



User Guide

Spyder Studio

For Spyder-S line

Software Version 6.1.3 | Document Revision 1.1 | September 2026

CHRISTIE®



Content

- 1. About This Guide & Legal Notices 9**
 - 1.1 About This Guide 9
 - 1.2 Revision History 10
 - 1.3 Copyright 10
 - 1.4 Trademarks 10
 - 1.5 General 10
 - 1.6 Warranty 10
 - 1.7 Limitation of Liability 10
 - 1.8 Software License 11
- 2. Safety & Regulatory 12**
 - 2.1 General safety instructions 12
 - 2.2 Symbols & conventions 12
 - 2.3 Environmental specifications 13
 - 2.4 Regulatory compliance 13
- 3. Product Introduction & Hardware Overview 15**
 - 3.1 What is Spyder Studio and Spyder-S? 15
 - 3.2 Frame types and per-Frame specs 16
 - 3.3 Front and rear panels 17
 - 3.3.1 Front panel 17
 - 3.3.2 Rear panel 17
 - 3.4 Linking Frames / Frame Groups 19
 - 3.5 The modular distributed architecture 21
 - 3.6 Package contents & rack mounting 22
- 4. Getting Started & Installation 24**
 - 4.1 Physical installation 24
 - 4.2 Power 24
 - 4.3 Network configuration 24
 - 4.4 Installing and updating the software 25
 - 4.4.1 Update Spyder Studio 26
 - 4.4.2 Update a Frame's firmware 27
 - 4.5 Connecting Studio to hardware 28
 - 4.5.1 Connect to a Frame 29

4.5.2 Work offline (no Frame or server needed).....	30
4.5.3 Optional — the Local Remoting Server	30
4.6 Securing the system	31
4.7 First configuration	31
4.8 Saving, restoring, and syncing projects	32
4.8.1 Backing up and restoring projects	32
4.8.2 Push and Pull data	33
4.8.3 HotSync.....	34
4.9 Application options	34
4.9.1 Advanced settings	36
5. Concepts & Architecture.....	42
5.0 Key terms	42
5.1 The Spyder Studio workspace.....	42
5.1.1 Canvas and Visual Image.....	44
5.2 PixelSpace	44
5.3 Layers and keyframes	44
5.4 Mix Effects — the "Single Layer Mixer"	45
5.5 Backgrounds — "Layer Zero"	45
5.6 Sources and inputs	46
5.7 Input modes.....	46
5.8 Output types	47
5.9 Resource and bandwidth budget	47
5.9.1 Frame groups	47
5.9.2 Layer provisioning across Frame groups	48
5.10 Signal flow (end to end).....	49
6. Configuration	50
6.1 Opening the Configuration Manager	50
6.2 Choosing a configuration mode	51
6.3 All In One workflow	52
6.3.1 Start with a new configuration	52
6.3.2 Load the current configuration	58
6.4 Free Form workflow	60
6.4.1 Start with a new configuration	60
6.4.2 Load the current configuration	75
6.5 Output roles and capability flags	77
6.6 Configuring OpMon outputs	79

6.6.1 OpMon presets.....	80
6.7 Configuring tiled outputs.....	81
6.7.1 Tile presets	83
6.8 PixelSpace properties: titling and test patterns	84
6.8.1 Name, origin and size.....	86
6.8.2 Titling.....	86
6.8.3 Test Pattern Generator	86
6.9 Managing EDIDs.....	88
6.9.1 Assigning an EDID to inputs	90
6.9.2 Preferred Detailed Timing	90
6.10 Reviewing the applied configuration	91
6.10.1 System Patch.....	91
6.10.2 System Layout	92
7. Operation.....	94
7.1 Managing sources.....	94
7.1.1 Read the Sources pane.....	94
7.1.2 Select a source — or create one.....	96
7.1.3 Drag a source onto the canvas — Mix Effect vs. layer	97
7.1.4 Source context menu and backup.....	97
7.1.5 Input Configuration.....	98
7.1.6 Sources from routers	100
7.2 Working with layers	100
7.2.1 Add content to a PixelSpace	100
7.2.2 Size and position a layer.....	100
7.2.3 Clone, mirror, and offset a layer.....	102
7.2.4 Apply layer attributes — crop, aspect ratio, pan & zoom, motion, transparency	103
7.2.5 Align layers with the Simulator toolbar	105
7.2.6 Layer context menu and keyboard shortcuts	109
7.2.7 Borders and shadows	110
7.3 Using treatments.....	114
7.3.1 Attribute categories	115
7.3.2 Creating and learning a treatment.....	115
7.3.3 Applying a treatment	116
7.3.4 Default treatment, files, and the context menu.....	116
7.3.5 Treatment properties and Recall Options	117
7.4 Using Mix Effects.....	118

7.4.1 Understand Top → Program and Bottom → Preview	118
7.4.2 Assign sources to Top and Bottom	119
7.4.3 Cut vs. Mix, and Duration	119
7.4.4 Worked example — transition between two sources	119
7.4.5 Mix Effect types — Normal, Aux, and Background	120
7.4.6 Freeze an ME	121
7.5 Working with backgrounds	121
7.5.1 Understand the three content types and the background group	122
7.5.2 Crop a background input	122
7.5.3 Mix or Cut backgrounds — the two-group rule	124
7.5.4 Substitute a still for a background	124
7.5.5 Add a background image	124
7.5.6 How many background stills fit	125
7.5.7 Test-pattern stills	125
7.6 Managing PixelSpaces	126
7.7 Managing outputs	126
7.7.1 General	128
7.7.2 Video Options	129
7.7.3 Test Pattern Generator	129
7.7.4 Advanced	130
7.8 Using routers	130
7.8.1 Router Patch (static wiring)	131
7.8.2 Router XY (live crosspoint)	132
7.8.3 The Add New Router dialog	133
8. Command Keys and Function Keys	135
8.1 Keys and scripts	135
8.2 Command Keys	135
8.2.1 Layout and pages	136
8.2.2 Recalling a command key	136
8.2.3 Creating a command key	137
8.2.4 The command-key context menu	138
8.2.5 Command Keys toolbar	139
8.3 Startup behavior	139
8.4 Function Keys	140
8.4.1 Layout	140
8.4.2 Running a function key	141

8.4.3 Creating a function key	141
8.4.4 Why the create menu varies.....	142
8.4.5 The function-key context menu	143
9. Scripts.....	144
9.1 The Script panel.....	144
9.1.1 Grid layout	144
9.2 Element types.....	145
9.3 Building and editing a script	146
9.3.1 Adding elements	146
9.3.2 How long an element plays	146
9.3.3 Moving and copying elements	146
9.3.4 Off elements.....	146
9.3.5 Keyframes, deleting, and undo	147
9.4 A Command Key's script.....	148
9.5 Running a script, keyframes, and triggers.....	148
9.6 Script panel context menus.....	149
9.7 Slave layers (Discreet mode).....	150
9.8 Live control behavior.....	151
9.8.1 Live Updates Enabled	151
9.8.2 Disconnect Layers	151
10. External Control & API.....	153
10.1 The external-control model.....	153
10.1.1 Replies and result codes.....	154
10.1.2 Layer ID convention	155
10.2 Connection and transport.....	156
10.3 Command reference.....	159
10.3.1 Mix Effects.....	159
10.3.2 Layers — content, transition, geometry	160
10.3.3 Sources, stills & inputs	163
10.3.4 Backgrounds.....	165
10.3.5 Keyframe / property get-set	165
10.3.6 Scripts & registers	165
10.3.7 Outputs.....	167
10.3.8 Routers & device commands.....	168
10.3.9 PixelSpaces & layout	169
10.3.10 Test patterns.....	169

10.3.11 System, status & hardware queries	169
10.3.12 Multiviewer (capability-gated)	170
10.3.13 OpMon & Tiled output (capability-gated)	172
10.3.14 Hardware dumps & stubs.....	172
10.4 Mix Effect control tokens.....	174
10.4.1 Transitioning — Mix and Cut	174
10.4.2 Argument validation and errors	174
10.4.3 Reading ME state.....	175
10.5 Companion integration	175
10.6 Console Simulator (UDP).....	175
11. Maintenance & Troubleshooting.....	177
11.1 Alerts and monitoring	177
11.1.1 Reading alerts	177
11.1.2 Saving and clearing alerts	178
11.2 Tools and diagnostics	178
11.2.1 UDP Protocol Tester	179
11.2.2 Maintenance utilities.....	179
11.2.3 User Diagnostics.....	180
11.3 Common problems.....	180
11.4 Factory reset and recovery	181
11.4.1 Perform a factory reset	182
12. Appendices, Glossary & Contact.....	183
Appendix A — Menu & toolbar reference	183
A.1 File.....	183
A.2 View	184
A.3 Layout	184
A.4 Server.....	185
A.5 Tools	186
A.6 Help	187
A.7 Debug (<i>option-gated</i>)	187
A.8 Save (<i>toolbar button</i>)	188
Appendix B — Auxiliary panes.....	188
B.1 Devices pane.....	188
B.2 System Layout pane.....	189
B.3 Dynamic Help pane	189
Appendix C — Capacity & bandwidth	189

- C.1 The three budgets.....190
- C.2 What draws against the bus.....190
- C.3 Frame groups: canvas bandwidth vs. Mix Effects190
- C.4 Mix Effect caps and Auto Provision191
- C.5 Worked reference — a four-Frame system.....191
- Appendix D — Third-Party & Open-Source License Notices192
- Appendix E — Supported formats192
 - E.1 Summary by connector192
 - E.2 SDI 12G and SFP formats192
 - E.3 HDMI 2.0 formats193
 - E.4 DisplayPort 1.4 formats.....195
- Glossary.....196
- Contact Information200

1. About This Guide & Legal Notices

This chapter introduces the guide — its scope, audience, and conventions — and states the legal notices that govern your use of Spyder Studio 6.1.3 and the Spyder-S hardware it controls.

1.1 About This Guide

- Purpose.** This guide describes how to configure and operate **Christie Spyder Studio 6.1.3**, the Windows control software for the **Spyder-S** modular windowing video processor, together with the Spyder-S hardware it drives.
- Audience.** Operators, system integrators, and automation/control developers who set up, run, or program a Spyder-S system.
- Applies to.** Spyder Studio software version **6.1.3** and the Spyder-S Series hardware. Document revision **1.1** (September 2026).

Conventions used in this guide:

Element	Meaning
Note:	Supplementary information.
Tip:	A helpful suggestion, not required.
Prerequisite:	Something that must be true before a procedure.
Notice:	A practice that risks lost work, data, configuration, or equipment damage — no personal-injury hazard.
Menu ▶ Item	A menu path — for example, Server ▶ Connection Manager .
<i>(future release)</i>	A roadmap feature not available in 6.1.3.

Note: The callouts above flag risk to your work, data, configuration, or equipment — never to persons. **Personal-safety hazards** use Christie's product hazard notation — **Danger**, **Warning**, and **Caution**, each with its own symbol and severity meaning — which is defined in Section 2.2 and applies to the safety instructions in Chapter 2 and to the markings on the product. **Notice** carries the same meaning in both places.

Related documents. These Christie documents accompany or supplement this guide:

- **Christie Software License Agreement** — the end-user license for Spyder Studio, presented for acceptance during installation and available separately from Christie (Section 1.8).
- **Christie Digital License Information** — the open-source component licenses (Appendix D).
- The Spyder-S **setup guide** (hardware installation and regulatory), and the **Spyder-S API Commands** reference (external control, Chapter 10).

1.2 Revision History

Revision	Date	Software version	Summary
1.1	September 2026	Spyder Studio 6.1.3	Updated for the 6.1.3 maintenance release: alternate output sync is enabled by default on new configurations (Sections 6.4.1.4 and 7.7.4).
1.0	July 2026	Spyder Studio 6.1.2	Initial release.

1.3 Copyright

Copyright © 2026 Christie Digital Systems USA, Inc. All rights reserved.

1.4 Trademarks

All brand names and product names are trademarks, registered trademarks or trade names of their respective holders.

1.5 General

Every effort has been made to ensure accuracy, however in some cases changes in the products or availability could occur which may not be reflected in this document. Christie reserves the right to make changes to specifications at any time without notice. Performance specifications are typical, but may vary depending on conditions beyond Christie's control such as maintenance of the product in proper working conditions. Performance specifications are based on information available at the time of printing. Christie makes no warranty of any kind with regard to this material, including, but not limited to, implied warranties of fitness for a particular purpose. Christie will not be liable for errors contained herein or for incidental or consequential damages in connection with the performance or use of this material. Our centers of excellence for manufacturing in Kitchener, Ontario, Canada and in Shenzhen, China are ISO 9001:2015 Quality Management System-certified.

1.6 Warranty

Products are warranted under Christie's standard limited warranty, the details of which are available at <https://www.christiedigital.com/help-center/warranties/> or by contacting your Christie dealer or Christie.

1.7 Limitation of Liability

IN NO EVENT WILL CHRISTIE BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, LOST REVENUES AND LOST DATA ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF CHRISTIE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHRISTIE SPECIFICALLY DISCLAIMS ANY WARRANTIES OR CONDITIONS, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE AND ACCOMPANYING DOCUMENTATION, IF ANY, PROVIDED HEREUNDER IS PROVIDED "AS IS". CHRISTIE HAS NO OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

CHRISTIE FURTHER DISCLAIMS ANY AND ALL REPRESENTATIONS RELATING TO THIRD PARTY SOFTWARE WHICH IS EMBEDDED IN THE SOFTWARE. Some jurisdictions do not allow exclusions or limitations of certain conditions or warranties. Only those exclusions or limitations that are lawful in Licensee's jurisdiction will apply to Licensee and, in such instances, Christie's liability will be limited only to the maximum extent permitted by law.

1.8 Software License

Your use of Spyder Studio is governed by the **Christie Software License Agreement** (the end-user license). You accept its terms by downloading, installing, activating, or using the software. The agreement is presented for acceptance during installation and is available as a separate document from Christie or your Christie dealer.

The Software also includes third-party open-source components covered by their own license terms; see **Appendix D**.

2. Safety & Regulatory

Read this chapter before you unpack, install, power, or operate the Spyder-S Frame. It covers the safety practices for handling the hardware, the warning symbols used throughout this guide, the environmental limits the equipment is rated for, and the regulatory approvals it carries.

2.1 General safety instructions

This section states the mandatory safety practices for the equipment: who may install and service it, electrical-shock and grounding precautions, lifting and rack-mounting precautions, ventilation requirements, and the conditions under which the unit must be disconnected from power.

- **Qualified personnel.** All service instructions are for use by **qualified service personnel only**. There are **no procedures, exchanges, or parts inside the unit intended to be performed by the user**. Unplug all power and power-supply cords from the apparatus before servicing.
- **Electrical safety.** Disconnect all power sources before performing maintenance or service. The Spyder-S has **redundant power supplies** (Spyder-S datasheet), so fully de-energizing the unit requires **unplugging all power and power-supply cords** from the apparatus, not just one. Maintain **reliable earthing** of the rack-mounted equipment, giving particular attention to supply connections other than direct connections to the branch circuit (for example, power strips). When connecting the unit to the supply circuit, consider the effect that overloading the circuits could have on overcurrent protection and supply wiring, and observe the equipment nameplate ratings.
- **Mechanical / handling.** The Spyder-S is a 1RU rack-mount unit weighing up to 8.5 lb (3.85 kg) — light enough for single-person handling. Support the chassis fully when racking, and mount the equipment so that uneven mechanical loading does not create a hazardous condition.
- **Ventilation & placement.** Do not block the chassis intakes or exhausts, and provide adequate airflow. Air flows **front to back** through the chassis — leave clearance at both the front (intake) and rear (exhaust) when racking. In a closed or multi-unit rack, the operating ambient temperature may exceed room ambient — install the equipment in an environment compatible with the maximum ambient temperature (T_{ma}) specified by Christie, and mount it so the airflow required for safe operation is **not compromised**.

2.2 Symbols & conventions

The table below is Christie's official hazard notation — each signal word, its symbol, and its meaning. These signal words denote **hazards to persons or property** and carry the severity meanings defined here wherever they appear in this chapter or on the product itself.

Danger, **Warning**, and **Caution** are reserved for hazards to persons and appear only in this chapter and on the product. Elsewhere in the guide, a practice that risks work, data, or equipment — but not injury — is flagged **Notice**, with the same meaning it carries here. The remaining documentation callouts (**Note**, **Tip**, **Prerequisite**) are listed in Section 1.1.

Symbol	Signal word	Meaning
	Danger!	If not avoided, the following will result in death or serious injury.

Symbol	Signal word	Meaning
	Warning!	If not avoided, the following could result in death or serious injury.
	Caution!	If not avoided, the following could result in minor or moderate injury.
	Notice.	If not avoided, the following could result in property damage.
	Information	Information messages provide additional information, emphasize, or provide a useful tip.

Note: The Spyder-S chassis and packaging may also carry regulatory and handling symbols — for example the protective-earth mark. The product labelling is authoritative for the complete set of symbols printed on the product.

2.3 Environmental specifications

This section gives the environmental envelope the Spyder-S is rated to operate and be stored in.

Specification	Value
Operating temperature range	0 to 40 °C (32 to 104 °F)
Storage temperature range (power off)	–40 to 60 °C (–40 to 140 °F)
Operating humidity (non-condensing)	10–90%
Maximum operating altitude	<3,000 m (<10,000 ft)
Cooling / airflow direction	Front to back

2.4 Regulatory compliance

The Spyder-S carries regional safety and EMC approvals. For the authoritative marks, refer to the product's regulatory label.

Area	Standard / mark	Reference
Product safety	IEC 62368-1:2018 (CB Scheme); UL 62368-1 & CAN/CSA C22.2 No. 62368-1:19, 3rd Ed.; EN IEC 62368-1:2020+A11:2020	UL Mark, UL File E530289; CB report E530289-A6003-CB-2 (see below)
EMC (Europe)	EN 55032:2015+A11:2020 & EN 55035:2017+A11:2020, EN 61000-3-2, EN 61000-3-3 (Class A)	See EMC certification below

Area	Standard / mark	Reference
EMC (USA)	FCC 47 CFR Part 15 Subpart B, Class A	See FCC notice below
EMC (Canada)	CAN ICES-3 (A) / NMB-3 (A) — ICES-003 Issue 7	See product marking below
EMC (Japan)	VCCI-CISPR 32:2016, Class A	See EMC certification below
Trade compliance (US)	TAA (Trade Agreements Act) compliant	Spyder-S datasheet (CD5108)

FCC (USA) — Class A notice. The following statement is reproduced verbatim from Christie's product documentation:

The product has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the product is operated in a commercial environment. The product generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of the product in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at the user's own expense. Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Canada — ICES. The product carries the marking:

CAN ICES-3 (A) / NMB-3 (A)

Product-safety certification. The Spyder-S is authorized by UL Solutions to carry the **UL Mark** (UL File **E530289**) and is certified under the **IECEE CB Scheme** to **IEC 62368-1:2018** (Audio/video, information and communication technology equipment — Safety requirements), with national certification to **UL 62368-1** and **CAN/CSA C22.2 No. 62368-1:19** (3rd Edition) and evaluation to **EN IEC 62368-1:2020 + A11:2020** for the European market. The certification (UL file **E530289**, CB Test Report **E530289-A6003-CB-2**) covers the video-processor unit as a complete end product — model series **E4F-A** (single power supply) and **E4F-B** (dual/redundant power supplies) — including the DisplayPort and SFP I/O board configurations. The equipment is classified **Class I**, Overvoltage Category II, Pollution Degree 2, IPX0, rated for use up to 40 °C and altitudes ≤ 2,000 m — this is the altitude at which the safety evaluation was performed, not the product's operating limit, which is given in Section 2.3. National differences have been evaluated for the US/Canada, EU, UK, Australia/New Zealand, China, and Saudi Arabia.

Note: The SFP transceiver modules supplied with the SFP I/O board are listed as critical components in the Spyder-S CB test report and carry their own TÜV mark.

EMC certification. The Spyder-S has been EMC-tested as a complete unit (DisplayPort variant **E4F-BD0-71** and SFP variant **E4F-BF0-61**, both with redundant power supplies) and is classified **Class A** for the USA (FCC 47 CFR Part 15 Subpart B), Canada (ICES-003 Issue 7), the European Union (**EN 55032:2015+A11:2020** emissions and **EN 55035:2017+A11:2020** immunity, with **EN 61000-3-2** and **EN 61000-3-3** for mains harmonics and flicker), and Japan (**VCCI-CISPR 32:2016**). Both units tested had dual power supplies; Christie has declared the single-power-supply versions compliant by similarity.

3. Product Introduction & Hardware Overview

This chapter introduces the Spyder-S processor and the Spyder Studio software that drives it, then describes the hardware you will install and operate: the Frame, its inputs and outputs, the rear-panel connectors, and how multiple Frames are linked into one system. It establishes the physical vocabulary the rest of the guide depends on. The processing model built on top of this hardware — canvas, PixelSpaces, layers, Mix Effects — is covered in Chapter 5.

3.1 What is Spyder Studio and Spyder-S?

Spyder-S Series is Christie's fifth-generation Spyder windowing video processor: the hardware that takes physical video inputs, composites and scales them across one or more displays, and drives physical outputs. Unlike a general-purpose computer, Spyder-S is built on **FPGAs and purpose-built hardware**, which is what gives it deterministic, ultra-low latency picture processing.

Spyder-S uses a **modular distributed architecture**: rather than one large chassis, the system is assembled from one or more **Frames** — 1RU units — linked together over a 100G network. Section 3.5 covers what that architecture means in practice.

Spyder Studio is the Windows control software for Spyder-S. It runs on a PC and connects to the processor over the network. Studio is the **client**; the Frame runs the **server** that executes the show.

While Studio is connected, your work reaches that server **as you do it** — applying a configuration, editing a source, building a Command Key all take effect on the Frame directly, with no separate commit step. **Push** (PC → Frame) and **Pull** (Frame → PC) are something different: bulk transfers of an *entire* data set, used for commissioning a Frame or reconciling two sides that have drifted, not for everyday work. Offline, Studio edits a purely local copy that reaches a Frame later. Section 4.8 covers the full model; Chapter 4 covers the connect flow and Chapter 7 live operation.

Two facts shape everything you do in Studio:

- **The hardware defines the limits.** Studio discovers the connected Frames and reports their capabilities — which connectors exist, which outputs can tile, how much bus bandwidth is free. A configuration is valid only if it fits inside the system's resource ceiling. See Chapter 6 for configuration and the resource budget.
- **Studio runs offline on its own.** Studio carries a local engine, so you can build, apply, and save a complete configuration with no Frame and no server connected ("Virtual Mode" — *Apply to local PC*). A separate **Local Remoting Server** option exists only for testing the connected workflow (Push/Pull, live apply) without hardware; see Section 4.5.

Note: Spyder-S automation commands are backward-compatible with the Spyder X80, so existing control integrations carry forward with only a few new commands for new features.

3.2 Frame types and per-Frame specs

A Spyder-S configuration is built from one or more **Frames**. Every Frame provides **4 inputs, 4 outputs, and 4 UHD layers**, with native **10-bit RGB 4:4:4** processing. The Frame model determines the connector type, the input/output **labeling**, and the physical **arrangement** (the rear panels are shown in Section 3.3):

Frame type	Connector / signal	Inputs	Outputs	Arrangement
HDMI 2.0 4x4	HDMI 2.0	A–D	A–D	2x2 stacked (in block + out block)
DisplayPort 1.4 4x4	DisplayPort 1.4	A–D	E–H	single row
SDI 12G 8Ch	Single-Link 1.5 / 3 / 6 / 12G	A–D	E–H	single row
SFP 8Ch	SDI 1.5 / 3 / 6 / 12G or HDMI 2.0	A–D	A–D	single row

Note — finding these models in Studio. The Configuration Manager does not use the public names above. Each Frame model appears there under its internal engineering designation — E4K-BBC-HDMI-4x4, E4K-BBC-DP-4x4, E4K-BBC-SDI-4x4, and E4K-BBC-SFP-4x4 (note the literal x). Match the connector type to pick the right one. See Section 6.3.

Per-Frame specifications. The following are common to a single Frame:

Specification	Per Frame
Inputs / outputs / layers	4 in / 4 out / 4 UHD layers
Super-scale	Input ports 1 and 2 — a layer fed from either scales to 32,768 px wide (Section 7.2.2)
Processing	Native 10-bit RGB 4:4:4
Refresh rates	23.98, 24, 25, 29.97, 30, 50, 59.94, 60 Hz
Color space / sampling / bit depth	YCbCr, RGB / 4:2:2, 4:4:4 (RGB is 4:4:4 only) / 8-bit, 10-bit; 12-bit YCbCr 4:2:2 on HDMI (Appendix E)
HDR	HLG
Resolutions	Up to 4096 × 2160 @ 60 Hz; background inputs up to 7680 × 1080 @ 60 Hz (HDMI) and 7680 × 2160 @ 60 Hz (DisplayPort) — see Appendix E; 7680 × 4320 @ 30 Hz (2 Ch) (<i>future release</i>)
Audio (<i>future release</i>)	Up to 16× channels, 20/24-bit, 44/48 kHz (8× on DisplayPort)
System link	4× QSFP28 — Frame-to-Frame 200G in / 200G out
Control	1× RJ45, 1 Gb Ethernet

Specification	Per Frame
Genlock / sync	Burst/Tri-Level (BBTL) in or out
Power	Redundant power supplies
KVM (future release)	HDMI 2.0 & DisplayPort Frames only; USB-C (USB 2.0 HID) — 4 host / 4 device on HDMI, 2 host / 2 device on DisplayPort

(The 4-in / 4-out count is also auto-filled by Studio from the chosen model in the Configuration Manager Frame Setup. Items marked (future release) — including KVM, audio, and the higher resolutions — are roadmap items and are not available in 6.1.3; max sampling and bit depth are supported per the relevant interface standard.)

3.3 Front and rear panels

This section describes the Spyder-S Frame's front information display and its rear-panel connectors.

3.3.1 Front panel

The **front panel** of a Spyder-S Frame is an **information-only status display**. It currently shows the Frame's **IP address** and the **Christie logo** for identification; additional on-panel status features are a (future release). The display is **not a touchscreen and has no operator buttons** — it shows information only. (The one control on the front panel is the recessed factory-reset button behind the small pinhole under the nameplate; see Section 11.4.) The Spyder-S Frame has no operator console; the system is operated entirely through **Spyder Studio** and external control (Chapters 7–10), not from the Frame itself.



Figure 3-1. Spyder-S Frame — front panel. Rack-mount ears, the information display (IP address and Christie logo), status LEDs, and Christie branding.

3.3.2 Rear panel

Every Frame's rear panel carries the same connectors except for the signal I/O. The **HDMI 2.0 Frame** below shows the full layout, with each connector numbered ①–⑧ on the panel. Across all Frame types, connectors ③–⑧ are **identical** — only the **inputs (①)** and **outputs (②)** differ (Section 3.2).



Figure 3-2. HDMI 2.0 Frame — rear panel (representative layout).

#	Connector	Description
①	Inputs	4 × HDMI 2.0 inputs, labeled A–D, in a 2×2 block.
②	Outputs	4 × HDMI 2.0 outputs, labeled A–D, in a 2×2 block. The LED beside each output's letter lights in the color of the output's mode and is off when the output is Unused (Section 6.5).
③	Genlock IN / OUT	Burst/Tri-Level (BBTL) sync — BNC in and out.
④	USB Host / Device	USB-C Host and Device ports (USB 2.0 HID).
⑤	DIAG HDMI	Diagnostics HDMI port — service and recovery output (see Section 11.4).
⑥	1 Gb CTRL	RJ45, 1 Gb Ethernet — control network.
⑦	4 × QSFP28 (100G)	System-link cages for Frame-to-Frame linking (Section 3.4).
⑧	Power (× 2)	Dual redundant AC power inlets.

Each rear panel also carries **DIAG / Device / Host** service connectors and a protective connector loop. The other Frame types share connectors ③–⑧ exactly as above; they differ only in their signal **inputs** ① and **outputs** ②:



Figure 3-3. DisplayPort 1.4 Frame — rear panel. ① 4 × DisplayPort 1.4 inputs (A–D) and ② 4 × outputs (E–H), in a single row. KVM (future release). Connectors ③–⑧ as in Figure 3-2.



Figure 3-4. SDI 12G Frame — rear panel. ① 4 × 12G-SDI inputs (A–D) and ② 4 × outputs (E–H), in a single row (BNC). Connectors ③–⑧ as in Figure 3-2.



Figure 3-5. SFP Frame — rear panel. ① 4 × SFP inputs (A–D) and ② 4 × outputs (A–D), in a single row. Connectors ③–⑧ as in Figure 3-2.

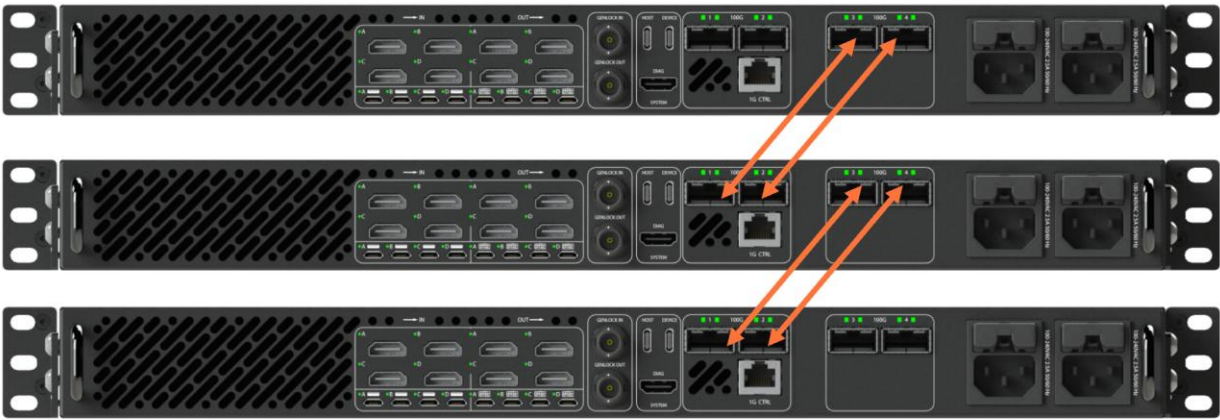
3.4 Linking Frames / Frame Groups

Multiple Frames operate together as a single processing system over the 100G network. Frames link through their **4x QSFP28** connectors (DAC or fiber — single-mode or multi-mode), with **200G in / 200G out** of bandwidth between Frames.

