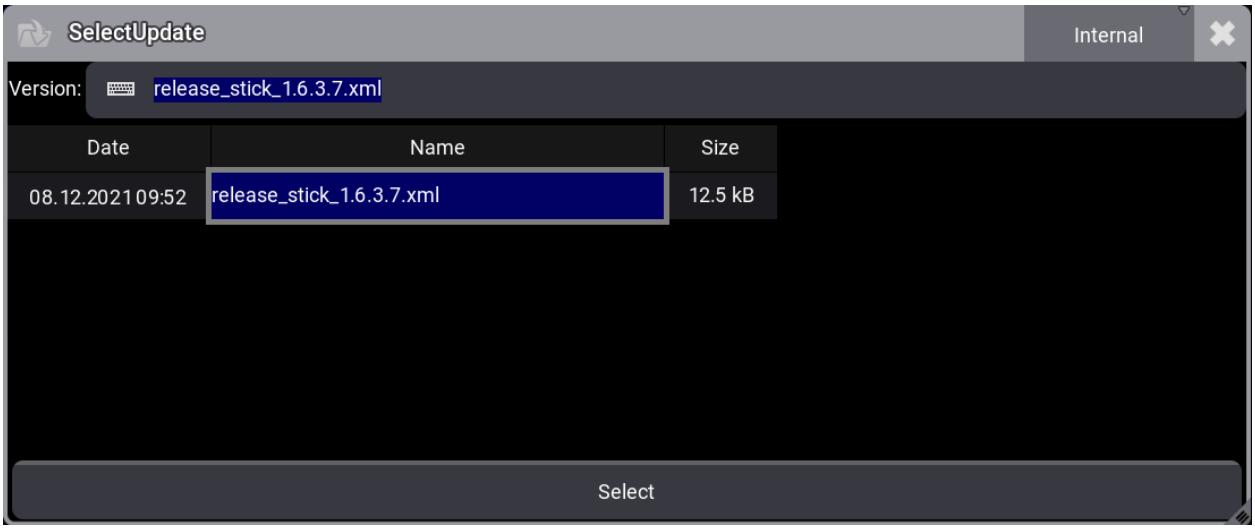
To delete update files from the hard drive:

1. To access the software update window, tap .
2. Tap Settings.
   The settings drop-down menu opens.



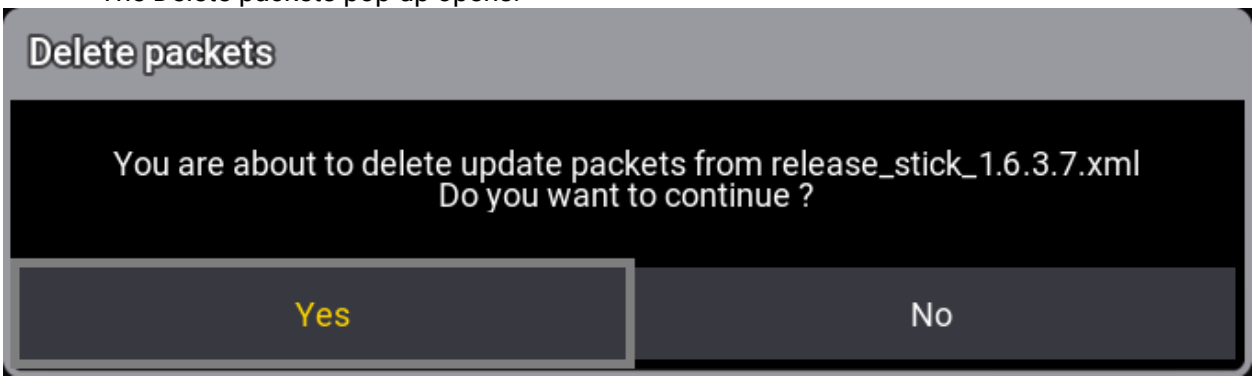
Settings drop-down

3. Tap Software Update.
   The Software Update window opens.
4. Tap Delete Update Files.
   The Select Update pop-up opens.



Select update pop-up

5. Select the to be deleted update file.
6. Tap Select.
   The Delete packets pop-up opens.



Delete packets pop-up

6. Tap Yes.

The selected update file is deleted from the hard drive.

1.52.7. Troubleshooting Update Process

During the update process, different problems may occur.

Corrupted Files on USB Flash Drive

Corrupted installer package

If the installer package is corrupted, try the following:

- Remove the USB flash drive from the grandMA3 device.
- Shutdown the grandMA3 device.
- Format the USB flash drive (FAT32).
- Copy the folders EFI, ma, and the update.scr file into the root directory of your USB flash drive.
- Update the grandMA3 device with the USB flash drive.

If the error message shows up again, try the following:

- Use a different USB flash drive.
- Format the USB flash drive (FAT32).
- Start a new download of the latest software version from www.malighting.com.
- Extract the zip file before copying the folders EFI, ma, and the update.scr file into the root directory of your USB flash drive.

Corrupted Show File

If the grandMA3 device does not stop updating, the show file saved on the grandMA3 device may be corrupted.

If the show file is corrupted, try the following:

- Remove the USB flash drive from the grandMA3 device.
- Reboot the grandMA3 device.
- Save the corrupted show file to another USB flash drive.
- Delete the corrupted show file from the grandMA3 device.
- Update the grandMA3 device with the other USB flash drive.

Individual Hardware Freezes

If the grandMA3 device starts updating, but fails to update one of the hardware sections, try the following:

- Remove the USB flash drive from the grandMA3 device.
- Reboot the grandMA3 device.

If the measures above do not help, try the following:

- Save any show file to another USB flash drive.
- Delete any show file from the grandMA3 device.
- Update the grandMA3 device with the other USB flash drive.

1.53. Fixture Types

Fixture types are used to visualize and control real-life fixtures on stage.

A fixture type is a footprint of a real fixture, smoke machine, laser, media server, dimmer, or anything to be controlled and visualized with a grandMA3 console.

A fixture type is described with the following parameters:

- modes
- channels
- physical properties

You can find a substantial amount of fixture types and manufacturers in the grandMA3 library.

Fixture types are also available for download from MA Fixture Share and GDTF Share webpages.

Beyond that, it possible to build a fixture from scratch with its own parameters and attributes using the fixture builder! The grandMA3 software includes its own fixture builder.
Using the GDTF Builder is also an option.

Links

For more information on Fixture Share, see **https://www.malighting.com/training-support/fixture-shares/**.

For more information on GDTF Share, see **https://gdtf-share.com/**.

For more information on GDTF Builder, see **https://fixturebuilder.gdtf-share.com/**.

Subtopics

- **Import Fixture Types**
- **Build Fixture Types**
- **Export Fixture Types**

1.53.1. Import Fixture Types

Fixture Types is used to import fixtures for later use.

It is possible to import fixture types from these libraries:

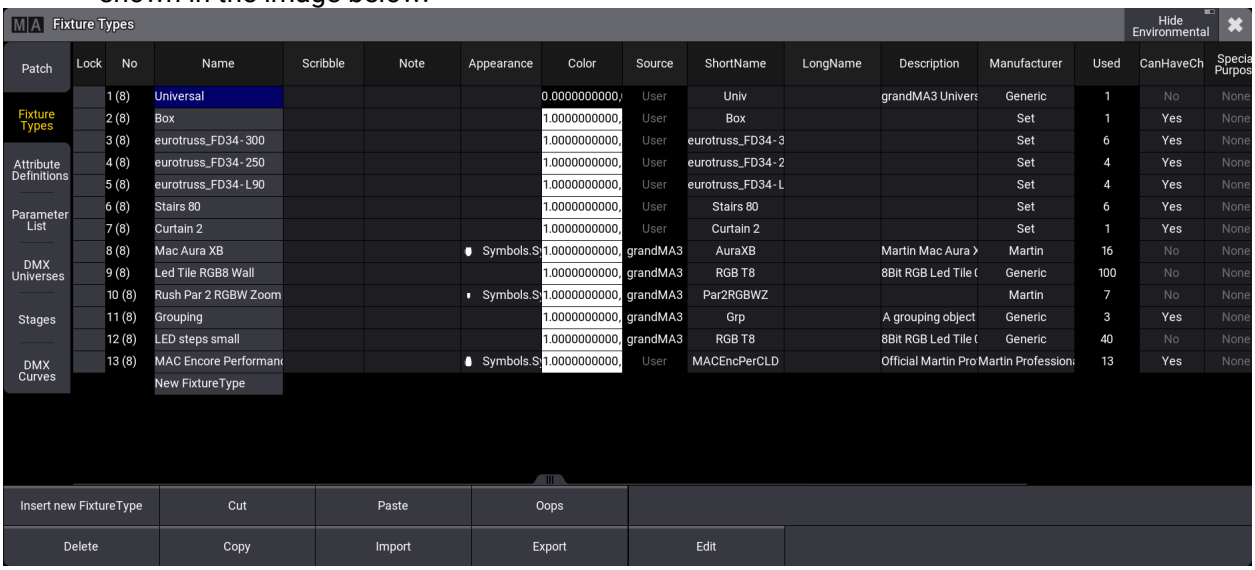
- **grandMA3** fixtures
- Converted **grandMA2** fixtures
- Fixtures using **GDTF** format

Fixture Types Menu

1. To access the Patch menu, press Menu and then tap Patch.

	Hint:
	When creating a new show, the Insert New Fixtures window opens. It is a good habit to import new Fixture Types from the Fixture Types menu. Tap on the top right corner of the Insert New Fixtures window to close it.

2. Tap Fixture Types in the bar on the left of the patch dialog. The Fixture Types menu opens as shown in the image below:



Fixture Types menu - Demoshow grandMA3

The menu is separated into different columns and rows. This is a short explanation of the different columns:

- **Lock, No, Name:**
  See **Parameter List** topic.
- **Scribble, Note, Appearance:**
  See **Scribble**, **Notes**, and **Appearances** topic.
- **Color:**
  Changes the color value. The selected color value is linked to the Fixture Types column in the Split View of the Patch window. It is also the image color of the used appearance.

- **Source:**
  Displays the origin of the fixture type. The source can be grandMA3 or grandMA2 for every fixture from the library (MA), or User for every fixture from the User () or Share (). User source includes fixture types from the GDTF share and imported grandMA3 fixture types.
- **ShortName:**
  Defines the name suggestion of fixtures the user wants to patch with this fixture type.
- **LongName:**
  Displays the full name of the fixture type/product.
- **Description:**
  Adds a description to a fixture. If a description is deposited in the **library**, It is automatically filled out.
- **Manufacturer:**
  Shows the name of the fixture type manufacturer.
- **Used:**
  Displays how many fixtures in the show use this fixture type.
- **CanHaveChildren:**
  This setting defines whether a fixture can have children. For most fixture types it is **No**.
- **Special Purpose:**
  Tap and hold a cell in this column to open the Select Special Purpose pop-up:

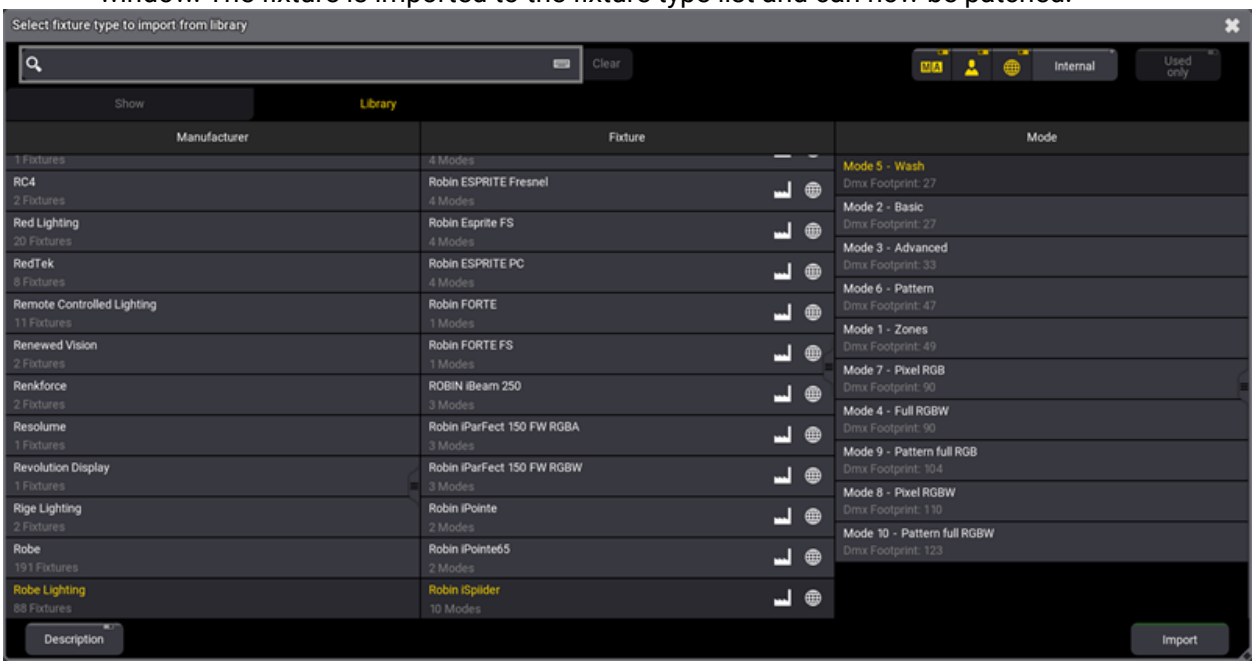
For more information, see **Bitmap** and **MArker**.

- **ShareGlobal:**
  When the same fixture type (attributes, their order, and their physical form and physical to values) is used several times within a show file, the patched fixtures of these fixture types can use the global presets that are only stored for one of these fixture types. ShareGlobal is set to Yes by default. ShareGlobal can not be changed in the Live Patch.
- **Version:**
  This displays the grandMA3 version number with which the fixture type can be used. The running grandMA3 version must be the same as or higher than the fixture type's version number.
- **XYZ:**
  XYZ can be enabled for a fixture type that supports this feature. **Yes** indicates that every fixture type mode has XYZ enabled. Partial indicates that at least one mode of the fixture type has XYZ enabled. XYZ is set to **No** per default. To enable XYZ, see **Activating XYZ for Fixture Types**.

Select and Import Fixture Types

Requirements: Steps 1 and 2 from the chapter above.

1. To import fixture types from the library, tap Import. The pop-up Select fixture type to import opens.
2. Select a fixture type and then tap Import in the bottom-right corner of the Select fixture type window. The fixture is imported to the fixture type list and can now be patched.



Select Fixture Type to Import From Library pop-up.

For a detailed description of Insert new Fixture window, see **Add fixtures to the show**.

Subtopics

- **Import GDTF**
- **Conflicts in Fixture Types**

1.53.1.1. Import GDTF

Import a GDTF file (General Device Type Format) containing a fixture type description. GDTF provides a fixture's digital footprint. Visit **https://gdtf-share.com** for more information.

The GDTF file is a zip file containing:

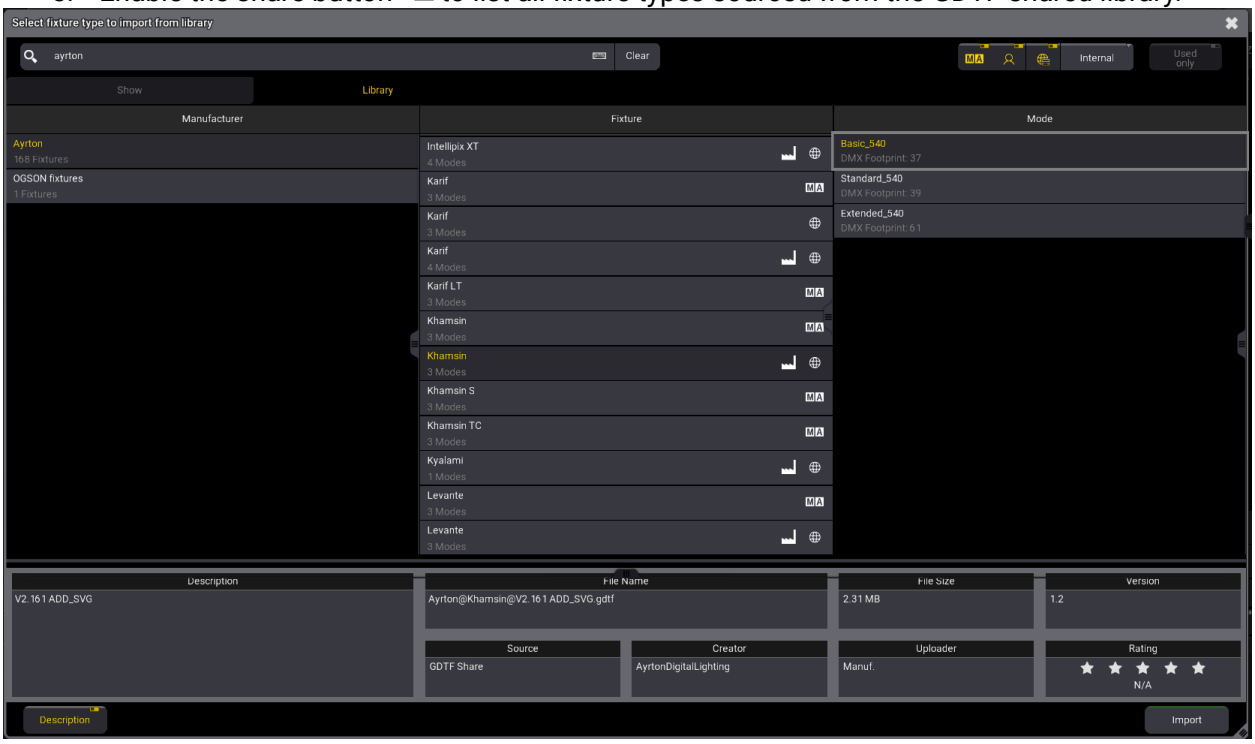
- Description
- Geometry data
- Gobo images

For a detailed description of GDTF, see the DIN specification 15800:2022 (**https://www.beuth.de/de/technische-regel/din-spec-15800/349717520**).

Import GDTF from the Console or Directly from the World Server

Access **grandMA3 fixture share** and **GDTF** libraries directly when there is an active connection to a World Server (for more information, see **World server**),

1. Open the patch menu by pressing Menu.
2. Tap Patch.
3. On the left side of the window, tap Fixture Types.
4. Tap Import at the bottom of the window. The Select Fixture Type to Import pop-up opens.
5. Enable the share button to list all fixture types sourced from the GDTF shared library.



Select Fixture to Import from Library pop-up

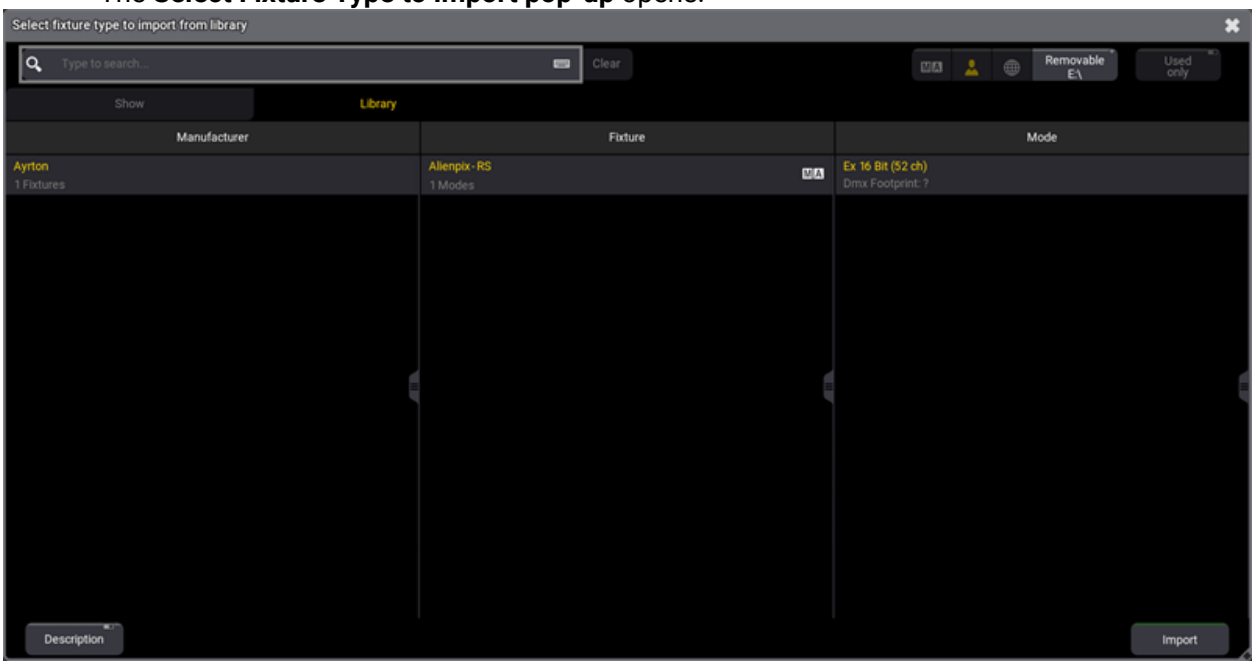
6. Select a fixture type and then tap Import. The fixture is imported to the fixture type list and can now be patched.

Enable Description for more information about the fixture type you want to import.

Import GDTF from a Flash Drive

Fixtures can be exported to a USB flash drive, and the fixtures can be imported to a show file even when there is no access to the World Server.

1. You can download the GDTF file on **https://gdtf-share.com/** or export it to a USB flash drive (and skip to point 4).
2. The file is located in the download folder.
3. Copy the GDTF file to a USB flash drive. The default path for GDTF fixture types is **grandMA3/gma3_library/fixturetypes.**
4. Insert the USB flash drive into the console.
5. Open the patch dialog and tap Fixture Types.
6. Tap Import.
   The **Select Fixture Type to Import pop-up** opens:



Select Fixture to Import from Library pop-up with USB drive selected

7. In the upper right corner, select the USB flash drive that contains the GDTF files.
8. Select the desired fixture type.
9. Tap Import. The fixture is imported to the fixture type list and can now be patched.

	Hint:
	The files must have a .gdtf or .GDTF extension to work.

For more information on how to patch, see **Add Fixtures to the Show**.

1.53.1.2. Conflicts in Fixture Types

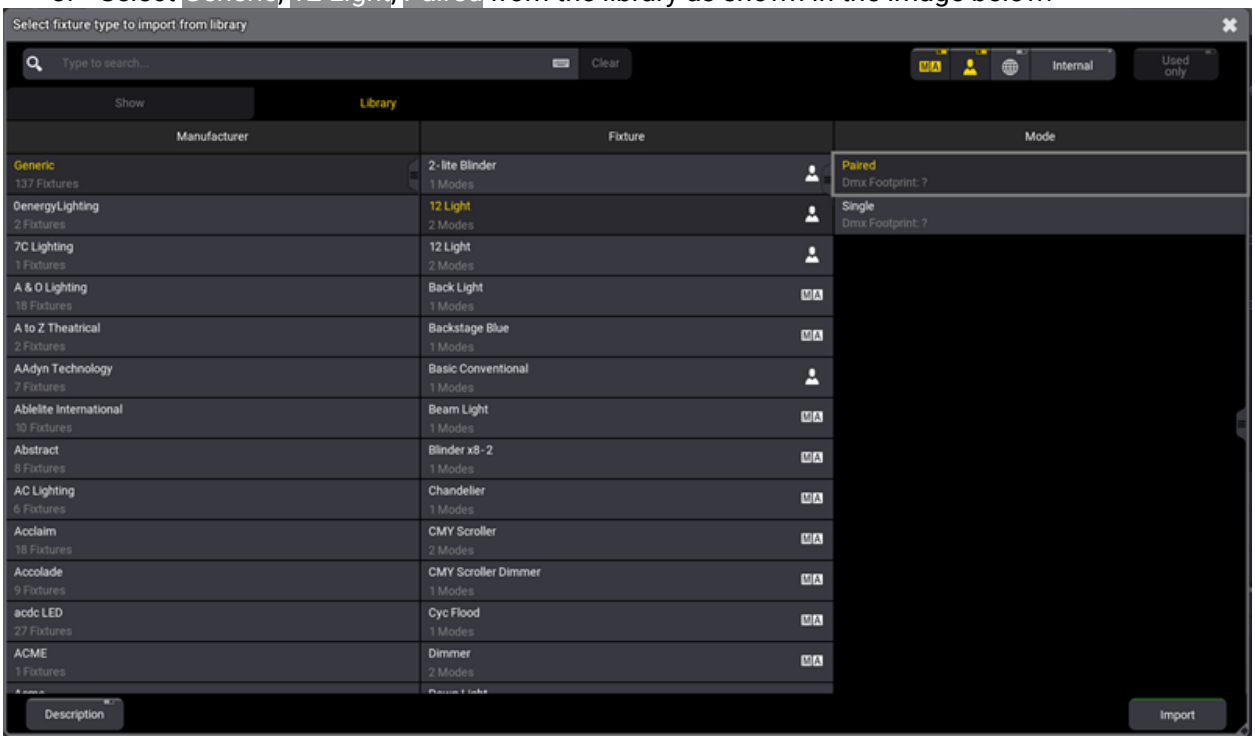
When importing or editing fixture types, it can happen that fixtures are displayed in a specific font color. The specific font color displays an existing conflict. The button Show Conflicts provides more information about this conflict. A conflict can be, for example, a DMX channel overlap!

Fixture Type with Orange Font Color

- When importing a fixture type from the library, a fixture is shown in an orange font color. This indicates a warning.
- Fixture types with an orange font color are accessible, but may not function properly.

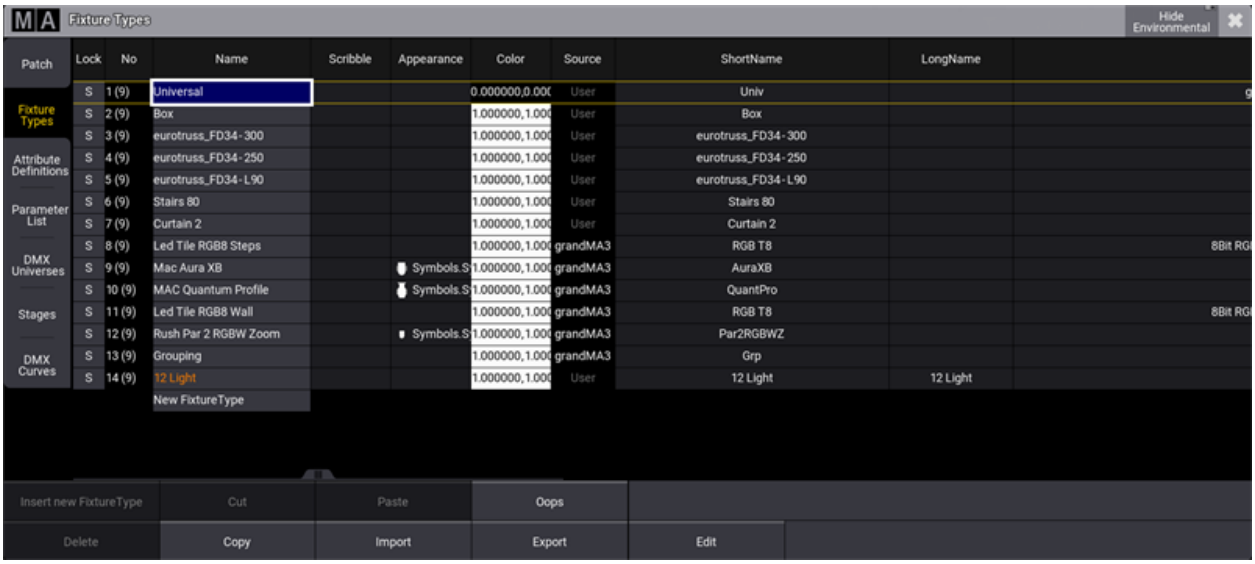
Example:

1. Open the Fixture Types menu, as shown in **Import fixture types**.
2. To select a fixture from the library, tap Import.
3. Select Generic, 12 Light, Paired from the library as shown in the image below:



A selection of a specific fixture type for a example of a warning

4. Tap Import, the select fixture type library closes. The selected fixture is displayed in an orange font color:



The example is displayed in the Fixture Types menu.

Fixture Type with Red Font Color

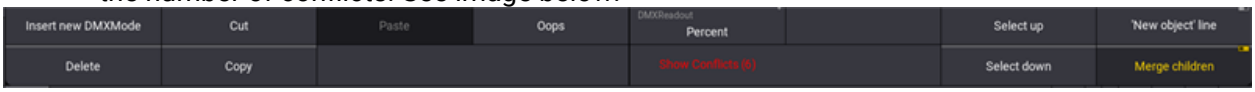
- Fixture types that have errors are displayed with a red font color in the Fixture Types-tab of the Patch menu.
- When a fixture type has errors, the corresponding fixtures are not displayed in the rest of the software, because they cannot function correctly.

Show Conflicts

Next to the font colors, it is also possible to get a more detailed description of the error / warning.

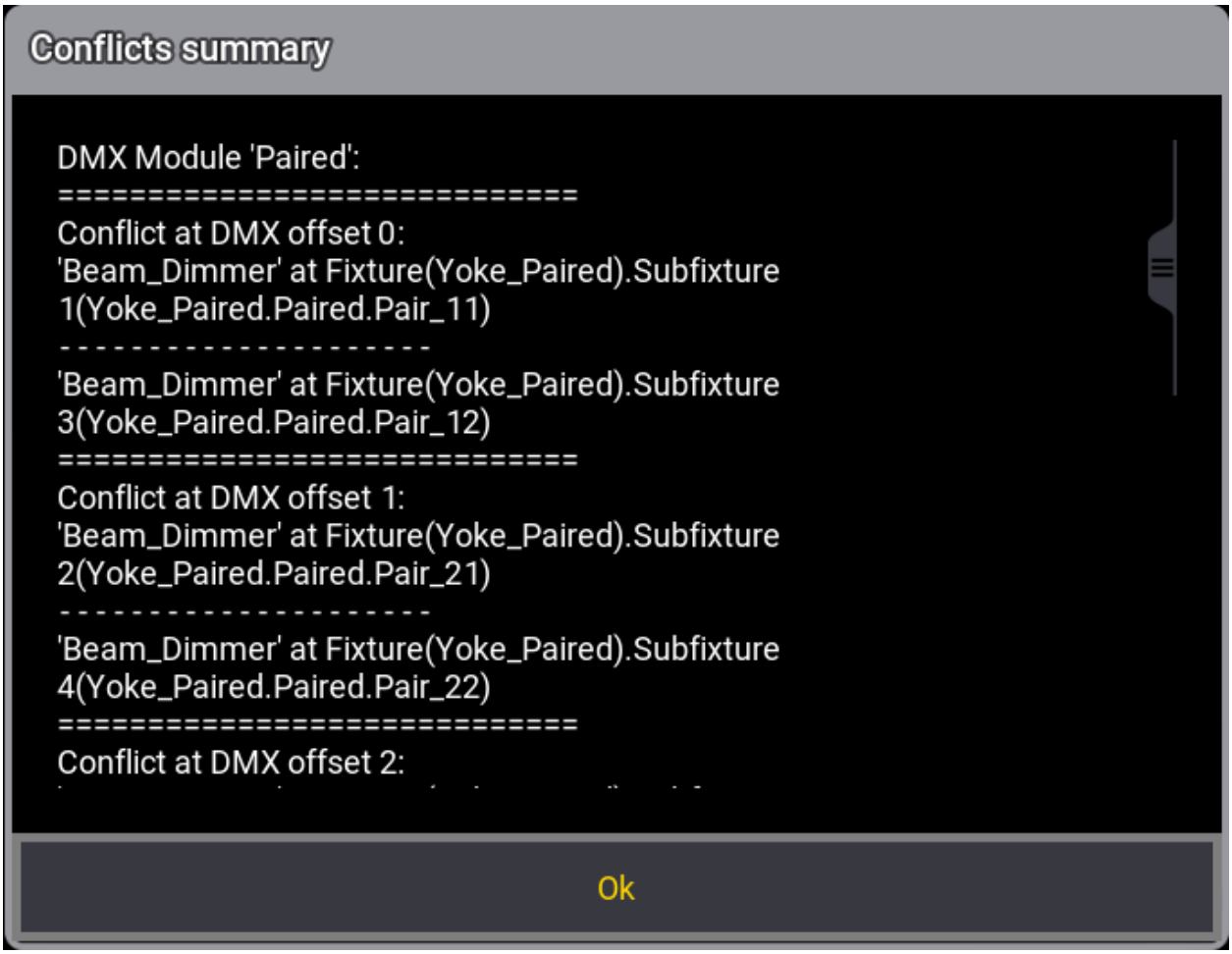
To get a more detailed summary of the error / warning:

1. Select the colored fixture type with a red or orange font color. To do this, go to the Fixture Types-tab in the Patch menu.
2. To open the editor, tap Edit. The Edit FixtureType Editor opens.
3. A button called Show Conflicts (x) is colored in a red font. The number in the parentheses shows the number of conflicts. See image below:



Show Conflicts button is only visible when an error / warning is emerging.

4. To open the Conflicts summary pop-up, tap Show Conflicts. The Conflicts summary pop-up opens:



Conflicts pop-up.

1.53.2. Build Fixture Types

grandMA3 allows building your own fixture types. Therefore, the grandMA3 software benefits from using a Fixture Type Editor. In this editor, it is possible to set up a custom-made fixture type using different inputs. For example:

- DMXModes
- Geometries
- PhysicalDescriptions
- Wheels

The following topics demonstrate how to use the Fixture Type Editor and provide a simple example of what a standard fixture type can look like.

	**Important:**
	Creating fixture types can be necessary to project real-life fixtures that have not been built yet.
	**Important:**
	Instead of creating a new fixture type it can be easier and faster to take an existing fixture type and adjust only the differences!
	**Hint:**
	It is also possible to build fixtures using the GDTF Builder and import them to grandMA3. For more information, see **Import GDTF**.

Subtopics

- **Insert Fixture Types**
- **Insert Geometries**
- **Insert Models**
- **Link Models to Geometries**
- **Insert DMX Modes and DMX Channels**
- **Environmental Fixture Types**

1.53.2.1. Insert Fixture Types

The following topics provide a step-by-step guide to building a fixture type of a basic moving head.

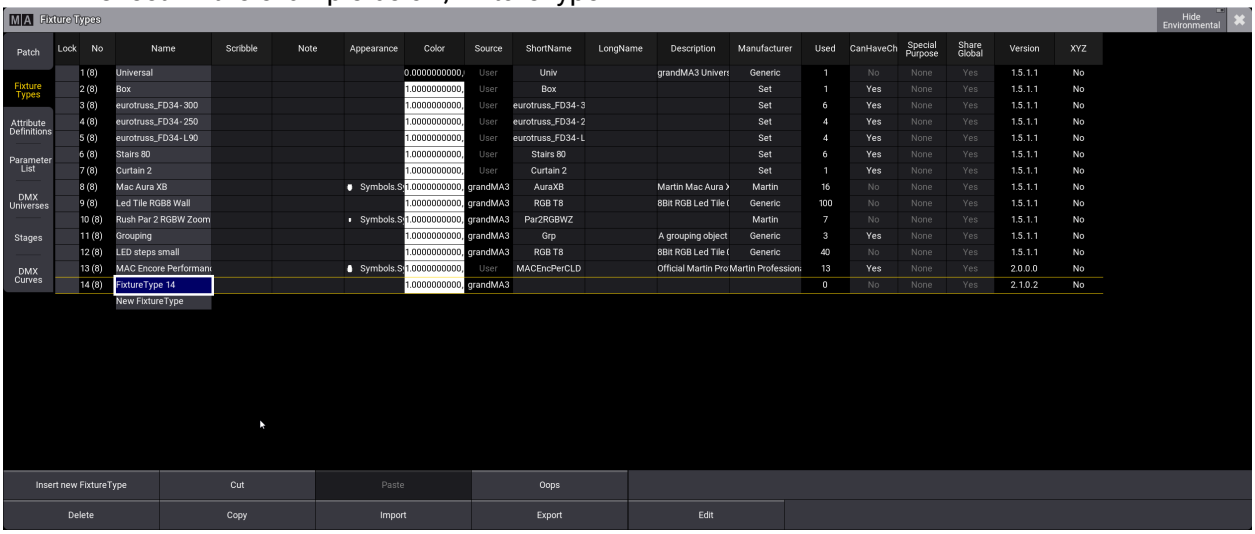
First of all, a fixture type has to be inserted.

Important:

Always make sure to save the settings you made; otherwise, they will get lost.
How to save settings in Fixture Types:
1. Leave the Patch.
2. The **pop-up Leaving the patch** appears.
3. Tap Ok.
4. Save the show file.
For more information, see **SaveShow Keyword**.

To back up data during or after the build of fixture types, leave the Fixture Type Editor and tap Export in the Fixture Types menu. An .xml file is exported. It is possible to import this file into a show file.

1. Open the Fixture Types tab in the Patch menu. To exit the Fixture Types tab, see **Import fixture types** topic.
2. In the **Name** column**,** tap New Fixture Type.
3. Tap Insert New Fixture Type in the bottom left corner. A new fixture-type row is inserted into the sheet. In the example below, FixtureType 14.



Fixture Types tab with the newly inserted FixtureType 14.

4. To edit or enter the basic data of the fixture type, fill out the following cells by right-clicking or tapping and holding the cell:
    - Name
    - Scribble
    - Appearance
    - ShortName

- Description
- Manufacturer

	**Hint:**
	To see a detailed description of every cell in the Fixture Types tab, see **Import Fixture Types** topic.

The next step is to create individual geometries for your fixture type: **Insert Geometries**.

1.53.2.2. Insert Geometries

The geometry of a fixture is the physical description of parts of the device.

This example of a basic moving head consists of four components:

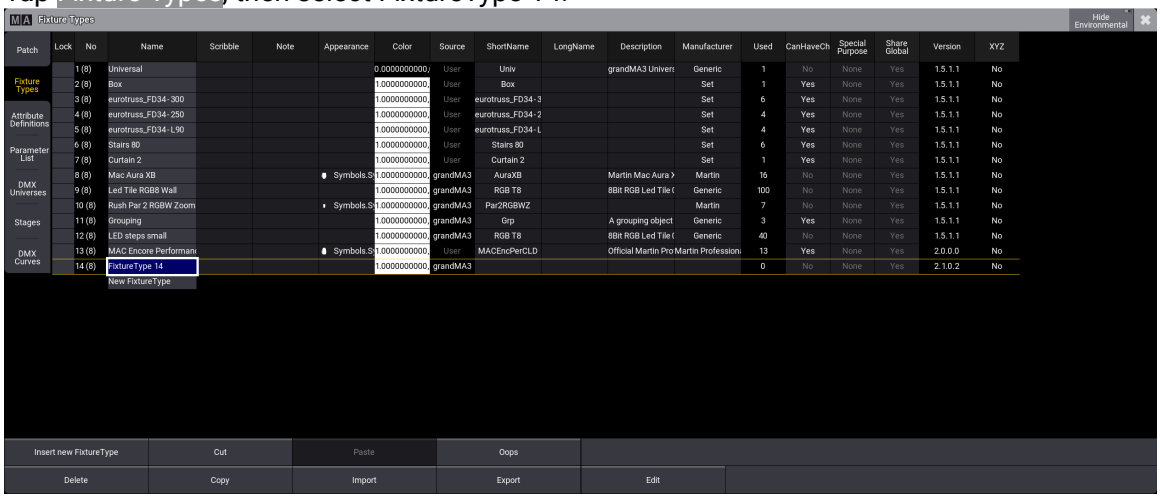
1. Base
2. Yoke
3. Head
4. Beam

All parts of the fixture need to be set up to link them to a specific DMX function later.

Requirement:

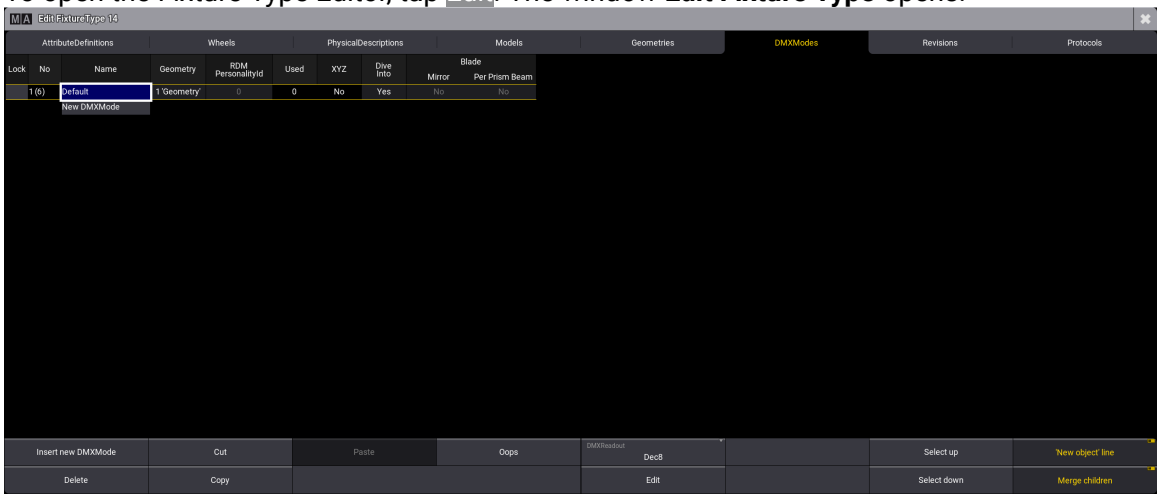
- **Insert Fixture Types**.

1. Tap Fixture Types, then select FixtureType 14.



Select the fixture type you want to edit

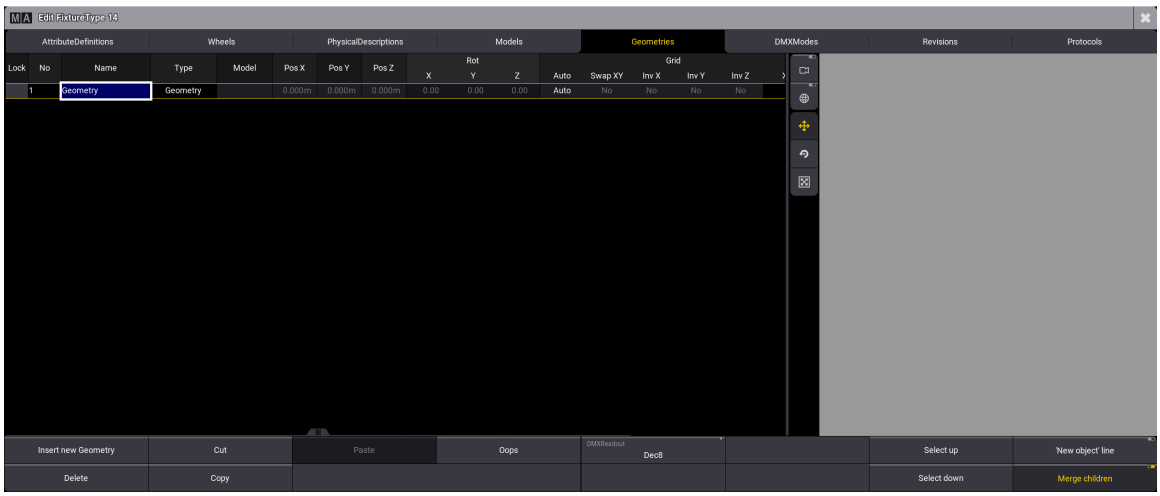
2. To open the Fixture Type Editor, tap Edit. The window **Edit Fixture Type** opens.