At full scale, a Spyder-S system delivers the following performance and capacity:

Capability	Specification
System	
Frames per system	Up to 16
Inputs / outputs	Up to 64 / 64
Layers	Up to 64
Bus bandwidth (system ceiling)	4.728 Gpps
Software stack	One unified stack and workflow across all Frames
Canvas — per Frame group	
Maximum size	70,778,880 px — e.g. 32,768 × 2,160 (8× 4K) or 98,304 × 720
Size limits	Width ≤ 98,304 · height ≤ 2,160 (2,600 at ≤ 30 Hz) · min 1,280 × 720
Canvas — whole system	
Size	200M+ px — validated in testing, not a hard maximum. Multiple Frame groups tile one canvas; the working limit comes from the bus budget and the per-group raster limits above.
Example	16 Frames as 4 Frame groups, each 6× 4K (4,096 × 2,160): 4 × 53,084,160 = 212,336,640 px
Performance	
Latency	1–2 frames (layers) · 2–3 frames (Mix Effects)
Frame link bandwidth	200G in / 200G out per Frame (4x QSFP28)

Each Frame group renders its own canvas independently, so the combined system canvas can exceed any single group's 70.7-megapixel limit. In the example above, four 6× 4K Frame groups form one 212-megapixel canvas — larger than the bus could drive as a single canvas — because the bus only pays for the widest group, not the sum.



Each orange link line = one QSFP28 (100G) cable
A single QSFP28 system-link cable - DAC or fiber (single-mode or multi-mode). Example shown at right.



Figure 3-6. Three Spyder-S Frames linked via QSFP28. Frames link through their rear 4 × QSFP28 (100G) ports (Section 3.3). The orange lines show the Frame-to-Frame system-link connections — 200G in / 200G out per Frame — that combine the Frames into one canvas. Each orange link line is a single QSFP28 (100G) system-link cable (DAC or fiber, single- or multi-mode), shown in the inset.

Connecting the system-link cables. Each Frame has four QSFP28 link ports, numbered **1–4** on the rear panel (connector ⑦). Cable the Frames **top-down** — from the **parent** Frame to the **child** Frames stacked below it — by one rule: **ports 3 and 4 of each Frame connect to ports 1 and 2 of the next Frame down.**

For a system of **one parent and two child Frames** (as in Figure 3-6):

Connect from	To
Parent Frame — port 3	Child Frame 1 — port 1
Parent Frame — port 4	Child Frame 1 — port 2
Child Frame 1 — port 3	Child Frame 2 — port 1
Child Frame 1 — port 4	Child Frame 2 — port 2

The two ports at the **top** of the chain — **parent ports 1 and 2** — and the two at the **bottom** — **child Frame 2 ports 3 and 4** — are left **open**. Extend the same pattern to add more Frames: each additional Frame's ports 1 and 2 receive ports 3 and 4 from the Frame above it.

In Studio, the mechanics of adding Frames, setting each Frame's rectangle on the shared canvas, and combining Frames into one display surface are part of configuration. The **Frame Groups** concept — how Frames combine into one or more canvases, extend the total canvas beyond a single group, and trade canvas bandwidth for Mix Effects — is introduced in Chapter 5 (Section 5.9.1); the step-by-step configuration is documented in Chapter 6.

3.5 The modular distributed architecture

Most video processors in this class are built as a single large chassis: you buy the capacity you might eventually need, house it, and power it from the first day. Spyder-S is built the other way round. The system is a **fleet of compact 1RU Frames** that link over the 100G system network to behave as one processing system — one canvas, one software stack, one operator workflow. That architecture is what delivers the platform's **flexibility, scalability, and performance**.

A Spyder-S system is sized by the number of Frames it contains — anywhere from a single 1RU Frame up to sixteen, giving 64 inputs, 64 outputs, and more than 200 million pixels on one canvas (Section 3.4). What any given system can do depends on how those Frames' resources are allocated across canvas, Aux outputs, layers, Mix Effects, and clones (duplicate copies of a layer's content — Section 7.2.3).

Figure 3-7 shows **three example configurations** from that range — one Frame, three Frames, and six Frames — to illustrate how capability grows as Frames are added:

- **The one Frame example.** Drives a **3 UHD canvas** with **1 UHD Aux** output, and provides up to **4 layers**, **6 UHD Mix Effects**, and up to **16 UHD clones** — substantial production capability in a single rack unit.
- **The three Frame example.** A **5 UHD canvas** with **3 UHD Aux** outputs, up to **12 layers**, **4 UHD Mix Effects**, and up to **48 UHD clones**.
- **The six Frame example.** An **8 UHD canvas** with **6 UHD Aux** outputs, up to **24 layers**, **4 UHD Mix Effects**, and up to **96 UHD clones**. This example splits its Frames into **two Frame groups**: because canvas bandwidth tracks the widest group rather than the whole canvas, that split is what leaves bus bandwidth free for the Mix Effects (Section 5.9.1, Appendix C).

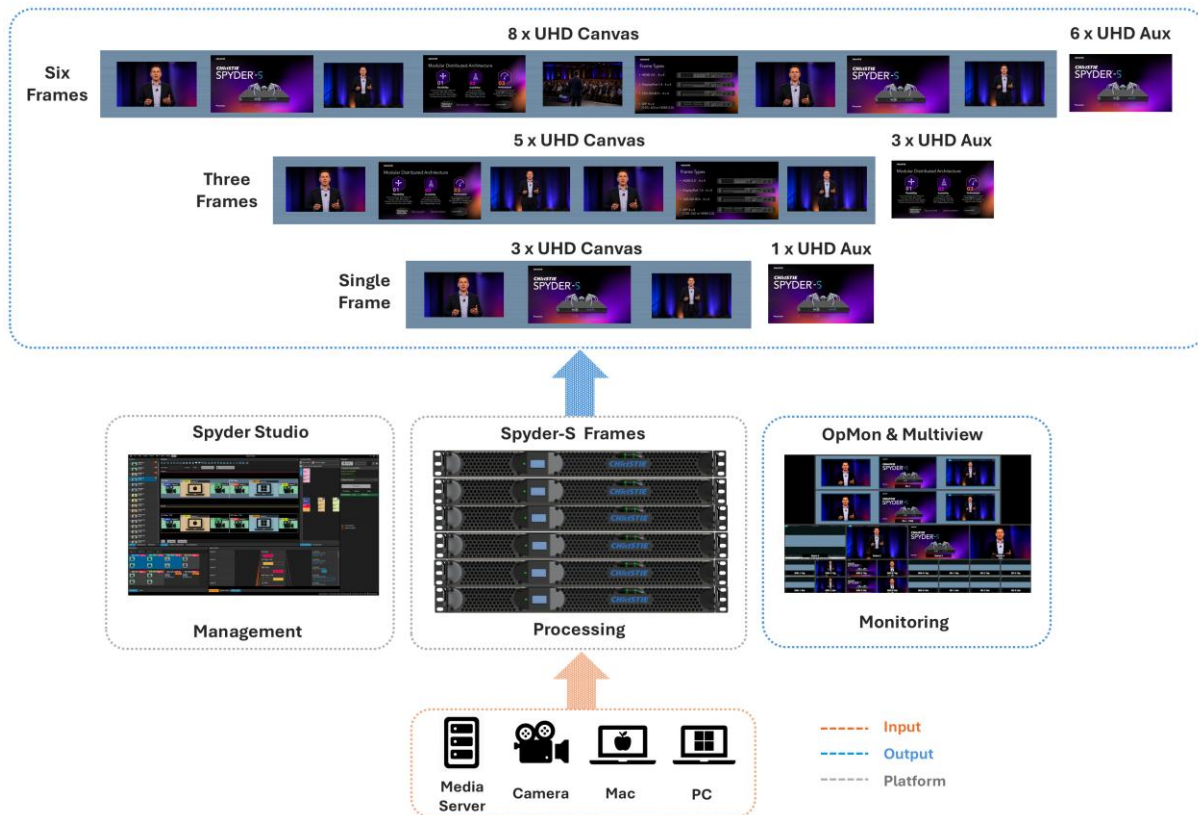


Figure 3-7. Three example configurations — one, three, and six Frames on the same architecture.

These are example allocations, not fixed specifications — the same Frame count can be configured differently, trading canvas width against Aux outputs or Mix Effects to suit the job. What does not change across them is the software or the operator workflow; only the amount of work the system can do.

Note: Mix Effect capacity does not rise with Frame count. The canvas and the Mix Effects draw on the same bus, so a wider canvas leaves less bandwidth for Mix Effects — which is why one Frame on a 3 UHD canvas supports six while six Frames driving 8 UHD support four. Splitting Frames into groups is how a large canvas recovers Mix Effect capacity; Section 5.9.1 explains the trade and Appendix C works through the arithmetic.

Each characteristic follows from the architecture.

Flexibility. Because I/O lives in the Frames rather than in fixed slots, a system carries exactly the connectivity a job needs. Frame types can be mixed in one system, so DisplayPort sources, SDI cameras, HDMI playback, and SFP fiber runs coexist without external converters (Section 3.2). Sources, layers, backgrounds, Mix Effects, and the four output roles — Canvas, Aux, OpMon, and Tiled — all draw on the same pool, so the same platform serves a single-screen breakout and a wide LED canvas.

Scalability. A single Frame provides 4 inputs, 4 outputs, and 4 UHD layers. Add Frames and those totals scale with them, up to **16 Frames** giving **64 inputs and 64 outputs** and more than **200 million pixels on one canvas** (Section 3.4, Appendix C). Capacity tracks the work: start with one Frame and add Frames as the jobs grow, rather than buying a fully populated chassis up front. Because the software and the workflow are identical at every size, an operator who learns a one-Frame system already knows a sixteen-Frame one.

Performance. Every Frame runs the same FPGA and purpose-built hardware rather than general-purpose computing, so latency is deterministic — 1–2 frames for layers, 2–3 frames for Mix Effects — and every Frame in the system delivers identical image quality and timing. Adding Frames adds capacity without changing how the system performs.

The resource and bandwidth model that decides whether any particular configuration is valid is covered in Section 5.9, with the arithmetic in Appendix C.

3.6 Package contents & rack mounting

Each Spyder-S Frame is a **1RU** unit measuring **18.75 × 8.24 × 1.73 in (476.25 × 209.29 × 43.94 mm)** and weighing **up to 8.5 lb (3.85 kg)**. Mount the Frame in a standard 19-inch rack, leaving adequate airflow clearance front and rear (see the handling and ventilation warnings in Chapter 2).

Power and environment requirements:

Specification	Value
Power	110–240 V, 220 W typical, up to 240 W max; redundant power supplies
Operating temperature	0 to 40 °C (32 to 104 °F)
Storage temperature (power off)	–40 to 60 °C (–40 to 140 °F)
Operating relative humidity	10–90%, non-condensing
Operating altitude	< 3,000 m (< 10,000 ft)

Package contents. Each Spyder-S Frame ships with:

- The **Spyder-S Frame**

- **2× power cords**
- **2× QSFP28 system-link cables**
- **A rack mount kit**
- **A USB drive** carrying the Spyder Studio installer and installation instructions (see Section 4.4)

4. Getting Started & Installation

This chapter takes you from an unpacked Spyder-S Frame to a running Spyder Studio session. It follows the natural order: install and power the hardware, put the Frame and PC on the same network, install the Studio software, connect Studio to the hardware, and then hand off to configuration.

If you are evaluating Studio without hardware in front of you, you can skip straight to Section 4.4 (install the software) and then build a configuration **offline** — Studio needs no Frame and no server to do that (Section 4.5, *Work offline*).

4.1 Physical installation

Mount the Frame and connect its video sources and displays.

The Spyder-S Frame is a **1RU rack-mount** chassis weighing **up to 8.5 lb (3.85 kg)** — light enough for a single-person install. Mount it in a standard 19-inch rack and leave clearance front and rear for airflow; the Frame draws cooling air through the chassis, so do not block the intakes or exhausts. Keep the unit within its operating temperature range of **0–40 °C (32–104 °F)**. See the handling and ventilation precautions in Chapter 2 and the rack specifications in Section 3.6.

Cabling order. A typical connection sequence is: (1) connect sources to the **input** connectors (HDMI/SDI **A–D**); (2) connect displays to the **output** connectors (HDMI **A–D** or SDI **E–H**); (3) connect an HDMI monitor to the **DIAG** port — with a keyboard and mouse on a USB hub at the USB-C **Device** port — for on-Frame diagnostics; (4) connect the **system-link** cables between Frames, port **3** → **1** and port **4** → **2** on the next Frame down (Section 3.4); (5) connect the **1 Gb control** network; (6) connect **power** (both supplies, ideally on separate circuits, for redundancy). Observe the mechanical-loading and circuit-overloading precautions in Chapter 2 when racking.

4.2 Power

Connect and apply power to the Frame.

The Spyder-S Frame uses **auto-ranging power supplies that accept 110–240 V**, so no voltage selector is needed. Power draw is **220 W typical, up to 240 W maximum**. The Frame ships with **redundant (dual) power supplies**. The Frame **runs on a single supply** — you only need both connected for **redundancy**. Connect both supplies, ideally to separate circuits, so the Frame keeps running if one input or supply fails. See the electrical-safety precautions in Chapter 2.

4.3 Network configuration

Spyder Studio reaches the Frame over an IP network, so the control PC and the Spyder server must be on the same reachable subnet.

Each Frame provides **one RJ45 1 Gb Ethernet port for control** — this is the connection Studio uses to discover and manage the Frame. Do not confuse it with the **system links**, which are **4x QSFP28** ports (DAC or fiber) used to interconnect Frames; those carry the inter-Frame system link, not Studio control traffic.

The connection is IP-based. Studio discovers servers on the network and shows the current server's IP in the status bar (for example, 192.168.50.96), alongside the PC's local IP — both are read out in the Connection Manager (Section 4.5).

A workable order is:

1. Put the control PC and the Frame on the same subnet (direct cable, switch, or managed network).
2. Confirm the PC has a valid IP on that subnet.
3. Launch Studio and open the Connection Manager to discover the server.

Assigning the Frame's IP address. Out of the box the Frame uses **DHCP** to obtain an address automatically. Christie recommends assigning a **static IP** instead, set from the Frame's on-screen Diagnostics:

1. Connect an HDMI monitor to the **DIAG** port and a keyboard and mouse via a USB hub on the USB-C **Device** port.
2. Move the mouse to the bottom of the screen and click **Settings** to open the **Network Settings** tab (which shows the **1G-AUX** information).
3. Turn **off DHCP** so the unit does not auto-acquire an address.
4. Enter the **IP address**, **Subnet mask** (typically 255.255.255.0), and **Gateway** for the device.
5. Click **Apply Settings**.

On the control PC, set the network card to an IP on the **same subnet** as the Frame; if Studio is already open, restart it for the change to take effect.

Note — network design. Spyder communicates with its clients over **broadcast traffic**, which can be heavy on an active network. Christie recommends a **closed network** for the Frame and all Studio clients, with unrelated high-traffic devices on a separate network. Avoid complex routers or managed switches unless they are correctly configured — misconfiguration can interrupt communication and break the client socket (restart the client application to restore it). On the PC, avoid a network card configured with multiple IP addresses or an enabled secondary wired/wireless connection.

Ports. External control uses the Frame's UDP control port, **11116** by default (Section 10.2); the Local Remoting Server listens on **TCP 11112** by default (Section 4.5).

4.4 Installing and updating the software

Spyder Studio is a Windows application installed on the control PC.

Minimum PC requirements. Install Spyder Studio on a computer with a display resolution of **1920 × 1080 or greater**.

Requirement	Minimum
Operating system	Windows 11, 64-bit
CPU	2.2 GHz dual-core
RAM	8 GB (16 GB recommended)
Disk	2 GB (plus space for user configurations and still-image files)

Installing.

1. Obtain the installer from **www.christiedigital.com** or the supplied Christie USB device. Christie recommends obtaining the latest version after receiving the product.

2. Close all other programs.
3. Double-click **Spyder Studio <version> Setup.exe**.
4. Install any prerequisite programs, then follow the Spyder Studio installation wizard.
5. To get the latest official release later, use **Help ► Check for updates** inside the software (Section 4.4.1).

Note: The application's data path is hard-coded to **C:\SPYDER**. All users must have full control (read and write access) to both the installation directory and **C:\SPYDER**.

Once installed, launch Spyder Studio. From here you can either **build a configuration offline** right away — no Frame or server required (Section 4.5, *Work offline*) — or **connect to a Frame** (Section 4.5) to drive real hardware. Either way, the next step is to build a configuration (Chapter 6).

4.4.1 Update Spyder Studio

Help ► Check for updates updates the Spyder Studio application on this PC — not the Frames. The PC needs internet access. The check follows the update channel set by **ApplicationUpdateChannel** (**Release** or **Beta**, Section 4.9.1).

1. Choose **Help ► Check for updates**.
2. If a newer version is available, **Update Available** asks whether to download it (Figure 4-1). Click **Yes**. The download runs in the background and is saved to your **Downloads** folder.
3. When the download finishes, **Install Update Now?** shows where the file was saved. Click **Yes** to start the installer, then follow the installation wizard.
4. After Spyder Studio is updated, its version no longer matches your Frames. Update each Frame from Connection Manager (Section 4.4.2).

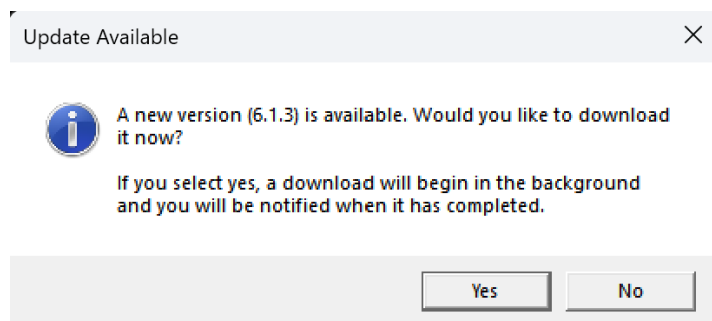


Figure 4-1. The Update Available message from **Help ► Check for updates**.

The check can also end with one of these messages:

Message	Meaning
No Updates Available	You are already using the latest version.
Update Check Failed	Studio could not reach the update server. Check the PC's internet access and try again later.
Download Failed	The update could not be downloaded. Try again later.

4.4.2 Update a Frame's firmware

After you update Spyder Studio (Section 4.4.1), update the firmware of your Frames to match it. When a Frame's firmware version doesn't match your Spyder Studio, Connection Manager flags the Frame's row **red**, and you must update the Frame before you can connect to it. Connection Manager lists the Frames on the network; Section 4.5.1 describes it in full. Studio updates the selected master Frame and any Frames linked to it that need the new version.

Prerequisite: Every Frame in the system is reachable on the network. If one is not, the button reads **Unable to connect** and the update cannot start (Section 11.3). Allow about 15–20 minutes, and have the system password ready if security is enabled (Section 4.6).

Notice: Do not close Spyder Studio, and do not power off any Frame, while the update is in progress.

1. Open **Server ► Connection Manager** and select the Frame's red row.
2. Click **Update Now**. If security is enabled, enter the password.
3. The **Update Confirmation** dialog shows the master Frame's IP address, the new version, and — when more than the master needs updating — the Frame IDs to be updated. Click **Yes**.
4. Wait for the update to finish. The button reads **Update in Progress...**, and the **Updating Frame** section shows an estimated time remaining, counting down from 15 minutes, and a progress bar (Figure 4-2). Studio backs up the Frame's data before the update and restores it afterwards, so your configuration is kept.
5. When **Power Cycle Required** appears, click **OK** and power-cycle the Frame: disconnect its power for about 5 seconds, then reconnect it. For a system with several Frames, disconnect power from all of them, reconnect, and power the master Frame on about 15 seconds after the others.
6. When the Frame is back on the network, its row is no longer red. Select it and click **Connect** (Section 4.5.1).

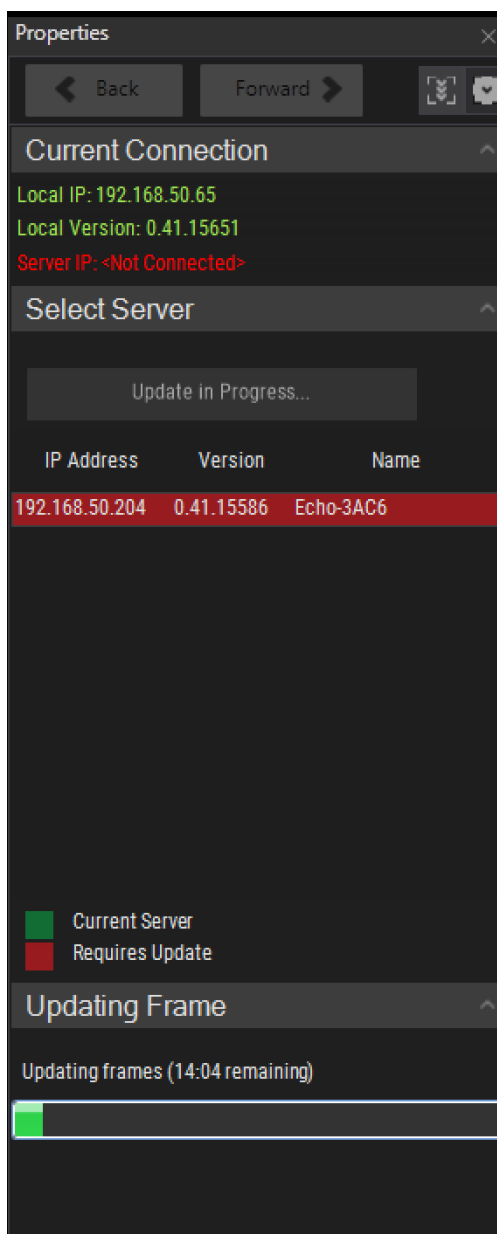


Figure 4-2. The Connection Manager during a firmware update — Update in Progress... and the Updating Frame countdown.

If the update fails, **Update Failed** reports *Failed to update remote frame(s)*. Check the Alert Viewer for details (Section 11.1).

4.5 Connecting Studio to hardware

Studio carries its own **local engine**, so it can build, apply, and save a configuration entirely **offline** — no server or Frame required (see *Work offline*, below). To drive real hardware, you connect Studio as a **client** to a Spyder **server** — a Frame on the network — and synchronize data with it.

4.5.1 Connect to a Frame

Open **Server ▶ Connection Manager**. It renders in the Properties panel (not as a modal dialog). Normally it shows two sections, **Current Connection** and **Select Server**; the third, **Updating Frame**, appears only while a firmware update runs:

Section	What it shows
Current Connection	The connected server, the PC's Local IP, and the local version. The active server is highlighted green .
Select Server	The list of discovered servers on the network, with the Connect / Disconnect button. Right-click ▶ Update Server List Now to re-scan. A Frame whose firmware needs updating is flagged red .
Updating Frame	A progress bar and status shown while a Frame firmware update runs (Section 4.4.2).

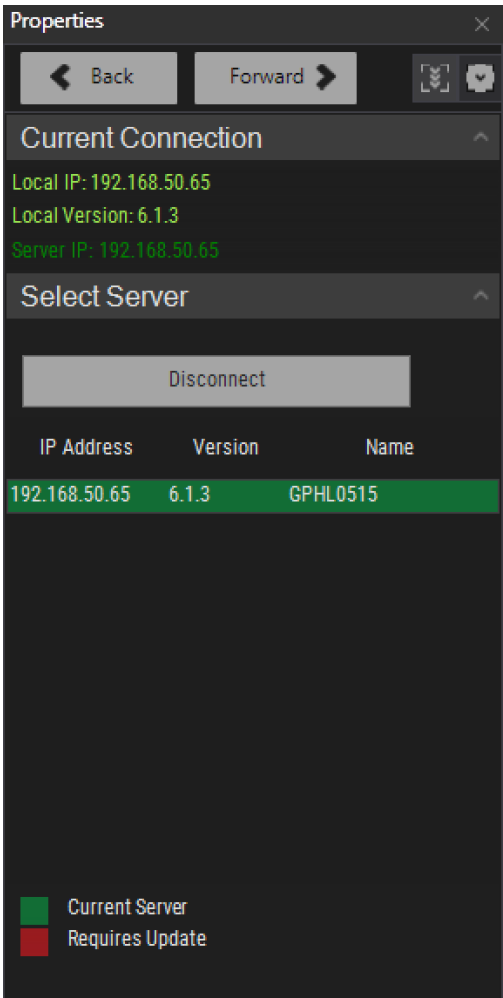


Figure 4-3. The Connection Manager — Current Connection and Select Server.

Pick a server and click **Connect**. The status bar then shows the connected server's IP.

Note: If a Frame's firmware version doesn't match the Studio client, its row is flagged **red** ("Requires Update") and the **Connect** button becomes **Update Now** — update the Frame before connecting (Section 4.4.2). The **green** swatch marks the current server.

Once connected, your work goes to the Frame as you do it. Applying a configuration, editing a source, sizing a layer, building a Command Key — each takes effect on the connected server directly. There is no commit step for ordinary work, and you do **not** need Push or Pull to keep the Frame in step with what you are doing. This is the normal live workflow, and it is what Chapter 7 assumes throughout.

The Server menu's transfer commands are a separate, deliberate path — they move an *entire* data set in one direction, and are enabled only when a remote server is connected:

- **Push Data To Server** (PC → Frame) — overwrites server data; you can include/exclude Inputs (EDIDs), Outputs, and Stills.
- **Pull Data From Server** (Frame → PC) — copies all data the other way and restarts the local system manager.
- **Perform HotSync** — copies the primary frame's data to a backup Spyder system, overwriting the backup.

Notice: Push and Pull each overwrite a full side. Back up your data (File ► Backup) before either. (Full sync behavior: Section 4.8.)

4.5.2 Work offline (no Frame or server needed)

You do **not** need a Frame — or any server — to build a configuration. Studio runs its own engine locally, so the full offline workflow happens in Studio alone:

1. Choose **File ► New** and build the Frames, canvas, PixelSpaces, and outputs (Chapter 6). Offline, the Frame set is **editable** — you pick the Frame models by hand.
2. **Apply** the configuration. With nothing connected, the Apply step reads **Apply to local PC now (Virtual Mode)** (instead of *Apply to Spyder hardware now*) — Studio builds the configuration locally on your PC.
3. Program layers, Mix Effects, Command Keys, and the rest against the local model.
4. Save your work to a ***.sprojx project file** (**File ► Backup**, Section 4.8.1), and reopen it later with **File ► Restore**.

This is the standard offline workflow: design a system away from hardware, save it as a project, and apply it to a Frame later. **"Virtual Mode"** is simply this disconnected state — *Apply to local PC* builds the configuration locally instead of committing it to a Frame.

4.5.3 Optional — the Local Remoting Server

The **Local Remoting Server** is a separate **Debug** option. **You do not need it for offline configuration work.** It runs Studio's local engine as a network server (NewRemoting TCP **11112**) so Studio can connect to it *as if it were a Frame* — letting you exercise the **connected** path (Push/Pull, HotSync, live apply, remote server events) without physical hardware.

1. Enable the Debug menu if it is hidden: open **Tools ► Options**, expand **Advanced ► User Diagnostics**, and set ShowDebugMenuItem to **True** (Section 4.9.1).
2. Choose **Debug ► Start Local Remoting Server**, then connect to it from **Server ► Connection Manager** like any server.

Note: Some menu items are hardware-gated and stay hidden even when connected to the Local Remoting Server (the expansion-link tools, for example), because they require supporting hardware.

4.6 Securing the system

Server ▶ Security enables **authentication** and sets a **password** on the system, so an unauthorized client can't connect and change it. Once you've connected Studio to a Frame (Section 4.5), this is one of the first things worth setting up — before handing the system over for live use.

Field	Purpose
Enable authentication requirement	Toggle switch — turn password protection on or off.
New Password / Confirm Password	Set or change the password. Must be at least 4 characters . Greyed until the toggle is on.
Submit / Cancel	Apply, or dismiss the dialog without changing anything.

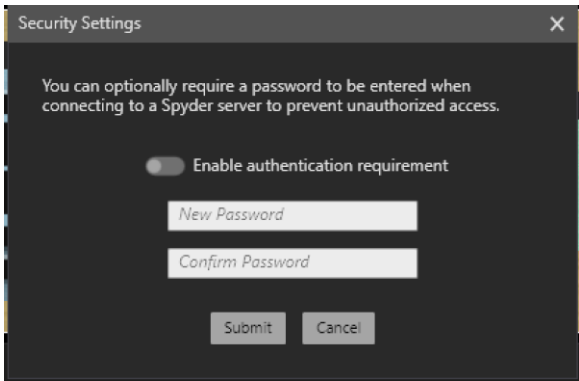


Figure 4-4. The Security dialog — enable authentication and set the connection password.

Submit stays clickable whether or not the authentication toggle is on; it is the **New Password** and **Confirm Password** fields that grey out until the toggle is enabled. The two entries are validated against each other when you click Submit.

1. Turn on **Enable authentication requirement**.
2. Enter a **New Password** (4 characters or more) and re-enter it in **Confirm Password**.
3. Click **Submit**.

Result: If New and Confirm differ, Studio shows *"The passwords you entered did not match."* and does not apply the change. Otherwise the password is set.

Notice: There is no password-recovery path in this dialog. Record the password through your own secure process before you click Submit — the guide documents the fields, not your credentials.

4.7 First configuration

Whether you are working offline or connected to a Frame, you define the system the same way. Choose **File ▶ New** to open the Configuration Manager and build your Frames, canvas, PixelSpaces, and outputs.

Before you start configuring, read **Chapter 5 (Concepts & Architecture)** so the terms — canvas, PixelSpace, layer, Mix Effect, background — are clear. Then follow **Chapter 6 (Configuration)** for the step-by-step build, including the All In One and Free Form paths and the resource/bandwidth budget. Live operation is covered in **Chapter 7**.

4.8 Saving, restoring, and syncing projects

Spyder Studio uses a **client/server model**. Studio (the PC client) works against the configuration and programming data; the Frame runs the server that executes the show.

While you are connected to a Frame, your work goes to the server as you do it. Applying a configuration, editing a source, building a Command Key, saving or updating a configuration — these take effect on the connected server directly. You do not need Push or Pull for ordinary work, and there is no separate "commit" step for everyday edits.

Push, Pull and HotSync are bulk data-set operations, not the normal path. Reach for them when you need to move an *entire* data set between a PC and a Frame — commissioning a new Frame from a PC-built show, bringing a PC into line with the Frame that holds the authoritative show, or bringing a backup Spyder system into line with the primary. They act on everything at once, which is exactly why they are dangerous (Section 4.8.2).

Working offline is the exception. With no server connected, Studio edits a purely local copy (Section 4.5, *Work offline*). That work reaches a Frame later — either by connecting and pushing it across, or by saving a *.sprojx project and restoring it on a machine that is connected.

Read this if you operate live. A **backup** (*.sprojx) is a portable file snapshot; **Push/Pull** overwrites a whole side of the client/server pair. Confusing the two is how shows get lost. Sections 4.8.1 and 4.8.2 draw the line.

4.8.1 Backing up and restoring projects

File ▶ Backup and **File ▶ Restore** save or reload the **entire project** as a single file. This is **local file backup** — a portable snapshot written to disk on the PC. It is distinct from **Server ▶ Push Data To Server** and **Pull Data From Server** (Section 4.8.2), which synchronizes data between the PC and the Frame.

A backup file is a ***.sprojx** project ("Spyder Project"). It captures the **configuration plus programming data** — Frames, PixelSpaces, outputs, sources, Mix Effects, Command Keys, Function Keys, scripts and treatments.

Backing up a project. Prerequisite: None. Backup writes a local file and does not require a connected Frame.

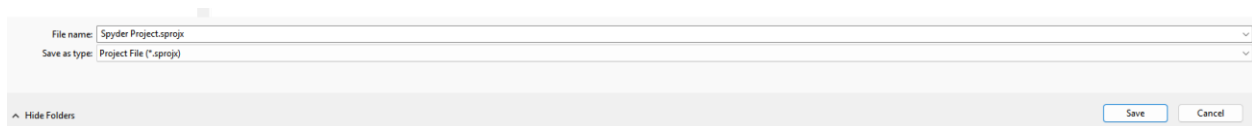


Figure 4-5. File ▶ Backup — the native Save dialog for a *.sprojx project file.

1. Choose **File ▶ Backup**. A **Save** dialog opens, defaulting to the name *Spyder Project* and the *.sprojx file type.
2. Choose a location and name, then save.
3. A busy dialog runs the backup.

Result: The project is written to the chosen *.sprojx file. Studio records the file name, the date and time, and the fact that it was a backup. The status bar "**Last Backup: ...**" read-out reflects this.

Tip: Back up before any Push, Pull or HotSync, and before any change you might want to walk back. A *.sprojx is small, portable, and the fastest way to return to a known-good state.

Restoring a project. Prerequisite: A *.sprojx backup file.

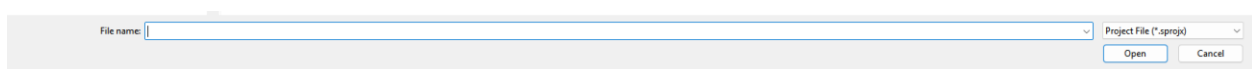


Figure 4-6. File ▶ Restore — the native Open dialog.

1. Choose **File ▶ Restore**. An **Open** dialog opens.
2. Select the *.sprojx project file and open it.
3. A busy dialog restores it.

Result: The project is reloaded from the file. Studio records the file name, date/time, and that it was a restore.

Notice: Restore replaces the current local project. Back up the current work first if it is not already saved.

4.8.2 Push and Pull data

Server ▶ Push Data To Server and **Server ▶ Pull Data From Server** synchronize data between the PC client and the Frame. Unlike Backup/Restore (which moves data to and from a file), Push/Pull moves data **between PC and Frame**, and each direction **overwrites the destination side wholesale**.

Notice: Both Push and Pull overwrite an entire side. Always run **File ▶ Backup** (Section 4.8.1) before either operation — a single *.sprojx is enough to recover.

These items are greyed out until a **remote** server is connected (Section 4.5).

Push Data To Server (PC → Frame). Copies **local PC data to the Spyder hardware**, overriding the server's configuration and programming data. The dialog warns: *"This action will copy local data files from your PC (client) to the Spyder hardware (server). It will override all configuration and programming data."*

The dialog is titled **Confirmation**, opens with *"WARNING! Creating an online Backup File before proceeding is advised."*, and is answered with **Yes** or **No**. Three checkboxes under *"Include the following data (may cause momentary video disruption)"* let you add data types to the push. **All three start unticked**, so a plain push sends neither EDIDs, nor output configurations, nor stills:

Checkbox	Data pushed
Input EDIDs	EDIDs
Output Configurations	Output configurations
Still Image Files	Background stills stored on the Frame

Note: "Input EDIDs" controls the **EDIDs** (Section 6.9), not input signal settings. Ticking any of the three may cause momentary video disruption on the affected outputs while the push applies — the dialog says so itself.

Result: Studio copies the data and then **restarts the remote system manager itself** — there is no second prompt and nothing to accept. It reports **Synchronization succeeded.** in an **Information** box (that spelling is the product's), with a **Don't show this message again** option; tick that and later pushes finish silently. If the restart does not come back, Studio reports **Failed to restart remote system manager after pushing data to server.** — see Section 11.3.

Pull Data From Server (Frame → PC). Copies **all** data from the Frame back to the PC, overwriting the local copy. After a confirmation, Studio copies remote to local and restarts the local system manager.

Result: The PC's local project matches the Frame. Use Pull when the Frame holds the authoritative show and you need to bring a PC into sync with it.

Restart Remote Frame. Server ▶ Restart Remote Frame reboots the connected Frame. This item is available only when a remote Frame is connected. Studio confirms (*"Are you sure you want to restart the current remote frame?"*), then restarts the Frame and disconnects. Use it after changes that require the Frame to re-initialize.

4.8.3 HotSync

Server ▶ Perform HotSync synchronizes a **primary** Spyder system with a **backup** system. The dialog states it plainly: the data on the backup Spyder system is **overwritten** with the current data on the primary frame.

The **HotSync** dialog has a **Backup Server** dropdown, two **Options** — **Push Routers** and **Apply Output Configs** — and a **Perform HotSync to Backup Server Now** button, with **Close** and **Apply** at the bottom.

HotSync remembers the options you used last time (stored in user options and managed by the app, not exposed in the Options grid — see Section 4.9).

Note: Like Push and Pull, HotSync acts on a connected remote server and is unavailable offline. Back up before running it (Section 4.8.1).

4.9 Application options

Tools ▶ Options opens the user options in the **Properties** panel — the application-wide preferences that tailor Spyder Studio to how you work. Options are **per-user**, **auto-save on change**, and persist across sessions; there is no separate Apply button for the panel as a whole.

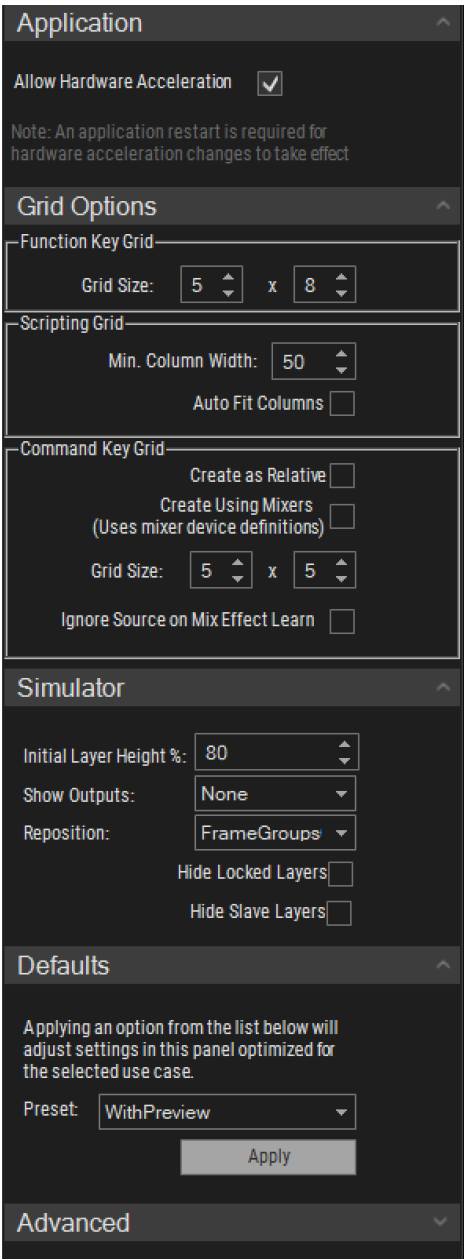


Figure 4-7. The Options grid — per-user application preferences.

The panel has five collapsible sections:

Section	What it controls
Application	Allow Hardware Acceleration — GPU acceleration for parts of the interface. Turn it off if you see UI rendering glitches. The panel notes that a restart is required for a change here to take effect.
Grid Options	The dimensions and creation defaults of the three grids: Function Key Grid (Grid Size, default 5 × 8), Scripting Grid (Min. Column Width, default 50; Auto Fit Columns), and Command Key Grid (Grid Size, default 5 × 5; Create

Section	What it controls
	as Relative; Create Using Mixers; Ignore Source on Mix Effect Learn).
Simulator	How the Simulator draws: Initial Layer Height % (default 80), Show Outputs (default None), Reposition (default FrameGroupsOnly), Hide Locked Layers (a layer is locked when a background input on a higher-numbered Frame in the same frame group would cover it — Section 6.4.1.3), Hide Slave Layers (when ticked, the Simulator draws only primary layers, not their slave layers — Section 9.7).
Defaults	A Preset dropdown plus Apply . Applying a preset adjusts the settings in this panel for a chosen use case — the panel's own wording is <i>"Applying an option from the list below will adjust settings in this panel optimized for the selected use case."</i>
Advanced	The full underlying settings list — see below.

4.9.1 Advanced settings

Advanced expands into a raw property grid of every stored application setting, grouped by area (Application Settings, Command Keys, Configuration Manager, Connection Settings, Display Simulator, Function Keys, Script, User Diagnostics). The friendly controls in the sections above are shortcuts onto the same underlying values — for example the **Allow Hardware Acceleration** checkbox writes the AllowHardwareRendering setting, and the **Simulator** section's Hide Locked / Hide Slave checkboxes write HideLockedLayers / HideSlaveLayers.

Use Advanced when you need a setting that has no dedicated control. Select a setting to see a short description of it below the grid. Clicking a number, text or drop-down value opens it for editing; press **Esc** to leave it without changing it.

The tables below list all 48 settings, by group. **Default** is the value Spyder Studio starts with; applying a **Defaults** preset (above) adjusts some of these settings.

Application Settings

Setting	Default	Values	What it does
AllowHardwareRendering	True	True / False	Lets parts of the interface use graphics hardware acceleration when it is available. Set it to False if you see rendering problems. The same setting as Allow Hardware Acceleration (Application section) ; a restart is needed for a change to take effect.
AlwaysSaveTestPatternToStillList	False	True / False	Always saves a generated test pattern to the Stills list. Not supported in Spyder Studio 6.1.3. Leave this setting at its default.

Setting	Default	Values	What it does
AlwaysShowOutputBlends	False	True / False	Shows the output blending panel whatever the output mode. Not supported in Spyder Studio 6.1.3. Leave this setting at its default.
ApplicationUpdateChannel	Release	Release, Beta	Which update channel Help ► Check for updates follows.
AutoLoadLayout	True	True / False	Restores the saved window layout at startup.
DeleteLocalStills	True	True / False	When you delete background stills on the connected server, also deletes this PC's cached copies of them.
FlashAlertsTabOnEventLevel	Warning	Fatal, Error, Warning, Information, Debug, Trace	Flashes the Alert Viewer tab when an alert at or above this level arrives and the panel is not visible (Section 11.1).
ImmediateSwitchBackupSource	False	True / False	Switches a source to its backup instantly, with no transition (Section 7.1.4).
IncludeStillsInBackup	True	True / False	Includes background stills in a project backup (.sprojx) file (Section 4.8.1).
InvertPatchLayers	False	True / False	Uses console colors for grids such as Sources, Stills and Command Keys.
MaxBasicPresets	100	Number	Maximum number of basic presets. Not supported in Spyder Studio 6.1.3. Leave this setting at its default.
MaxEffects	100	Number	Maximum number of effects.
MaxPlayItems	100	Number	Maximum number of play items. Not supported in Spyder Studio 6.1.3. Leave this setting at its default.
MaxSalvos	100	Number	Maximum number of salvos.
MaxSources	100	Number	Maximum number of sources.

Setting	Default	Values	What it does
MaxStills	100	Number	Maximum number of stills.
MaxTreatments	100	Number	Maximum number of treatments.

Command Keys

Setting	Default	Values	What it does
CmdKeyHeight	5	Number	Number of command keys shown vertically. The second Grid Size value under Command Key Grid .
CmdKeyWidth	5	Number	Number of command keys shown horizontally. The first Grid Size value under Command Key Grid .
DefaultMixers	False	True / False	New command keys are learned as mixers. The same setting as Create Using Mixers .
DefaultRelative	False	True / False	New command keys are relative. The same setting as Create as Relative .
IgnoreSourceOnMixEffectLearn	False	True / False	A key captured from Preview or Program does not store a source on its Mix Effect elements, so recalling it leaves each Mix Effect's current source unchanged. The same setting as Ignore Source on Mix Effect Learn .

Configuration Manager

Setting	Default	Values	What it does
DefaultOutputFormat	1920x1080 @ 59.94 Hz	A list of output formats (resolution @ refresh rate)	The resolution a new output starts at in the Configuration Manager.

Connection Settings

Setting	Default	Values	What it does
LastServer	(empty)	Server address	The last connected server address. Set automatically when you connect to a server.

Setting	Default	Values	What it does
PreferredNIC	<i>(empty)</i>	Blank, or one of this PC's network adapters	The network interface to bind to. Restart Spyder Studio for a change to take effect.

Display Simulator

Setting	Default	Values	What it does
DisplayPanZoomInSimulator	True	True / False	Shows pan / zoom on layers in the Simulator.
GenericBackColor	0, 0, 0 (black)	Color	The grid background color.
HideLockedLayers	False	True / False	Hides locked layers in the Simulator. The same setting as Hide Locked Layers . For what makes a layer locked, see Section 6.4.1.3.
HideSlaveLayers	False	True / False	Hides slave layers in the Simulator (Section 9.7). The same setting as Hide Slave Layers .
IgnoreRenewalGroupForLayerAdd	False	True / False	Allows a layer to be added to a PixelSpace when the layer's Frame group is different from the PixelSpace's.
KeyFrameDefaultHeightScale	0.8	Number	The initial height of a new layer, as a fraction (0.8 = 80%). The same setting as Initial Layer Height % .
MouseMoveWindow	10	Number (pixels)	How far the mouse must move after you press the button before a layer starts to move in the Simulator.
OutputOverlayColor	255, 255, 255 (white)	Color	The color the output overlays are drawn in. Restart Spyder Studio for a change to take effect.
RepositionPixelSpaces	FrameGroupsOnly	None, FrameGroupsOnly, AllPixelSpaces	How PixelSpaces are arranged in the Simulator on parallel-configured systems. The same setting as Reposition .
ResizeFrom	FromCenter	FromCenter, FromCorner	Whether a layer resizes from its center or from a corner when you size it in the Simulator.

Setting	Default	Values	What it does
ShowInvalidRouterPatching	False	True / False	Checks the routing path from source to layer through the router crosspoints. For troubleshooting only; it may slow the system down.
ShowOutputOverlays	None	None, BehindLayers, AboveLayers	Draws the system's outputs as wireframes over layers and PixelSpaces. The same setting as Show Outputs .
ShowPacketsProcessed	False	True / False	Shows the frames per second (fps) in the Simulator.
ToolbarAlignmentMorph	True	True / False	Morphs (animates) keyframe moves made with the Simulator toolbar on layers in Program.

Function Keys

Setting	Default	Values	What it does
FunctionKeyHeight	8	Number	Number of function keys shown vertically in a column. The second Grid Size value under Function Key Grid .
FunctionKeyWidth	5	Number	Number of function keys shown horizontally in a row. The first Grid Size value under Function Key Grid .

Script

Setting	Default	Values	What it does
AlwaysShowPreviewScriptingModeBar	True	True / False	Always shows the preview scripting mode bar in the Script panel (Section 9.5).
ScriptAutoFitColumnsToClientWidth	False	True / False	Fits the Script panel's columns to the width of the panel. The same setting as Auto Fit Columns .
ScriptMinimumColumnWidth	50	Number	The minimum width of a Script panel column; enter 0 for no minimum. The same setting as Min. Column Width .

Setting	Default	Values	What it does
ScriptRowHeight	40	Number	Has no visible effect in this release.
ScriptViewFollows	True	True / False	The Script view follows execution.

User Diagnostics

Setting	Default	Values	What it does
ShowDebugMenuItem	False	True / False	Shows the Debug menu (Start Local Remoting Server, Logical Fiddler, Show Packets Processed).
UserDiagnosticsMode	Enabled	Disabled, EnabledWithExceptions, Enabled	The run-time mode for real-time protection.

Notice: Advanced exposes settings that have no guard rails. Change one only when you know what it does; the curated sections above cover everything needed for normal operation.

Tip: `ShowDebugMenuItem` is what puts the **Debug** menu on the menu bar, and **Debug ▶ Start Local Remoting Server** runs a server on the PC itself. That lets the Alert Viewer's Remote tab (Section 11.1), the UDP Protocol Tester, and the Console Simulator work with no physical Frame attached.

5. Concepts & Architecture

Spyder Studio drives a Spyder-S image processor. Before operating it, you need the processing model: how a signal travels from a physical input to a pixel on a display, and which resources each step consumes. This chapter defines that model. Every later chapter assumes the terms established here.

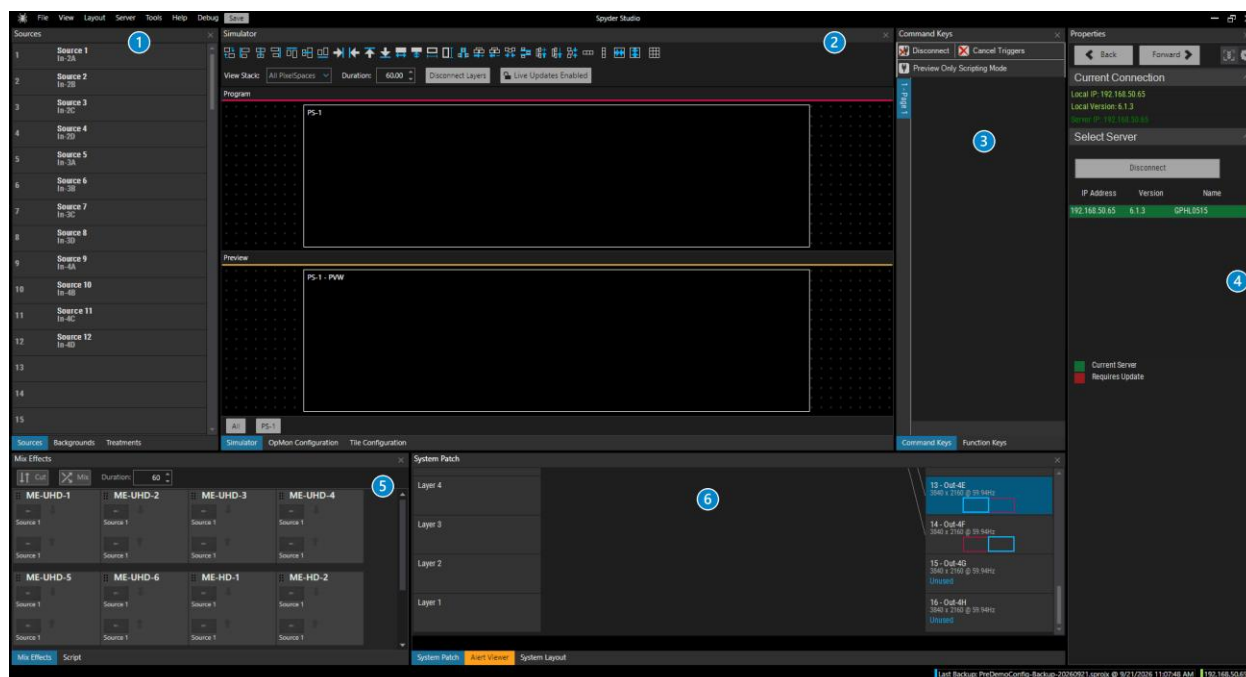
Read this first. Operators who skip the concepts tend to fight the tool — placing content on the wrong rail, exhausting layers they didn't know they spent, or expecting a transition the resource model can't deliver. Five minutes here prevents most of it.

5.0 Key terms

Term	One-line definition
Canvas	The total coordinate space across all PixelSpaces in a configuration.
Visual Image (VI)	A composed picture on the canvas — what the audience sees.
PixelSpace	A logical display region (a screen) with its own resolution, Program view, and Preview view.
Layer	A single piece of moving content placed on a PixelSpace, controlled by keyframes.
Keyframe	The size, position, and attribute state of a layer at a point in time.
Source	A named input signal available to place on a layer.
Mix Effect (ME)	A two-input mixer that produces one transitionable layer ("Single Layer Mixer").
Background	Full-canvas content behind all layers, per PixelSpace ("Layer Zero").
Output	A physical connector, configured as Normal, Aux, OpMon, or Tiled.
Program / Preview	The on-air rail and the staging rail of a PixelSpace.

5.1 The Spyder Studio workspace

Before the concepts, it helps to know where each of them lives on screen. The figure below is the Spyder Studio workspace with its six main regions numbered; the rest of this chapter explains the ideas behind them, and Chapters 6 and 7 explain how to drive them.



- | | | |
|--------------------------------------|---------------------------------|---|
| 1 Sources / Backgrounds / Treatments | 2 Simulator (Program / Preview) | 3 Command Keys / Function Keys |
| 4 Properties | 5 Mix Effects / Script | 6 System Patch / Alert Viewer / System Layout |

Figure 5-1. The Spyder Studio workspace, with the six main regions numbered. The key beneath the screenshot names each region.

#	Region	What it holds	Covered in
1	Sources / Backgrounds / Treatments	The content you place: input sources, full-canvas backgrounds, and saved layer "looks".	5.6, 5.5, 7.3
2	Simulator (Program / Preview)	The canvas itself — a live map of every PixelSpace, with the on-air Program rail above the staging Preview rail. This is where you compose.	5.1–5.4
3	Command Keys / Function Keys	Recallable looks and one-touch actions built from what you composed.	Chapters 8, 9
4	Properties	The settings of whatever is currently selected — a layer, an output, a PixelSpace, or a tool. Its content changes with the selection.	7.2, 7.7
5	Mix Effects / Script	The transition engine (one column per Mix Effect, Top and Bottom buses) and the cue-based Script grid.	5.4, 9.1

#	Region	What it holds	Covered in
6	System Patch / Alert Viewer	The signal flow of the applied configuration, and the system's own message log.	6.10, 11.1

Two things about this layout are worth knowing up front. **Panels are dockable and hideable** — everything in the View menu can be shown, hidden, or rearranged, so your screen may not match the figure exactly; Layout ▸ Save As Custom stores an arrangement you like. And **the Properties panel (4) is contextual** — it is the single place almost every setting in this guide is edited, so when a procedure says "in the Properties panel", it means region 4 showing whatever you just selected.

5.1.1 Canvas and Visual Image

The **canvas** is the single coordinate space that holds every PixelSpace in a configuration — region 2 above is your view of it. A **Visual Image (VI)** is a composition on that canvas: the arrangement of layers, backgrounds, and effects that resolves to what the audience sees.

You build a VI by placing content onto PixelSpaces. The canvas is what lets a layer span two adjacent screens, or move from one PixelSpace to another: both screens live in the same coordinate space.

5.2 PixelSpace

A **PixelSpace** is a logical display region — one screen as the operator thinks of it. Each PixelSpace has:

- A **resolution** and an origin (Horizontal/Vertical Start) on the canvas. A PixelSpace can be any size the configuration supports — from a single output up to the canvas limits in Appendix C. The examples throughout this guide use a 7680 × 2160 PixelSpace (two 3840-wide outputs side by side), which is simply the demonstration system's layout, not a required or standard size.
- A **Program view** (on-air) and a **Preview view** (staging). You compose on Preview, then transition it to Program.
- Its own **background Mix Effect** (see Section 5.5).

A configuration can hold several PixelSpaces. The bottom of the Simulator shows an **All** tab plus one tab per PixelSpace, and the View Stack selector filters which PixelSpace's Program/Preview pair you see. PixelSpace creation and sizing happen in configuration; see Section 6.3.1.2 (All In One) and Section 6.4.1.3 (Free Form). Per-PixelSpace properties (titling, test pattern generator) are covered in Section 6.8.

5.3 Layers and keyframes

A **layer** is one piece of content on a PixelSpace — a source or a Mix Effect. A layer's geometry and look at any instant is a **keyframe**: size, position, crop, aspect ratio, pan & zoom, border/shadow, and transparency.

Position is stored **normalized** (−1.0 to +1.0) and resolved to pixels against the PixelSpace. The Properties panel shows both: a normalized X/Y *and* an Absolute X/Y in pixels. This is why the same layer keyframe behaves consistently across PixelSpaces of different sizes.

Note — super-scale. Not every layer scales equally far. A layer fed from **input port 1 or 2 of any Frame** is *super-scaled* and can be stretched to **32,768 px** wide — the full width of a Frame group's maximum canvas. Layers fed

from inputs 3 and 4 cannot. Because the capability follows the physical input, it is settled when you patch the Frame, not when you build the layer (Section 7.2.2).

Note: Layers are a finite hardware resource. Each Spyder-S Frame provides a fixed pool of UHD layers, and that pool is shared by everything that needs a layer. Backgrounds do **not** draw from it (Section 5.5), but Mix Effects and Aux content do (Sections 5.4, 5.8). See Section 5.9 for the budget.

Layer operations — adding content, keyframing, cloning, mirroring, cropping, alignment — are covered in Section 7.2.

5.4 Mix Effects — the "Single Layer Mixer"

A **Mix Effect (ME)** is a two-input mixer whose output occupies a **single layer** on the canvas. Think of it as a Single Layer Mixer: the transition happens *inside* the ME resource, so the PixelSpace spends one layer to show a full A-to-B transition.

An ME has two inputs:

- **Top** — by operator convention the **Program** (on-air) content.
- **Bottom** — by operator convention the **Preview** (staging) content.

Note: "Top" and "Bottom" are the ME's two input buses. "Program" and "Preview" are the PixelSpace's two rails. They are different things that line up in normal use: the Top input feeds Program, the Bottom input feeds Preview. Internally the ME also computes a "top" (the currently opaque side); operators do not set that directly.

An ME transitions between its two inputs two ways:

- **Cut** — an instant swap. A Cut is a Mix with a duration of one frame.
- **Mix** — a linear cross-fade over the **Duration** (default 60 frames). Pressing Mix again transitions back.

You trigger a transition from the GUI (the **Mix** / **Cut** buttons), from a Function Key, or from an external API command (Section 10.4). All three call the same underlying mix.

Mix Effects come in three kinds, all built on the same model:

ME type	Purpose	Freeze?
Normal	Standard program/preview mixing on a PixelSpace.	No
Aux	Drives an Aux output with its own mixer (same class as Normal).	No
Background	The per-PixelSpace background mixer (Section 5.5).	Yes

Full operation is in Section 7.4; the external-control tokens (MESR / METR / MELA / MELS, with Cut = METR <id> D1) are in Section 10.4.

5.5 Backgrounds — "Layer Zero"

A **background** is full-canvas content that sits **behind every layer** on a PixelSpace — the scenic plate the layers float over. Call it Layer Zero: it underlies the layer stack but is not itself one of the UHD layers.

This is the key resource fact: a background consumes **inputs, not layers** — a full-screen scenic background costs you input channels, but it does not spend one of your precious UHD layers. Where those inputs sit still matters: within

a frame group, layers on Frames numbered lower than a Frame holding a background input are drawn behind the background, so Spyder locks them. Use the first inputs in each frame group for backgrounds and no layers are lost (Section 6.4.1.3).

A background input is also only shown over its own frame group — see Section 6.4.1.3.

A background is built as a **background group**: an HCount × VCount grid of inputs tiled to fill the PixelSpace. A 7680 × 2160 PixelSpace fed by two 3840 × 2160 inputs is a 2 × 1 group consuming two background inputs. Each input can be cropped (crop drives the computed size; see Section 7.5).

Backgrounds transition with the same Mix/Cut model as any ME (Section 5.4), with one rule:

Note: Background Mix and Cut require **two** background groups in the same PixelSpace. With a single group the bottom bus has nothing to hold, so there is nothing to transition to — both Mix and Cut are inert. Add a second group to enable transitions. Top group → Program, Bottom group → Preview, exactly as in Section 5.4.

A **still image** can substitute for a background's video, reusing the same inputs and obeying the same crop and two-group rules. A **test-pattern still** is shown directly on the PixelSpace's outputs instead (Section 7.5.7). Each Frame holds up to 16 background stills, shared by the PixelSpaces whose background inputs are on it (Section 7.5.6). Full operation is in Section 7.5.

5.6 Sources and inputs

A **source** is a named, placeable input signal (for example *Cam 1* on input *In-3A*). Sources auto-populate from the connected Frame.

Placing a source is covered in Section 7.1. How you place it onto the canvas matters:

- **Drag onto a PixelSpace** → the source enters **through a Mix Effect** and occupies one layer.
- **Ctrl-drag onto a PixelSpace** → the source is placed **directly as a layer**, bypassing the ME.

This distinction is a direct consequence of the layer-vs-ME model in Section 5.4.

5.7 Input modes

Each physical input is configured by **connector type** (DP, HDMI, SDI, SFP) and signal format. Two facts shape capacity planning:

- **Dual-link / paired connectors reserve two physical channels** per logical input. A background group cell on a dual-link source costs two channels, not one.
- Input signal attributes are read from the signal and displayed per input in the layer properties' **Input Configuration** section (Section 7.1.5). Color Space and Sampling, Bit Depth, Color Range, Color Standards, and Genlocked can each be **overridden** there; HDR is reported but not overridable on the input.

5.8 Output types

Each physical output is configured for one role. The configuration tool reports each output's capability as a flag — **N** / **A** / **O** / **T**:

Flag	Output type	Meaning
N	Normal	A scaled program output for a PixelSpace.
A	UnscaledAux	An unscaled Aux feed. (A = UnscaledAux, <i>not</i> generic Aux.)
O	OpMon	An output monitor / multiviewer tiling Program, Preview, and inputs.
T	Tiled	One output driving a grid of tiles (LED wall / tiled display).

Notice: The capability flags come from hardware discovery. An output only offers the roles its card supports — a port that cannot tile will not show **T**. Plan output roles against the actual card population, not the model name.

OpMon configuration is in Section 6.6 and Tiled configuration in Section 6.7; live per-output adjustment is in Section 7.7.

5.9 Resource and bandwidth budget

Spyder-S allocates three budgets you can exhaust independently:

- **Layers** — a fixed pool of **four UHD layers per Frame**, spent by direct layers, Mix Effects, and Aux content. Backgrounds do **not** spend layers. Each layer additionally supports up to four clones (Section 7.2.3), which do not themselves spend layers.
- **Inputs** — physical input channels, spent by sources *and* by background groups. Dual-link sources spend two channels each.
- **Bus bandwidth** — the total internal pixel throughput, **4.728 Gpps** across the bus. Every active layer, background input, and output draws against it.

Tip: When you run out of "layers" for a full-screen plate, move that plate to a **background** (Layer Zero). It frees a layer at the cost of input channels — often the right trade for static scenic content.