Fixture Type Editor

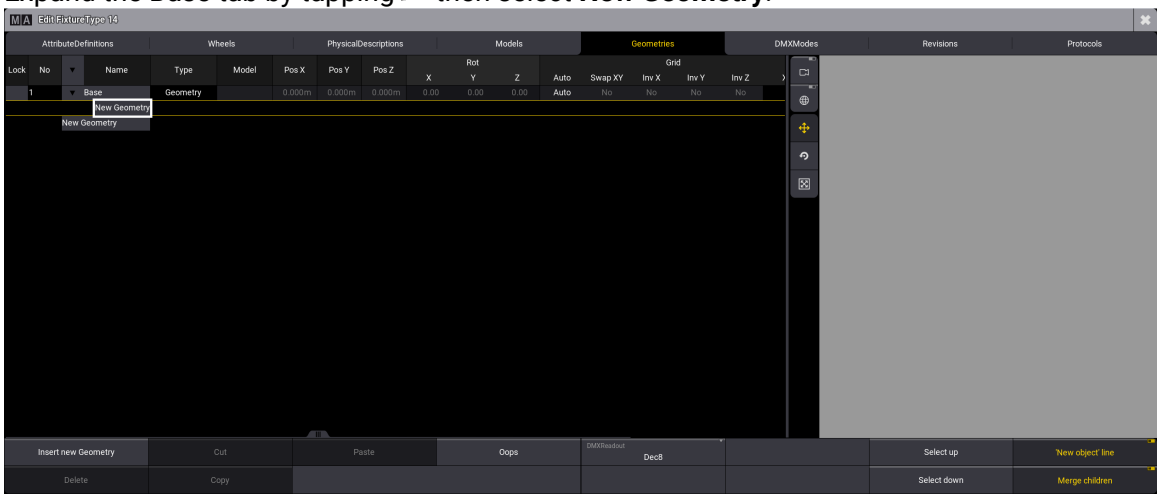
3. Tap Geometries.

The Geometries window opens.



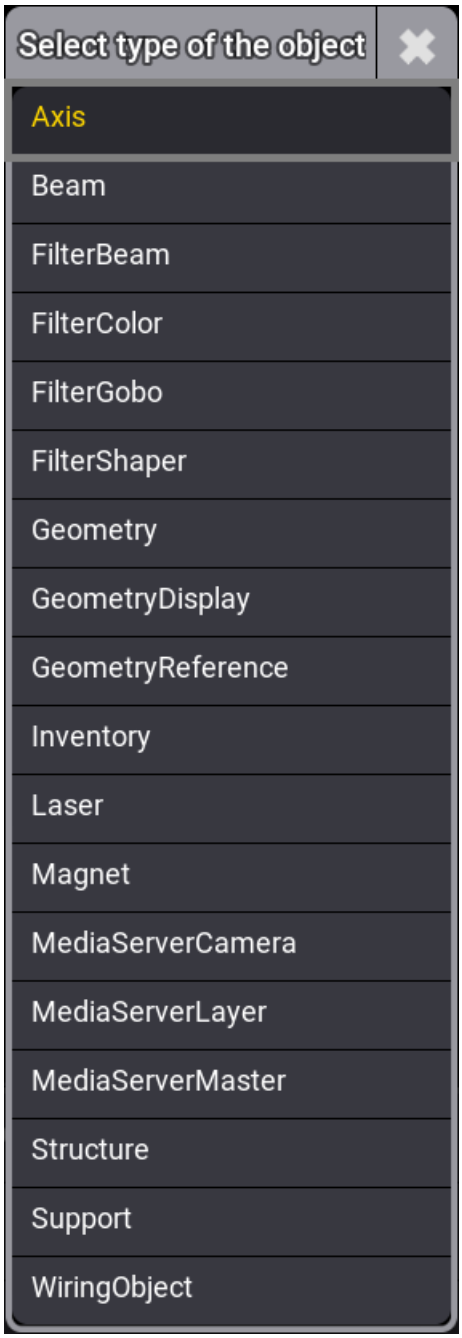
*FixtureType Editor Geometries window with the **Geometry Viewer** on the right side*

4. Under the Name column, tap and hold Geometry and rename it Base.
5. At the bottom right of the window, tap 'New object' line.
6. Expand the Base tab by tapping ▶ then select **New Geometry**.



7. Insert a new geometry
8. Tap Insert new Geometry.

The pop-up Select type of the object opens.



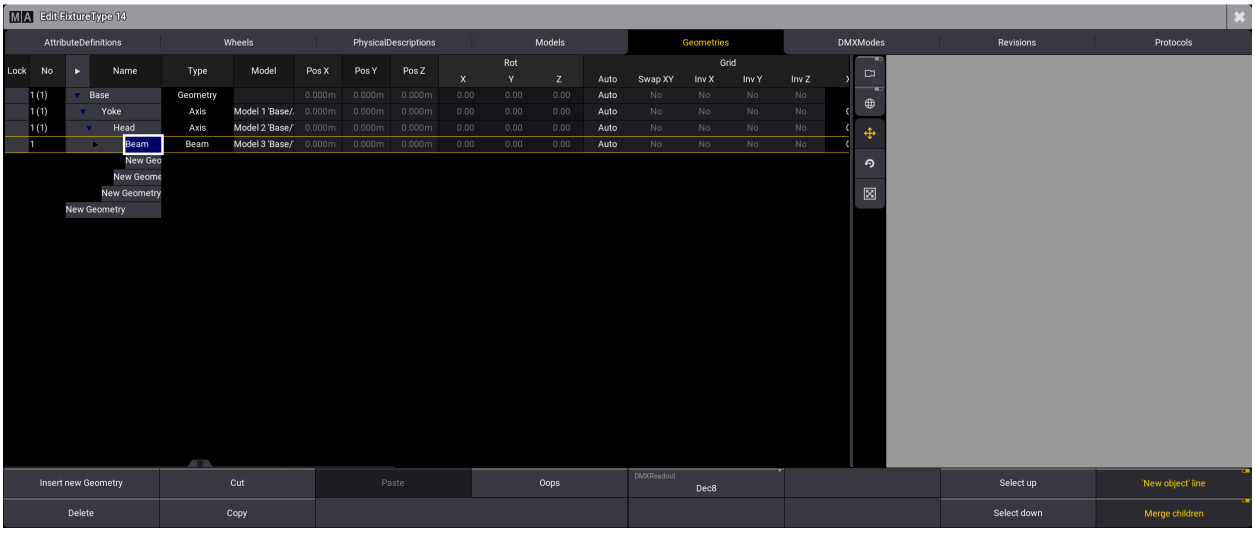
Select type

- Select **Axis**.
- Axis is now displayed in the column Type.

9. In the column Name, rename Axis to Yoke.
   - Press Edit and tap Axis, or press and hold Axis.
   - The virtual keyboard opens.
   - Enter Yoke.
10. Expand the cell Yoke.
    - Yoke's child New Geometry opens.
    - Repeat steps **6 to 8** and rename Axis to Head.
11. Expand the cell Head.
    - Head's child New Geometry opens.

- Repeat steps **6 to 8** selecting Beam.

Geometries are inserted, as shown in the image below:



Inserted geometry and its children

After inserting the fixture type's geometries, the next step is to create models as a physical foundation for the fixture's different parts: **Insert Models**.

1.53.2.3. Insert Models

Each geometry has a separate description of a model.

Every model needs a mesh. Meshes are 3D models that represent objects in a virtual environment.

A mesh is a .3ds file (the 3D image format used by Autodesk 3D Studio) or a .gltf/.glb file (the open standard 3D image format).

Meshes are stored in the Meshes pool after they have been imported to the show.

- For more information about importing meshes, see **Meshes**.

	Important:
	After they have been imported, meshes must be linked to a model of a fixture type.
	Hint:
	Meshes are automatically added to the mesh pool after importing an MVR file or when a fixture type is imported to the show. For more information, see **MVR**.

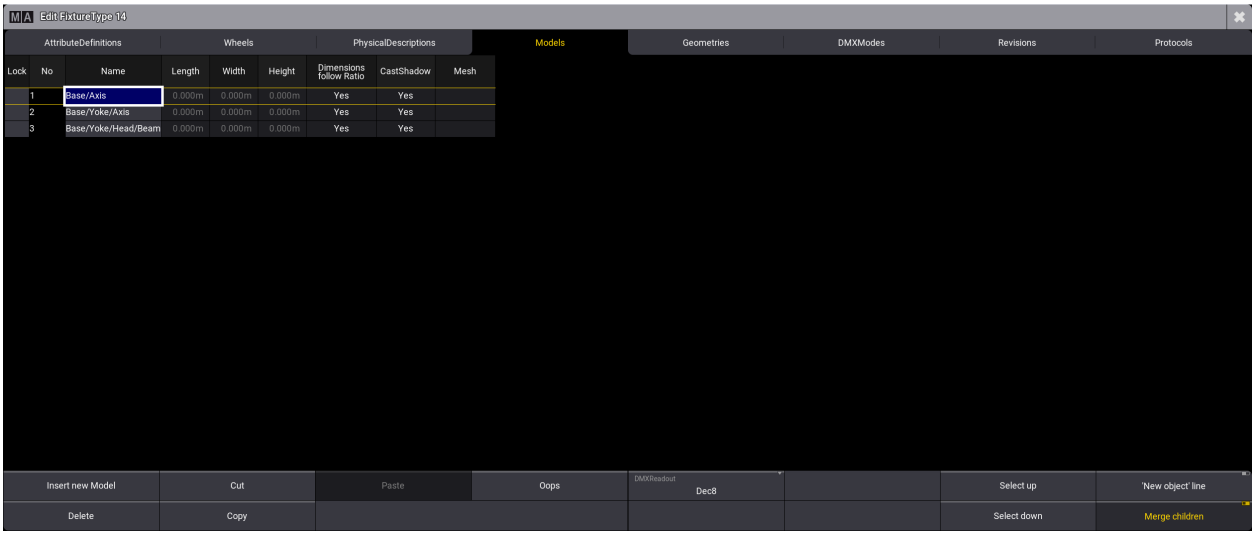
Requirement:

- **Insert Geometries**.

Models are built upon geometries.

1. Tap Models.

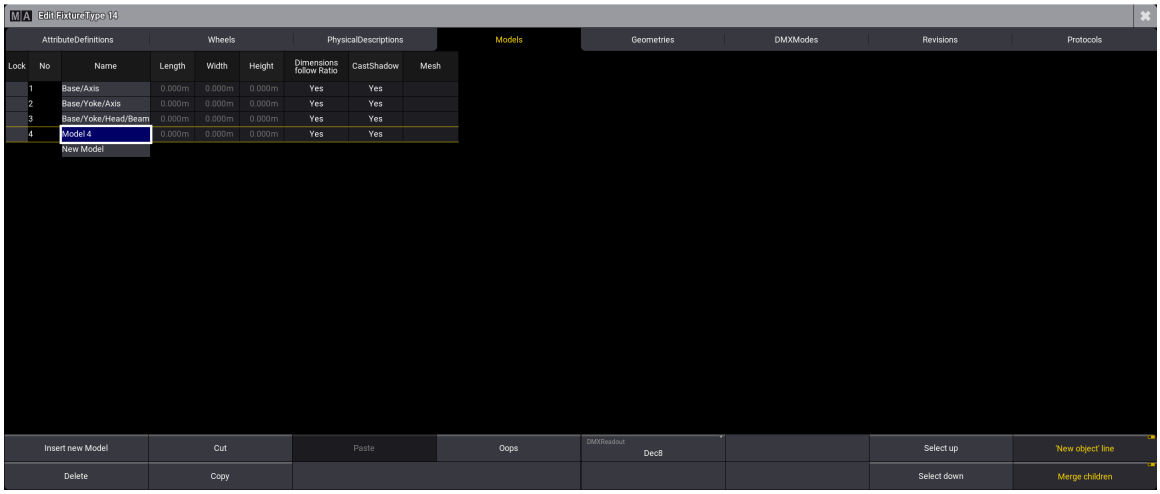
   The menu Models opens.



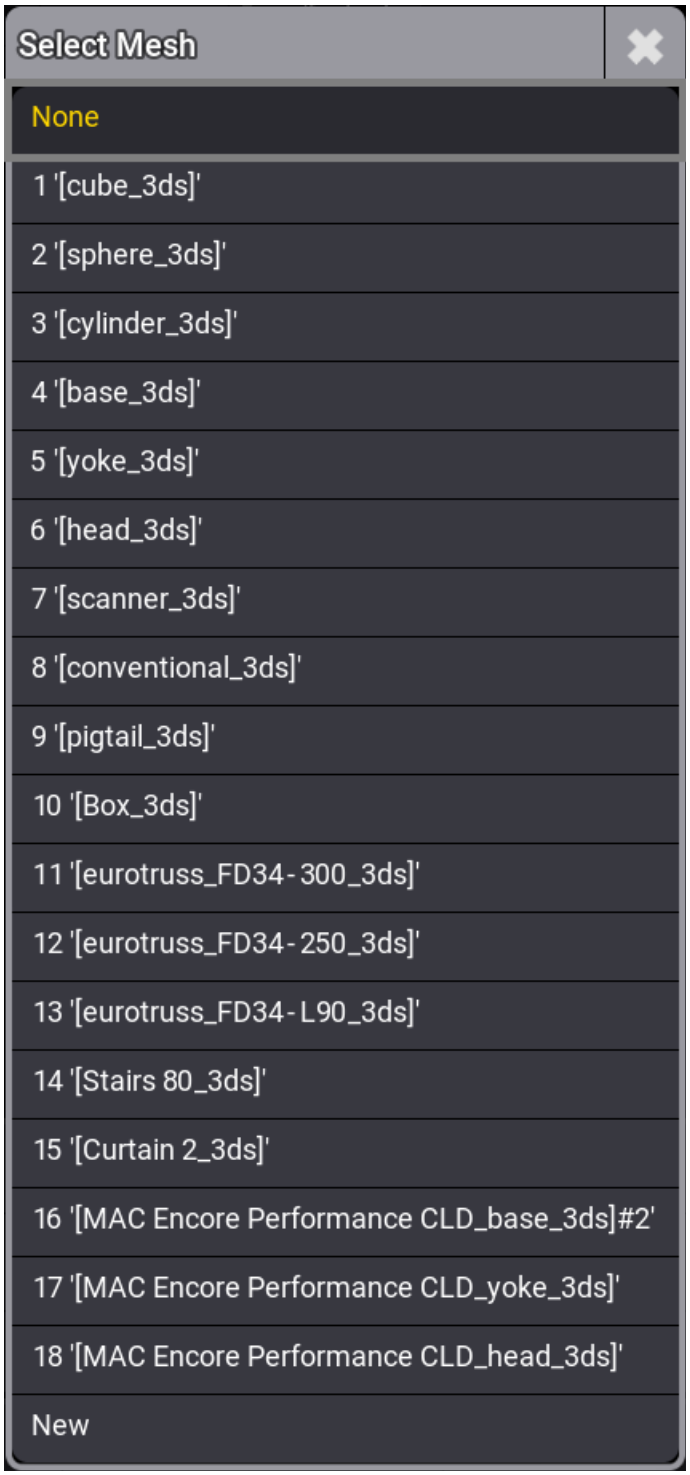
Open Models tab in the Fixture Types Editor

2. At the bottom right of the window, tap 'New object' line.

3. Select New Model and tap Insert new Model.
   - Model 4 is added.



   Insert Model 4 in the Models tab
4. Rename Model 4 into Base.
   - Press Edit and tap Model 4, or press and hold Model 4.
   - The virtual keyboard opens.
   - Enter Base.
5. Set Base's Mesh to [base_3ds].
   - Press Edit and tap the cell in the Mesh column or press and hold that cell.
   - The pop-up Select Mesh opens.



Select Mesh pop-up - Select [base_3ds].

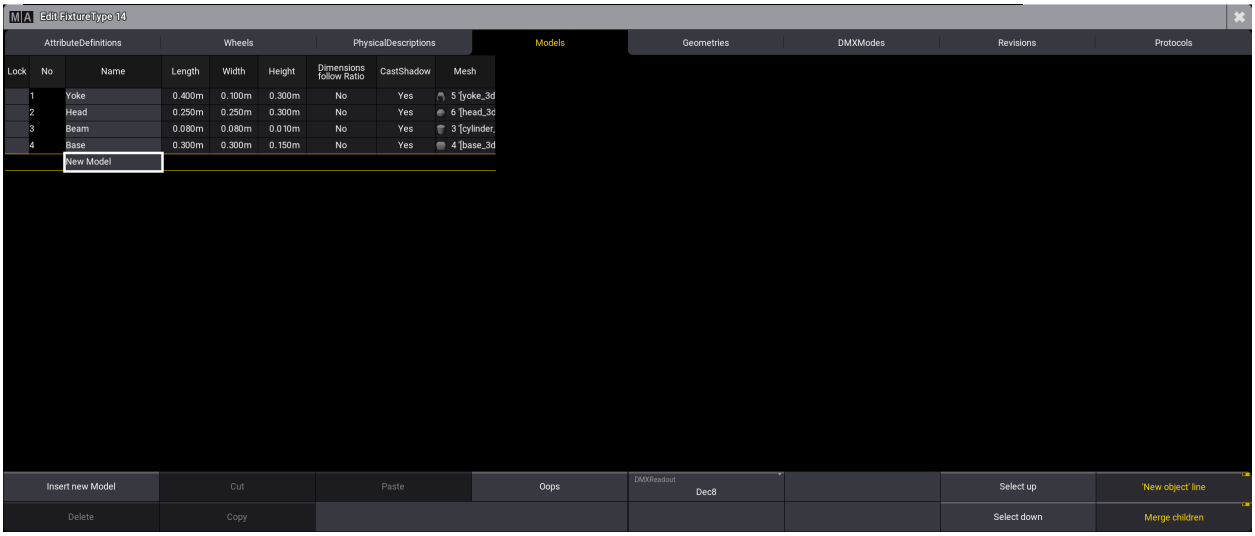
6. Rename Base/Axis into Yoke and set its Mesh to [yoke_3ds].
7. Rename Base/Yoke/Axis into Head and set its Mesh to [head_3ds].
8. Rename Base/Yoke/Head/Beam into Beam and set its Mesh to [cylinder_3ds].

   To represent the fixture correctly, Length, Width, and Height need to be set.
9. Enter the measurements of the fixture, as shown in the table below.

	Important:
	To prevent the models from automatically resizing themselves, set the **Dimensions follow Ratio** column to **No** before entering the

	measurements.

Name	Length	Width	Height
Yoke	0.4 m	0.1 m	0.3 m
Head	0.25 m	0.25 m	0.3 m
Beam	0.08 m	0.08 m	0.01 m
Base	0.3 m	0.3 m	0.150 m



Inserted models and measurements

The models are now automatically assigned to the geometries in the column Model, except for Base, which was inserted directly into the models.

The next step is to connect the models and geometries you made so they are displayed correctly: **Link Models to Geometries**.

1.53.2.4. Link Models to Geometries

To correctly display the models in the 3D Viewer, the next thing to do is to link models to geometries.

Requirement:

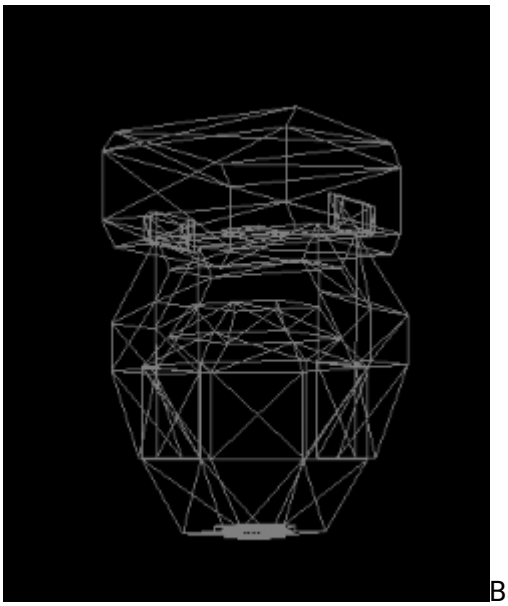
- **Insert Geometries** and **Insert Models**.

1. Tap the tab Geometries.
2. To link the model Base to geometries, tap, and hold the cell in the column Model.
   - Pop-up Select Model opens.
   - Select Base.
3. Set the offset of position in single geometries.
   - In Base, leave the default value of Pos Z.
   - In Yoke, set Pos Z to -0.265 m.
   - In Head, set Pos Z to -0.100 m.
   - In Beam, set Pos Z to -0.150 m.