5.9.1 Frame groups

In a multi-Frame system, Frames are organized into one or more **Frame groups** (also called *renewal groups*). Each Frame group drives its own **canvas rectangle**, and a single group's canvas is capped at **8× 4K** (8 × 4096 × 2160 px; height up to 2160). You mark where a new group begins during configuration (the **New Group** / renewal-master flag, Section 6.3.1). The system's **total canvas is the tiled arrangement of every group's rectangle**, so the total surface can extend well beyond one group's 8× 4K limit.

All processing shares **one bus, 4.728 Gpps** (Appendix C). The canvas, the Mix Effects, and the per-input system overhead all draw from it and compete — a larger canvas leaves less bandwidth for Mix Effects. The decisive rule: **canvas bandwidth is set by the single widest group, not by the sum of the groups** (each group's canvas is renewed independently). That is what makes Frame groups valuable in two distinct ways:

- **Recover Mix Effect bandwidth — same canvas.** Split a canvas into more, *smaller* groups: the widest group shrinks, canvas bandwidth drops, and the freed bus is auto-provisioned to Mix Effects. An 8× UHD canvas across one group costs ≈ 4.0 Gpps (no MEs); the same 8× UHD across two groups ≈ 2.0 Gpps; across four groups ≈ 1.0 Gpps — each step hands bandwidth to MEs.
- **Enlarge the canvas — beyond 8× 4K.** Add groups to tile a larger total canvas than any one group can drive. Because the cost tracks only the widest group, a **16× UHD** canvas built from four **4× UHD** groups costs just ≈ 2.0 Gpps and still leaves room for Mix Effects — whereas the same 16× UHD as two **8× UHD** groups costs ≈ 4.0 Gpps and leaves none.

Rule of thumb: *many small groups* give you a **large canvas and Mix Effects at once**; *few large groups* spend the bus on canvas. The bandwidth math is worked through in Appendix C.

Resources split into **global** (shared across all Frame groups) and **local** (private to each group):

Scope	Resources
Global — shared across groups	Mix Effects, ME inputs, multiviewer / OpMon tiles, the operator monitor (OpMon), and the linked canvas
Local — private to each group	Background inputs, layers, and canvas outputs

Note: This is why the **Layer Provisioning** mode (Section 5.9.2) exists. Because layers are a *local* resource, **Global Linking** links them one-to-one across groups — a layer can appear in any group, with the total capped by the smallest group — while **Discreet** keeps each group's layers to itself. **Backgrounds are per-PixelSpace** (Section 5.5), drawing on each group's local inputs; Mix Effects are global, so one ME bank serves the whole system.

Tip: Frame groups are created in configuration via the **New Group** flag (Section 6.3).

5.9.2 Layer provisioning across Frame groups

When a system spans **multiple Frame groups**, how layers are shared between them is governed by the **Layer Provisioning** mode (set from **Server > Global Parameters > Layer Provisioning** — see Section 6.3.1). The mode decides how many layers you have and what happens when a layer moves between Frame groups:

Mode	Layers available	Behavior
Global Linking	Limited by the Frame group with the fewest layers	All layers are permanently linked one-to-one across Frame groups. Simplest model; every layer exists everywhere, but the total is capped by the smallest Frame group.
Discreet	All individual layers	A layer moved into a Frame group outside its own hardware is not visible unless you manually link it with a Command Key (Section 9.7). Maximizes layer count at the cost of manual linking.

Slave layers. A layer is a hardware resource of its own Frame group. To appear in another Frame group, it needs a **slave layer** — a layer in that group that is linked to it and follows it. Moving the layer, or making any other keyframe change to it, is applied automatically to all of its slave layers.

- **Global Linking** provisions the slave layers for you: each layer has one slave layer statically provisioned in every Frame group, so any layer can span all Frame groups with no manual setup. The trade-off is that the

available layer count is reduced to that of the smallest Frame group, and Global layers cannot be unlinked for Discreet use.

- **Discreet** makes every layer available, but a layer is only displayed within its own Frame group. A layer moved outside its group still appears in the Simulator, but not on that group's outputs, until you add a slave layer to it in a command key (Section 9.7).

Note: When a system runs multiple Frame groups (Global Linking or Discreet), a layer that spans Frame groups must either use a **Mix Effect** or take its source from an upstream external **routing switcher** connected to all Frames. With a supported external router, Spyder manages the crosspoint switching as needed.

Notice: Switching between Global Linking and the dynamic (Discreet) modes can **invalidate existing configuration data, including Command and Function Keys**. Choose the provisioning mode when you build the configuration; if you must change it later, plan to rebuild the configuration and clear stale data.

Capacity math and worked allocation examples are in Appendix C.

5.10 Signal flow (end to end)

Putting it together, a pixel travels:

1. **Physical input** → a **Source** (Section 5.6), configured by input mode (5.7).
2. The source is placed on a **Layer** via a **Mix Effect** (5.4) or directly (5.6), and shaped by a **keyframe** (5.3).
3. Layers composite over the **Background** (5.5) within a **PixelSpace** (5.2), on the **Preview** rail.
4. A **transition** (Cut / Mix) promotes Preview to **Program**.
5. Program resolves to one or more **Outputs** (5.8) — Normal, Aux, OpMon, or Tiled.

Everything draws against the layer, input, and bandwidth budgets of Section 5.9. The rest of this guide is how you drive each step: Configuration (Chapter 6) establishes the system's structure; Operation (Chapter 7) runs content through it.

6. Configuration

Configuration defines the structural layer of a Spyder-S system: which Frames are present, how they form a canvas, which PixelSpaces sit on that canvas, and how each physical output is used. This is the **Preconfig** model — the system's structure is established first, and everything in Chapter 7 operates inside it.

How often you revisit this chapter depends on the deployment:

- **Live events and rental / staging** — configuration is typically done per show or per venue, as the rig changes. You configure, run the show, and reconfigure for the next one.
- **Fixed installations** — a control room, visitor centre, house of worship, command centre, or corporate lobby is usually configured **once at commissioning** and then left alone. Day-to-day users only operate within it, and the configuration is revisited only when the physical rig changes (a display added, an output re-purposed) or a firmware update requires it.

The wizards are the same either way; only the frequency differs. If you are commissioning a permanent install, expect to spend proportionally more time here and to save the finished configuration as a *.sprojx project (Section 4.8.1) so it can be restored exactly if a PC is ever replaced.

Everything in Studio is built on the configuration created here. The chapter assumes the processing model from Chapter 5: canvas, PixelSpaces, layers, Mix Effects, inputs, and the output roles (Section 5.8). It does not repeat those definitions — it shows how to commit them to hardware.

You build a configuration in the **Configuration Manager**, which offers **two separate wizards**:

- **All In One** (Section 6.3) — an array-oriented wizard with **three** pages: Frame Setup → System Mapping → Apply. The fastest path for standard output arrays; System Mapping auto-allocates outputs, inputs, and Mix Effects for you.
- **Free Form** (Section 6.4) — a canvas-oriented wizard with **five** pages: Frame Setup → Canvas Setup → PixelSpaces Setup → Outputs Setup → Apply. Manual, canvas-level placement of Frame groups, PixelSpaces, and outputs, for maximum flexibility.

The two wizards are independent workflows with different pages — within a single configuration pass, choose one on the **Configuration Mode** screen (Section 6.2) and follow it to its Apply page rather than switching mid-pass. (Once a configuration has been **applied**, you can revisit it with **either** wizard — see the note in Section 6.2.) Each wizard has two entry paths — **Start with a new configuration** and **Load the current configuration** — documented separately below.

6.1 Opening the Configuration Manager

Open the Configuration Manager from **File ▶ New**. It opens as its own window over the workspace, on the **Configuration Mode** start screen (Section 6.2). The Configuration Manager is used in both **Virtual (offline)** and **Connected** modes.

Offline vs. connected. One thing governs how the wizard behaves: whether Studio is connected to hardware. Everything else follows from it.

State	What it means for configuration
Offline (Virtual Mode, not connected)	You build a virtual Frame set by hand. Frame Setup is editable ; the final Apply page writes a local virtual configuration you can work in away from hardware.

State	What it means for configuration
Connected (bonded to a Frame or server)	The Frame set mirrors the discovered hardware. Frame Setup is read-only ; the final Apply page writes the configuration to the hardware on the network .

This split appears again at each wizard's Frame Setup and Apply pages.

Navigating the wizard. **Back** and **Next** sit at the bottom of every page; **Cancel** closes the window without changing the live configuration. Leaving a page validates it, and each page shows a footer message with a validation icon — **Good**, **Warning**, or **Error**. **Next is blocked into the later pages while the configuration is invalid** — you cannot reach Apply until validation passes. Pressing **Back** from the first page returns to the Configuration Mode start screen.

Note: The whole configuration must fit inside one fixed budget — a **total bus bandwidth of 4.728 Gpps**. Every PixelSpace, background input, output, and Mix Effect draws against it. Both wizards validate this budget continuously and block Apply while it is exceeded. See the resource budget in Section 5.9.

Note: If a page will not advance, read its footer. Common causes are a bandwidth Warning — the footer reads "[Warning] Bandwidth 'System': Total bus bandwidth exceeded by {x} Gpps" — or, in Free Form, having no PixelSpace yet.

6.2 Choosing a configuration mode

The Configuration Manager opens on the **Configuration Mode** screen: "Select a path below to configure the Spyder-S platform according to use case." This screen is the fork between the two wizards. Make two choices here — the **mode** and the **entry path** — then click through.

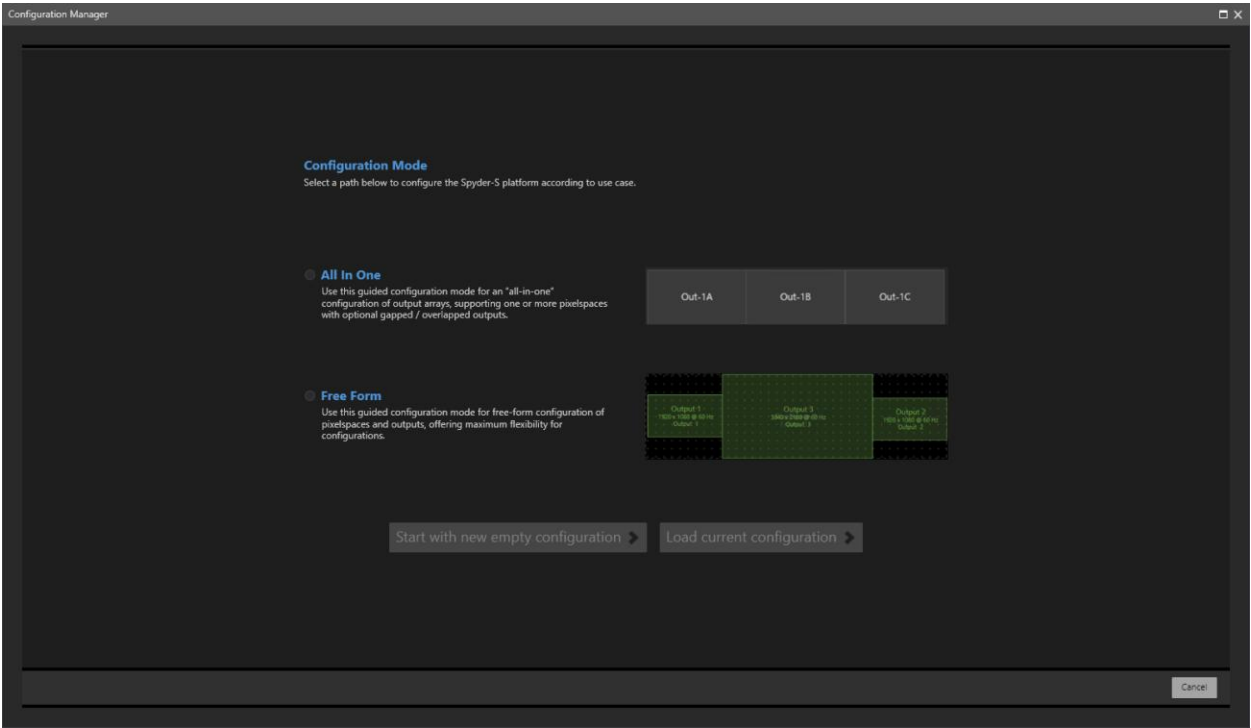


Figure 6-1. The Configuration Mode start screen — choose All In One or Free Form, then an entry path.

Choose a mode:

Mode	Use case	Pages
All In One	Array-oriented configuration of output arrays, supporting one or more PixelSpaces with optional gapped or overlapped outputs.	Frame Setup → System Mapping → Apply (3)
Free Form	Free-form placement of PixelSpaces and outputs on a canvas, for maximum flexibility.	Frame Setup → Canvas → PixelSpaces → Outputs → Apply (5)

The **All In One** thumbnail shows a simple three-output array (Out-1A / Out-1B / Out-1C); the **Free Form** thumbnail shows outputs arranged non-linearly on a canvas.

Choose an entry path with one of the two buttons beneath the modes:

- **Start with new empty configuration** — begins the selected wizard with sensible defaults and **no programming data** ("empty" means no command keys, sources, or other show data — not a blank rig). The Frame set it opens with depends on mode: **offline**, a first-ever configuration starts with **one default HDMI Frame** (add more with **Add Frame**), and after you Apply the Frame set **persists** — a later empty start opens with the Frame count from your last applied configuration until you clear it; **connected**, it mirrors the Frames actually connected. Documented for All In One in Section 6.3.1 and for Free Form in Section 6.4.1.
- **Load current configuration** — loads the configuration currently in effect into the selected wizard, so you can edit it. Documented for All In One in Section 6.3.2 and for Free Form in Section 6.4.2.

Note — updating an applied configuration. The two wizards are independent within a single pass, but once a configuration has been **applied** you can revisit it with **either** wizard: reopen the Configuration Manager, choose All In One or Free Form, and use **Load current configuration**. A configuration built in All In One can later be loaded and updated in Free Form, and vice versa. Note that **Free Form exposes more granular control** — if you load a Free-Form-built configuration into All In One, those finer controls are not available there. **For advanced configurations, and for updating them, use Free Form** for both the initial build and later changes.

Cancel closes the Configuration Manager without changing the live configuration.

Note: The empty start behaves differently per mode. **All In One** seeds one PixelSpace (PS-1, a 1 × 1 array with Preview on) and is **valid immediately** — you can Apply at once, then customize. **Free Form** populates the Frames and canvas but starts with **zero PixelSpaces**, so it is **invalid** until you add one — the Apply page stays greyed out until at least one PixelSpace exists.

6.3 All In One workflow

The **All In One** wizard has three pages — **Frame Setup** → **System Mapping** → **Apply Configuration**. System Mapping does the heavy lifting: you describe output arrays and the allocation engine assigns outputs, inputs, and Mix Effects and reports the budget.

6.3.1 Start with a new configuration

From the Configuration Mode screen, select **All In One** and click **Start with new empty configuration**. The wizard opens on Step 1 with the current Frame set — **one HDMI Frame** on a first-time offline configuration (or the Frames from your last applied configuration) — and one seeded PixelSpace (PS-1).

6.3.1.1 Step 1 – Frame Setup

Frame Setup defines the Frames in the configuration: "Setup Spyder frames to be included in this configuration:" The page shows the Frames table and, offline, lets you add, remove, and re-model Frames.

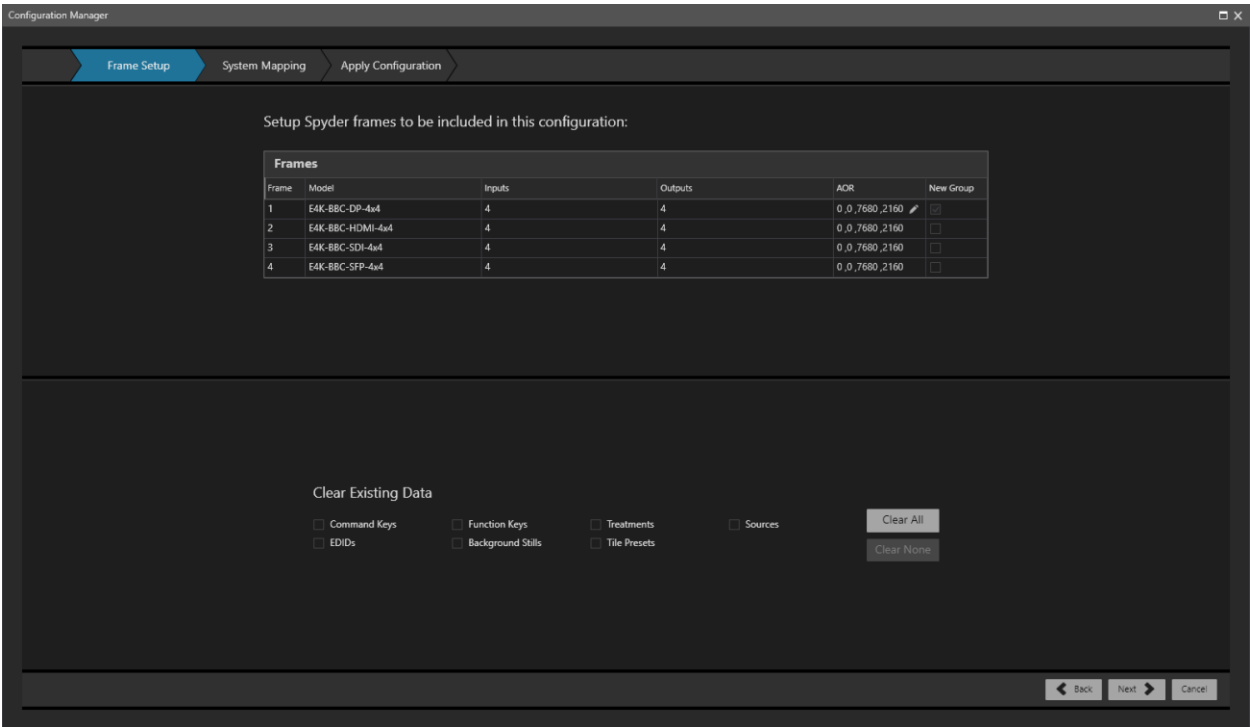


Figure 6-2. All In One, Step 1 — Frame Setup: the Frames table and Clear Existing Data options.

The Frames table columns (All In One):

Column	Field	Notes
Frame	Number, from 0	Frame number. Frame 0 is the master Frame and cannot be deleted.
Model	Dropdown	One of four E4K-BBC cards (below).
Inputs / Outputs	Counts, auto-filled	Auto-filled from the model — 4 / 4 on E4K-BBC.
AOR	X, Y, Width, Height	Area of Raster — the Frame's rectangle on the canvas. The edit pencil opens an X / Y / Width / Height popup.
New Group	Toggle	Marks the Frame as the start of a new Frame group — a new master.

The four selectable **Frame models**:

Model	Connector
E4K-BBC-DP-4x4	DisplayPort
E4K-BBC-HDMI-4x4	HDMI
E4K-BBC-SDI-4x4	SDI
E4K-BBC-SFP-4x4	SFP

Whether the table is editable depends on connection state (Section 6.1):

- **Offline** — editable. Use **Add Frame** (top-right) to append a Frame; it defaults to the previous Frame's model (HDMI if none) and auto-fills 4 inputs / 4 outputs. Its **AOR** (Area of Raster) tiles it next to the previous Frame at the **configured Frame width** — in a default configuration each Frame is **3840 × 2160**, so the second Frame's AOR starts at X = 3840 (a wider saved canvas scales these values accordingly). It also gets a delete icon. Changing its **Model** dropdown recomputes its connector, layers, inputs, and outputs. Delete any Frame except the master (Frame 0), which has no trash icon. Toggle **New Group** to start a new Frame group.
- **Connected** — read-only. The Frame list mirrors the actual hardware; there is no Add Frame, no Model dropdowns, and no delete icons.

Clear Existing Data. Frame Setup also carries a **Clear Existing Data** block — seven checkboxes, plus **Clear All** and **Clear None**, which tick and untick all seven at once. Nothing is removed while you tick the boxes: **the deletion happens when you Apply the configuration**, and it is not undoable. Each box deletes one category of show data:

Tick this	and Apply deletes
Command Keys	The saved command keys (Section 8.2).
Function Keys	The saved function keys (Section 8.4).
Treatments	The saved treatments (Section 7.3).
Sources	The source definitions in the Sources pane (Section 7.1).
EDIDs	The EDIDs held in the EDID Manager (Section 6.9).
Background Stills	The background still images stored on the Frame (Section 7.5).
Tile Presets	The saved tile presets used by Tile Configuration (Section 6.7).

These are independent — ticking **Sources** leaves your command keys alone, and vice versa. Note that a command key or a script that referred to something you have just deleted will no longer find it.

For a brand-new configuration there is usually no show data to clear, so leave these unticked unless you are deliberately wiping a category. **Clear None** is greyed out when nothing is ticked.

Layer provisioning (multiple Frame groups). If you mark a Frame with **New Group**, the configuration spans more than one Frame group, and the **Layer Provisioning** mode decides how layers are shared. Set it from **Server** ▶ **Global Parameters** ▶ **Layer Provisioning** — at any time, including before Apply:

Layer Mode	Layers available	Behavior
Global Linking	Capped by the Frame group with the fewest layers	All layers are permanently linked one-to-one across Frame groups.
Discreet	All individual layers	A layer moved into a Frame group outside its own hardware is not visible until you link it with a Command Key (Section 9.7).

Note — how canvas size and Frame count trade against each other. There is no single "maximum canvas." The limit slides, and it is worth understanding why before you size a system.

Two things compete for the same **4.728 Gpps** bus (Section 5.9):

- **The canvas**, which costs roughly **0.5 Gpps per UHD of width**. An 8 × UHD canvas therefore spends about 4.0 Gpps — most of the bus.
- **Per-input system overhead**, which grows with the **number of inputs**. Every Frame you add brings four more inputs, so overhead climbs as the system grows.

Frame count does not cap the canvas directly. It caps it *indirectly*, by eating the headroom the canvas would otherwise use. The result is a sliding boundary:

Single-group canvas	Widest raster	Fits the budget up to
8 × UHD	30,720 px	11 Frames
8 × DCI-4K	32,768 px	5 Frames
7 × UHD (or 6 × DCI-4K)	26,880 px	the full 16 Frames

Read it either way round: a modest Frame count buys you the widest canvas the platform can render, while a full 16-Frame system has to settle for a slightly narrower one. Cross the boundary and the footer raises *"Total bus bandwidth exceeded by {x} Gpps"*, greying out Apply until you reduce the canvas or remove a Frame.

To go wider than this, split the Frames into groups (toggle **New Group**). Canvas cost tracks only the **widest** group, not the sum of all of them (Section 5.9.1) — so a canvas divided across groups costs what its widest group costs, and all 16 Frames can then serve a total canvas far larger than any single group could carry. This is the intended path to very large canvases, not a workaround.

Click **Next** to advance to System Mapping. **Back** returns to the Configuration Mode screen; **Cancel** exits.

6.3.1.2 Step 2 — System Mapping

System Mapping is the central All In One page. It defines PixelSpaces, background sources, global settings, and Mix Effects, and shows live resource utilization. The numbers you see — PixelSpace sizes, output and input assignments, every Gpps figure — are produced by the allocation engine, not entered by hand.

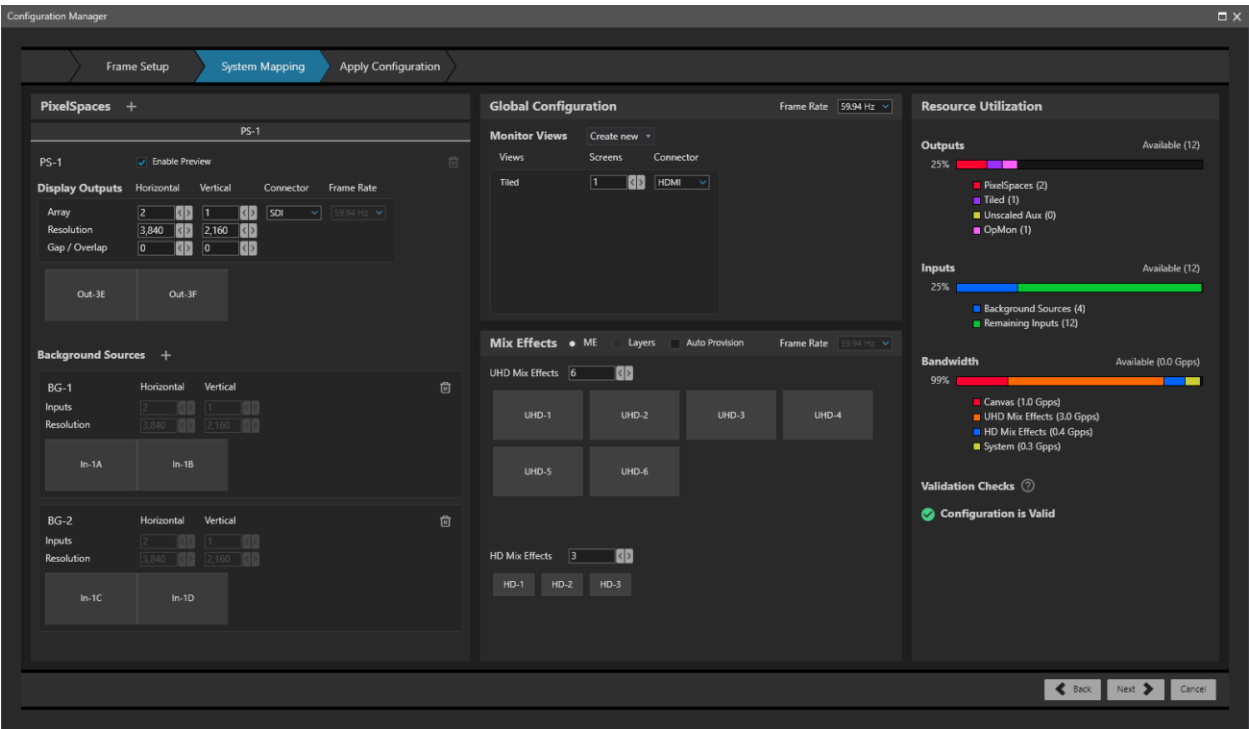


Figure 6-3. All In One, Step 2 — System Mapping: PixelSpaces, Mix Effects, and live Resource Utilization.

The page has three columns: **PixelSpaces / Background Sources** (left), **Global Configuration** (middle), and **Resource Utilization** (right).

PixelSpaces (left column). Add PixelSpaces with the + control (remove with ; the last remaining PixelSpace cannot be removed). Each PixelSpace has:

Field	Default	Meaning
Enable Preview	on	Creates the matching preview PixelSpace; also drives OpMon auto-creation.
Display Outputs ▸ Array (H × V)	1 × 1	A tiled grid of outputs (HCount × VCount).
Resolution (per tile)	3840 × 2160	HActive × VActive.
Connector	—	DisplayPort, HDMI, or SDI — picks which physical output type feeds the PixelSpace.
Frame Rate	system rate	Per-PixelSpace rate.
Gap / Overlap (H × V)	0 × 0	Spacing between tiles; negative = overlap , clamped to at most one full tile.

The PixelSpace rectangle is the union of its tiles. An H × V array consumes **H × V physical outputs**, each matching the chosen connector. For example, a 2 × 1 array of 3840 × 2160 DisplayPort tiles produces a 7680 × 2160

PixelSpace on two outputs. **Background Sources** are added per PixelSpace (+) as an HCount × VCount grid of inputs; **each input cell consumes one physical input**.

Global Configuration (middle column). System-wide properties and the Mix Effect pool:

Control	Notes
Frame Rate	The global canvas rate; defaults to 59.94 Hz .
ME / Layers toggle	MixEffects mode creates Normal MEs; Layers mode creates Aux MEs.
Auto Provision	Auto-packs MEs into spare bandwidth (see below).
(ME) Frame Rate	System rate or its half.
UHD Mix Effects	Numeric field with up/down arrows; maximum 12.
HD Mix Effects	Numeric field with up/down arrows; maximum 48.

With **Auto Provision** on, the manual ME counts are ignored and greyed: the engine packs as many UHD Mix Effects as the leftover bus bandwidth allows (up to 12), then fills the remainder with HD Mix Effects (up to 48). On a typical 4-Frame rig this yields 6 UHD + 3 HD. The **Monitor Views** ▶ **Create new** control adds **Tiled** (multi-viewer) or **Unscaled Aux** outputs; OpMon outputs are **not** added here — they are auto-derived from any PixelSpace with **Enable Preview** on.

Note: Frame rate is a budget input, not a cosmetic setting. Bandwidth scales with rate: a UHD Mix Effect costs roughly $0.498 \times (\text{rate} / 60)$ Gpps and an HD Mix Effect roughly $0.125 \times (\text{rate} / 60)$ Gpps. Set the system frame rate early — changing it later re-runs the entire allocation and can push a valid configuration over the bus ceiling.

Resource Utilization (right column). Three live bars against the 4.728 Gpps bus, plus a validation result:

Bar	Segments
Outputs	PixelSpaces (Normal) · Tiled · Unscaled Aux · OpMon.
Inputs	Background Sources · Remaining Inputs.
Bandwidth	Canvas · UHD Mix Effects · HD Mix Effects · System, in Gpps.

Beneath the bars, a **Validation Checks** section reports the result. A configuration that passes shows a green check and *"Configuration is Valid"*. A configuration that does not shows *"Warnings (n)"* with each problem itemized beneath it — for example *"System: Total bus bandwidth exceeded by 0.48 Gpps"*. While a warning stands, **Next** is greyed and the Apply page cannot be reached.

Tip: Watch the Bandwidth bar as you add PixelSpaces and inputs. The space the canvas and backgrounds leave free is exactly what Auto Provision has to spend on Mix Effects.

Click **Next** to advance to Apply. **Back** returns to Frame Setup.