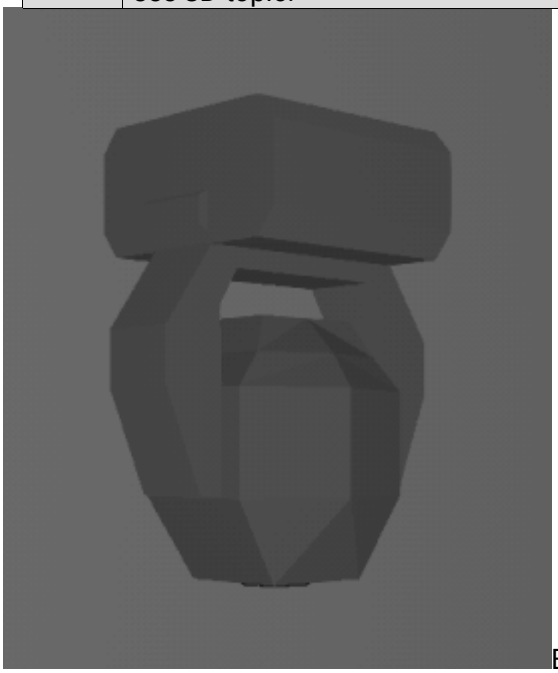
	Hint:
	The offset of the position depends on the measurements of the models.
	Hint:
	These cells can also be edited using the **Geometry Viewer**.

Set the offset of Pos Z

This is the result in the 3D window.


Basic moving head displayed in wireframe in the 3D window

	Hint:
	To display the basic moving head in wireframe in the 3D window, see **3D** topic.


Basic moving head rendered in the 3D window

Geometry Viewer

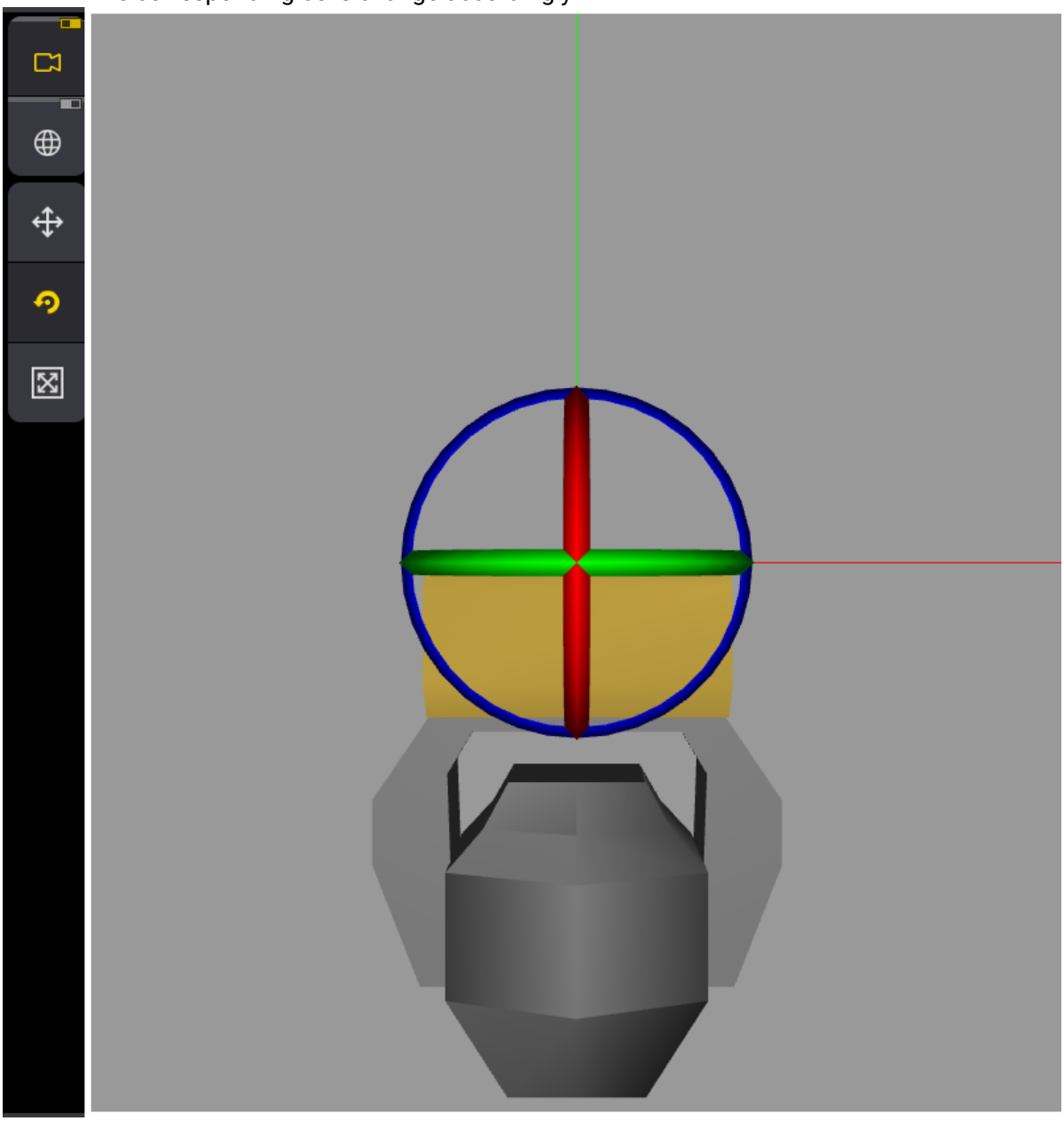
The geometry viewer on the right side of the window displays a real time 3D visualization of the selected geometry tree.

It allows you to view and to edit the geometries of the fixtures.

	Hint:
	To display the geometries in the geometry viewer, add meshes in the models tab first. See **Insert Models**.

To edit the geometries:

1. Tap on the part of the fixture you want to edit or select a geometry in the table.
   This part is now selected and is displayed with a yellow color in the 3D area.
2. Tap and hold one of the colored indicators to move, scale or rotate the geometries.
   The corresponding cells change accordingly.



Geometry Editor with the selected geometry Base in rotation mode and camera focus enabled

(Camera focus):
If this button is enabled, the selected geometry is displayed in the center of the 3D area.
If this button is disabled, the center of the whole fixture is pinned to the center of the 3D area.

(World model transformation):
If this button is enabled, the axis follows the world.
If this button is disabled, the axis follows the selected geometry.

(Translation mode):
Lets you move the selected geometry.
Tap and hold the arrow of the selected axis and move it. When the geometry is in the desired position, release your finger.

(Rotation mode):
Lets you rotate the selected geometry.
Tap and hold one of the colored rings and move it. When the geometry is in the desired position, release your finger.
While you rotate the geometry along its own axis, the degree value of the rotation is displayed in the top left corner of the 3D area. Additionally, a colored disc shows a visual representation of the angle by which the geometry is rotated.

(Scale mode):
Lets you scale the selected geometry.
Tap and hold one of the axis indicators and move it. When the geometry has the desired size, release your finger.
To scale all three axes at the same time, tap and move the block in the center of the axis indicators.

	Hint:
	Scaling the geometries changes the values for length, width, and hight in the Models tab.
	Hint:
	If you scale a geometry whose model is also linked to other geometries, those geometries are scaled as well.

The last step of building a basic moving head, is inserting DMX modes and channels and linking them to specific geometries: **Insert DMX Modes**.

This is important for fully and accurately managing your fixture type.

1.53.2.5. Insert DMX Modes and DMX Channels

The next step is to insert DMX modes and DMX channels and linking them to the geometries.

Specific functions and attributes can only be executed by specific physical parts of a fixture. The pan movement for example is executed by the yoke of the basic moving head.

Therefore you have to link DMX channels with their specific functions and attributes to the corresponding geometrical parts of the fixture type.

Requirement:

- **Insert Geometries**

Description of DMX modes:

- A DMX mode consists of one or several DMX channels.
- The DMX channel defines different attributes and functions of the fixture.
- The DMX mode is the parent, and the DMX channel is its child.

	**Important:**
	The build of a fixture type is based on the hierarchic structure of parent-child. A parent comes first, and the child follows its parent. There may be several children.
	**Hint:**
	Many manufacturers provide DMX charts that define channels. It is possible to download the DMX charts for a specific fixture type from the manufacturer's website in most cases. Use a DMX chart to enter the channels in the fixture type table.

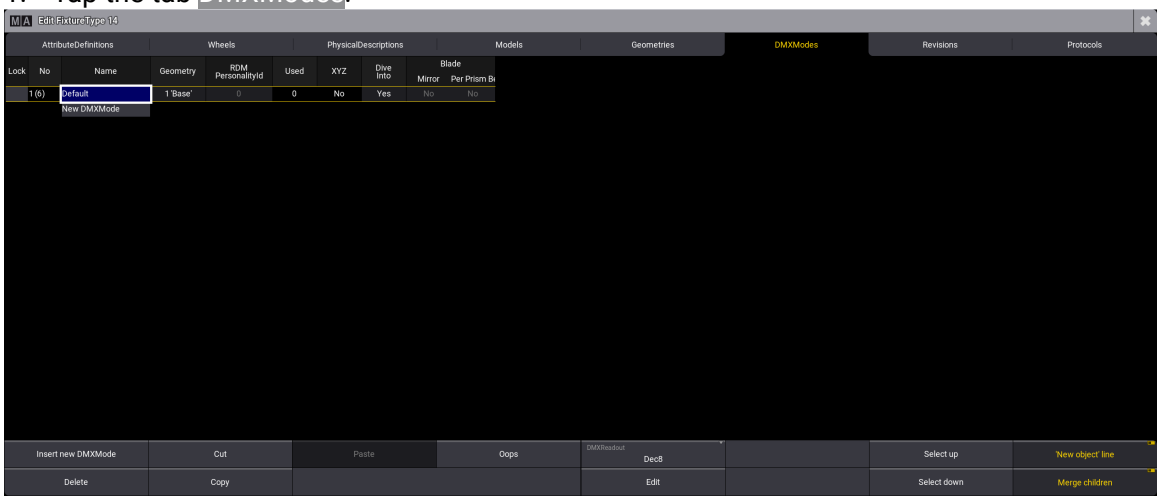
Preparation

This basic moving head is based on the following DMX chart, showing a relative patch address and the related function:

Relative Patch Address	Function
1	Pan (8 bit) Pan movement by 540°
2	Pan Fine (16 bit)
3	Tilt (8 bit) Tilt movement by 270°
4	Tilt Fine (16 bit)
5	Dimmer
6	Red

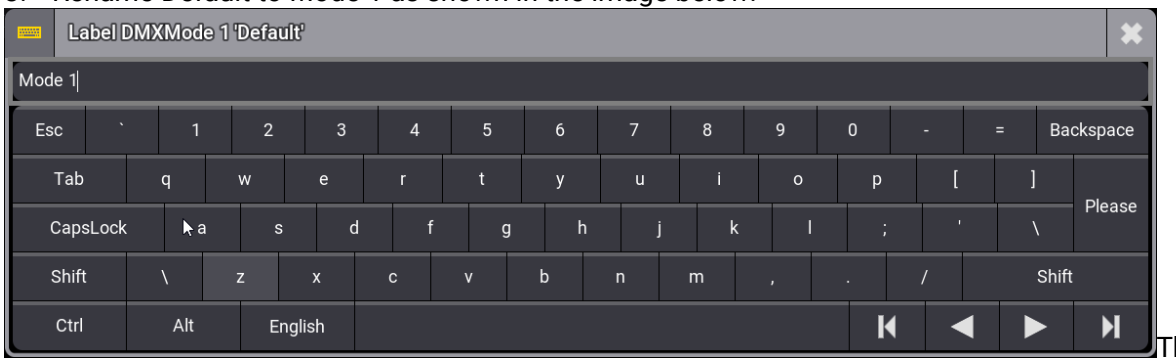
Relative Patch Address	Function
7	Green
8	Blue

1. Tap the tab DMXModes.



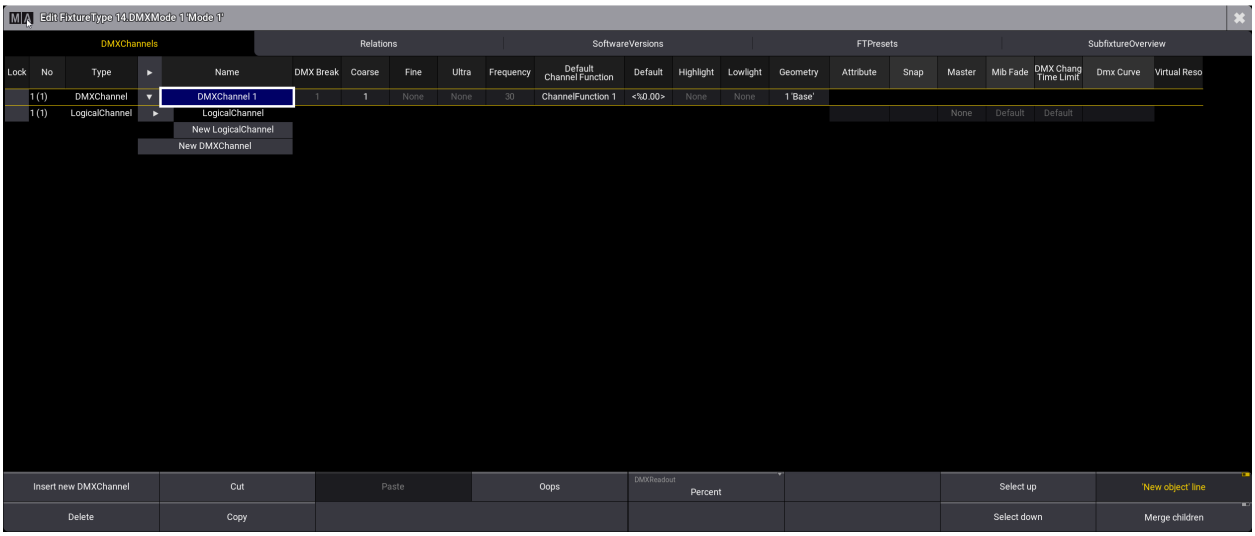
The editor for building a fixture

2. To change the name of the cell, tap and hold or right-click the cell labeled Default. A text editor opens.
3. Rename Default to Mode 1 as shown in the image below:



This is the text editor

4. To insert DMX channels, select Mode 1 and tab Edit again. The DMX Channel Editor with a tree structure opens:



DMXMode Editor with one DMX channel

The most important columns in this editor are:

- **Coarse**, **Fine**, **Ultra**, **Default**, **Highlight**:
  See **Parameter List** topic.
- **Attribute:**
  This displays the functionality of the DMX channel.
- **Master:**
  This defines if the value of the channel is affected by a group master or the grand master.
- **Physical From:**
  This displays the physical value of a fixture. Physical From can not be changed in the Live Patch.
- **Physical To:**
  This displays the physical value of a fixture. Physical To can not be changed in the Live Patch.
- **DMX From:**
  Sets the DMX value from the lower end of the DMX range on the channel function. DMX From can not be changed in the Live Patch.
- **DMX To:**
  DMX To in the channel function is automatically calculated depending on the DMX From value of the following channel function or the end of the DMX channel.

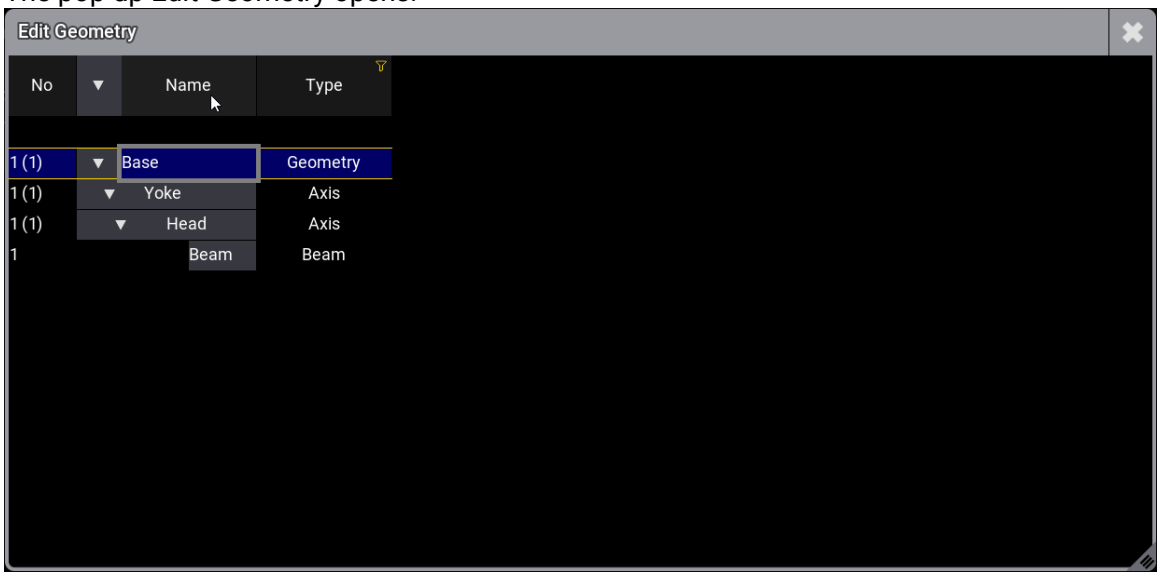
The example of a basic moving head requires 6 DMX channels for all of its different Attributes (Pan; Tilt; Dimmer; Red; Green; Blue)

Enter DMX Channel 1 – Pan

	Hint:
	For the following example, it makes sense to change the DMXReadout from Dec8 to Percent, as there are only 50 % and 100 % values in this example. Since it is more accurate, for other cases it can make more sense to use Dec8 or Dec16.

To enter the first channel:

1. In the row DMXChannel 1, enter:
   - **Coarse**: 1
   - **Fine**: 2
   - **Default**: 50 %
   - **Highlight**: none
2. Edit the column Geometry
   In the row DMXChannel 1, tap and hold or press Edit and tap the cell.
   The pop-up Edit Geometry opens.



   Edit Geometry pop-up Select geometry Yoke.
3. Make sure the Merge children button is disabled.
4. Expand DMXChannel 1 for further entries in LogicalChannel:
   - Tap ▶ in the row DMXChannel 1.
5. In the row LogicalChannel select:
   - **Attribute**: Pan
6. The DMXChannel 1 is renamed to Yoke_Pan.
   - **Master**: none
7. Expand LogicalChannel, now called Pan, for further entries in ChannelFunction:
   - Tap ▶ in the row LogicalChannel.
8. In the row ChannelFunction select:
   - **Attribute**: Pan
   - **Physical From**: -270
   - **Physical To**: 270

	Hint:
	As soon as you select an attribute, the DMX channel is renamed after the attribute: **Geometry_Attribute**.

DMX channel 1 is entered.

Enter DMX Channel 2 – Tilt

To enter the second channel:

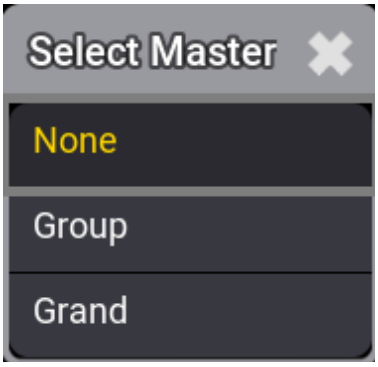
1. Tap Yoke_Pan.
2. To enable NewDMXChannel, tap 'New object' line button**.**
3. Tap New DMXChannel and tap Insert new DMXChannel.
   New DMXChannel is renamed to DMXChannel 2.
4. To enter the second channel, enter in the row DMXChannel 2:
   - **Coarse**: 3
   - **Fine**: 4
   - **Default**: 50 %
   - **Highlight**: none
5. Edit the column Geometry and select Head.
6. Open LogicalChannel. For more information, see **step 4** in description for the first DMX channel.
7. - **Attribute**: Tilt
8. DMXChannel 2 is renamed after the attribute.
   - **Master**: none
9. Open ChannelFunction. For more information, see **step 7** in description for the first DMX channel.
   - **Attribute**: Tilt
   - **Physical From**: -135
   - **Physical To**: 135

DMXChannel 2 is entered.

Enter DMX Channel 3 – Dimmer

To enter the third channel:

1. Tap New DMXChannel and tap Insert new DMXChannel.
   New DMXChannel is renamed to DMXChannel 3.
2. To enter the third channel, enter in the row DMX Channel 3:
   - **Coarse**: 5
   - **Default**: 0 %
   - **Highlight**: 100 %
3. Edit the column Geometry and select Beam.
4. Open LogicalChannel. For more information see **step 4** in description for the first DMX channel.
5. - **Attribute**: Dimmer
6. DMXChannel 3 is renamed after the attribute.
7. To control the value of the attribute using the Grand Master, tap in the column **Master.**
   - The **pop-up Select Master** opens



Select Master pop-up- Tap Grand.

8. Open ChannelFunction. For more information, see **step 7** in description for the first DMX channel.
   - **Attribute**: Dimmer
   - **Physical From**: 0
   - **Physical To**: 1

DMXChannel 3 is entered.

Enter DMX Channel 4 – Red

1. Tap New DMXChannel and tap Insert new DMXChannel.
   New DMXChannel is renamed to DMXChannel 4.
2. To enter the fourth channel, enter in the row DMXChannel 4:
   - **Coarse**: 6
   - **Default**: 100 %
   - **Highlight**: 100 %
3. Edit the column Geometry and select Beam.
4. Open LogicalChannel. For more information, see **step 4** in the type LogicalChannel.
5. - **Attribute**: ColorRGB_R
6. DMXChannel 4 is renamed after the attribute.
7. **Master**: none
8. Open ChannelFunction. For more information see **step 7** in the type ChannelFunction.
   - **Attribute**: ColorRGB_R
   - **Physical From**: 0
   - **Physical To**: 1

DMXChannel 4 is entered.

Enter DMX Channel 5 – Green

1. Tap New DMXChannel and tap Insert new DMXChannel.
   New DMXChannel is renamed to DMXChannel 5.

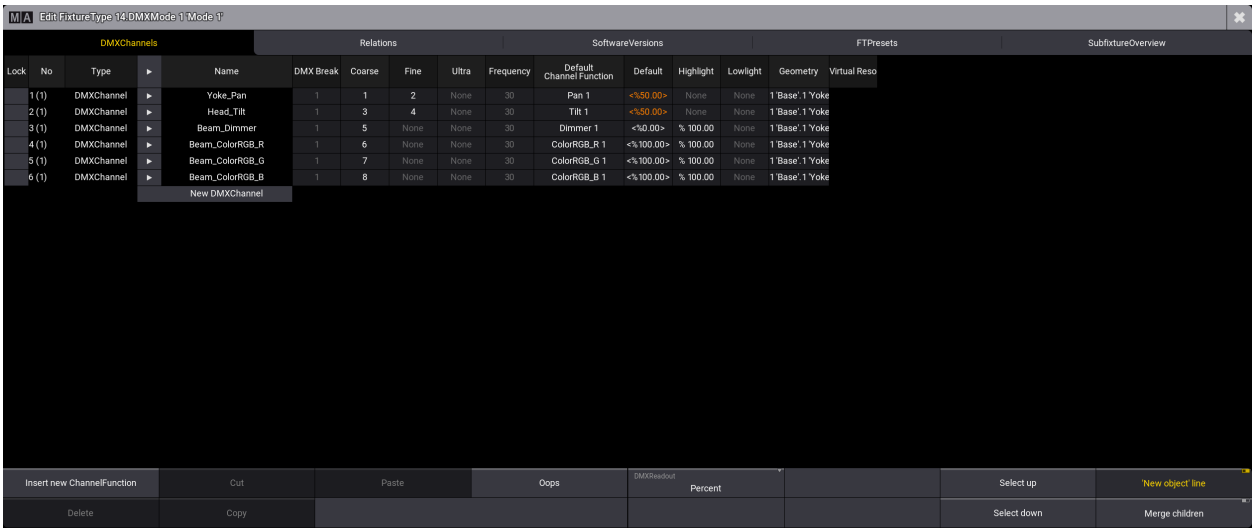
2. To enter the fifth channel, enter in the row DMXChannel 5:
   - **Coarse**: 7
   - **Default**: 100 %
   - **Highlight**: 100 %
3. Edit the column Geometry and select Beam.
4. Open LogicalChannel. For more information, see **step 4** in the type LogicalChannel.
5. - **Attribute**: ColorRGB_G
6. DMXChannel 5 is renamed after the attribute.
7. **Master**: none
8. Open ChannelFunction. For more information see **step 7** in the type ChannelFunction.
   - **Attribute**: ColorRGB_G
   - **Physical From**: 0
   - **Physical To**: 1

DMXChannel 5 is entered.

Enter DMX Channel 6 – Blue

1. Start with steps **1 to 3**, as described in DMXChannel 2.
2. To enter the sixth channel, enter in the row DMXChannel 6:
3. - **Coarse**: 8
   - **Default**: 100 %
   - **Highlight**: 100 %
4. Edit the column Geometry and select Beam.
5. Open LogicalChannel. For more information, see **step 4** in the type LogicalChannel.
6. - **Attribute**: ColorRGB_B
7. DMXChannel 6 is renamed after the attribute.
8. **Master**: none
9. Open ChannelFunction. For more information, see **step 7** in the type ChannelFunction.
   - **Attribute**: ColorRGB_B
   - **Physical From**: 0
   - **Physical To**: 1

DMX channel 6 is entered, and DMX Mode 1 is inserted. The DMX Mode Editor should look like this:



DMXMode Editor with six Channels: Pan; Tilt; Dimmer; Red; Green; Blue

Now all attributes of the basic moving head are covered with these six channels.

The example of building a basic moving head with all its attributes (Pan; Tilt; Dimmer; ColorRGB) and geometries (Base; Yoke; Head; Beam) is now completed.

	**Hint:**
	The **SubfixtureOverview** tab on the right side shows a preview of all attributes for every subfixture. With this tool the user can see if all attributes of a fixture type are covered.
	**Hint:**
	Check whether your fixture type is displayed and working correctly. To do this open the **Fixture Sheet** and **3D Viewer** and try out all the implemented functions.

1.53.2.6. Environmental Fixture Types

A way to make a stage more realistic is to use environmental fixture types. These fixture types are anything other than lighting fixtures, such as stages, trusses, seating, or scenery.

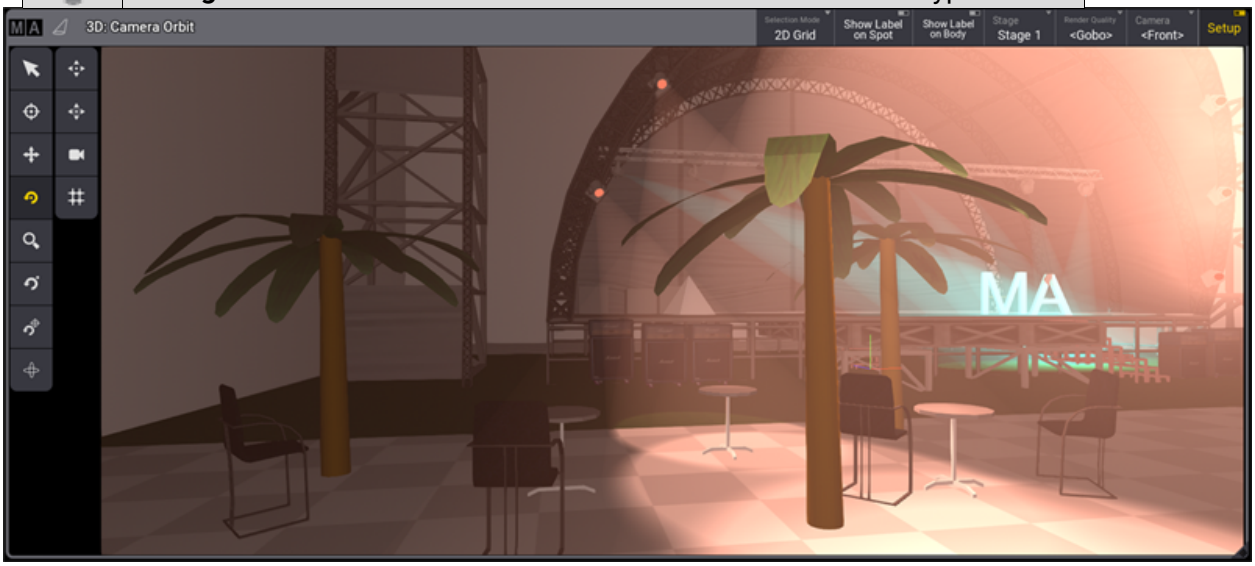
Environmental fixture types are part of the fixture library in the Patch menu. The environmental fixture types can be found in the "Set" manufacturer folder of the Insert New Fixtures overlay.

For more information on how to add new fixtures, see **Add fixtures to the show**.

Environmental fixtures do not have DMX addresses.

The Hide Environmental button hides all fixture types that do not have at least one DMX address assigned to a DMX channel. This button can be found in the Patch menu, the Live Patch menu, the Fixture Type section, and the Insert New Fixture dialog.

	Hint:
	www.gdtf-share.com contains even more environmental fixture types.



A stage with environmental fixtures is displayed in the 3D window.

1.53.3. Export Fixture Types

It is possible to export the fixture types to the following locations:

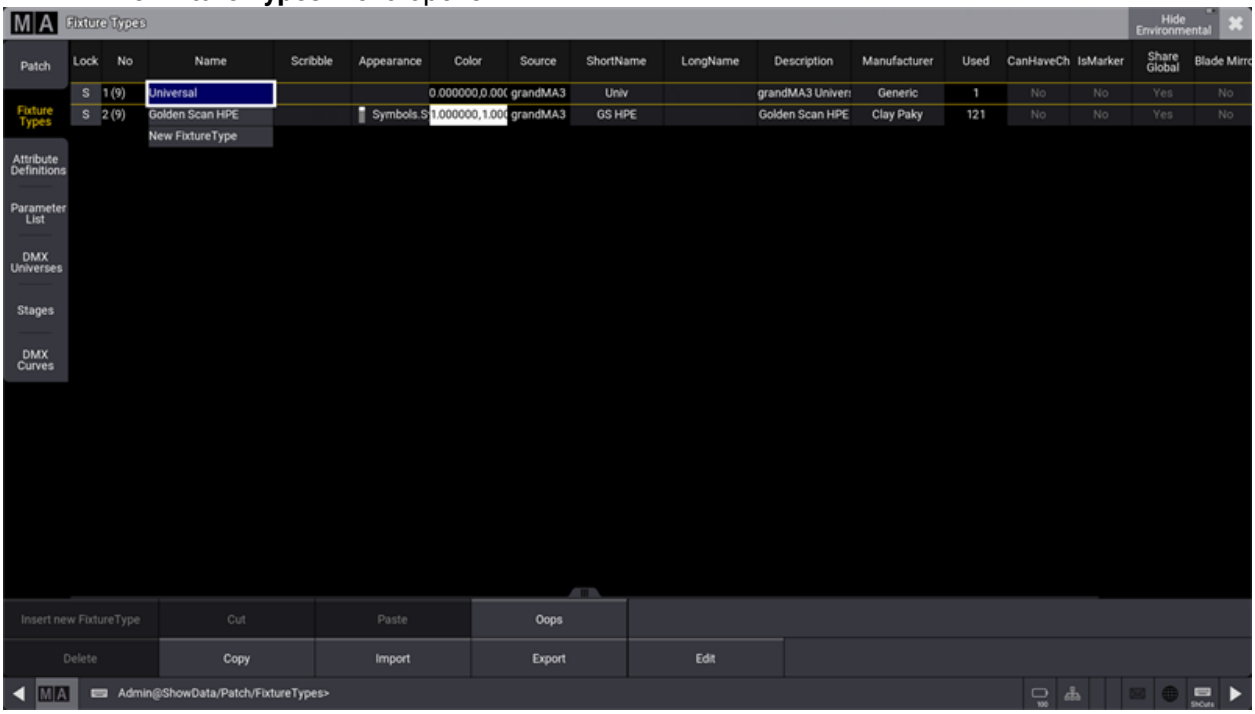
- Internal drive
- External USB drive

It is possible to export the fixture types as the following file formats:

- grandMA3 format
- GDTF

Requirement: Open the Patch.

1. Tap Fixture Types in the bar on the left of the patch dialog.
   The **Fixture Types** menu opens.



Open Fixture Types

2. Tap at the to be exported fixture type in the list.
3. Tap Export.
   The **Export Fixture Type** pop-up opens.

Export a fixture type

4. To select the destination drive, tap at Internal in the title bar.
   If an USB stick is inserted, the USB stick can be chosen.
5. To export the fixture type as grandMA3 format for only grandMA3 use case, tap at User.
   To export the fixture type as GDTF to use the fixture type within different programs that can read GDTF, tap at GDTF.
6. Enter a name.

	Hint:
	If no name is entered, the fixture type file will be exported as manufacturer@name.xml or manufacturer@name.gdtf.

7. Tap Export.

The selected fixture type is exported to the selected drive.

When the selected drive is internal, the fixture type is exported to the computer's local drive **(C:)\ProgramData\MALightingTechnology\gma3_library\fixturetypes**.

When the selected drive is external, the fixture type is exported to the external USB Drive **(E:)\grandMA3\library\fixturetypes.**

1.54. File Management

In addition to saving and loading complete show files, the grandMA3 software supports the exchange and portability of smaller portions of show data; for example, a collection of macros or even a single preset.

To maintain a predictable organizational structure, the software uses well-defined sets of folders within the hard drive of the console, the hard drive of a computer running grandMA3 onPC, and any USB drive connected to either a console or an onPC station.

The following topics cover data transfer between a client and the console, as well as the default folder structure created and utilized by the software. For more information about the handling of complete show files, see the **Show File Handling** topic.

Important:

The file name must not contain the following characters: \ " $ * ? ^ | / : < > `

Subtopics

- **SFTP Connection to a Console**
- **Folder Structure**

1.54.1. SFTP Connection to a Console

The consoles can be accessed using an SFTP client program.

One such program could be FileZilla Client from the FileZilla Project (external link to internet webpage).

The SFTP client needs to be installed on a computer in the same network as the console.

To access the folder structure of the console, the **IP address** of the device is needed. A username and password are also required. SFTP uses TCP port **22**.

To access the grandMA3 area of the console's hard drive, please use **madata** as both username and password.

	Hint:
	The grandMA2 area of the grandMA3 consoles can also be accessed using SFTP. Please use **data** as both username and password to access this part.
	Be very careful about making any changes in the folders or files. This is direct access to the device. Deleting or moving files may cause serious harm to the device.

Once the connection is made, it is possible to browse through the folders inside the device. The first level of folders has three folders. The "actual" folder gives access to the current software version. The "gma3_library" folder contains exported objects, including screenshots. The "installation_packages" contains zip files with the necessary installation files.

Learn more about the folder structure in the **Folder Structure topic**. Read more about screenshots in the **Screenshots topic**.

1.54.2. Folder Structure

The grandMA3 software uses a default set of folders to help maintain a predictable structure when saving data to any internal or external drives.

	Hint:
	This topic, and other topics in the manual, lists a particular folder name as **gma3_x.y.z**. In this name, the “x.y.z” portion takes the place of the actual installed version number. If the installed version is v1.4.2.1, the name of this folder is **gma3_1.4.2**.

Main Folder

The main folder where all grandMA3 user data is stored by default appears using different labels depending upon the type of drive. This folder can be found under the following names and locations:

On USB Drives:

The main folder on any USB drive connected to either a console or a computer running grandMA3 onPC appears at the root of the drive with the name **grandMA3**.

The full folder structure is created once the drive is connected to the station and selected in the system. There are many ways to select the drive, including the Backup menu, any menu allowing import or export, or the **Select Drive [Drive Number]** command. For more information on the **Drive** keyword, see the **Drive keyword** topic.

On Internal Drives of Computers Running grandMA3 onPC:

The full folder structure is created as part of the software installation process.

The main folder on the internal drive of any computer running grandMA3 onPC appears with the name **MALightingTechnology**.

On computers running macOS, the main folder is found at the following location:
[System HD]/Users/[User Name]/MALightingTechnology

On computers running Windows, the main folder is found at the following location:
C:\ProgramData\MALightingTechnology

On the Internal Drive of the Console:

When logged into the console via SFTP, the console presents the main grandMA3 folder as the root directory, using the name **/**. For more information about SFTP, see the **SFTP Connection** topic.

Shows and Backups

Show files and backup show files are saved in dedicated folders, which appear in the following locations:

On USB Drives:

grandMA3/shared/shows
grandMA3/shared/backups

On Internal Drives of Computers Running grandMA3 onPC:

MALightingTechnology/gma3_x.y.z/shared/shows
MALightingTechnology/gma3_x.y.z/shared/backups

On the Internal Drive of the Console:

/actual/shared/shows
/actual/shared/backups

Warning:
Moving, altering, or deleting any files other than show files or backup show files within the **gma3_x.y.z** folder of an onPC station or the **actual** folder of a console may make the software unstable or render the system unable to boot.

Library

When exporting smaller portions of data from the show file (including objects such as fixture profiles, user profiles, and macros), these files are saved into the **gma3_library** folder of the selected drive unless an alternate destination is specified during the export. Most object types have dedicated default sub-folders within the library.

The complete list of automatically generated library folders and sub-folders includes:

- agendas
- appearances
- certificates
- colorthemes
- datapools
    - executorconfigurations
    - executorpages
    - filters
    - groups
    - layouts
    - macros
    - matricks
    - plugins
    - presets
    - sequences
    - timecodes
    - worlds

- fixturetyperesources
    - gobos
    - meshes
- fixturetypes
- inout
    - artnet
    - dcremotes
    - dmxremotes
    - midiremotes
    - osc
    - outputconfigurations
    - sacn
- media
    - images
    - sounds
    - symbols
    - videos
- mvr
- netkeys
- patch
    - dmxcurves
    - stages
- scribbles
- userprofiles
    - cameras
    - renderqualities
    - screenconfigurations
    - viewbuttons
    - views
- users

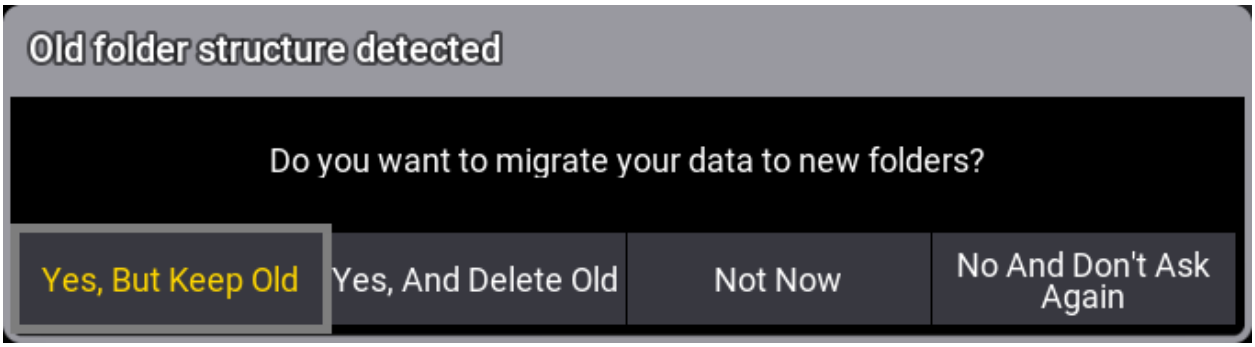
Exceptions to the Default Library Structure

Exporting any objects, that don no have a dedicated sub-folder within the gma3_library, saves the files directly to the **gma3_library** folder unless an alternate destination is specified during the export.

Exporting either the parent of multiple objects, which each have their dedicated folders, (for example, exporting DMXProtocols as one file rather than separate exports of Art-Net and sACN) or children of objects where those children do not have their dedicated folder (for example, exporting one or more cues rather than a whole sequence), automatically adds a descriptive sub-extension to the file name while saving the file to the most appropriate folder.

Automatic Folder Structure Migration

The default folder structure used by version 1.5 and later of the grandMA3 software is different than the structure used in previous versions. When the software recognizes a USB drive, which includes the older structure but not the newer structure, the software presents a pop-up with options for automatically migrating the existing data to the new structure.



Old folder structure pop-up

Options in the pop-up include:

- Yes, But Keep Old: Tap to copy files to the new structure while also maintaining copies in the old structure. This allows both older and newer software versions to continue using the drive.
- Yes, And Delete Old: Tap to move files to the new structure and delete the old structure.
- Not Now: Tap to make no changes to the current structure. The next time the USB drive is recognized by the software, the same pop-up will appear.
- No And Don’t Ask Again: Tap to make no changes to the current structure. The software will not present this pop-up when the USB drive is recognized in the future.

1.55. Show Creator

The Show Creator menu lets you **Import** and **Export** various object types, **Create Presets**, **Create Groups**, **Store Presets to Fixture Types, and Create Presets from Fixture Types.**

To learn more about **Import** and **Export**, see **Import Keyword** and **Export Keyword**.

To learn more about creating various object types, for example, **Groups**, see **AutoCreate Keyword**.

To learn more about how to **Store Presets to Fixture Types**, see **AutoStore Keyword**.

Subtopics

- **Import / Export**
- **Create Groups**
- **Create Presets**
- **Create Presets from Fixture Types**
- **Store Presets to Fixture Types**
- **Partial Show Read (PSR)**

1.55.1. Import / Export

The **Import / Export** menus present a graphical workflow for exporting objects from the current show file as a smaller file with minimal additional show data.

	Hint:
	When exporting objects that reference other objects in the show, the referenced objects are automatically included in the export. For example, when exporting a macro with a defined appearance, the exported file includes the referenced appearance.

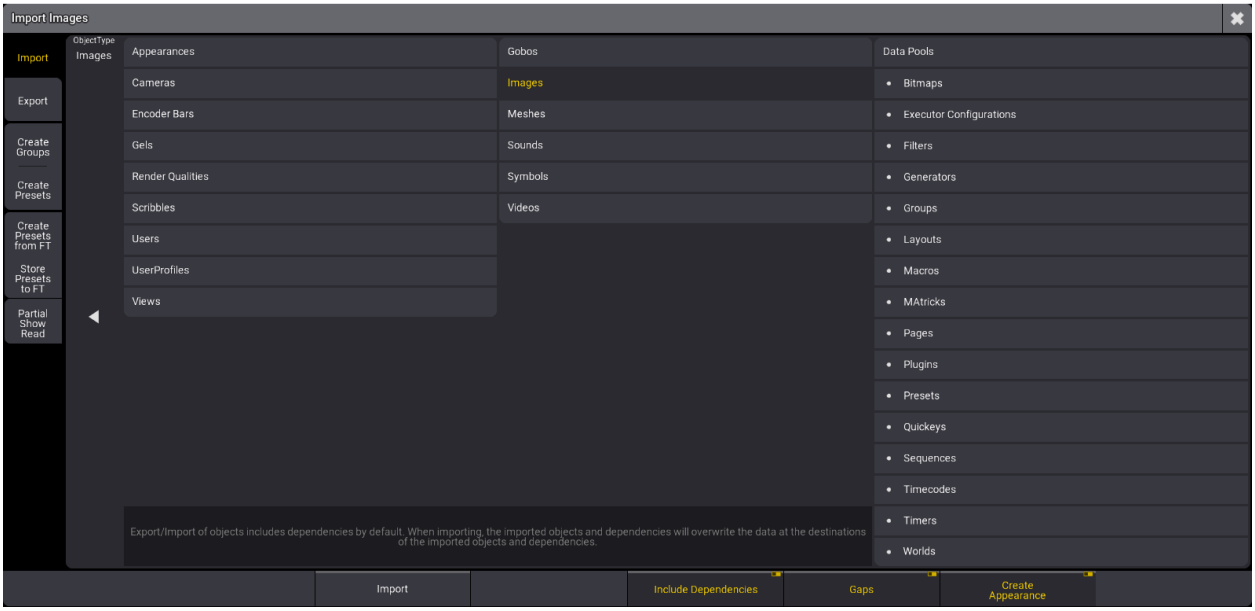
The **Import** menu also allows objects from these smaller files to be imported into the current show file. For example, this menu provides an easy way to copy a user profile or a selection of macros from one show file to another.