6.3.1.3 Step 3 — Apply Configuration

The final page is a Christie Spyder-S screen with a single action. **Which button appears depends on the connection state:**

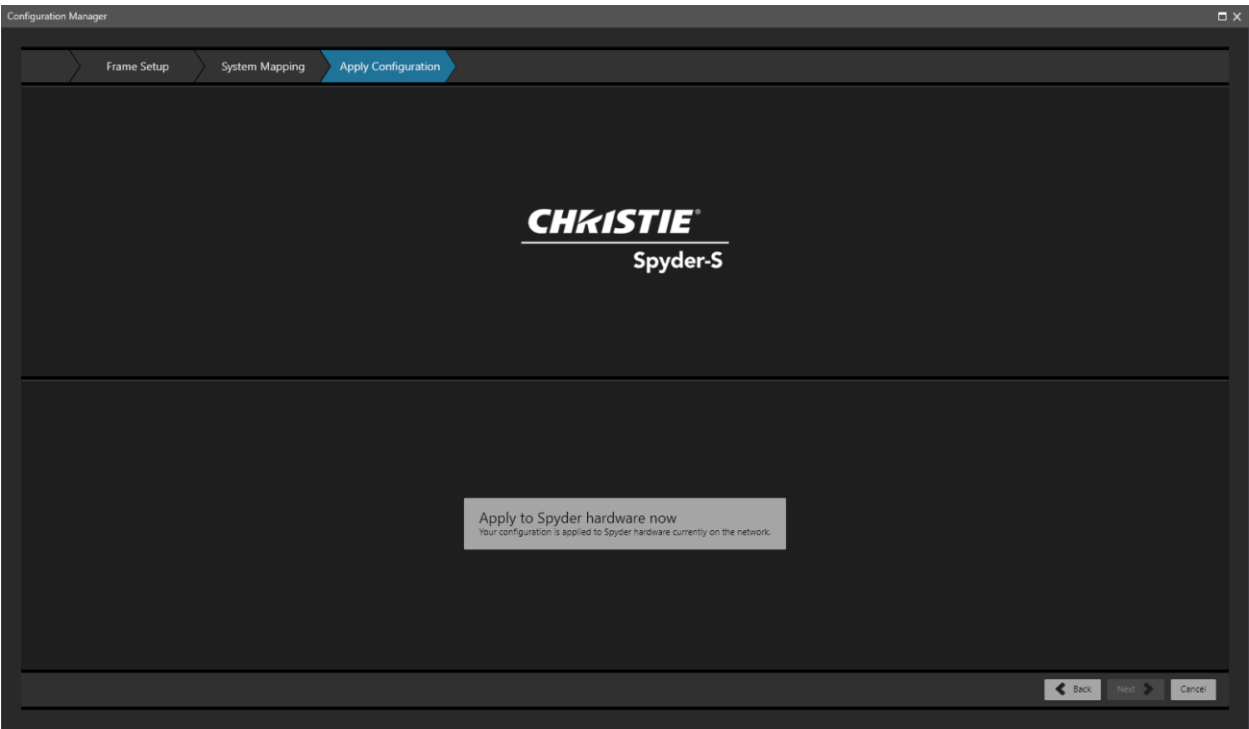


Figure 6-4. All In One, Step 3 — Apply Configuration.

Button (exact label)	When	Effect
Apply to local PC now (Virtual Mode)	Offline / not connected	"Builds a virtual configuration on your local PC. It lets you work while away from Spyder Hardware."
Apply to Spyder hardware now	Connected	"Your configuration is applied to Spyder hardware currently on the network."

The button is enabled only when the configuration **is valid** (Good). Apply runs on a background thread behind an "Applying Configuration..." overlay; on a **Good** result the window auto-exits. On failure, the footer reads "There was a problem applying configuration. Please review Alert Viewer for more details."

Notice: Apply commits the **Clear Existing Data** selections from Frame Setup (Section 6.3.1.1). For a new configuration these are normally unchecked.

6.3.2 Load the current configuration

From the Configuration Mode screen, select **All In One** and click **Load current configuration**. The wizard opens on the same three pages — **Frame Setup** → **System Mapping** → **Apply** — but each page is **prepopulated with the configuration currently in effect** so you can edit it rather than build from scratch. Walk the three steps below.

6.3.2.1 Step 1 – Frame Setup (loaded)

Frame Setup opens with the **current Frame set already listed**. The table columns, Frame models, and editability rule are exactly as in a new configuration (Section 6.3.1.1):

Column	Field	Notes
Frame	Number, from 0	Frame 0 is the master Frame and cannot be deleted.
Model	Dropdown	One of the four E4K-BBC cards (DisplayPort / HDMI / SDI / SFP).
Inputs / Outputs	Counts, auto-filled	4 / 4 on E4K-BBC.
AOR	X, Y, Width, Height	Area of Raster; edit pencil opens the X / Y / W / H popup.
New Group	Toggle	Starts a new Frame group.

- **Offline** — the loaded Frames are **editable**: add, delete (except the master), re-model, or re-group them, just as with a new configuration.
- **Connected** — read-only; the list mirrors the discovered hardware.

Clear Existing Data — read this when editing a live configuration. The seven checkboxes (Command Keys, Function Keys, Treatments, Sources, EDIDs, Background Stills, Tile Presets) plus **Clear All** / **Clear None** wipe the checked data types **when you Apply**.

Notice: When your edit **changes an input category** — for example, adding background inputs that were not previously configured — check the relevant **Clear Existing Data** boxes (typically **Sources** and **Background Stills**) so that stale sources and backgrounds do not remain after Apply. Clearing commits on Apply and affects existing show data, so clear only what you intend to rebuild.

Click **Next** to advance to System Mapping.

6.3.2.2 Step 2 – System Mapping (loaded)

System Mapping opens with the **current PixelSpaces, background sources, global settings, and Mix Effects already populated**. The three columns and every control behave exactly as for a new configuration (Section 6.3.1.2):

- **PixelSpaces / Background Sources** (left) — the existing PixelSpaces are listed. Edit their Array, Resolution, Connector, Frame Rate, and Gap / Overlap; add PixelSpaces with + or remove with (the last cannot be removed). Add or remove Background Sources per PixelSpace.
- **Global Configuration** (middle) — the loaded Frame Rate, Bit Depth, ME / Layers mode, Auto Provision state, and ME counts. Editing them re-runs the allocation.
- **Resource Utilization** (right) — the Outputs, Inputs, and Bandwidth bars and the "✓ Configuration is Valid" result update live as you edit.

Notice: Changing the system frame rate on a loaded configuration re-runs the entire allocation and can push a previously valid configuration over the 4.728 Gpps ceiling. Watch the Bandwidth bar.

Click **Next** to advance to Apply. **Back** returns to Frame Setup.

6.3.2.3 Step 3 — Apply Configuration (loaded)

The Apply page is identical to the new-configuration case (Section 6.3.1.3). The button reflects the connection state — **Apply to Spyder hardware now** (connected) or **Apply to local PC now (Virtual Mode)** (offline) — and is enabled only when the edited configuration is **valid**.

Notice: Apply commits the **Clear Existing Data** selections you made in Step 1. On a loaded configuration this is where stale sources, backgrounds, keys, or presets are actually removed — confirm your selections before applying.

6.4 Free Form workflow

The **Free Form** wizard has five pages — **Frame Setup** → **Canvas Setup** → **PixelSpaces Setup** → **Outputs Setup** → **Apply Configuration**. Instead of describing arrays, you position Frame groups, PixelSpaces, and outputs graphically on a canvas with a full alignment toolbar.

6.4.1 Start with a new configuration

From the Configuration Mode screen, select **Free Form** and click **Start with new empty configuration**. The wizard opens on Step 1 with the Frames and canvas populated but **zero PixelSpaces** — so the configuration is invalid (and Apply stays greyed) until you add at least one PixelSpace on Step 3.

6.4.1.1 Step 1 — Frame Setup

Frame Setup defines the Frames in the configuration: *"Setup Spyder frames to be included in this configuration:"* Free Form's Frame Setup uses the simpler column set — no AOR or New Group columns; in Free Form, grouping is done on the **Canvas Setup** page (Step 2):

Column	Field	Notes
Frame	Number, from 0	Frame 0 is the master Frame and cannot be deleted.
Model	Dropdown	One of four E4K-BBC cards (below).
Inputs / Outputs	Counts, auto-filled	Auto-filled — 4 / 4 on E4K-BBC.

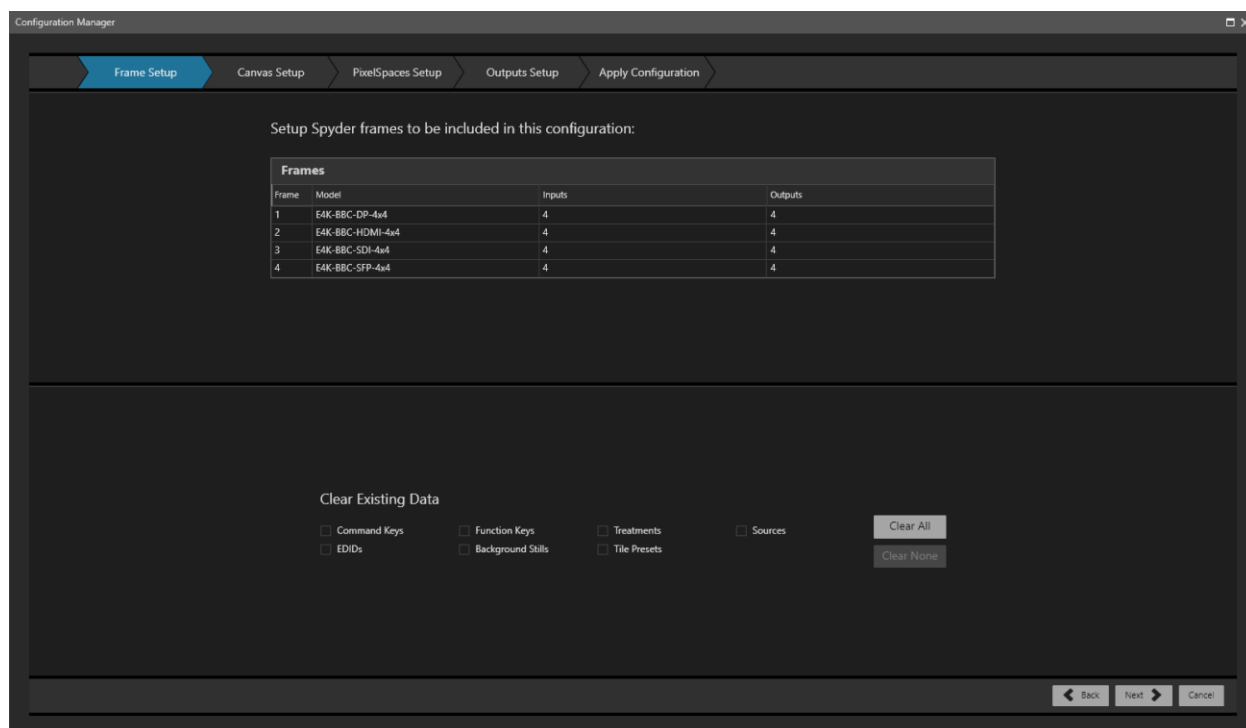


Figure 6-5. Free Form, Step 1 — Frame Setup: the Frames table and Clear Existing Data options.

The four selectable **Frame models**:

Model	Connector
E4K - BBC - DP - 4x4	DisplayPort
E4K - BBC - HDMI - 4x4	HDMI
E4K - BBC - SDI - 4x4	SDI
E4K - BBC - SFP - 4x4	SFP

Editability follows the connection state (Section 6.1):

- **Offline** — editable. **Add Frame** appends a Frame (defaults to the previous model, 4 in / 4 out, with a delete icon); the **Model** dropdown re-models it; delete any Frame except the master.
- **Connected** — read-only; the list mirrors the discovered hardware.

Clear Existing Data. The same seven checkboxes — **Command Keys**, **Function Keys**, **Treatments**, **Sources**, **EDIDs**, **Background Stills**, **Tile Presets** — plus **Clear All** and **Clear None**. Each deletes one category of show data **when you Apply**; Section 6.3.1.1 lists what each one removes.

For a brand-new configuration, leave these unticked unless you are deliberately wiping a category.

Click **Next** to advance to Canvas Setup. **Back** returns to the Configuration Mode screen; **Cancel** exits.

6.4.1.2 Step 2 — Canvas Setup

Canvas Setup defines the canvas the PixelSpaces and outputs are arranged on. It is a graphical editor over the per-Frame-group canvas rectangles.

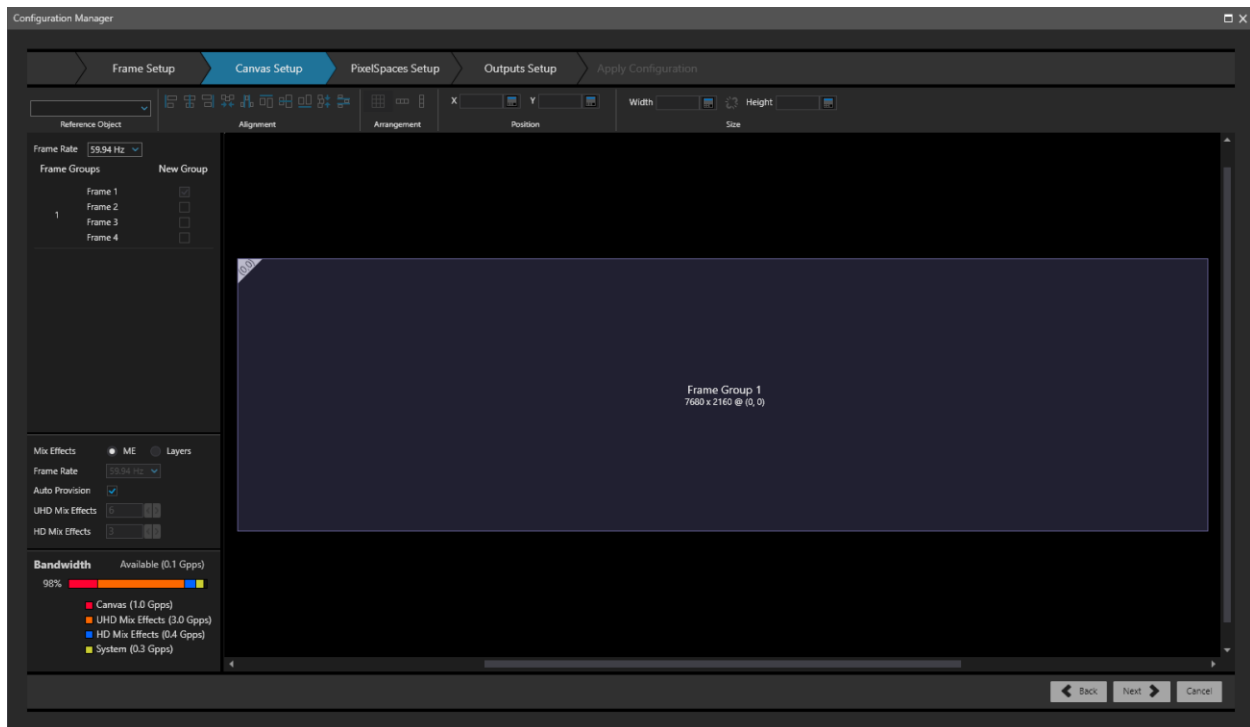


Figure 6-6. Free Form, Step 2 — Canvas Setup with the default single Frame group: all four Frames share one 7680 x 2160 canvas.

What this page controls. Each canvas rectangle belongs to one Frame group, so the **number** of canvases follows from how you group Frames — and grouping is done right here, with the **New Group** checkboxes described below. On this page you can:

- **Create a canvas** by starting a new Frame group (check **New Group** on a Frame).
- **Resize** any canvas rectangle, via the Size fields or by dragging on the canvas.
- **Reposition** any canvas rectangle relative to the others.

What you cannot do is add or delete a canvas *independently* of the Frame grouping — a canvas exists because a Frame group exists, so removing one means un-grouping the Frames that own it.

- **Frame Groups.** The left panel lists every Frame with a per-Frame **New Group** checkbox and the resulting group number. Checking **New Group** starts a new Frame group — a new canvas rectangle — with that Frame as its master. This is where Free Form's grouping happens; Free Form's Frame Setup page has no New Group column.
- **Positioning.** Select a Frame-group rectangle to fill the **Position X / Y** and **Size Width / Height** fields, or drag and resize it directly on the canvas — see *Rearranging the groups* below.
- **Global Frame Rate.** Set the system frame rate here (default **59.94 Hz**). Set it early — changing it re-runs the allocation.
- **Mix Effects.** Choose **ME Mode** (provisions Mix Effects) or **Layers Mode** (provisions Aux MEs). **Auto Provision** checked fills UHD first, then HD; unchecked, allocate UHD / HD manually while watching the bandwidth indicator stay under 100%. *Layers Only Mode* = Auto Provision unchecked **and** UHD = HD = 0.
- **Bandwidth Allocation bar** (bottom of the left panel) is color-coded: **red = Canvas**, **orange = UHD Mix Effects**, **blue = HD Mix Effects**, **yellow = System**. Canvas size and ME capacity trade directly — a larger canvas leaves fewer MEs; grouping Frames recovers ME capacity for a given canvas size.

Working with more than one Frame group. The default is a single group holding every Frame — the one canvas rectangle in Figure 6-6. To split the system, tick **New Group** on the Frame the next group should start at. That Frame becomes the new group's master and takes the Frames below it with it, so ticking **New Group** on Frame 3 of a four-Frame system gives two groups — Frames 1–2 and Frames 3–4 — and a second canvas rectangle, auto-placed to the right of the first:

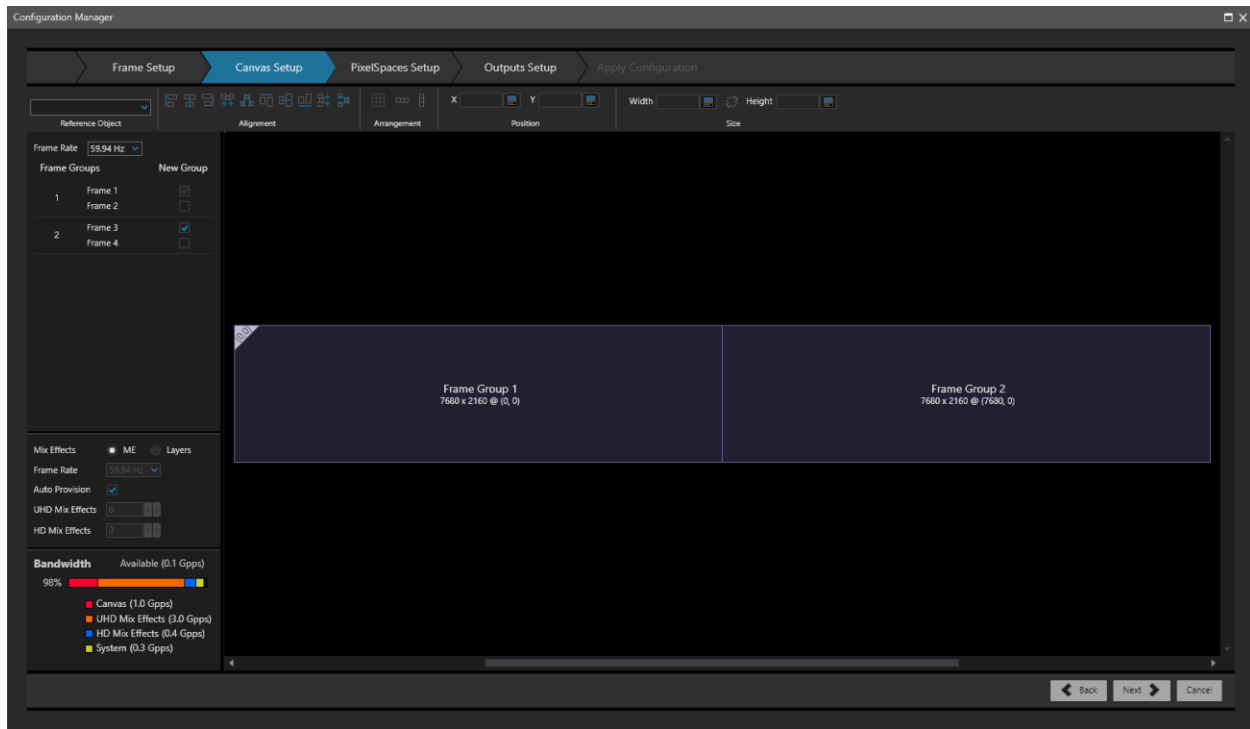


Figure 6-7. Canvas Setup with two Frame groups. Frames 1 and 2 form Group 1, Frames 3 and 4 form Group 2, and each group has its own 7680 x 2160 canvas. Group 2 is placed at X = 7680, so the two tile into a 15360 x 2160 total canvas.

Two details are worth noting in that figure:

- **Frame 1's New Group box is ticked and greyed out.** The first Frame is always a group master, so you cannot un-group it — which is why a system always has at least one Frame group.
- **The Bandwidth panel has not moved.** Compare it with Figure 6-6: the total canvas has doubled to 15360 × 2160, yet **Canvas** still costs **1.0 Gpps** and the Mix Effect allocation is untouched. Canvas bandwidth is charged on the **widest single group**, not on the tiled total — which is exactly why grouping is the lever that buys back Mix Effect capacity on a wide system.

To merge a group back, untick the same box: the Frames rejoin the group above and the extra canvas rectangle disappears.

Rearranging the groups. Click a group's rectangle to select it — it turns blue, picks up resize handles, and its row highlights in the Frame Groups list — then drag it to a new place on the canvas. The groups are not locked into a row: drag Frame Group 2 underneath Frame Group 1 and the canvas becomes two stacked bands, and drag it back out to the right to return to a side-by-side pair.

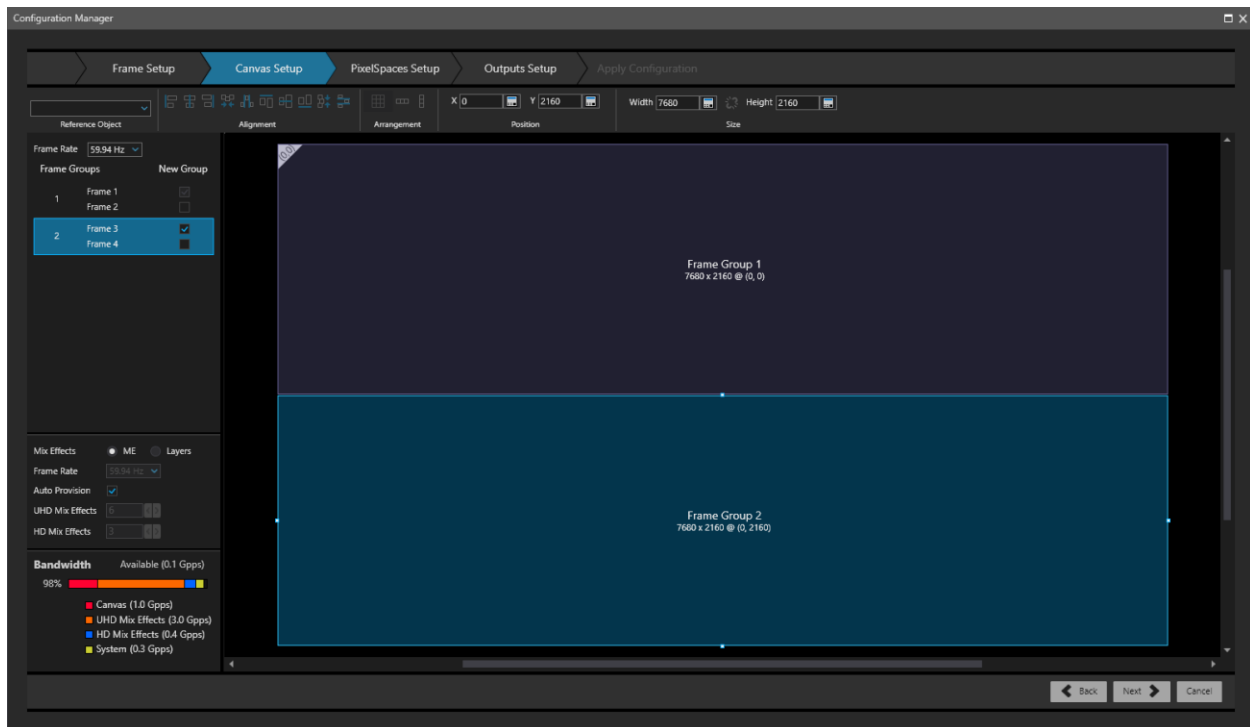


Figure 6-8. The same two Frame groups rearranged: Frame Group 2 has been dragged underneath Frame Group 1 and now sits at (0, 2160). The Position and Size fields track the selected group, and the Bandwidth panel is unchanged — rearranging groups costs nothing.

Three things to know before you drag:

- **Edges snap.** Release a group near another group's edge and it lands flush against it — dropping Frame Group 2 roughly below Frame Group 1 snapped it exactly to **(0, 2160)**, with no gap to clean up.
- **A drop away from an edge does not snap, and groups may overlap.** Released in open space, a group simply stays where you dropped it, even if that puts it on top of another group — the page will not stop you or warn you.
- **Use Position X / Y for exact placement.** Typing the coordinates is the reliable way to undo an overlap or to place a group precisely; the alignment toolbar (Align, Grid Arrange, Stack, Remove Spacing, Distribute) does the same job for several groups at once.

Auto Provision: who chooses the Mix Effect counts. On a **new** configuration **Auto Provision is ticked**, and the engine decides: it fills **UHD Mix Effects first, then HD**, using whatever bandwidth the canvas and backgrounds leave free. The two count boxes are greyed because they are outputs of that calculation, not inputs to it. Untick it and the boxes become editable, so you can set the mix yourself while watching the Bandwidth bar stay under 100 %.

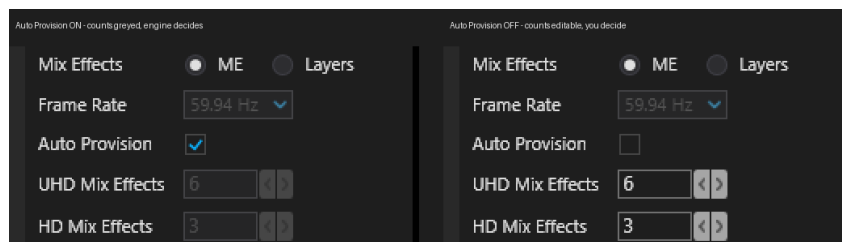


Figure 6-9. Auto Provision ticked and unticked. Ticked, the engine sets the counts and greys them; unticked, you set them. Everything else on the page is identical, so the tick and the greying are the only things to look for.

⚠ **Opening an existing configuration arrives with Auto Provision UNTICKED**, holding the counts that configuration was built with. That is the opposite of the new-configuration default, so check the box before assuming the engine is still managing the allocation for you.

Rearranging never changes the bandwidth budget. Side by side or stacked, **Canvas** still costs the same, because it is charged on the widest single group rather than on the shape the groups make together.

Canvas limits:

Limit	Value
Minimum size	1280 × 720
Maximum width	98304
Maximum height	2160 (2600 at ≤ 30 Hz)
Maximum pixels	70,778,880

The height limit applies to **each Frame group's own canvas**, not to the shape the groups make together: a single group cannot be set taller than 2160 at 59.94 Hz — type a larger value and the field reverts — but two 2160-high groups can be stacked, as in Figure 6-8, for an overall 7680 × 4320.

Tip: How you split Frames into groups (via the **Frame Groups** list's **New Group** checkboxes on this page) sets both the **total canvas size** (the tiled sum of the group rectangles) and how much bus bandwidth is left for **Mix Effects** (canvas bandwidth tracks the single widest group). See Section 5.9.1 for the trade-offs and Appendix C for the math.

The page validates the bandwidth budget; if used exceeds 4.728 Gpps it raises the bus-exceeded Warning. Click **Next** to advance to PixelSpaces Setup. **Back** returns to Frame Setup.

6.4.1.3 Step 3 — PixelSpaces Setup

PixelSpaces Setup places PixelSpaces over the canvas with the same graphical editor.

Prerequisite: Frame Setup and Canvas Setup are complete (the canvas exists).

A new configuration arrives here **empty** — no PixelSpaces, the canvas outline bare, and the Inputs meter reading **0 of 16. Apply Configuration is greyed out** until at least one PixelSpace exists, because a configuration without one is invalid.

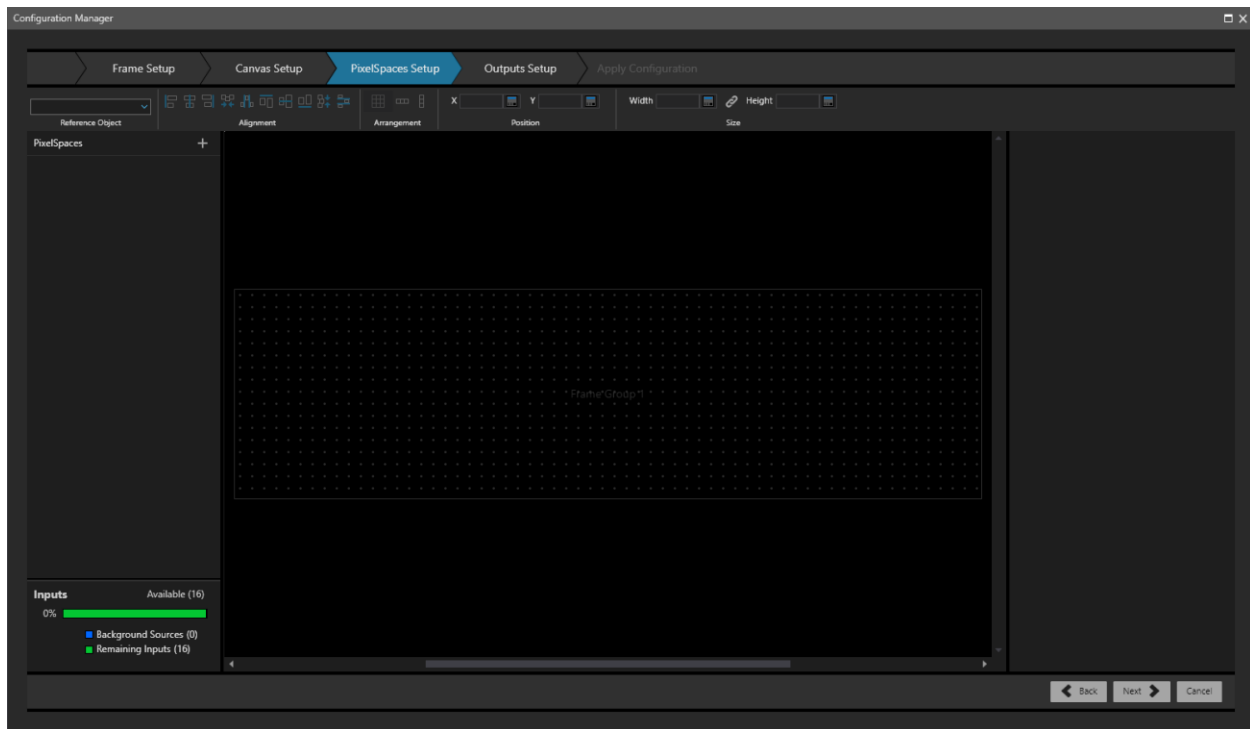


Figure 6-10. PixelSpaces Setup as a new configuration opens: no PixelSpaces yet, the Inputs meter at 0 of 16, and Apply Configuration greyed out until you add one.

1. Click **+ Add PixelSpace** to open the **Add PixelSpace** dialog.
2. Set **Name** (auto-fills *PS-1*), **Enable Preview**, **Array** (Horizontal × Vertical), **Resolution** (H × V, default 3840 × 2160), and **Gap / Overlap** (H × V). A live **PixelSpace Size** read-out shows the resulting dimensions (for example, 1 × 1 at 3840 × 2160 = a 3840 × 2160 PixelSpace).
3. Click **Add**. The new PixelSpace is auto-placed beside the last one, wrapping to a new row if it would overrun the canvas right edge.

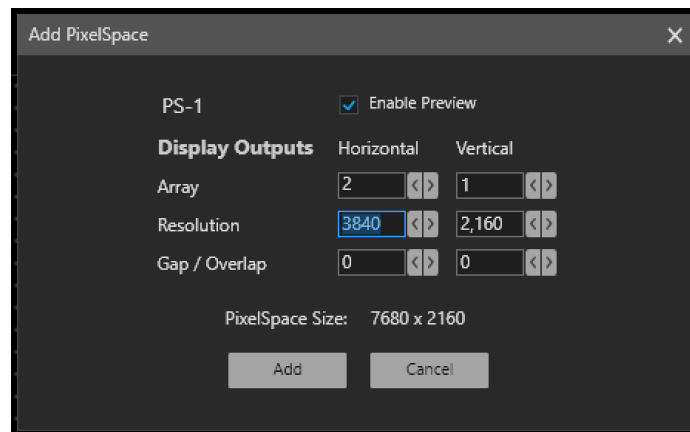


Figure 6-11. The Add PixelSpace dialog. Array x Resolution is what sets the PixelSpace size: a 2 x 1 array of 3840 x 2160 outputs gives the 7680 x 2160 shown in the live PixelSpace Size read-out.

4. Position and size it on the canvas — drag it, or use the **Position X / Y** and **Size Width / Height** fields and the alignment toolbar.

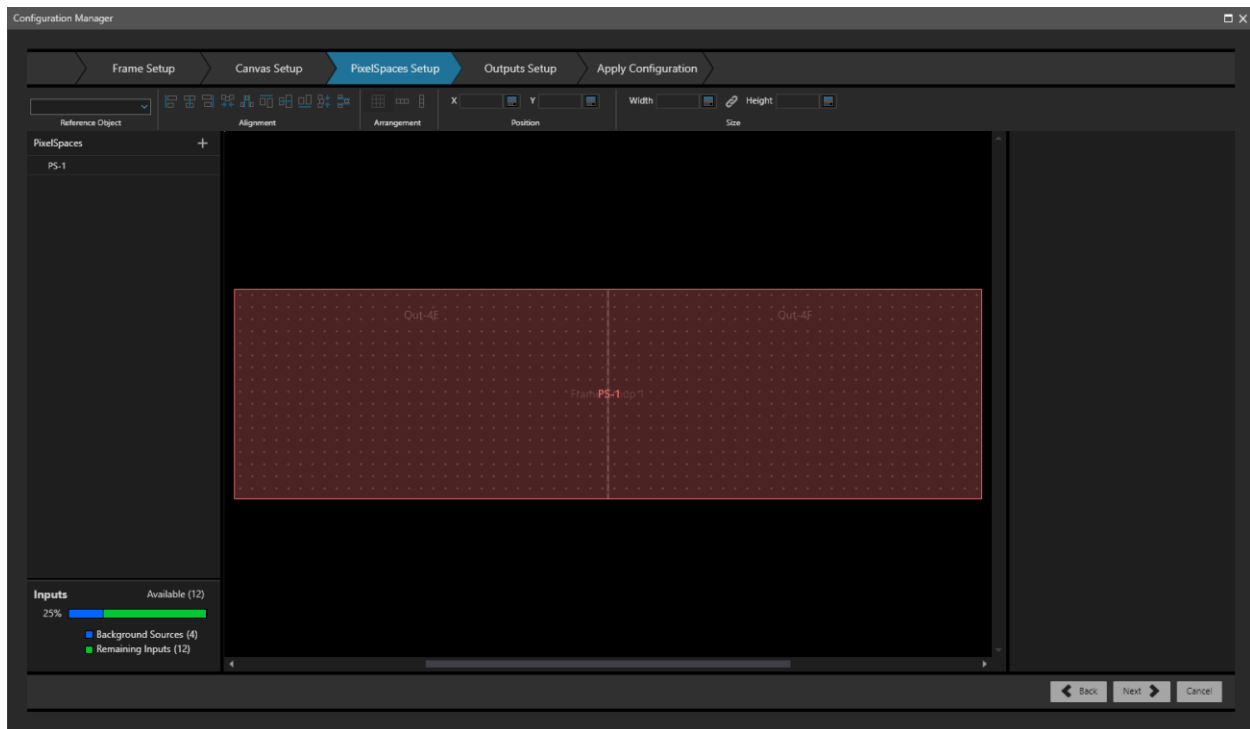


Figure 6-12. The result: PS-1 in place on the canvas, drawn red because it is not selected. Select it to open its detail panel (Figure 6-13).

Selecting a PixelSpace opens an inline detail panel with a **General** group (Name, Enable Preview) and a **Background Sources** group.

Configuring background sources. A **background source** defines which physical inputs feed a PixelSpace's background. Single-click a PixelSpace so it turns from **red** (unselected) to **blue** (selected), then click **+ Add BG Source**. A Background Group is added with a default **Input Res** chosen to fill the PixelSpace — for a 7680 × 2160 PixelSpace that is 3840 × 2160, two inputs side by side — and the platform assigns the first available inputs.

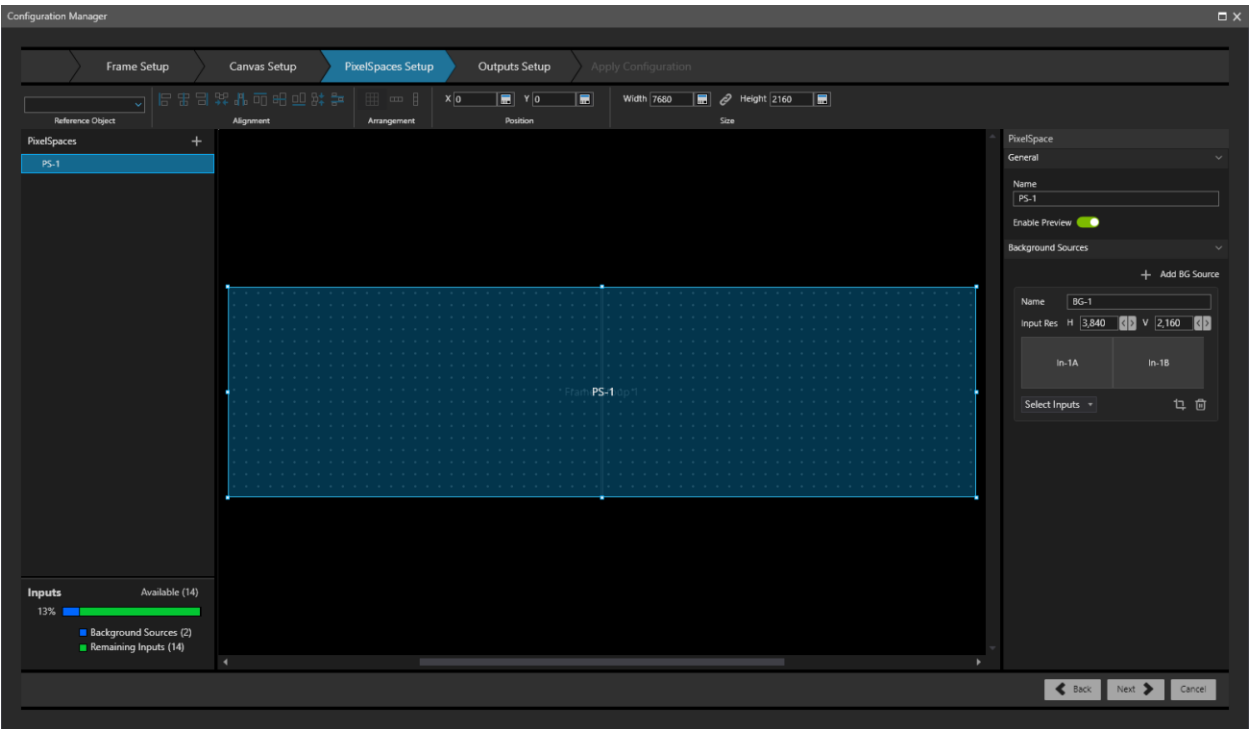


Figure 6-13. PixelSpaces Setup with one background source configured. PS-1 is selected, so it is drawn blue and fills the canvas, and its detail panel opens on the right showing the Background Sources group. The Inputs meter at the bottom left reports two of the sixteen inputs in use.

Each background group is a card in that panel, with these controls:

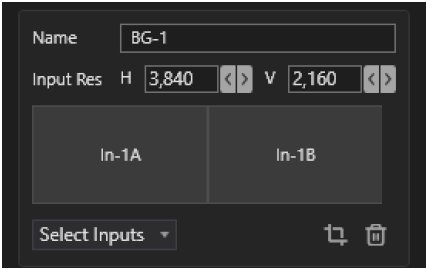


Figure 6-14. The background-source card in close-up: name, input resolution, the assigned input tiles, and the Select Inputs, Cropping, and Delete controls.

Control	What it does
Name	The group's name, auto-filled BG-1 , BG-2 , ... Names are unique across the whole configuration, so the second group on a second PixelSpace becomes BG-3 , not BG-2 again.
Input Res H × V	The resolution of one background input. Changing it changes the size and the number of inputs used.
Input tiles	One tile per input, labelled with the input it uses (In-1A , In-1B , ...), laid out as they map onto the PixelSpace.

Control	What it does
Select Inputs	A checkbox list of the inputs you may assign — see Figure 6-15.
Cropping	Opens the Crop controls — see Figure 6-16.
Delete	Removes the background group.

Select Inputs lists each input with its **Frame Group**, **input**, and **connector**, so you can tell at a glance which physical connector you are about to use. The inputs this source already holds are ticked:

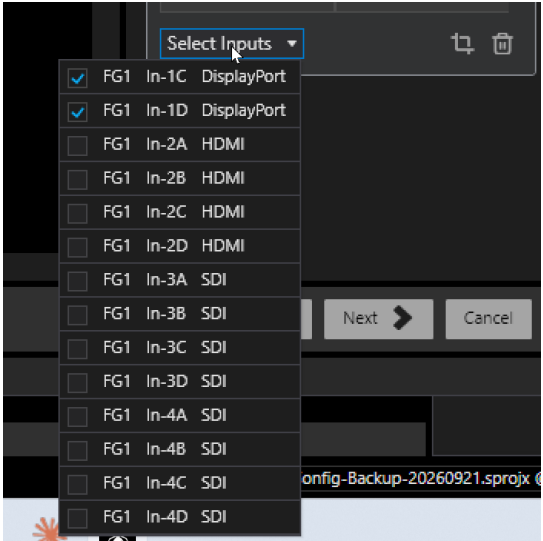


Figure 6-15. Select Inputs on a rig with DisplayPort, HDMI, SDI and SFP Frames — SFP inputs list as SDI. The source holds In-1C and In-1D; In-1A and In-1B are absent because another background source already holds them.

Inputs already assigned to another background source — on this PixelSpace or any other — are **not** listed at all, so the list only ever offers inputs you can actually take. The list also leaves out inputs whose connector cannot carry the source's **Input Res**, and it only offers inputs from the frame groups that the PixelSpace covers. When a DisplayPort input is carried on a pair (see below), the second input of the pair is shown ticked and greyed out.

Swapping an input. The number of ticked inputs always equals the number of tiles. When the list is full, ticking another input does nothing — **untick an input first, then tick the new one**. The change takes effect when the list closes, and the new input takes the **same place on the PixelSpace** as the input you unticked, including any crop. To put a particular input in a particular place, untick the input that is currently there. Swap one input at a time: if you untick and tick several inputs before closing the list, the new inputs fill the freed places in list order rather than in the order you clicked them.

Cropping is not an inline field — it is behind the **Cropping** button. Clicking it opens the **Crop** controls over the page: **Top / Left / Right / Bottom**, and **X Position / Y Position**. **Width** and **Height** are shown for reference but cannot be edited here. There is no OK or Cancel; the values apply as you set them, and the controls close when you click elsewhere.

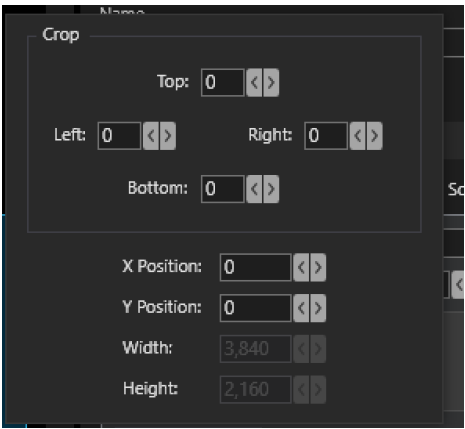


Figure 6-16. The Crop controls, opened from a background source's Cropping button. Width and Height are shown for reference but cannot be edited here.

A background source is not confined to one Frame or one connector type — a single source can span HDMI inputs on one Frame and SDI inputs on another.

Note: A background input is only shown over **its own frame group**. If a tile sits over the part of the PixelSpace that belongs to another frame group, that part of the output is **black**. Configuration Manager does not warn you, and when a PixelSpace spans more than one frame group it can place inputs this way by itself. Open **Select Inputs** and check that each tile's input belongs to the frame group (**FG1**, **FG2**, ...) underneath it.

How many inputs a background source uses. Inputs are allocated as a grid, $\text{ceil}(W / H_{\text{Active}}) \times \text{ceil}(H / V_{\text{Active}})$. On a 7680 × 2160 PixelSpace, an Input Res of 1920 × 1080 therefore uses **4 × 2 = eight** inputs, while 3840 × 2160 uses two.

Watch the Inputs meter at the bottom-left of the page. It reads **Available (n)** with a bar split into **Background Sources** and **Remaining Inputs**, and it is the only place that shows what your background sources are really costing — including the case below, where one background input quietly consumes two.

One background input can consume two physical inputs. The maximum a single background input can carry depends on the connector:

Connector	Maximum for one background input	Physical inputs used
DisplayPort	7680 × 2160	one input up to 4096 × 2160; two above that
HDMI	7680 × 1080	always one

The supported background input formats are summarised in Appendix E.1.

When a DisplayPort background input goes above 4096 × 2160 it is carried on a **pair** of inputs — **A + B**, or **C + D**. Both inputs of a pair are always on the same Frame, so a pair never spans two frame groups. Only the first of the pair is labelled on the tile, so a source on DisplayPort Frame 1 shows a tile marked **In-1A** while silently also using In-1B. The Inputs meter is where you see the true cost:

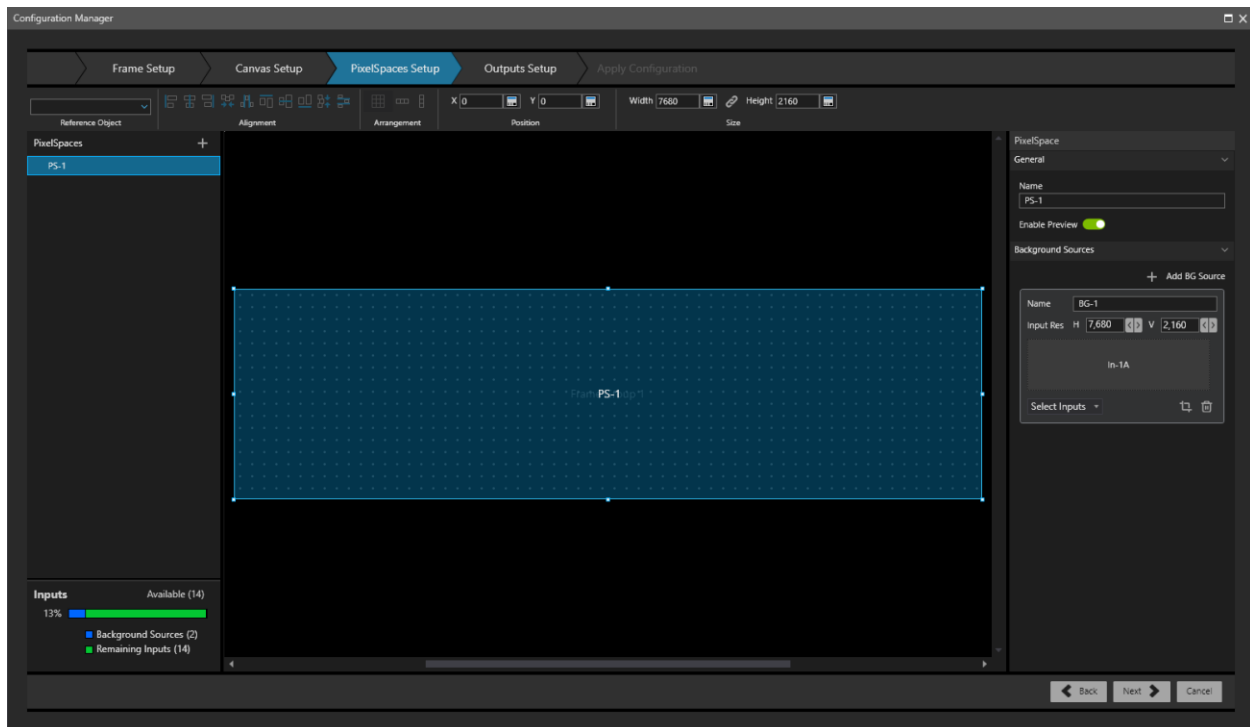


Figure 6-17. One background input at 7680 x 2160 on DisplayPort. It shows a single tile, In-1A — but the Inputs meter reports Background Sources (2), because above 4096 x 2160 the input is carried on a pair.

Note: This is a limit on total picture, not on width alone. A DisplayPort input at **7680 × 1080** is far wider than 4096 and still uses a **single** input, because it stays within the same budget as 4096 × 2160.

If the resolution will not fit, no inputs are assigned at all — the tiles disappear, the footer reads "*Warning: No inputs available for Background Source {name}*", and **Apply Configuration** is greyed out until you fix it. HDMI never falls back to a pair, so asking an HDMI background input for more than 7680 × 1080 always fails this way.

How the inputs are placed and cropped. The tiles fill the PixelSpace from its top-left corner, left to right and then top to bottom, one Input Res cell per tile. When the Input Res does not divide the PixelSpace exactly, only the **rightmost column** is cropped on its right edge and only the **bottom row** on its bottom edge — no other tile is cropped. For example, an Input Res of 3000 × 2160 on a 7680 × 2160 PixelSpace places three tiles at X 0, 3000 and 6000; the last one is cropped by 1320 on the right, leaving 1680 visible. To crop or position a tile further, select it and open **Cropping**; the tiles follow the rules in *Cropping a tile by hand* below.

Adjusting the fit. Once a source exists you have three ways to change how it covers the PixelSpace:

- **Adjust the PixelSpace size** — Width / Height changes auto-crop and reposition the background inputs; the manual **Crop** and **Position** values behind the **Cropping** button override the automatic crop.
- **Change the Input Res** — this changes the size and number of inputs automatically. The whole grid is laid out again from the top-left corner and **all crops are reset**: the source keeps its current inputs (lowest input first, as many as still fit), adds the next free inputs if it needs more, and releases any it no longer needs.
- **Change the assigned inputs** — tick and untick them in **Select Inputs** (see *Swapping an input* above).

Cropping a tile by hand. Whichever side you crop, the tile stays anchored at its top-left corner (its **X / Y Position** does not change) and the remaining picture moves **left** and **up** into it — a Left or Top crop moves the picture itself, and a Right or Bottom crop just shortens it. Tiles to its left or above it do not move. Tiles to its **right in the same row** move **left**, and tiles **below it in the same column** move **up**, by the crop amount; reducing the crop moves

them back. The tiles that move keep their own crops, so a matching strip at the right or bottom edge of the PixelSpace is left **uncovered** — Studio does not fill it.

Mixing and cutting backgrounds needs two groups. With only **one** Background Group on a PixelSpace, its background cannot be Mixed or Cut from Program to Preview — there is no second background to transition to. You can see it in the Mix Effects pane: that PixelSpace's background Mix Effect carries a single content slot, where an ordinary Mix Effect has two.

Click **+ Add BG Source** again to add a second group. The background Mix Effect then shows two slots — **BG-1** on Program and **BG-2** on Preview — and Mix and Cut work as they do for any other Mix Effect (Section 7.5.3):

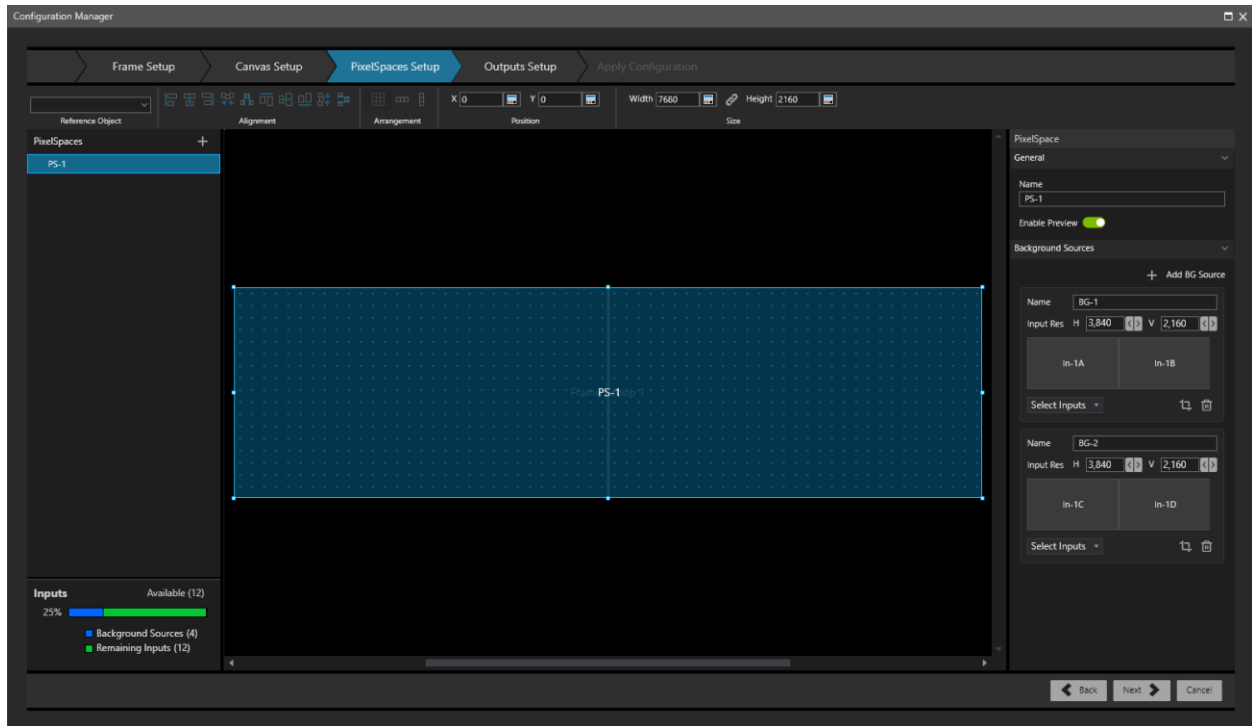


Figure 6-18. The same PixelSpace with two background groups. BG-1 takes In-1A and In-1B, BG-2 takes In-1C and In-1D, and the Inputs meter has gone from two to four.

Budget for it: the second group takes its own full set of inputs, so a PixelSpace that used two inputs now uses four.

Which Frame you use for backgrounds decides which layers you keep. Within a frame group, a layer on a Frame that is numbered **lower** than the highest Frame holding a background input is drawn **behind** the background, so Spyder **locks** it. For example, in a frame group of Frames 1 and 2, a background on In-2A locks the four layers on Frame 1 — even though Frame 1 holds no background input.

Tip: Use the **first inputs in each frame group** for backgrounds — start with the lowest-numbered Frame in the group (for example In-1A, In-1B, ... on Frame 1). No layers are locked, and every other input stays available for layers.

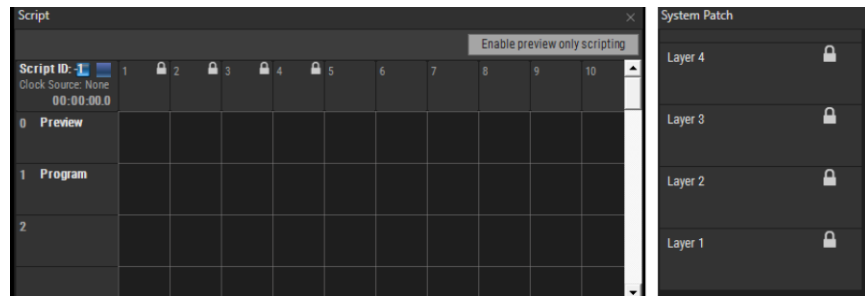


Figure 6-19. Locked layers. Both background sources use inputs on Frame 2, so the layers on Frame 1 (Layers 1-4) are locked: each shows a padlock in the Script header and in System Patch.

Notice: Configuration Manager does not warn you that layers will be locked; check for padlocks after you apply. When you drag a source onto the canvas, Spyder skips locked layers and uses the next unlocked one. Selecting a locked layer shows *"This layer is currently locked, adjustments to this layer are unavailable"* in **Properties**, with an **Unlock Layer** link. If you unlock a layer and use it, its content is hidden **behind the background** on Program, although it still shows on Preview.

The page requires **at least one PixelSpace**, or it returns a Warning. Click **Next** to advance to Outputs Setup. **Back** returns to Canvas Setup.

6.4.1.4 Step 4 – Outputs Setup

Outputs Setup is the full output editor: a **Default Resolution** dropdown and a per-output list beside the canvas.

A new configuration arrives with **every output Unused** and nothing on the canvas — you place them yourself. The capability letters are already populated, because they come from the cards rather than from anything you have done yet.

- **Default Resolution** — the resolution assigned to outputs as they are dropped onto the canvas: 1280×720 · 1920×1080 · 2048×1080 · 3840×2160 · 4096×2160 (default **3840×2160**).
- **Output list** — each row shows Frame Group, Output (editable name), Connector, Mode, and the **N / A / O / T** capability flags (see Section 6.5), and ends in a colored dot that shows the output's mode (colors in Section 6.5). An unplaced output shows **Unused**, with a gray dot.
- **Placing outputs** — drag an output from the list onto the canvas; outputs tile left-to-right within the target Frame. **X / Y** is editable for any output; **Width / Height** is editable only for Scaled outputs. Precision-placement controls (Reference Object, Alignment, Arrangement, Position) sit in the top toolbar.
- **Frame Group scope** — on a **one-group** canvas, outputs can be placed anywhere. On a **multi-group** canvas, an output can only be placed in the canvas area of its **own** Frame Group: a valid drop area shows **green**, other groups show **red** and reject the drop.

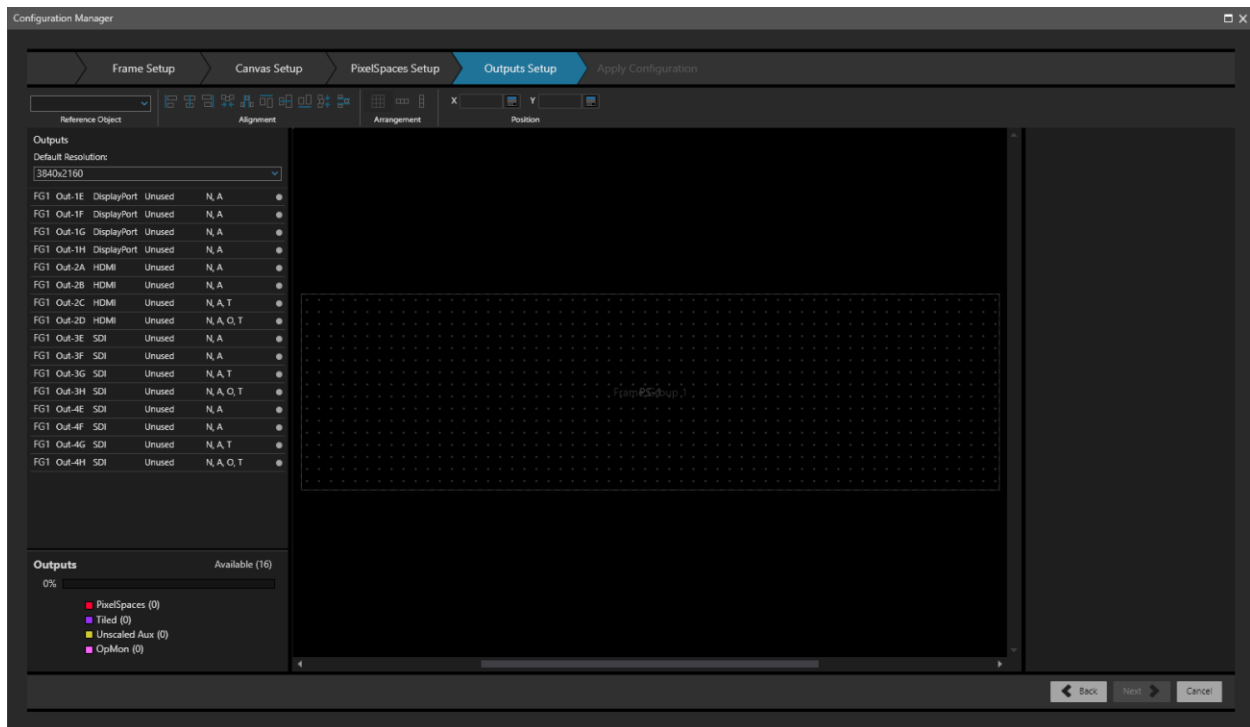


Figure 6-20. Outputs Setup as a new configuration opens: every output Unused, the Outputs meter at 0 %, and the N / A / O / T capability letters already filled in from hardware discovery.