Tap the video below to see the example.

Open the **Import / Export** menus using the following steps:

1. Press the Menu key or tap near the top of the **Control Bar**.
2. Tap Show Creator. The Show Creator menu opens.

Tap the Import or Export tab in the upper-left corner to access the desired interface.

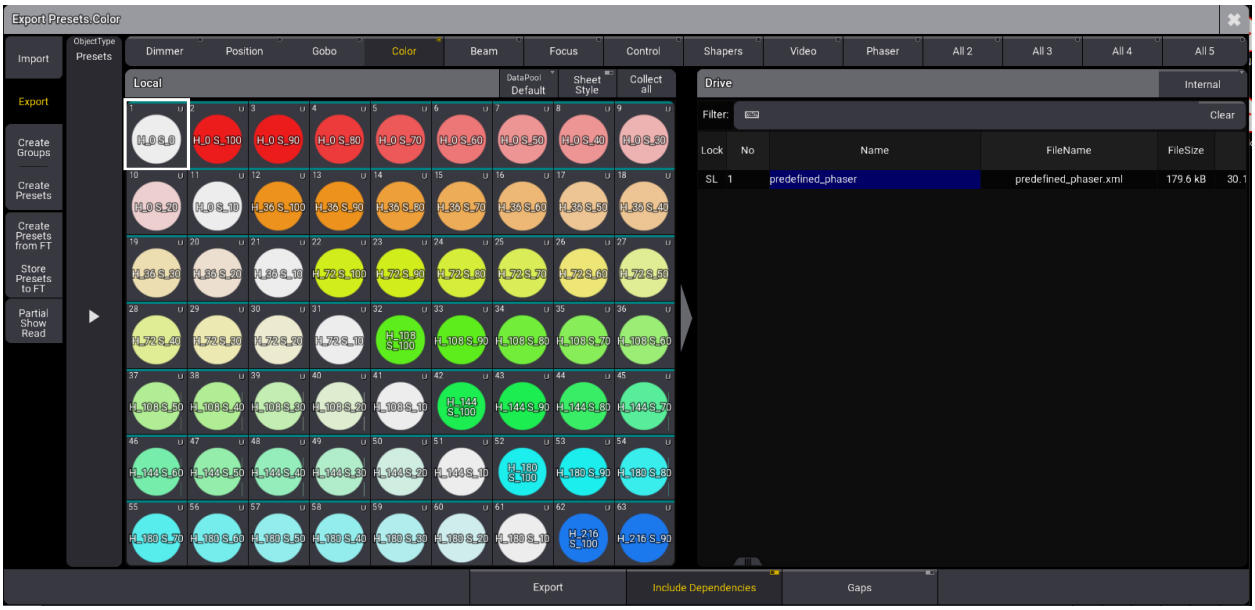


Import menu

Object Types

These menus allow for the import and export of a wide variety of object types. Tap ObjectType on the left side of the main area to open or close the full list of available object types:

- **Appearances**
- **Bitmaps**
- **Cameras**
- **Data Pools**
- **Encoder Bars**
- **Executor Configurations**
- **Filters**
- **Gels**
- **Generators**
- **Gobos**
- **Groups**
- **Images**
- **Layouts**
- **Macros**
- **MAtricks**
- **Meshes**
- **Pages**
- **Plugins**
- **Presets**
- **Quickeys**
- **Render Qualities**
- **Scribbles**
- **Sequences**
- **Sounds**
- **Symbols**
- **Timecodes**
- **Timers**
- **Users**
- **UserProfiles**
- **Views**
- **Videos**
- **Worlds**



Export menu prepared to export color presets

The main area of both the **Import** and **Export** menus shows the source of the data on the left and the destination on the right. Therefore, the **Export** menu displays the specified object types from the current show file in the area on the left, titled **Local**, and the contents of the relevant folder of the selected drive on the right, titled **Drive**. In the **Import** menu, these two areas appear on opposite sides.

- An enabled Gaps button imports or exports the gaps of the XML file. For more information, see **/Gaps Option Keyword**.
- Enabeling Include Dependencies will import and export dependencies. For more information, see **/NoDependencies keyword**.

Drive Selection

To choose a different destination drive for the exported data, tap the Internal button at the right side of the title bar of the **Drive** section or swipe to open a list of available drives. In the **Import** menu, this drive setting defines which drive the console uses as the data source.

Local Section Title Bar Tools

The title bar of the **Local** section includes a few buttons with helpful tools.

- Tap DataPool to cycle through available data pools within the current show file or swipe to open a list of data pools. This option is only available when importing or exporting an **ObjectType,** which is incorporated into a **DataPool** within the show file.
  These **ObjectTypes** are listed below DataPools in the right column of the ObjectType section.
- Tap Sheet Style to display objects in the **Local** section as a sheet with relevant details for each object. Tap Sheet Style again to view objects as they appear in their pools. Some objects, like UserProfiles, never appear in pools. Therefore they will always appear in the sheet format regardless of the state of the Sheet Style button.
- Tap Collect all to select all objects of the current type for export. Once tapped, this button changes to read Uncollect all. Tap this button to deselect all objects. These functions do not appear in the **Import** menu.

- Tap Clear Collection to reset the collection of objects. This option is only available in the **Import** menu.

Object Sub-Types

Some entries in the ObjectType selection include several collections of objects, which appear as different pools. These include:

- Presets: The **Import** and **Export** menus handle and display each preset pool individually.
- Gels: The **Import** and **Export** menus handle and display each gel book as an individual pool.

With the Presets or Gels ObjectType selected, a row of radio buttons appears across the top of the main area showing all of the relevant object sub-types. Tap one of these buttons to display the desired pool below.

Multiple Object Selection

Tap any object in the displayed pool to select it for export. A brown border appears around each selected object. To select multiple objects in a pool, tap each of the desired objects. Tap any selected object to deselect it.

When Sheet Style is enabled, select multiple adjacent objects by drawing a lasso of overall desired objects. To select just one object, tap only the desired object. To toggle selection for any objects in the sheet without changing the selection status of any other objects, hold Ctrl on the keyboard while tapping the desired objects in the sheet.

Export Workflow

1. Tap the Export tab in the upper-left corner.
2. Set the desired ObjectType.
3. Set the desired destination drive.
4. Select all of the desired objects to export from the **Local** area.
5. Enter the desired file name in the Name field of the **Drive** area.
6. Tap Export at the bottom of the menu.

Import Workflow

1. Tap the Import tab in the upper-left corner.
2. Set the desired ObjectType.
3. Set the desired source drive.
4. Select all of the desired files to import from the **Drive** area.
5. Select the desired destination in the pool in the **Local** area.
6. Tap Import at the bottom of the menu.

	**Hint:**
	Certain object types (including **Gobos**, **Images**, **Symbols**, and **Videos**) can be included in **Appearances**. To automatically create new **Appearances** based on these objects as they are importing, toggle on the Create Appearances option before tapping Import.

Import and Export Using Command Line Syntax

For information about exporting and importing show data using command line syntax without the use of these menus, see the **Export keyword** and **Import keyword** topics.

1.55.2. Create Groups

Creating groups based on your current patch is easy and quick.
To do so, press Menu and tap Show Creator. On the left side of the **Show Creator** window, tap Create Groups.

Tap the video below to see the example.

The **Create Groups** menu opens.

Create Groups menu

The left side of the window displays fixture types that are available based on your patch. Tapping Fixture Types in the title bar switches to **Classes** and **Layers**. To learn more, see **Classes and Layers**.

The right side of the window displays the group pool. Tap DataPool in the title bar to select the desired **Data Pool** from the dropdown list; tap Sheet Style to change the display to sheet style.

To create groups:

1. In the left area, select the fixture type, class, or layer for which you want to create a group.
2. In the right area, tap an empty pool object.
3. Tap Create Groups /All to create a group with all fixtures for the selected objects or Create Groups /Single to create single fixture groups. To learn more, see the **AutoCreate** keyword.

When tapping Advanced on the Create Groups title bar, a center area appears. It allows the creation of groups based on a reduced set of fixtures.

Create Groups menu with Advanced mode enable

Create groups with Advanced enabled:

1. Select the objects you want to create a group in the left area. The center area now displays the fixtures for the selected objects.
2. From the center area, select the fixtures for which you want to create a group, for example, all odd fixtures.
3. Tap an empty pool object in the right area.
4. Tap Create Groups /All.
5. A new group with odd fixtures is created.

1.55.3. Create Presets

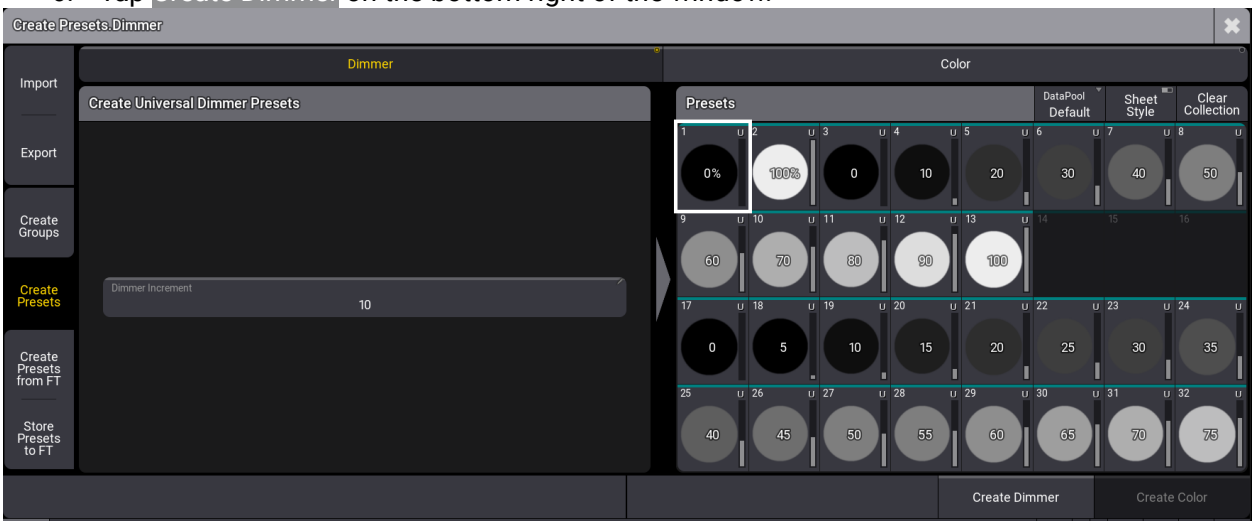
You can easily create universal dimmer or color presets.

To learn more about presets, please read the **Create New Preset** topic.

Create Universal Dimmer Presets.

This is the workflow to create universal dimmer presets:

1. Press Menu, then tap Show Creator.
2. On the left side of the window, tap Create Presets.
3. On the window's feature group bar, tap Dimmer.
4. Tap Dimmer Increment to define the increasing percentage from one preset to the next. Presets are created from 0% to 100%. The allowed increment input range is from 5% to 50%.
5. In the **Presets** area, tap an empty pool object, and it will define the starting point where presets will be generated.
6. Tap Create Dimmer on the bottom right of the window.



Create dimmer presets menu

Create Universal Mixcolor Presets

Universal color presets will be created by combining hue and saturation.
When Book is enabled in the title bar, the left area of the window displays a list of manufacturer swatch books from which to create presets.

To learn more, please read the **Using the Color Picker** topic.

Color presets are created by the combination of **Hue** and **Saturation**.

This is the workflow to create universal mixcolor presets:

1. Press Menu, then tap Show Creator.
2. On the left side of the window, tap Create Presets.
3. On the window's feature group bar, tap Color.
4. In the left area of the window, tap Amount Hue and enter a value between 3 and 36.
5. Tap Amount Saturation and enter a value between 1 and 20.
6. Tapping Sort By will define if the presets will be ordered by Hue or Saturation values.
7. Tap an empty object pool in the right area.
8. Tap Create Color at the bottom right of the window.

Create color presets menu when Book is not enabled

Tapping Book on the title bar will switch this area to the manufacturer swatch books view.

The left area displays a list of manufacturers and corresponding gels available to create presets.

Tap View to display the gels with **Big Icons**, **List**, or **Small Icons**.

You can sort the gels by **Name** or **Key** by tapping **Sort By**.

This is the workflow to create universal mixcolor presets when Book is enabled:

1. Select a manufacturer.
2. Select the gels from which you want to create presets. The whole manufacturer's list will be created as presets if no gels are selected.
3. Tap an empty pool object.
4. Tap Create Color at the bottom right of the window.

Create Color Presets menu when Book is enabled in the title bar of the left area

1.55.4. Create Presets from Fixture Types

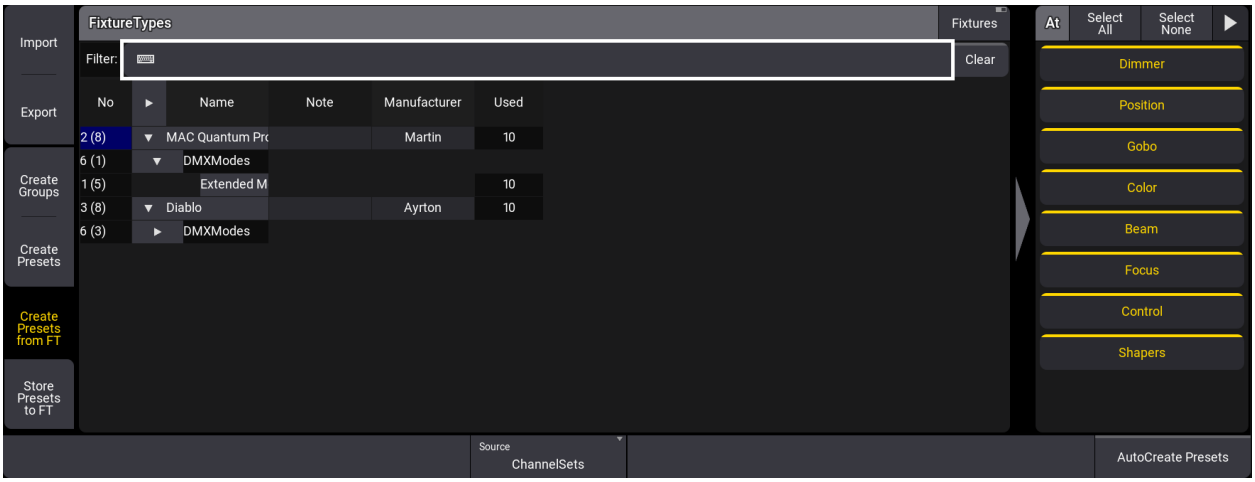
Create Presets from Fixture Type (FT)

Here is how to create presets using presets that were previously stored to fixture types:

To open the Show Creator menu, press Menu and tap Show Creator. The Show Creator menu opens.

- Requirement: Patched fixture types that were previously stored using the **Store Presets to FT** menu or AutoStore keyword. For more information, see **Store Presets to FT**.

On the left side of the Show Creator menu window, tap Create Presets from FT.



Create Presets from Fixture Types (FT) menu in normal mode

The left area displays only the **Fixture Types** and modes that contain **Fixture Type Presets**.
When Fixtures is enabled in the title bar of this area, the patched fixtures are displayed instead of the fixture types. Here also, only fixtures whose fixture types contain fixture type presets are listed.

The right area lets users define which presets will be created from fixture types.
By tapping ▶ on the title bar in the right area, the **At Filter** is temporarily activated, and the feature group area is expanded to display the attributes filter.

Create Presets from FT menu with Fixtures enabled and expended At Filter area

This is the workflow to create presets from fixture types:

1. Select fixtures or fixture types.
2. Select the feature groups, features, or attributes.
3. Tap AutoCreate Presets at the bottom right of the window.

A pop-up will ask to Overwrite or Merge existing presets or Cancel the operation:

Tapping Advanced in the main title bar will display the **Fixture Types** in the left area, the available **Fixture Type Presets** for the selected **Fixture Types** in the center, and the **Presets** pool on the right.

Create Presets from FT menu when Advanced is enabled in the title bar

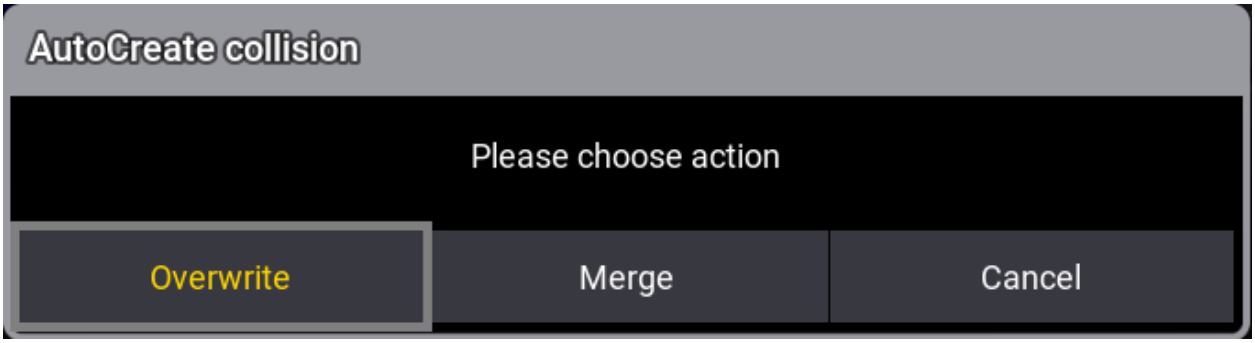
The feature groups tabs are displayed on top of the window. For more information, see **Feature Group Control Bar**.

This is the workflow to create presets when Advanced is active:

1. Select the fixtures or fixture types.
2. Select a feature group from the tab at the top of the window.
3. Select the desired **Fixture Type Presets** from the center area.
4. Tap an empty pool object in the **Presets** area.
5. Tap AutoCreate Presets at the bottom right of the window.

	Important:
	If no preset target is specified, the newly created presets may be stored at existing presets with the same name.

A pop-up will ask to Overwrite or Merge existing presets or Cancel the operation:



1.55.5. Store Presets to Fixture Types

It is an excellent habit to save already stored and named presets directly to the Fixture Types used in your show file; you can then create presets from these Fixture Types when you use them again on another show file.

	Important:
	Fixture Type Presets are not part of the GDTF specification. They can only be transferred to other show files using grandMA3 Fixture Types during export and import.
	Known Limitations:
	1. Only presets that have a name, can be used for creating FixtureType Presets. 2. All presets are stored to a fixture type, but it should only be possible to store global and universal presets to fixture types. 3. Presets that contain attributes of different feature groups discard these attributes during import.

Use the **AutoStore** keyword to store presets to fixture types. For more information, see the **AutoStore** keyword.

Presets can also be stored to fixture types using the **Show Creator** Menu.

Store Presets to Fixture Type (FT)

- Requirement: Fixture Types used in your show file and stored and custom-named presets.

To open the Show Creator menu, press Menu and tap Show Creator. The Show Creator menu opens.
Tap Store Presets to FT.

The left side area lets users define which presets of the selected feature group will be stored in the fixture types.
By tapping ▶ on the title bar, the **At Filter** is temporarily activated, and the feature group list is expanded to display the attributes filter.

The right side area displays the list of Fixture Types used in your show file.
Tap Fixtures on the title bar to display the patched fixtures.

Store Presets to Fixture Types menu

This is the workflow to store presets to fixture types when Advanced is not enabled:

1. Select the feature groups or attributes.
2. Select the fixture types or fixtures if Fixtures is enabled in the title bar of the right area.
3. Tap AutoStore Presets at the bottom right of the window.

A pop-up will ask to Overwrite or Merge existing Fixture Type Presets or Cancel the operation.

Store fixture type presets pop-up

If Overwrite is selected, only the presets of the selected feature group are overwritten.

Store Presets to Fixture Type (FT) when Advanced is enabled

Tapping Advanced in the main title bar will replace the feature group area with a **Presets** pool area.

The **Fixture Types** now occupy the center area, and the right area displays **Fixture Type Presets** for the selected fixture types.
The feature groups tabs are displayed above the window. For more information, see **Feature Group Control Bar**.

Store Presets to FT menu when Advanced is enabled

This is the workflow to store Fixture Type Presets when Advanced is active:

1. Select a feature group from the bar on top of the window.
2. Select the presets from the left area.
3. Select the fixture types or fixtures from the center area.
4. Tap AutoStore Presets at the bottom right of the window.

A pop-up will ask to Overwrite or Merge existing Fixture Type Presets or Cancel the operation.

Store fixture type presets pop-up

If Overwrite is selected, only the presets of the selected feature group are overwritten.

1.55.6. Partial Show Read (PSR)

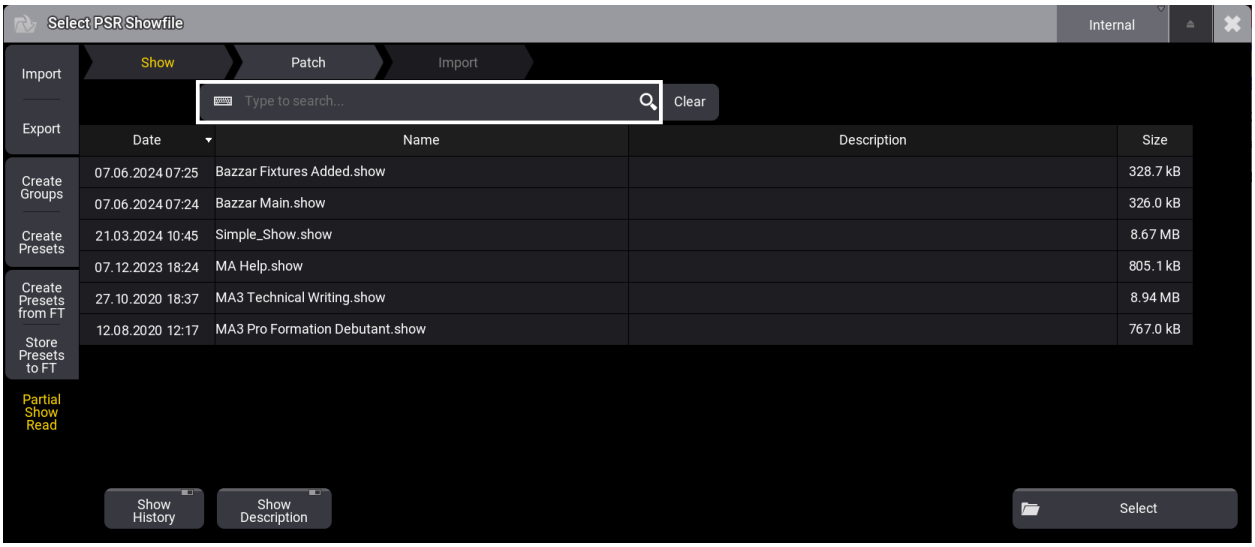
The Partial Show Read menu transfers objects, especially objects containing fixture data, from a different show to the currently loaded show file.

For more information, See **PSR**.

	Important:
	The show file to be PSR'ed must have been saved to the same software version as the running show file on the station where you performed the PSR.
	The partial show read can't be oopsed; it is strongly recommended that the loaded show file be renamed before proceeding with PSR.

To access the PSR menu:

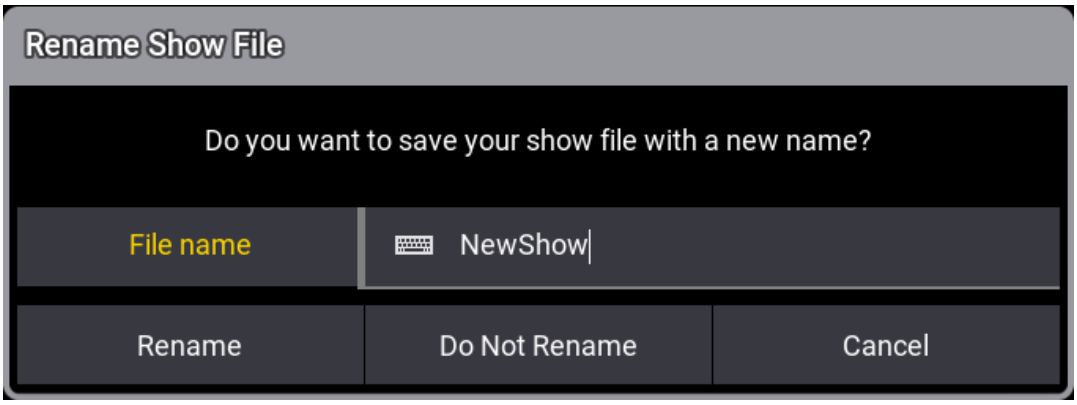
Press Menu, tap Show Creator, then tap PSR. The PSR menu opens:



The arrow-shaped tabs at the top will guide you through the different steps.

1. Tap the Show tab on the bar at the top of the menu.
2. Select a drive (on the right side of the title bar)
3. Select a show file.
4. Tap Select at the bottom of the window or tap Patch in the arrow shape tab bar on top.

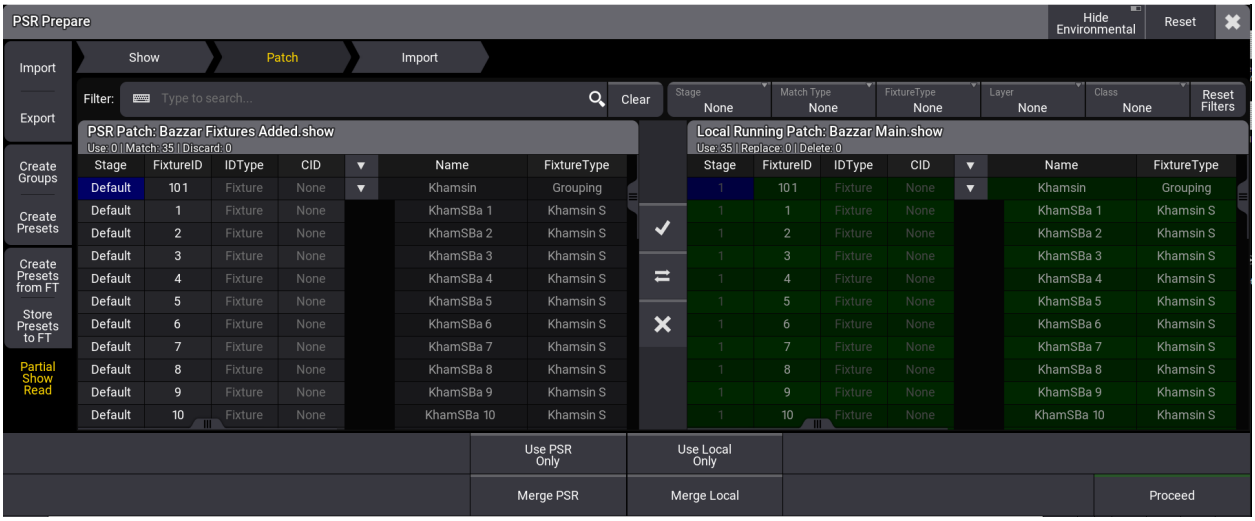
The Rename Show File pop-up opens:



Enter the new name and tap Rename. Tap Do Not Rename if you don't want to rename the show file. If you don't enter a new name, tapping Rename will result in the same outcome as tapping Do Not Rename.

PSR Prepare Menu

If the selected file is suitable, the PSR Prepare menu will open.



When entering the PSR Prepare menu, the software matches the fixtures in the following order:

1. Fixture ID
2. Channel ID
3. Name
4. GUID

The menu is divided into two areas:

- On the left, **PSR Patch** displays the patch of the show file to be imported and its name in the title bar.
- On the right, **Local Running Patch** displays the currently loaded show file and its name in the title bar.

Editing the corresponding cell of the **PSR Patch** allows you to change the properties for **Stage**, **FixtureID**, **IDType**, and **CID**.
Changed properties are displayed in a cyan font. Cells that cannot be edited have a darker background color.

When an IDType is renamed in the PSR Show, two names may appear in one cell of the IDType of the **PSR Patch**. They are displayed with this naming scheme: Name in **PSR Show** (Name in **Local Show**). The names of the IDTypes of the **PSR Show** can't be taken over into the **Local Show**.

When fixtures are not set to the same IDType/CID combination or one of them has no Fixture ID, they will be set to the new IDType PSR and get a consecutive CID.

Rows with corresponding fixtures on both sides are called matched.

Use the tools between both patch areas to define which fixtures will be used:

-  Use the selected fixtures in the resulting patch.
-  Use the fixtures from the opposite patch in the resulting patch.
-  Remove the selected fixtures from the resulting patch.

Marked fixtures have a green background. After the PSR Prepare process, they will exist in the currently loaded show file with their respective fixture types.
Different fixtures can be used from the right and left sides, but not fixtures on the same row.

Environmental fixtures can be hidden by tapping Hide Environmental in the title bar.

Use the search field to filter the patches by fixture names.
Other filters are available:

- Stage: Displays the fixtures of the selected stage.
- Match Type: Displays the fixtures depending on their matching state:

1. **None**: No filters applied.
2. **Matched**: Displays only fixture pairs that are matched together.
3. **Unmatched**: Displays only fixtures that do not match together.
4. **Conflicted**: Displays the conflicting fixtures.

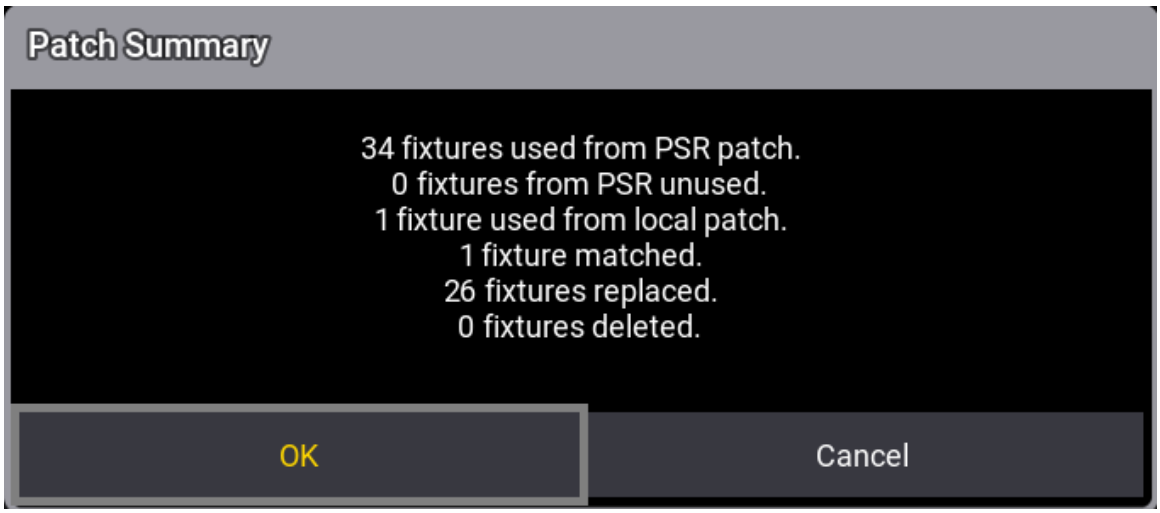
- Fixture Type: Displays only the fixtures of the selected fixture type.
- Layer: Displays only the fixtures of the selected layer.
- Class: Displays only the fixtures of the selected class.
- Reset Filters: Resets all filters.

For a fast selection of fixtures to be taken, tap one of the four buttons at the bottom of the window:

- Use PSR Only: The resulting patch contains only fixtures from the **PSR** show file.
- Use Local Only: The resulting patch contains only fixtures from the **Local** show file.
- Merge PSR: Add all unmatched or unselected fixtures from the **PSR** show list to the resulting patch.
- Merge Local: Add all unmatched or unselected fixtures from the Local show list to the resulting patch.

Tap Reset in the menu title bar to reset all changes and filters.

When done, tap Proceed in the bottom right corner of the window or Import in the tab bar. The PSR Patch Summary pop-up opens:



Tap OK to proceed with the PSR Import menu or Cancel.

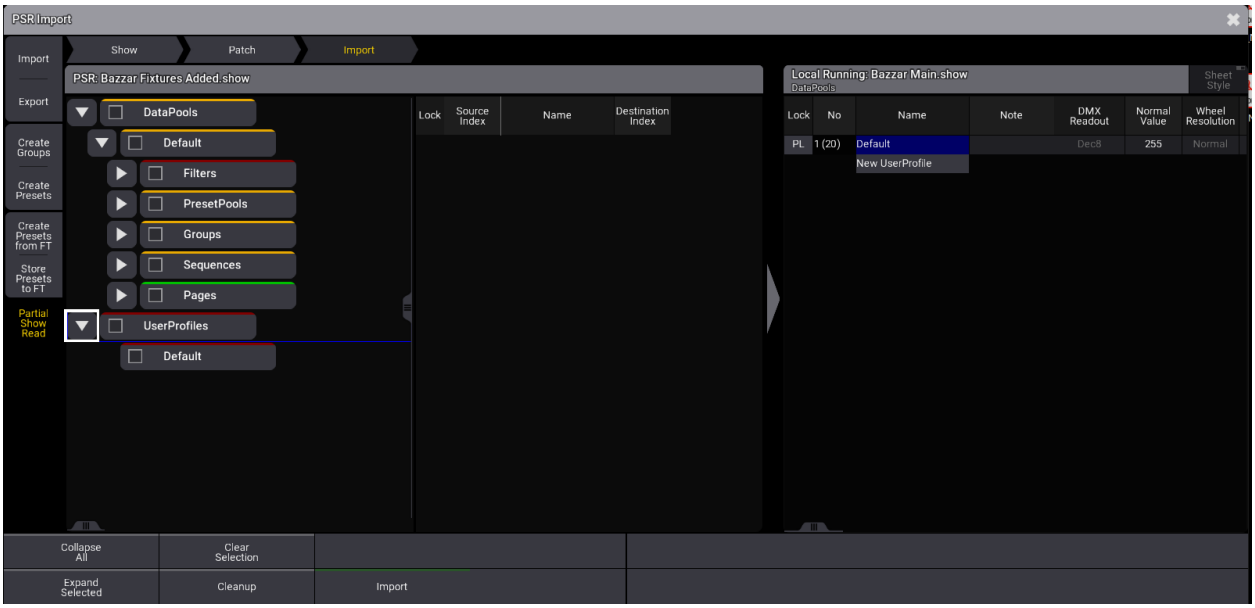
PSR Import Menu

The menu is divided into three areas:

1. The left area (PSR) displays the different pools of the showfile to be PSR'ed, and the title bar displays the name of the show file. Navigate the object tree to the objects you want to import. Tap ▶ to unfold a level in the tree. Mark the objects to be imported by tapping their checkboxes. Selected objects get a yellow checkmark and are displayed in the center area. Parent objects get a gray check mark to indicate that some child objects are selected. When selecting an object that has a reference, the reference is also marked for import and gets a purple checkmark. A number behind the checkbox indicates the hierarchy level of an object.

The colors of the indicator bar indicate the following states:

- Green: This element contains newer data than the local show file.
- Orange: The children of this element are a mixture of older and newer data in the PSR show file.
- Red: This element contains only older data in the PSR show file.



PSR Import menu

2. The center area displays the last selected object type in the left area and allows the definition of the **Import Index** (which spot in the pool it will be imported to). Edit a cell in the **Import Index** column or tap a pool object in the right area (Local). When the Import Index is set to <Auto>, the same object exists in both show files, and the object will be merged with the matching object in the locally running show file.
3. The right area (Local Running) displays the corresponding pool in the local show file. The title bar also displays the name of the show file. The orange collected frames indicate where the objects will be imported. When tapping into the pool in the Local Running area, only selected objects in the PSR area will be imported.

Import non-default DataPool

To import a data pool other than the default one, select a non-default data pool by tapping its checkboxes. Then tap on the referring parent DataPools object in the tree structure. The right area (Local Running) now displays DataPools.

Select a free pool object in the right area. A pop-up window appears:

Confirm by tapping **OK**. A new DataPool with the non-default Datapool of the PSR is now created in the locally running showfile.

Tap Import to transfer the selected object into the local showfile. Objects that were imported are marked as imported in the object tree.

Tap Cleanup to remove imported items from the PSR.

System-locked objects can't be imported. For example, world 1 or filter 1.

	Known Limitation: • The programmer gets clear when entering Import. • When importing the fixtures, references from fixtures to custom appearances will not be preserved. • The console must be in standalone or Idle master state to perform a Partial Show Read.
	Hint: Use the PSR keyword in the programmer to select fixtures of the ID type PSR. For more information, see **PSR keyword**.

1.56. Control other MA Devices

This section describes how to control other MA devices via the console.

Subtopics

- **grandMA3 Nodes**
- **MA Network Switch**
- **RemoteHID**

1.56.1. grandMA3 Nodes

To adjust the settings in the grandMA3 Nodes, it may be convenient to control them from a connected console or onPC.

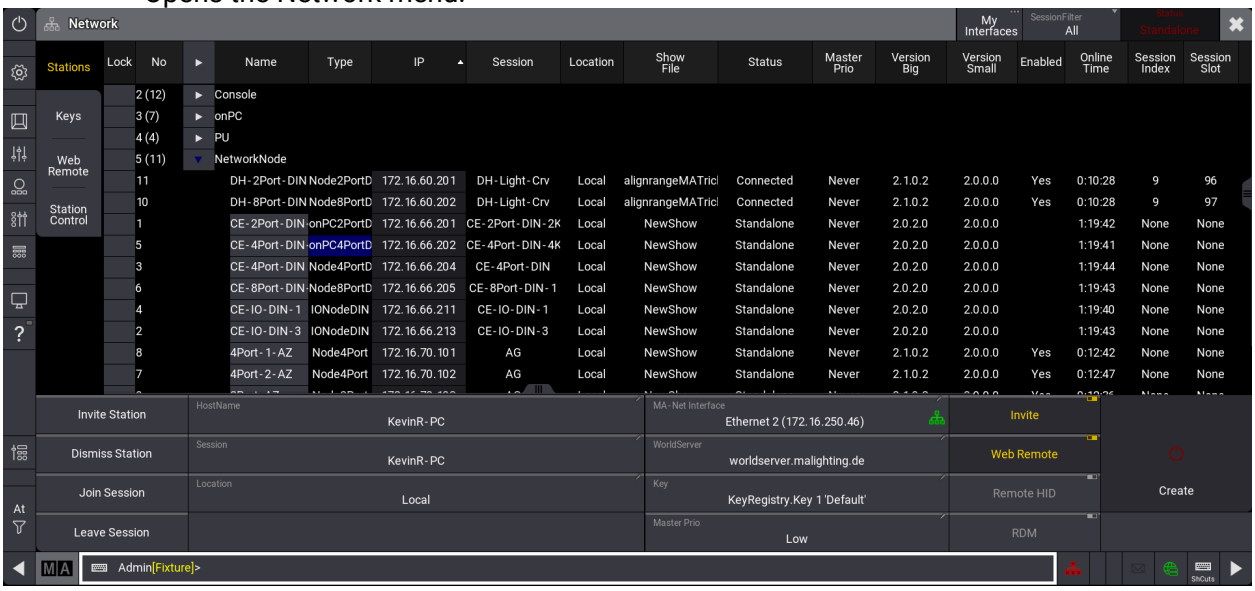
For more information, read the **Session topic** in the **Networking section**.

	Hint:
	If you want to use Art-Net or sACN using grandMA3 nodes, make sure to switch the console to **Mode2**. For information on how to configure the xPort Nodes in Mode2 see **Configure xPort Nodes in the Console**.

Change Name and Set IP Address

To change the name and to set the IP address of a grandMA3 Node, open the Network Interface Menu:

- Press Menu.
  - Opens the **menu select pop-up**.
- Tap Network.
  - Opens the Network menu.

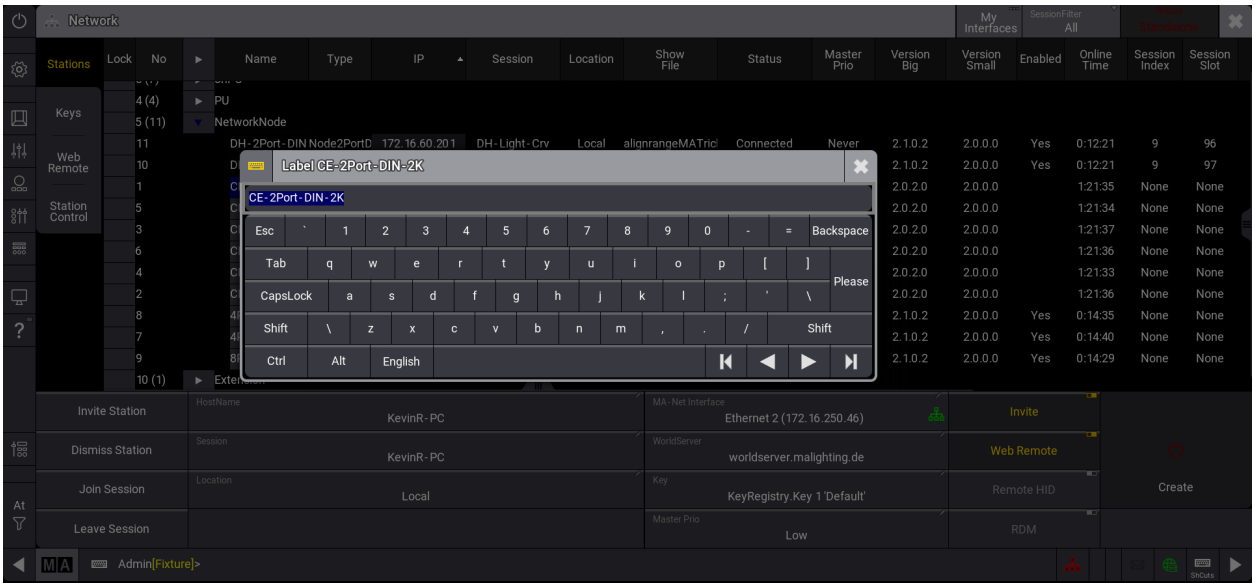


Network menu

	Important:
	You can only change the name or IP address of a node if it is not in another session.