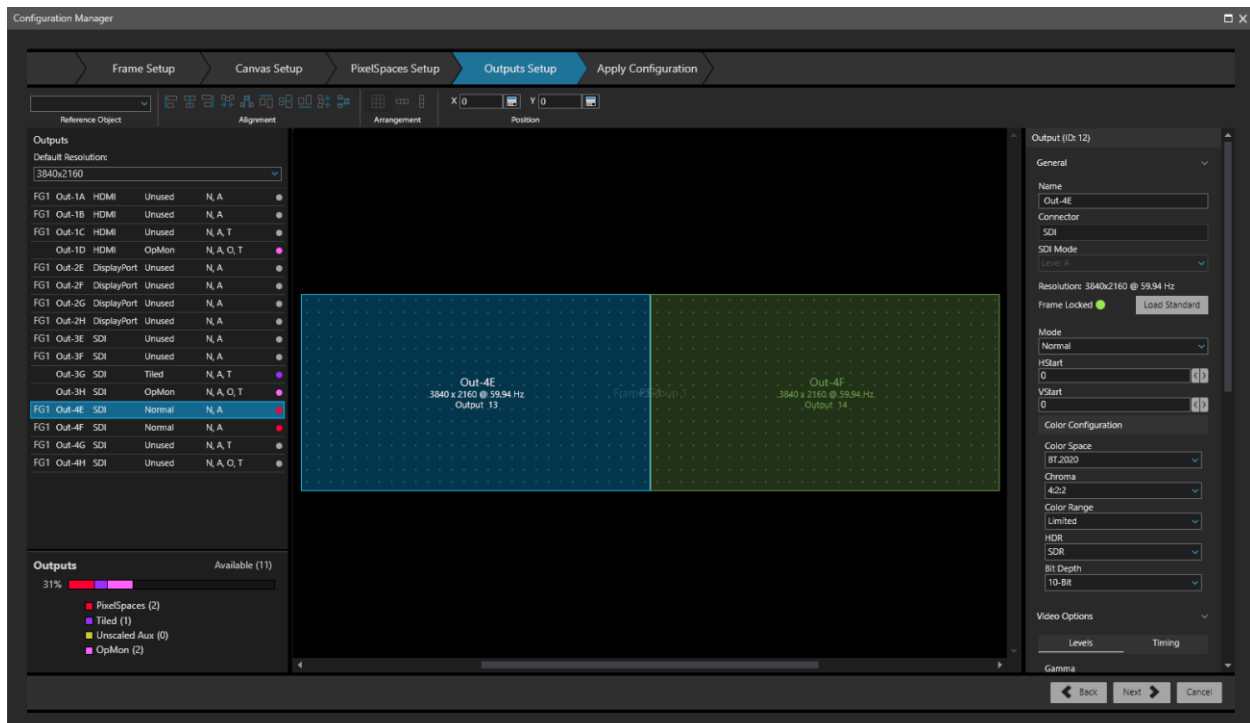


Figure 6-21. The same page once outputs have been placed: Out-4E and Out-4F (Normal) carry the PixelSpace, Out-3G is Tiled, Out-1D and Out-3H are OpMon, and the Outputs meter updates to match. Out-4E is selected — blue in the list and on the canvas — and its Output properties open on the right. The dot at the end of each row shows the mode: red Normal, purple Tiled, magenta OpMon, gray Unused.

Selecting an output opens its inline **Output properties** panel with three collapsible groups. (This is the configurator's cut-down editor — the live panel reached from System Patch or System Layout adds a **Test Pattern Generator** group, Section 7.7.3.)

Group	Contents
General	Name, Connector, read-only Resolution, Frame Locked + Load Standard, Mode (capability-filtered), HStart / VStart, and Color Configuration (Color Space, Chroma, Color Range, Bit Depth).
Video Options	Levels (Gamma only in Spyder Studio 6.1.3) and Timing (Enable custom timing → Vertical / Horizontal Offset).
Advanced	A property grid exposing every output property — full timing, HDR , and the connector- and mode-dependent fields (for example SDI Mode).

Note: Resolution is read-only; change it via **Load Standard**.

Note: Applying a configuration created in 6.1.3 or later enables **alternate output sync** on every output, whatever its connector type, which keeps the outputs correctly synchronized. The property itself is **UseAlternateOutputSync** in the Advanced group (Section 7.7.4). Configurations created before 6.1.3 keep their existing setting until you create and apply a new one.

The page requires **at least one active output**, or it returns a Warning. Click **Next** to advance to Apply. **Back** returns to PixelSpaces Setup.

6.4.1.5 Step 5 — Apply Configuration

The final page is a Christie Spyder-S screen with a single action. **Which button appears depends on the connection state:**

Button (exact label)	When	Effect
Apply to local PC now (Virtual Mode)	Offline / not connected	"Builds a virtual configuration on your local PC. It lets you work while away from Spyder Hardware."
Apply to Spyder hardware now	Connected	"Your configuration is applied to Spyder hardware currently on the network."

The button is enabled only when the configuration **is valid** (Good) — which, for Free Form, requires at least one PixelSpace and at least one active output. Apply runs behind an "Applying Configuration..." overlay and the window auto-exits on success. On failure, the footer directs you to the Alert Viewer.

Notice: Apply commits the **Clear Existing Data** selections from Frame Setup (Section 6.4.1.1). For a new configuration these are normally unchecked.

6.4.2 Load the current configuration

From the Configuration Mode screen, select **Free Form** and click **Load current configuration**. The wizard opens on the same five pages — **Frame Setup** → **Canvas Setup** → **PixelSpaces Setup** → **Outputs Setup** → **Apply** — each

prepopulated with the configuration currently in effect so you can edit it. Because a live configuration already has PixelSpaces and outputs, it loads **valid**. Walk the five steps below.

6.4.2.1 Step 1 – Frame Setup (loaded)

Frame Setup opens with the **current Frame set already listed**, using Free Form's column set (Frame / Model / Inputs / Outputs):

- **Offline** — the loaded Frames are **editable** (add, delete except the master, re-model).
- **Connected** — read-only; the list mirrors the discovered hardware.

Clear Existing Data — read this when editing a live configuration. The seven checkboxes (Command Keys, Function Keys, Treatments, Sources, EDIDs, Background Stills, Tile Presets) wipe the checked data on Apply.

Notice: When an edit **changes an input category** — for example adding background inputs not previously configured — check the relevant boxes (typically **Sources** and **Background Stills**) so stale data does not remain after Apply. Clear only what you intend to rebuild.

Click **Next** to advance to Canvas Setup.

6.4.2.2 Step 2 – Canvas Setup (loaded)

Canvas Setup opens with the **Frame-group canvas rectangles already positioned**. Every control behaves as for a new configuration (Section 6.4.1.2): re-position and resize the group rectangles with the alignment toolbar or the **Position X / Y** and **Size Width / Height** fields; the **Global Frame Rate**, **Mix Effects / Layers** mode, and **Auto Provision** state are loaded from the current configuration; the color-coded **Bandwidth Allocation** bar (Canvas / UHD ME / HD ME / System) updates live.

Notice: Regrouping Frames or resizing the canvas on a loaded configuration re-runs the allocation and can push it over the 4.728 Gbps ceiling. Watch the Bandwidth Allocation bar. The canvas limits (min 1280 × 720; max width 98304; max height 2160, or 2600 at ≤ 30 Hz; max 70,778,880 pixels) still apply.

Click **Next** to advance to PixelSpaces Setup. **Back** returns to Frame Setup.

6.4.2.3 Step 3 – PixelSpaces Setup (loaded)

PixelSpaces Setup opens with the **current PixelSpaces already placed** on the canvas. Editing works exactly as for a new configuration (Section 6.4.1.3):

- Select a PixelSpace to open its inline **General** and **Background Sources** detail; re-position or resize it on the canvas or via the Position / Size fields.
- Add a PixelSpace with **+ Add PixelSpace** (Name, Enable Preview, Array, Resolution, Gap / Overlap, with the live **PixelSpace Size** read-out) or remove one.
- Add or edit **Background Sources**: single-click a PixelSpace so it turns from red to blue, then **+ Add BG Source**. The default Input Resolution rules (7680 × 2160 → 3840 × 2160; 3840 × 2160 → 1920 × 1080) and the three adjustment options (adjust PixelSpace size, change Input Resolution, change assigned Inputs) apply as before. Remember that a **second** Background Group is required to Mix / Cut a background between Program and Preview.

At least one PixelSpace must remain, or the page warns. Click **Next** to advance to Outputs Setup. **Back** returns to Canvas Setup.

6.4.2.4 Step 4 – Outputs Setup (loaded)

Outputs Setup opens with the **current outputs already placed** on the canvas and the rest available in the list. Editing works exactly as for a new configuration (Section 6.4.1.4):

- Re-assign the **Default Resolution** for newly-dragged outputs; drag additional outputs from the list onto the canvas (tiling left-to-right within a Frame; multi-group placement is restricted to the owning Frame Group — green valid, red rejected).
- Edit **X / Y** for any output and **Width / Height** for Scaled outputs; use the precision-placement toolbar (Reference Object, Alignment, Arrangement, Position).
- Select any output to open its three-group **Output properties** panel (General, Video Options, Advanced) and change Name, Mode (capability-filtered — see Section 6.5), Color Configuration, timing, or HDR.

At least one active output must remain, or the page warns. Click **Next** to advance to Apply. **Back** returns to PixelSpaces Setup.

6.4.2.5 Step 5 — Apply Configuration (loaded)

The Apply page is identical to the new-configuration case (Section 6.4.1.5). The button reflects the connection state — **Apply to Spyder hardware now** (connected) or **Apply to local PC now (Virtual Mode)** (offline) — and is enabled only when the edited configuration is **valid**.

Notice: Apply commits the **Clear Existing Data** selections you made in Step 1. On a loaded configuration this is where stale sources, backgrounds, keys, or presets are actually removed — confirm your selections before applying.

6.5 Output roles and capability flags

Both wizards configure each physical output for one role, and both report each output's hardware capability as a set of letter flags, filtering the selectable modes to match. This reference applies to the All In One System Mapping page and the Free Form Outputs page alike.

The per-output list shows each output's supported modes as letters:

Flag	Mode	Meaning	Color
N	Normal	A scaled program output for a PixelSpace.	Red
A	UnscaledAux	An unscaled Aux feed (UnscaledAux, <i>not</i> plain Aux).	Yellow
O	OpMon	An output monitor / multiviewer.	Magenta
T	Tiled	One output driving a grid of tiles.	Purple

The modes an operator can select are **Normal**, **Unscaled Aux**, **OpMon**, **Tiled**, and **Unused**. The first four carry a capability letter; **Unused** (gray) marks an output not yet placed. An Unused output is not blank: a display connected to it shows a default test pattern labeled with the output's name (for example *Out-1A*), which helps you identify the port. **Mode is set in the output's Properties panel**, whose dropdown is filtered to the modes the port's hardware capability allows.

Each mode has a color, and the same color is used in three places: the dot at the end of the output's row in the Outputs Setup list, the Outputs availability chart below that list, and the output's LED on the rear panel of the Spyder-S Frame (Section 3.3.2). An operator can see how every output is set by looking at the back of the rack.

Mode	Dot in the output list	Availability chart	Output LED
Unused	Gray (#A5A5A5)	Not shown	Off
Normal (on the canvas)	Red (#FD0031)	PixelSpaces	Red
Unscaled Aux	Yellow (#FCB36)	Unscaled Aux	Yellow
OpMon	Magenta (#FF61FF)	OpMon	Magenta
Tiled	Purple (#9B2FFE)	Tiled	Purple

Notice: The flags come from hardware discovery, not the model name. An output offers only the roles its card supports — a port that cannot tile will not show **T**, and its Mode dropdown will not offer Tiled. Plan output roles against the actual card population.

You may see Tiled described as "outputs 3 and 4" and OpMon as "output 4 only". Treat those as a **rule of thumb, not a rule**. Capability is not decided by position: it comes from the card reported by hardware discovery, and different card types report different roles.

On the reference system used throughout this chapter, the rule of thumb holds on three Frames and fails on the fourth:

Frame	Card	Output 1	Output 2	Output 3	Output 4
1	DisplayPort	N, A	N, A	N, A	N, A
2	HDMI	N, A	N, A	N, A, T	N, A, O, T
3	SDI	N, A	N, A	N, A, T	N, A, O, T
4	SFP	N, A	N, A	N, A, T	N, A, O, T

The DisplayPort Frame supports Normal and Unscaled Aux outputs only. On its own it does not provide Tiled or OpMon outputs. When the DisplayPort Frame is combined with any other Frame type in the system, Tiled and OpMon outputs are available from those Frames — for example outputs 3 and 4 of an HDMI, SDI or SFP Frame, as shown in the table above.

► **Read the letters on the row, never the position.** They are the only thing that tells you what a given port will accept, and the Mode dropdown is filtered to them.

These are the same names the live output Properties panel uses, so a role means the same thing in the wizard and in operation (Section 7.7.1).

⚠ **A blank Mode dropdown means the output does not match the card.** The Mode dropdown is filtered to the capability letters, so if an output's current mode is not among them the list has nothing to show and the box appears empty. That happens when a **loaded configuration still carries an output the installed card no longer provides** — for example a configuration built when a Frame held a different card, then re-opened after the card was changed. Two outputs in the same mode on the same page make it obvious: the one whose card supports the role displays it, the stale one displays nothing.

How to check: open the same wizard with **Start with new empty configuration**, which builds its output list from the hardware actually present. Any output that appears in your loaded configuration but not in that fresh list is a leftover, and its Mode, connector and flags describe a port that is no longer there. Rebuilding the configuration clears them.

Tiled and OpMon outputs leave their Frame Group. In the output list their **Frame Group** cell is blank, while Normal *and* Unused outputs both show their group (for example FG1). The Frame Group column marks which group's canvas an output may be placed on — an Unused output is still placeable there, whereas a Tiled or OpMon output is a whole-output role that never occupies canvas area, so it belongs to no group. This is why the two roles also disappear from the canvas view.

6.6 Configuring OpMon outputs

An **OpMon (Output Monitor)** turns one multiviewer-capable output into a single monitoring screen: the Program and Preview PixelSpaces across the top, with rows of tiles below that you fill with inputs, background inputs, and Mix Effect buses. You configure it after applying a configuration, on the **OpMon Configuration** tab, which shares the Simulator pane region with **Simulator** and **Tile Configuration**.

Prerequisite: An output set to **OpMon** mode. Only outputs whose capability flags include **O** can be (Section 6.5). If no output is in OpMon mode, the tab shows "Please configure at least one output to OpMon mode to use this configuration window."

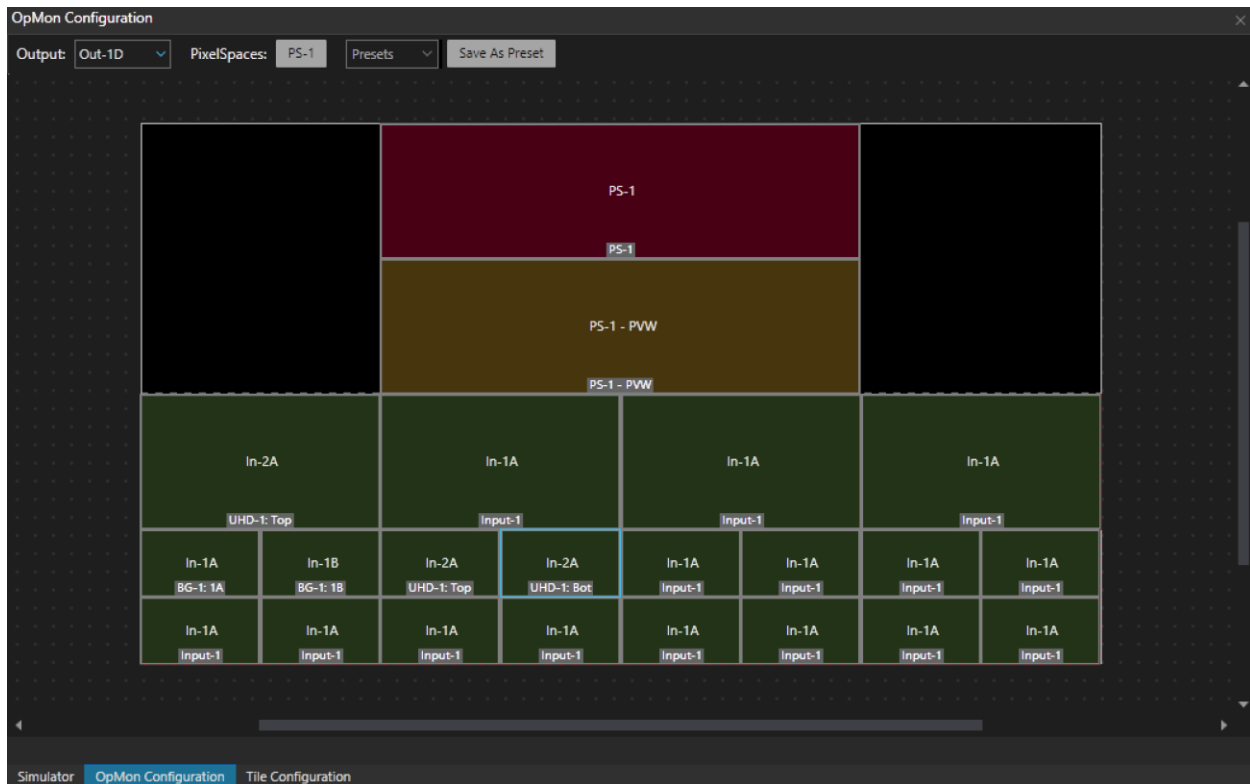


Figure 6-22. The OpMon Configuration tab for Out-1D. PS-1 and PS-1 - PVW fill the top; below are a row of four large tiles and two rows of eight small tiles. The first four small tiles show the examples in this section: BG-1 on two tiles (In-1A and In-1B) and UHD-1 on two tiles (Top and Bottom).

Each tile shows two labels: the input at the centre (for example **In-1B**), and what the tile carries at the bottom (for example **BG-1: 1B**, **UHD-1: Top**, or **Source 3**).

To open the OpMon layout:

1. Open the **OpMon Configuration** tab.
2. In the toolbar, choose the output in the **Output** dropdown.

To show an input on a tile:

1. Drag a source from the **Sources** pane onto the tile. The tile's bottom label shows the source name.

To show a background input on a tile:

1. Drag a background source from the **Backgrounds** pane onto the tile. The tile shows the source's first input, for example **BG-1: 1A**.
2. To show a different input of the same background source, right-click the tile and choose **Background Input**, then the input. The current input is ticked.

For example, to show both halves of BG-1, drag BG-1 onto two tiles, then set the first tile to **In-1A** and the second to **In-1B**.

To show a Mix Effect bus on a tile:

1. Drag a Mix Effect from the **Mix Effects** pane onto the tile. The tile shows the Mix Effect's **Top** bus, for example **UHD-1: Top**.
2. To show the **Bottom** bus instead, right-click the tile and choose **Toggle Top/Bottom Content**. Choose it again to switch back.

For example, to watch both buses of UHD-1, drag UHD-1 onto two tiles, leave the first on **Top**, and toggle the second to **Bottom**.

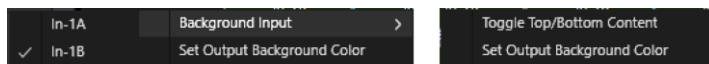


Figure 6-23. The tile right-click menus. Left: a background tile, with *Background Input* open to choose between *In-1A* and *In-1B*. Right: a Mix Effect tile, with *Toggle Top/Bottom Content*.

To set the output background color: right-click in the layout and choose **Set Output Background Color**. The color applies to the whole output.

Expected result: each tile shows the content you assigned, labeled with its input and content.

6.6.1 OpMon presets

A preset saves the current layout of an OpMon output so you can recall it later.

To save a preset:

1. Click **Save As Preset**.
2. In **Enter Preset Name**, type a name and click **OK**. The name cannot be empty and must be different from every other preset; otherwise Studio asks again.

The **Presets** dropdown lists your presets. Point to a preset to open its commands:

Command	What it does
Apply	Loads the preset onto the selected output.
Edit Name	Renames the preset.

Command	What it does
Re-Learn	Replaces the preset with the output's current layout.
Delete	Deletes the preset after you confirm "Are you sure you want to delete multiviewer preset: <name>".

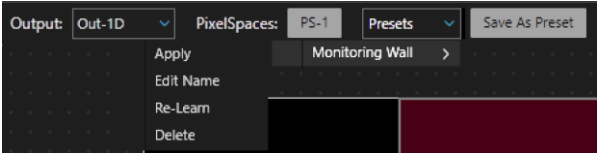


Figure 6-24. The Presets dropdown with one preset, Monitoring Wall, and its commands.

If a preset cannot be loaded, Studio shows "Error loading OpMon preset! Please ensure pixelspaces are available on the output's frame."

6.7 Configuring tiled outputs

A **Tiled** output is a single output made of a grid of tiles, each showing an input, a background input, or a Mix Effect bus — the way you drive a tiled display such as an LED wall, or monitor many sources on one screen. You configure it after applying a configuration, on the **Tile Configuration** tab, the third tab of the Simulator pane region.

Prerequisite: An output set to **Tiled** mode. Only outputs whose capability flags include **T** can be (Section 6.5). If no output is in Tiled mode, the tab shows "Please configure at least one output to Tiled mode to use this configuration window."

How many tiles you get. Each tile is a fixed **960 × 540**, so the grid is the output's resolution divided by that. Changing the output resolution changes the tile count:

Output resolution	Grid	Tiles
1920 × 1080	2 × 2	4
3840 × 2160	4 × 4	16

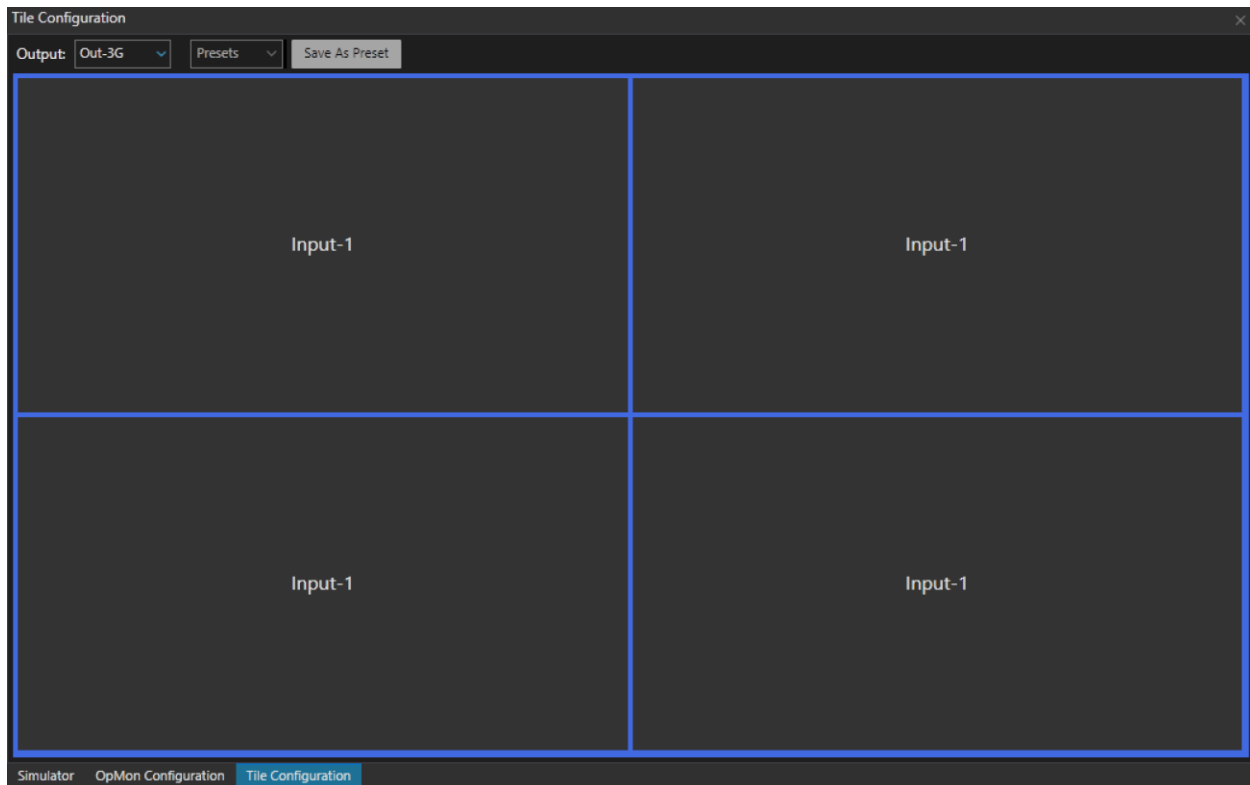


Figure 6-25. Tile Configuration for Out-3G at 1920 x 1080 — a 2 x 2 grid of four tiles, each showing Input-1 until you assign content.

To change the tile count:

1. Select the tiled output in **System Patch** or **System Layout**.
2. In the **Properties** panel, click **Load Standard**.
3. Choose **1920 x 1080** for a 4-tile grid, or **3840 x 2160** for a 16-tile grid, then click **Apply**.

After the change, every tile shows **Input-1** until you assign content. Studio does not fill the tiles for you.

To open the tile layout:

1. Open the **Tile Configuration** tab.
2. In the toolbar, choose the output in the **Output** dropdown. The list shows only outputs in Tiled mode.

To show an input on a tile:

1. Drag a source from the **Sources** pane onto the tile. The tile shows the source name, for example **Source 3**.

To show a background input on a tile:

1. Drag a background source from the **Backgrounds** pane onto the tile. The tile shows the source's first input, for example **BG-1: 1A**.
2. To show a different input of the same background source, right-click the tile and choose **Background Input**, then the input. The current input is ticked.

For example, to show both halves of BG-1, drag BG-1 onto two tiles, then set the first tile to **In-1A** and the second to **In-1B**.

To show a Mix Effect bus on a tile:

1. Drag a Mix Effect from the **Mix Effects** pane onto the tile. The tile shows the Mix Effect's **Top** bus, for example **UHD-1: Top**.
2. To show the **Bottom** bus instead, right-click the tile and choose **Toggle Top/Bottom Content**. Choose it again to switch back.

For example, to watch both buses of UHD-1, drag UHD-1 onto two tiles, leave the first on **Top**, and toggle the second to **Bottom**.

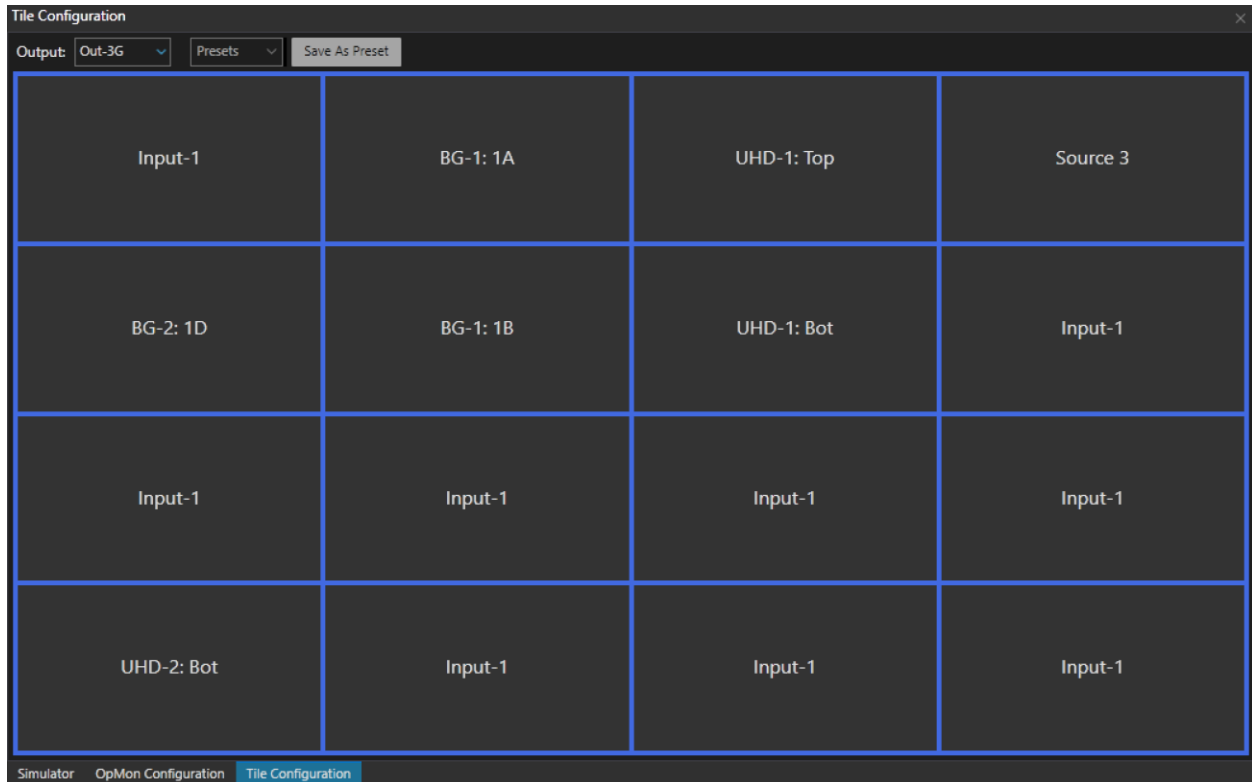


Figure 6-26. Out-3G at 3840 x 2160 — a 4 x 4 grid of sixteen tiles showing the examples in this section: BG-1 on two tiles (1A and 1B), UHD-1 on two tiles (Top and Bot), and Source 3.

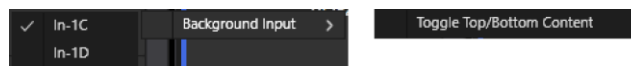


Figure 6-27. The tile right-click menus. Left: a background tile, with Background Input open to choose between In-1C and In-1D. Right: a Mix Effect tile, with Toggle Top/Bottom Content.

Expected result: each tile shows the content you assigned.

Note: The tile grid is per output — switch the **Output** dropdown to configure a different tiled output.

6.7.1 Tile presets

A preset saves the current tile layout of a Tiled output so you can recall it later.

To save a preset:

1. Click **Save As Preset**.
2. In **Enter Preset Name**, type a name and click **OK**. The name cannot be empty and must be different from every other preset; otherwise Studio asks again.

The **Presets** dropdown lists your presets. Point to a preset to open its commands:

Command	What it does
Apply	Loads the preset onto the selected output, returning each tile to its saved content.
Edit Name	Renames the preset.
Re-Learn	Replaces the preset with the output's current layout.
Delete	Deletes the preset after you confirm "Are you sure you want to delete multiviewer preset: <name>".

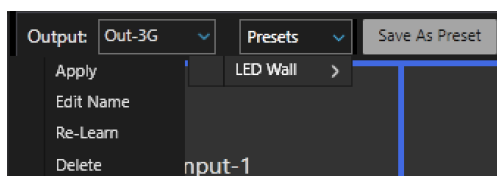


Figure 6-28. The Presets dropdown with one preset, LED Wall, and its commands.

6.8 PixelSpace properties: titling and test patterns

Each PixelSpace has a **Properties** panel with three groups: the PixelSpace's name, origin and size; **Titling**; and the **Test Pattern Generator**. **Titling is not supported in Spyder Studio 6.1.3** (Section 6.8.2).

Prerequisite: An applied configuration with at least one PixelSpace.

To open a PixelSpace's properties, do either of the following:

- Click an empty area of the PixelSpace on the **Program** or **Preview** canvas (an area not covered by a layer).
- Click the PixelSpace in **System Patch** or **System Layout**.

The **Properties** panel opens, headed **PixelSpace (ID: n)**.

PixelSpace (ID: 0)

Name
PS-1

Horizontal Start
0

Vertical Start
0

Width
7680

Height
2160

Titling

Enable Titling
☐

Location:
OutsideOfLayer

Position:
BottomRight

Size:
Medium

Background:
0, 0, 0

Foreground:
255, 255, 255

Test Pattern Generator

Pattern:
Color Bars

Type:
Color Bars, Grayscale Low

Options

☐ Outline
☐ Grid

☐ Center 0
☐ Center X

Background:
0, 0, 0

Foreground:
255, 255, 255

Clear
Apply

Figure 6-29. The PixelSpace Properties panel for PS-1 — the PixelSpace group, Titling, and the Test Pattern Generator, all at their defaults.