Change the Name

1. Edit the name of the respective node by right-clicking with a mouse or by using **gestures**.



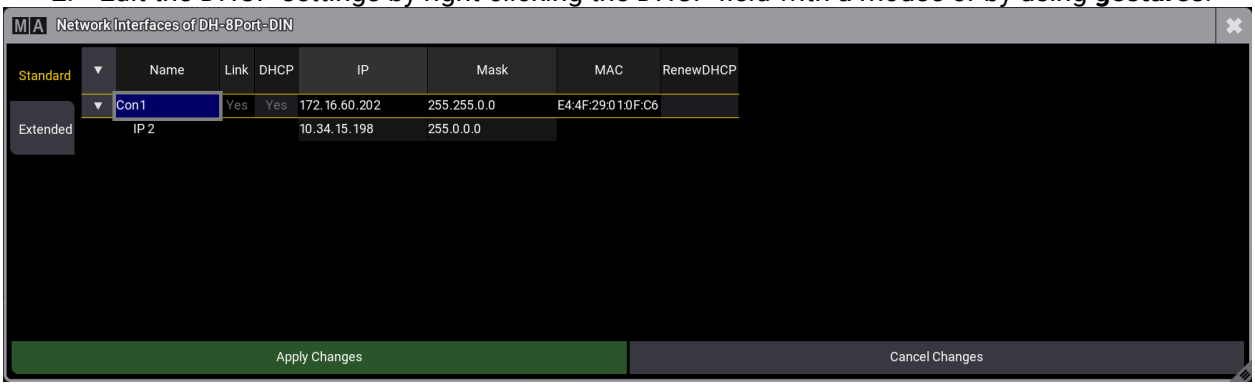
Label Node pop-up

2. Enter the desired name.
3. To confirm the changes, press Please.

The Node has a new name.

Change the IP Address

1. Change the IP address of the respective node by right-clicking with a mouse or by using **gestures**.
2. Edit the DHCP settings by right-clicking the DHCP field with a mouse or by using **gestures**.



Network Interfaces pop-up of a Node

3. Set DHCP to **Yes** or **No**.
4. Edit the IP address of the respective node by right-clicking with a mouse or by using **gestures**.
   The Edit IP pop-up opens.
5. Enter the new IP address.
6. Tap Apply Changes to confirm the changes.

The new IP address is set.

For more information about the network menu, read the **Network Update topic** in the **Update the Software section**.

	Hint:
	It is also possible to update the Nodes via **manual update**.

Change the Output Configuration

To adjust the output configuration settings in the grandMA3 Nodes, control them from a connected console or onPC.

For more information, read the **DMX port configuration topic** in the **DMX In and Out section**.

To connect a grandMA3 Node with the console, set the Node to grandMA3 mode.

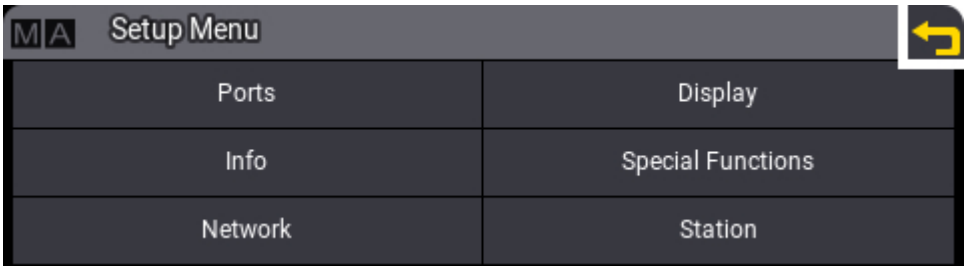
Change Modes

From grandMA3 to Mode2

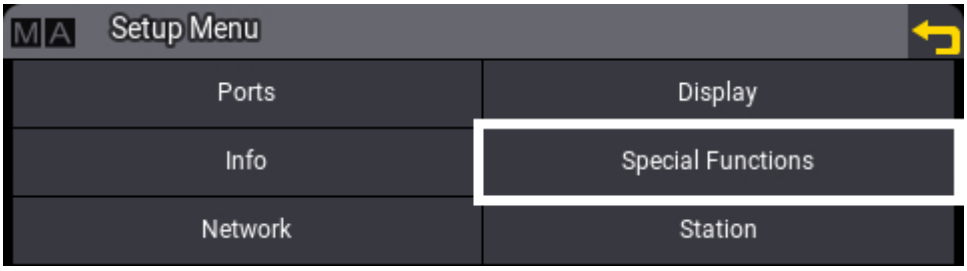
- To change the Node from grandMA3 to Mode2, follow these instructions:

grandMA3 screen:

- To switch to Mode2, press the rotary knob:

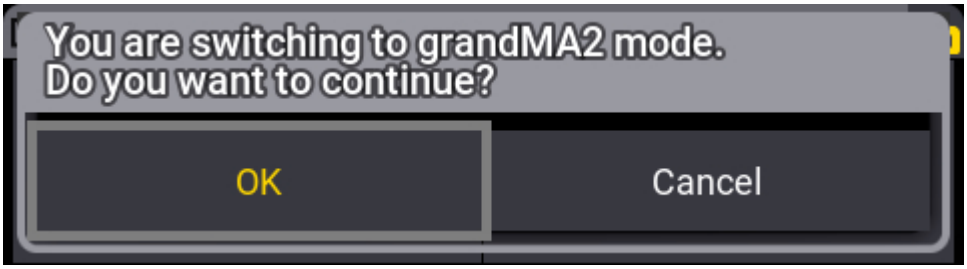


- Turn the rotary knob to select Special Functions and press the rotary knob to confirm:



- Select Switch to grandMA2 and press the rotary knob to confirm:

- Select OK and press the rotary knob to confirm:



The Node reboots in Mode2.

For more information about how to switch a xPort Node from Mode2 to grandMA3, see **change modes**.

For more information about changing to grandMA3 Mode2, read the **Mode2 topic** in the section grandMA3 Mode2 of the **grandMA2 User Manual**.

1.56.2. MA Network Switch

The MA Network Switch is the perfect device for a full lighting control solution. The combination of console, switch, and networking devices provides the easiest solution even for complex lighting systems.

You can control the MA Network Switch by using the web interface or directly with a grandMA3 console in Mode2 or a grandMA2 onPC. For more information about how to control the MA Network Switch by using the web interface see the **MA Network Switch manual**.

To learn how to control the MA Network Switch from a grandMA3 console in **Mode2** or a grandMA2 onPC, read the topic **Control the MA Network Switch** in the **grandMA2 User Manual**. There, you will find out how to add the MA Network Switch to the network configuration, configure ports, or edit groups and presets.

For more information about changing to grandMA3 Mode2, read the **Mode2 topic** in the section grandMA3 Mode2 of the **grandMA2 User Manual**.

1.56.3. RemoteHID

The Remote HID feature allows using the local HID devices (typically mouse and keyboard) to control other grandMA3 stations that are available in the network as if the local HID devices are connected to the remote device.

A use case for this feature is controlling onPC stations that are used for visualization right at a console. Instead of placing several mice and keyboards on the table, only one set of mouse and/or keyboard is needed.

	Restriction:
	It is not a remote desktop feature. The display of the remote device needs to be visible.

To control another station from the local console, follow these steps:

1. Enable RemoteHID in Menu - Network menu on the remote station.
2. Execute the RemoteHID command on the local station:

RemoteHID IP [remote_ip_address]

- Or -

RemoteHID [remote_station_type] ["remote_hostname"]

3. When the connection has been established, the screen of the local station changes to olive green. During the connection, only the executors, including the 100mm faders and Go+[Large], Go-[Large], and Pause[Large] remain usable on the local station. All other elements are blocked.

The remote function can be ended using one of the following options:

- Press MA + MA + Off
- Use the keyboard shortcut Shift + Ctrl + Alt + E

Example

A preprogramming setup where there are a console and a powerful graphics computer running grandMA3 onPC optimized for 3D visualization. They are connected in a session on a local LAN network.

The grandMA3 onPCs hostname is "3D" and RemoteHID is enabled.

From the console, there is a need to change a setting in the 3D window on the grandMA3 onPC.

In the command line, type:

Now, the mouse and keyboard connected to the console can be used to operate the computer.

Finish the connection by pressing MA + MA + Off

	Important:
	grandMA3 onPC on Windows controlled via RemoteHID, is not only limited to control the grandMA3 onPC application. Furthermore, the mouse and keyboard can also access remotely the entire system.
	Restriction:
	RemoteHID can only be used on grandMA3 consoles and grandMA3 onPC on Windows.
	Restriction:
	At the moment, it is not possible to use the touch displays of grandMA3 consoles to control the mouse remotely, so a USB mouse must be connected to the grandMA3 console to move the mouse on the remote device.

1.57. Troubleshooting

In case your grandMA3 device does not work as expected, the topics below include helpful general troubleshooting steps as well as detailed solutions to some possible issues.

Subtopics

- **Clean Start**
- **Update Does Not Work**
- **Station Does Not Connect**
- **Panic Macro**

1.57.1. Clean Start

A clean start clears the current show and all user settings as the grandMA3 software boots. This may be a helpful troubleshooting step in cases where the software boots normally but encounters an issue when the show file loads.

Clean Start a grandMA3 console, Replay Unit, or Processing Unit

	**Important:**
	This process relies on the Ctrl or control key commonly found at or near the bottom-left or bottom-right corner of an alpha-numeric keyboard. For devices without an integrated keyboard, an external USB keyboard is required.

To perform a clean start on a grandMA3 console, replay unit, or processing unit:

1. Make sure the integrated keyboard is functional or connect an external USB keyboard.
2. Boot or reboot the device, watching for the mode selection pop-up.
3. When the mode selection pop-up appears, press and hold the Ctrl key.
4. While holding the Ctrl key, tap the desired mode in the mode selection pop-up.
5. Once the mode selection pop-up disappears, release the Ctrl key.
6. The device completes the boot process with no show file loaded and all user settings reset to factory defaults.

Clean Start grandMA3 onPC

- To perform a clean start on a Windows system, run the grandMA3 onPC x.x.x.x Clean Start app from the MA Lighting folder.
- To perform a clean start on a macOS system, press and hold the option key on the keyboard and then open the grandMA3 onPC application. An alert pop-up appears, confirming the clean start operation. Click Continue to proceed with the clean start. Otherwise, click Cancel.

The software starts with no show file loaded and all user settings reset to factory defaults.

1.57.2. Update Does Not Work

There may be several reasons for a software update to fail:

Broken Installer Package

	Restriction:
	The installer package zip file may be broken after the download.

1. Delete the old installer package.
2. Download a new installer package.

Installation from the Zip File

	Restriction:
	Do not install the software from the zip file.

1. Unzip the file to your computer's hard drive.
2. Install the software from the uncompressed folder.

Broken USB Flash Drive

	Restriction:
	Older USB flash drives may be worn and cannot be read correctly.

- Format the USB flash drive.
- Replace the USB flash drive with a new one.

	Hint:
	New out-of-the-box USB flash drives may come without a partition table.

1. Check the USB flash drive by copying and pasting other files and folders from one computer to another.
2. Copy the installation packages if the USB flash drive works perfectly.
3. Perform software update.

Broken USB Port

	Restriction:
	The USB port may be broken on the grandMA3 device or the computer.

- Use a different USB port.
- Update the USB port driver.
- Replace the broken USB port.

Broken USB Cable

	Restriction:
	The USB cable may be broken.

- Use a different USB cable.

1.57.3. Station Does Not Connect

If a station or a wing does not connect, reset the station settings with a **Clean Start**.

If the command wing or fader wing fails to connect to the onPC software, try switching the product off and then on again.

Reasons that can prevent an onPC command wing or fader wing communication with a computer:

- A damaged USB port on the command wing or fader wing.
- A damaged USB port on the computer.
- A damaged USB cable.
- Recently installed operating system updates or power settings.
- An incomplete or corrupted onPC software installation.

1.57.4. Panic Macro

	**Important:**
	Make sure to save the show file before executing the Panic macro.
	**Restriction:**
	Do not use the Panic macro during a live show.
	**Important:**
	Be aware that the use of the panic macro can have a severe impact on the show.

In different cases, for example, if a show file is not working as expected before a show, it can be useful to use the "Panic" macro.

The panic macro will restore and reset general settings to default. These general settings are listed below:

1. Warning pop-up
2. Warning pop-up
3. Clear the Programmer (ClearAll)
4. Change Command Line back to root destination
5. Switch off Web Remote
6. Switch off Remote HID
7. Switch off RDM
8. Switch off Agenda
9. Switch off DC/ MIDI/ DMX Remotes
10. Switch off OSC Input
11. Switch off OSC Output
12. Switch off PSN Input
13. Go to World 1
14. Select Filter 1
15. Switch off Freeze
16. Switch off Preview
17. Switch off Plugins
18. Switch off Sequences
19. Switch off Timecodes
20. Switch off all Executors
21. Switch off Timecode Slots
22. Switch off Master Mode of Groups
23. Set the Master Fader of all Groups to 100%
24. Switch Off DMX Tester
25. Set the Grand Master to 100%
26. Set the World Master to 100%

27. Set the Highlight Master to 100%
28. Switch off Highlight
29. Set the Lowlight Master to 100%
30. Switch off Lowlight
31. Set Solo Master to 100%
32. Switch off Solo
33. Set Grand Rate to 1
34. Set Grand Speed to 60 BPM
35. Set Programmer Time to 0s
36. Switch off Programmer Time
37. Set Programmer X Fade to 0s
38. Switch off Programmer X Fade
39. Set Executor Time to 0s
40. Switch off Executor Time
41. Set Executor X Fade to 0s
42. Switch off Executor X Fade
43. Set all Speed Masters to 60 BPM
44. Set the Blind Master to 100%
45. Switch off Blind
46. Enable Sync for Phasers in the Programmer
47. Disable SingleStep for Phasers in the Programmer
48. Switch off Align
49. Set Value Readout to Natural
50. Set Wheel Mode to Additive
51. Request all Universes
52. Set Merge Mode of all Universes to HTP
53. Switch off Art-Net and sACN Output
54. Switch off Art-Net and sACN Input
55. Clear Programmer for all Users
56. Select Encoder Bar 1
57. Switch to first Feature Group
58. Reload the User Interface
59. Switch off all Macros

Hint:

To see the list from above in the software, press Edit and then tap the Panic macro in the macro pool.

- To learn more about macros, see **Macros**.

Import and Execute Panic Macro

1. Search for Panic in the import list. For more information about importing macros, see **Create Macros**.
2. After import, tap Panic in the Macros pool window. The text editor opens.
3. To confirm the panic macro, press Please Please.

4. The panic macro is executed.

1.58. Glossary

The following collection of abbreviations, acronyms and fundamental lighting terms can be found in the grandMA3 user manual.

To open the glossary in the grandMA3 software:

1. Press Help and enter **Glossary** into the command line.
2. Press Please. The Glossary opens.

A

ACN - Architecture for Control Networks
Is a suite protocol. It uses a lot of elements that are currently not supported by grandMA3. But the ACN protocols also have a version for transporting DMX data. For more information on ACN, see **sACN menu**.

ANSI - American National Standards Institute

Art-Net
Art-Net 4 Ethernet communication standard is developed by Artistic Licence. For information about Art-Net in the grandMA3 software, see **Art-Net menu**.

Attributes
Attributes are the building blocks of fixture types. For more information, see **Attribute definitions**.

B

bit - binary digit

bpm - beats per minute

C

CIE - Commission internationale de l'éclairage

CLI - Command-line interaction

Command Line
The syntax to control the console can be entered in the Command Line. For more information, see **Command Line**.

Conventional lights
Lighting device with dimmer only.

CPU - Central Processing Unit

Cue
Cues a stored values. A set of cues is a Sequence. For more information, see **Cues and Sequences**.

D

dBu - decibel unloaded

DC - Direct Current

DHCP - Dynamic Host Control Protocol
Is a system where IP addresses are distributed from a DHCP server. For more information on how to set IP addresses, see **Interfaces and IP**.

DMX - Digital Multiplex
Communication protocol to send a digital signal from a controller to a receiver. A single DMX cable can transmit one Universe.

DMX Channel
A single DMX Channel has a value between 0 and 255.

DMX Universe
A DMX Universe contains 512 DMX channels.

Dual Encoder
The five dual encoders on a console or a command wing are used to adjust the different attributes of the fixtures or other functions. For more information, see **Dual Encoders**.

E

Encoder Bar
Displays the attributes or functions linked to the encoders and includes shortcuts for quick access to a selection of user profile settings and programming tools. For more information, see **Encoder Bar**.

Executors
Executors (Exec) are physical keys, knobs, and faders on the grandMA3 hardware. They can also be represented as on-screen virtual executors. For more information, see **Executors**.

Extension
Allows to extend the amount of physical executor handles for specific grandMA3 products. For more information, see **Connect grandMA3 extension**.

F

Fader Wing
An additional fader for the grandMA3 onPC system. For more information, see **grandMA3 onPC fader wing**.

Fixture
A complete lighting unit including at least a housing and a lamp.

Fixture Type
Fixture types are used to visualize and control real-life fixtures on stage. For more information, see **Fixture Types**.

fps - frames per second

G

GB - Gigabyte

GDTF - General Device Type Format
Is a file format used to describe devices. For information about GDTF, see GDTF Share.

GlobalMaster
This is the session master station in all networked systems. For more information, see **Locally Networked Devices**.

GPI - General Purpose Interface

Grand Master
Is used to limit the output of the intensity of all the fixtures patched in a show. For more information, see **Grand Master**.

Groups
Groups contain a selection of fixtures. This includes the order and grid position of the selection. This can be used as a programming tool, where it is a fast way to select the fixtures. For more information about Groups, see **Groups**.

H

Hex - Hexadecimal

Highlt - Highlight
See **Highlt key**.

HTP - Highest Takes Precedence
The highest value will be output, mostly used with Dimmer. For more information about priorities, see **LTP**.

Hz - Hertz

I

IdleMaster
When a console is ready to be connected to another station, the console is in IdleMaster mode. For more Information, see **Standalone Device**.

IEC - International Electrotechnical Commission

IP - Internet Protocol

K

Keywords
Defined word in the software with a specific usage.

L

LED - Light Ermitting Diode
An intelligent light with a LED source.

Level Wheel
The level wheel on the right side of a console or command wing is always assigned to dimmer attributes.

log - logarithm

Lua
Lua is a scripting language designed to support general procedural programming. For more information, see **What is Lua**.

LTC - Linear Time Code

LTP - Latest Takes Precedence
Most recent value will be output, mostly used for all attributes besides Dimmer. For more information about priorities, see **DMX port configuration**.

M

mA - milliampere

Macro
Macros are commands stored in a pool object. For more information, see **Macros**.

Master
Masters are physical representations of various timing and level overrides that exist in the software. For more information, see **Masters**.

MB - Megabyte

MIB - Move In Black
For more information, see **Move In Black**.

MIDI - Musical Instrument Digital Interface
A standard communication protocol. For more information about MIDI, see **MIDI**.

Moving Light
Lighting instrument with additional attributes other than dimmer. It is possible to control the position of a Moving Light from a console.

MSC - MIDI Show Control
Sends functions via MIDI signal from an executor to another device, for example a console or MIDI sequencer.

Multi-Instance Fixture
A lighting instrument with multiple individually controllable sections.

MVR - My Virtual Rig
MVR is a file format that is used to share data for a patch. For more information, see **MVR**.

O

OS - Operating System

OSC - Open Sound Control
OSC is a client and server system that defines a message address pattern used to address elements in the receiving server. For more information, see **OSC**.

P

Pan
Horizontal movement axis of a fixture.

Parameters
A parameter can use several DMX channels. For more information, see **Parameters**.

Phasers
Dynamically changes the output for attributes using a set of information in two or more steps. For more information, see **Phasers**.

PoE - Power over Ethernet

Playback
Recalling stored information physically (for example, fader) or graphically (user interface) on a console. A preset, a sequence with cues, or a timecode show can be played back. For more information, see **Playback Window**.

Presets
Presets are containers for values. For example, when the same values are used in different cues several times. For more information, see **Presets**.

Prev - Previous
See **Prev key**.

Programmer
The programmer is a temporary memory, where the active values are placed. The values can then be stored or released. For more information, see **Programmer**.

Prvw - Preview
See **Prvw key**.

PSN - PosiStageNet
grandMA3 stations can receive PSN data. For more information, see **PSN**.

PSR - Parcial Show Read
PSR transfers objects of show files. For more information, see **PSR**.

PWM - Pulse Width Modulation

R

RAM - Random Access Memory

RCA - Radio Corporation of America

RDM - Remote Device Management
Permits intelligent bidirectional communication between devices from multiple manufacturers using a modified DMX512 data link. For more information, see **RDM**.

RPM - Rounds per Minute

S

S/PDIF - Sony/Philips Digital Interface

sACN - Streaming Architecture for Control Networks
For more information, see **ACN**.

SelFix - SelectFixture
See **SelFix key**.

Sequ - Sequence
See **Sequ key**.

SMPTE - Society of Motion Picture and Television Engineers

SSD - Solid-State Drive

SSE - Streaming SIMD Extensions

STP - Shielded Twisted Pair

Syntax
The command line syntax is used to create valid commands. For more information, see **Syntax**.

T

TC - Timecode
Timecode is a time signal used to let off the recordings of a playback or trigger cues. For more information, see **Timecode**.

TCP - Transmission Control Protocol

Tilt
Vertical movement axis of a fixture.

Tracking
Tracking is the principle to store only the changes in the cues. For more information, see **Tracking**.

U

UDP - User Datagram Protocol

UI - User Interface
See **User Interface**.

UUID - Universally Unique Identifier

V

V - Volt

VRAM - Video Random Access Memory

W

Web remote
See **Web remote**.

X

XFade - Crossfade
For more information, see **Assign object to an executor**.

XLR - External Line Return

XML - Extensible Markup Language

1.59. grandMA3 List of Trademarks

The grandMA3 Software supports other brands and highlights their intellectual property in its software.

The following trademarks used by grandMA3 are protected in the US and other countries:

- Rosco colors are protected by Rosco Laboratiories, Inc.

Rosco Laboratories, Inc.
Rosco
CalColor
Cinegel
Cinelux
Mix
Mixbook
R26
R80
Roscolux
Supergel
True Rosco Color
VS

- macOS is protected by Apple Inc.
- Windows is protected by Microsoft Corporation