6.8.1 Name, origin and size

Field	What it sets
Name	The PixelSpace's name, shown on the canvas and in System Patch.
Horizontal Start	The X position of the PixelSpace's top-left corner.
Vertical Start	The Y position of the PixelSpace's top-left corner.
Width	The PixelSpace's width, in pixels.
Height	The PixelSpace's height, in pixels.

Type a value and press **Enter**, or use the arrows. The change applies as soon as it is committed — there is no Apply button. The numeric fields accept 1 to 65535; a larger value is set to 65535 without a warning.

Caution: Changing these fields changes a live PixelSpace. They are normally set when you configure the system (Section 6.3 or 6.4); change them here only when you mean to.

6.8.2 Titling

Notice: Titling is not supported in Spyder Studio 6.1.3. The Titling group appears in the PixelSpace Properties panel, but changing its settings has no effect. Leave the settings at their defaults.

The group holds these settings:

Setting	Options	Default
Enable Titling	On / off	Off
Location	OutsideOfLayer, InsideOfLayer	OutsideOfLayer
Position	TopLeft, TopCenter, TopRight, BottomLeft, BottomCenter, BottomRight	BottomRight
Size	Small, Medium, Large	Medium
Background	Any color	0, 0, 0 (black)
Foreground	Any color	255, 255, 255 (white)

6.8.3 Test Pattern Generator

The Test Pattern Generator puts a test pattern across the whole PixelSpace.

To put a test pattern on the PixelSpace:

1. Choose a **Pattern**, then a **Type**. The preview shows the result.
2. Tick any **Options**, and set the **Background** and **Foreground** colors.

3. Click **Apply**.
4. To remove the pattern, click **Clear**.

Pattern	Types
Color Bars	Color Bars, Grayscale Lower (default); Color Bars, Horizontal; Color Bars, Pluge Lower; Color Bars, SMPTE
Color Field	Single Color; 8-Bit Color Ramps; 8-Bit RGB Checkerboard
Gray	Gamma (ON/OFF - 50%); Gray 16 Steps Top to Bottom; Gray 16 Steps Bottom to Top; Gray 32 Steps Top to Bottom; Gray 32 Steps Bottom to Top; Gray 16 Steps Split; Gray 32 Steps Split
Grid	Grid (32x32); Grid (64x64); Grid (128x128)
Grill	Grill Horizontal; Grill Vertical
Sweep	Sweep Left to Right; Sweep Right to Left; Sweep Top to Bottom; Sweep Bottom to Top

The **Options** draw over the pattern. All four are off by default.

Option	Adds
Outline	A border around the edge of the PixelSpace.
Grid	A grid over the pattern.
Center O	An ellipse centred on the PixelSpace.
Center X	Diagonal lines from corner to corner.

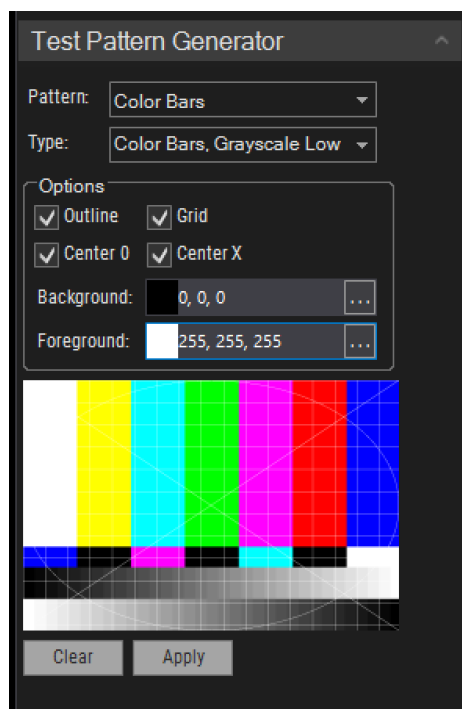


Figure 6-30. The Test Pattern Generator with Color Bars and all four Options ticked — the preview shows the outline, grid, ellipse and diagonals over the bars.

Note: Test patterns are not drawn in the Simulator.

Note: This generator covers the whole PixelSpace. Each output also has its own generator, which covers that output only — see Section 7.7.3.

Note: **Titling** and **Test Pattern Generator** appear only when the master Frame supports them. If they are missing, only the PixelSpace group is shown.

6.9 Managing EDIDs

Server ▶ EDID Manager manages the **EDIDs** the Spyder presents to its sources, so that each source sees the signal format you want. This is part of configuring the system — decide it alongside your PixelSpace and output setup, before going live. The view has two lists: **EDIDs** on the left and **Inputs** on the right.

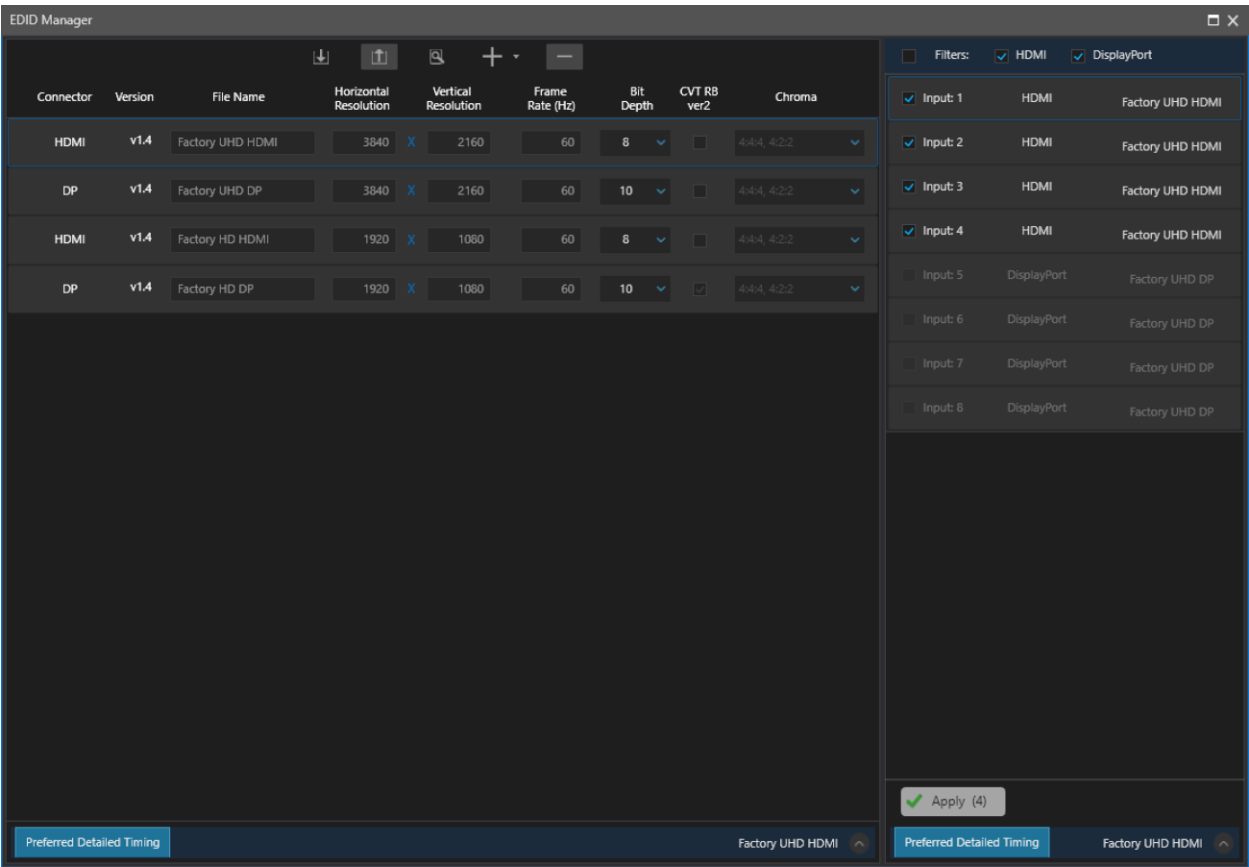


Figure 6-31. The EDID Manager. Factory UHD HDMI is selected on the left, so the four HDMI inputs that use it are checked on the right and the DisplayPort inputs are greyed out.

Each EDID occupies one row of the left-hand list, with these fields: **Connector**, **Version**, **File Name**, **Horizontal Resolution**, **Vertical Resolution**, **Frame Rate (Hz)**, **Bit Depth**, **CVT RB ver2**, and **Chroma**.

Factory EDIDs are read-only. The system ships with four of them — *Factory UHD HDMI*, *Factory UHD DP*, *Factory HD HDMI*, and *Factory HD DP*. Every field in a factory row is greyed out, and so are the **Export** and **Remove** buttons while a factory row is selected. Only EDIDs you create yourself can be edited, exported, or deleted, and you edit them **in place** in the row — there is no edit mode to switch on first.

The toolbar above the EDID list carries five actions, left to right:

Action	Effect
Import EDID From File Location	Load an EDID from a file.
Export EDID To File Location	Save the selected EDID to a file. Unavailable for factory EDIDs.
Read EDID from Display Device	Read the EDID reported by an attached display. A dialog asks for the Output and the Connector to read from.
Add Edid	Create a custom EDID. The button carries a dropdown offering Add new custom HDMI EDID and Add new custom DP EDID ; the new row appears at the end of the list, editable immediately. A new custom HDMI EDID is

Action	Effect
	named <i>HDMI 1</i> and starts at 3840 x 2160, 60 Hz, 8-bit, with 4:4:4 and 4:2:2 chroma.
Remove Edid	Delete the selected EDID, after a Confirm Remove EDID prompt. If the EDID is applied to any inputs, the prompt adds that those inputs <i>"will be reverted back to factory default"</i> . Factory EDIDs cannot be removed.

In a custom EDID, **Bit Depth** offers –, 6, 8, 10, 12, 14, 16 and RES, and **Chroma** is a list of two check boxes, **4:4:4** and **4:2:2**. If you change an EDID that is already applied to inputs, the status next to **Apply** reads *"Changed EDID needs to be reapplied."* — apply it again to update those inputs.

Note: EDIDs are part of the data pushed by **Input EDIDs** in Push Data To Server (Section 4.8.2).

6.9.1 Assigning an EDID to inputs

Each row of the **Inputs** list shows the input number, its connector, and the EDID currently applied to it. When you select an EDID on the left, the inputs that already use it are checked, and inputs with a different connector are greyed out (Figure 6-31). The **Filters** row above the list narrows it by connector type, and the check box to the left of **Filters** checks or clears every input in the list.

To assign an EDID to inputs:

1. Select the EDID in the left-hand list.
2. Check the inputs you want to use it.
3. Click **Apply**. Its label shows how many inputs are checked.
4. At the **Confirm?** prompt — *"You are about to apply the selected EDID to the selected input(s). Would you like to continue?"* — click **Yes**.

Expected result: *"Successfully Applied"* appears next to **Apply**, and the Inputs list shows the EDID against each input you checked.

Note: After you remove a custom EDID that was applied to inputs, the Inputs list keeps showing the removed EDID until you close and reopen the EDID Manager. It then shows the factory EDID.

6.9.2 Preferred Detailed Timing

Both lists have a **Preferred Detailed Timing** bar along the bottom, labelled with the name of the EDID it shows. Click the arrow at the right of the bar to expand it.

- Under the EDID list, it shows the timing of the selected EDID: **Active Pixels**, **Blank Pixels**, **Front Porch**, **Sync Width** and **Polarity**, horizontal and vertical, followed by **Total Pixels**, **Refresh Rate** and **Pixel Clock (MHz)**.
- Under the Inputs list, it shows the same values, read-only, for the EDID on the selected input.

To edit the timing of a custom EDID, check **Enable Advanced Properties**. Studio first shows an **Advanced Timing**

Warning: *"This will allow you to modify properties on the EDID that may not conform to VESA standards and can result in performance and quality degradation."* After you click **OK**, Active Pixels, Blank Pixels, Front Porch, Sync Width, Polarity and Pixel Clock can be edited. Total Pixels and Refresh Rate are calculated from them.

Preferred Detailed Timing		HDMI 1	
☒ Enable Advanced Properties	Horizontal		Vertical
Active Pixels	3840		2160
Blank Pixels	160		62
Front Porch	48		3
Sync Width	32		5
Polarity	Positive		Negative
Total Pixels	4000		2222
Refresh Rate	133.313 KHz		59.996 Hz
Pixel Clock (MHz)	533.25		

Figure 6-32. Preferred Detailed Timing for a custom EDID, with Enable Advanced Properties checked.

6.10 Reviewing the applied configuration

After a configuration is applied — by either wizard — you can review and adjust the resulting wiring without re-running the Configuration Manager. Two panes show it:

- **System Patch** — a 5-column patch grid (Input/Layer → Window Patch → Display/PixelSpace → Output Patch → Output) showing the full signal flow, with right-click actions per element.
- **System Layout** — a top-to-bottom signal-flow diagram (Frame Group → Outputs → PixelSpaces → Preview → Layer), hidden by default; enable it with **View ▶ System Layout**.

Click a PixelSpace or an output in either pane to open its properties in the **Properties** panel.

6.10.1 System Patch

System Patch reads from left to right:

- **Layers** — one row per layer, with Layer 16 at the top. A layer with content shows its source, its Mix Effect, a **red** dot while it is on Program and an **amber** dot while it is on Preview, and a small drawing of where it sits in the PixelSpace.
- **Patch lines** — red and yellow lines connect each layer with content to the Program and Preview PixelSpace.
- **PixelSpaces** — each PixelSpace with a red bar, and its preview (for example *PS-1 - PVW*) with an amber bar.
- **Outputs** — one row per output: its number and name (for example *13 - Out-4E*), its resolution and refresh rate, and either **Unused** or a small drawing of the PixelSpace in red with the part that output shows outlined in blue. Grey lines connect each PixelSpace to the outputs it drives.



Figure 6-33. System Patch — layers on the left, the PixelSpace and its preview in the middle, outputs on the right.

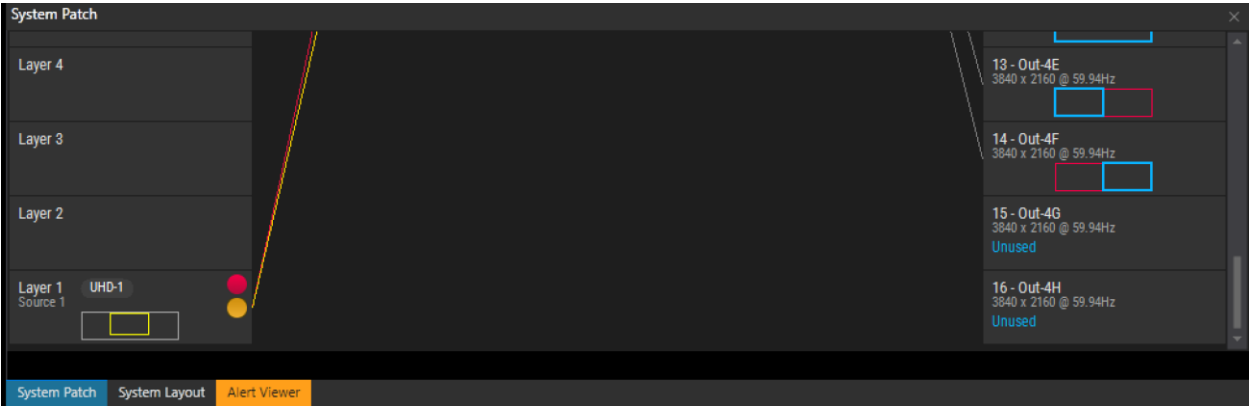


Figure 6-34. System Patch scrolled to Layer 1, which is on Program and Preview. PS-1 drives outputs 13 - Out-4E and 14 - Out-4F; each output drawing outlines its half of the PixelSpace in blue.

Right-click an element to act on it:

Right-click	Menu
A PixelSpace	Add, Delete, Clear TestPattern (removes a test-pattern still from the PixelSpace, Section 7.5.7)
A layer	Always Relative, Lock

6.10.2 System Layout

System Layout shows the same wiring as a grid, read from top to bottom, with one column per output:

Row	Shows
Frame Group	The frame group the outputs belong to.
Outputs	Each output's number and name. An output that is not in use shows Unused .
PixelSpaces	The Program PixelSpace that drives the outputs above it.

Row	Shows
Preview	That PixelSpace's preview.
Layer	The layers, two per column, with Layers 15 and 16 at the top. A layer with content shows its source and the same red and amber dots as System Patch.

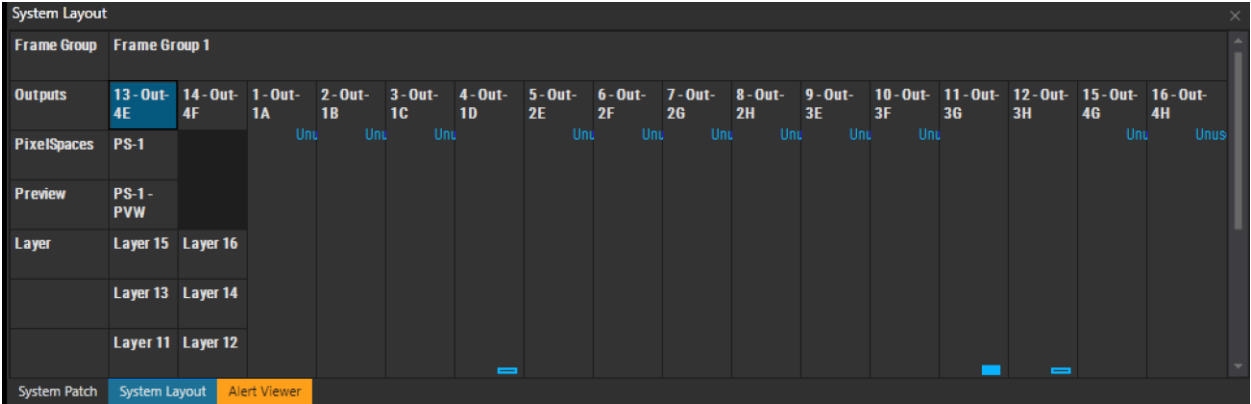


Figure 6-35. System Layout — PS-1 and its preview drive outputs 13 - Out-4E and 14 - Out-4F; the other outputs are unused.

Right-click an element to act on it:

Right-click	Menu
A PixelSpace	Delete PixelSpace
A layer	Always Relative, Lock

Live operation on the applied configuration is covered in Chapter 7; ongoing maintenance is in Chapter 11, and capacity math with worked examples is in the appendices (Chapter 12).

7. Operation

Configuration (Chapter 6) establishes the system's structure; Operation is the work done inside it — whether that is running a live show or driving a permanently installed system day to day. This chapter covers those tasks: putting sources on screen, shaping layers, transitioning content with Mix Effects, managing backgrounds, and managing PixelSpaces and outputs. It assumes the processing model from Chapter 5 — Mix Effect as "Single Layer Mixer," Background as "Layer Zero," and the layer/input/bandwidth budgets — and references it rather than repeating it.

7.1 Managing sources

A **source** is a named, placeable input signal — the building block you put on a layer to get content on screen. Sources live in the **Sources** pane, the left-hand dockable panel that shares a tab region with two siblings: **Sources** • **Backgrounds** • **Treatments** (select via the tabs at the bottom).

When Studio is connected to a Spyder Frame, the Frame's configured sources sync into this list automatically. Inputs defined on the Frame appear without manual entry, so the pane mirrors the system's source data. See Section 5.6 for the source/input concept; creating a source by hand is covered in Section 7.1.2, and a source's input signal settings are edited in its layer's **Input Configuration** properties (Section 7.1.5).

7.1.1 Read the Sources pane

The pane is a numbered grid of source cells. Each populated cell shows three things, plus an on-air indicator when the source is in use:

- **Source name** — a user-definable name, for example *Cam 1* or *PC 1*. Name sources for what they are; the names appear on layers, in Mix Effects, and in external-control commands.
- **Input label** beneath the name — for example *In-3A*. Read it as **In-<Frame><input>**: In-3A is **input A on Frame 3**, In-2D is input D on Frame 2. The digit is the Frame number as it appears in the Configuration Manager, and the letter is the physical input connector on that Frame's rear panel (Section 3.3.2) — so the label traces a source straight back to a cable in the rack.
- **Thumbnail** — a small identifying graphic, part of the source's definition (see the Note below). It is there to help you pick the right source at a glance; it is **not** a live preview of the incoming signal.
- **Tally dots**, top-right of the cell — a **red** dot while the source is on **Program** and an **amber** dot while it is on **Preview**. A source on both carries both dots; a source that is on neither carries none.



Figure 7-1. The Sources pane — named source cells with input labels and identifying thumbnails. PC 1 and Cam 1 carry red and amber tally dots.

Empty numbered cells are placeholders for creating new sources (Section 7.1.2). Each source references an **Input Configuration** (the signal format and processing for its input, edited from the layer properties once the source is on a layer — Section 7.1.5) and carries the **Source properties** shown in the Properties panel.

Note: Selecting a source in the Sources pane shows only the **Source** section of the Properties panel — its definition (Name, Thumbnail, Router, Input, Config ID, and the Preferences / Backup tabs). The full layer property stack (Input Configuration, Key Frame, Border / Shadow, Advanced) appears only once the source is placed on a layer, because those are *layer* properties. Layer properties are covered in Section 7.2.

The Source section's fields:

Field	What it does
Name	The source's name. Rename the source here.
Thumbnail / Browse...	The identifying image shown in the source cell. Browse... opens a file dialog listing image files.
Router	The router that feeds the source — Spyder Internal (Frame <n>) for a Frame's internal router (the number is zero-based; see Section 7.1.2) or an external router. The list also offers <Direct to Layer> . Add opens the Add New Router dialog (Section 7.8.3); Edit shows the selected router's properties in the Properties panel.
Input	The router input the source is fed from.
Config ID	Read-only.
Preferences ▶ Preferred Treatment	<None> or one of the saved treatments (Section 7.3).
Preferences ▶ Preferred Layer	<None> or Layer 1 ... Layer n . When you drop the source on an empty part of the canvas, it goes to this layer if the layer is free, instead of the next free layer.
Backup ▶ Type	The source's backup (Section 7.1.4).

7.1.2 Select a source — or create one

The pane supports three direct gestures:

Action	Result
Single-click a source	Selects it; its properties appear in the Properties panel.
Single-click an empty cell	Starts a New Source in that slot (below).
Right-click a source	Opens the context menu (Section 7.1.4).

To create a source:

Prerequisite: A configuration is applied, so the internal router and its inputs exist.

1. **Single-click an empty numbered cell.** The Properties panel switches to the **New Source** editor, with three sections: **General** (the source **Name** — default *Source N* — and its **Input Type**), **Router** (the routing that feeds it), and **Add Source(s)**.
2. Type a **Name** that says what the signal is — the name appears on layers, in Mix Effects, and in external-control commands.
3. Under **Router**, pick the router and set the **Input** number the source is fed from. Each Frame has its own internal router, listed as **Spyder Internal (Frame <n>)**; **Add / Edit** define an external router (Section 7.8). **Input Type** follows the selected input's connector; it is not set independently while creating.

⚠ **The number in the router name is zero-based**, so it is one lower than the Frame number you see everywhere else. **Spyder Internal (Frame 0)** is the router for **Frame 1**, (*Frame 1*) is Frame 2, and so on — a source on **In-2A** reports its router as *Spyder Internal (Frame 1)*. Match the router to the input label, not to the digit in the name.

4. Click **Create Source Now**.

Result: The source appears in the clicked cell, showing its name and the patched input's connector label (for example *In-1A*). The editor stays open with the **Input** number advanced by one, so a run of sources on consecutive inputs can be created back-to-back — click **Create Source Now** for each. Click **Back** (top of the Properties panel) to leave without creating, or select anything else when done.

Tip: To remove a source created by mistake, right-click it and choose **Delete** (Section 7.1.4).

7.1.3 Drag a source onto the canvas — Mix Effect vs. layer

Dragging a source from the pane onto the **Simulator canvas** is the primary way to put it on screen. The modifier key chooses the path, and the choice has a direct resource consequence (Section 5.4, 5.6):

- **Regular drag** → **through a Mix Effect**. The source is applied to an available Normal Mix Effect, which is then shown on a layer. This is the default, because the system's layer mode is normally *MixEffects*.
- **Ctrl + drag** → **directly as a layer**. Holding Ctrl forces *Layers* mode, so the source is shown straight on the layer with no Mix Effect in between.

The drop target determines what happens:

Drop target	Result
Empty canvas / a PixelSpace	Adds the source on the next available layer (via ME or direct, per the modifier) — or on the source's Preferred Layer if that layer is free (Section 7.1.1).
An existing layer	Replaces that layer's content. If the layer has a Mix Effect, the source goes to that ME's Top (Program) bus. If the layer has no Mix Effect, the source goes straight onto the layer — but only when the source's input is on the same Frame as that layer; each Frame has its own layers. Otherwise the drop is refused — no message appears and nothing is logged in the Alert Viewer.

Dropping in System Patch or System Layout. You can also drop a source on a row in **System Patch** or a cell in **System Layout**: a PixelSpace (for example **PS-1**) puts it on that PixelSpace's Program, a Preview (for example **PS-1 - PVW**) on its Preview, and a layer follows the **An existing layer** rule above.

Note: A regular drag costs a layer *and* an ME; a Ctrl-drag costs only a layer. Choose the path against your layer and ME budgets (Section 5.9). If you don't need a transition on this content, a direct layer is the cheaper placement.

7.1.4 Source context menu and backup

Right-clicking a source opens its context menu:

Item	Action	Enabled when
Add new Source	Create a new source in the clicked slot	Always
Copy Source	Duplicate the source to a new slot	A source is clicked

Item	Action	Enabled when
Insert	Insert a slot	Always
Delete	Remove the source	Always
Use Backup	Switch the source to its configured backup	Source has a backup (Type ≠ None) and is not already using it
Exit Backup	Switch back from the backup to the primary	Source is currently using its backup

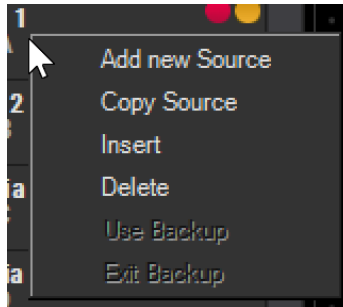


Figure 7-2. The source context menu — Add new Source, Copy Source, Insert, Delete, Use/Exit Backup.

A **backup source** is a fallback that a source automatically falls over to when its primary signal is lost — redundancy content that stands in for it. It is configured on the Source properties **Backup** tab, which carries a **Type** dropdown and, depending on the type, a picker for the fallback content:

- **Type = Source** — a **Source:** picker appears; the chosen source becomes the backup.
- **Type = None** — no backup; only the dropdown shows, and **Use Backup** / **Exit Backup** stay greyed out.

Note: The **Use Backup** and **Exit Backup** items are enabled only when a backup is actually configured. With **Type = None** they are greyed.

Note: The current release does not support stills on layers, so a source cannot fail over to a backup still — the Type dropdown offers only **None** and **Source**.

Failover behavior. Whether **Use Backup** switches a live source immediately or eases the change in is controlled by the **ImmediateSwitchBackupSource** option (Tools ▸ Options ▸ **Advanced**; Section 4.9). **Exit Backup** returns the source to its primary.

7.1.5 Input Configuration

The **Input Configuration** section of the layer properties holds the signal settings for the layer's **input**. Select a layer, then expand **Input Configuration** in the Properties panel. The formats each connector accepts are listed in Appendix E.

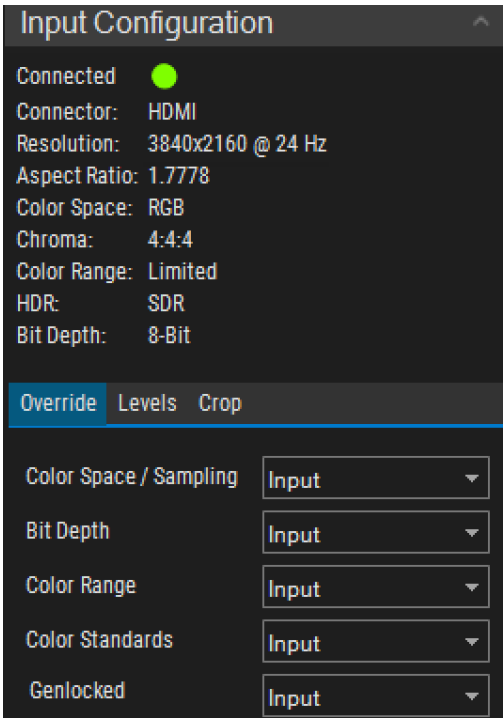


Figure 7-3. The Input Configuration section — the signal read-out and the Override, Levels, and Crop tabs.

The top of the section reads back the incoming signal: a **Connected** indicator, the **Connector** type, and the detected **Resolution**, **Aspect Ratio**, **Color Space**, **Chroma**, **Color Range**, **HDR**, and **Bit Depth**. With no signal present, the read-out fields are blank.

Below the read-out, three tabs adjust the input:

Tab	Controls
Override	One dropdown per signal attribute — Color Space / Sampling , Bit Depth , Color Range , Color Standards , and Genlocked . Each defaults to Input , meaning the value detected from the signal is used; select a value to force it when a source misreports or needs pinning. The choices are: Color Space / Sampling — RGB 4:4:4, YCbCr 4:4:4, YCbCr 4:2:2; Bit Depth — 8-Bit, 10-Bit; Color Range — Full, Limited; Color Standards — BT.709, BT.2020; Genlocked — Genlocked, Asynchronous.
Levels	Picture-level adjustment for the input — Gamma (default 1.00), as a value box with a slider. Defaults restores the tab.
Crop	Trims the incoming picture from the Left and Right only — each a value with a slider. Top and Bottom are shown greyed out: input crop does not support them. To crop all four edges, use the layer's Crop attribute (Section 7.2.4). Reset All clears the crop. A crop set here stays with the layer you set it on: another layer showing the same source keeps its own crop values.

Note: These are the per-input signal attributes introduced in Section 5.7. The remaining collapsible section, **Advanced**, exposes low-level input and source properties as a raw property grid — **Gamma**,

UseAlternateInputSync, and **SourceName**. A name typed into **SourceName** does not rename the source; rename it in the **Source** section's **Name** field.

7.1.6 Sources from routers

A source can also come through an external router in front of the Frame's inputs. Adding a router and patching or switching its crosspoints are covered in **Section 7.8**.

7.2 Working with layers

A **layer** is one piece of content on a PixelSpace — a source or a Mix Effect — controlled by keyframes (Section 5.3). You arrange layers in the **Simulator**, the central canvas that shows where each layer sits on Program and Preview. The Simulator shows two stacked views — **Program** (top, on-air) and **Preview** (bottom, staging) — each rendering its PixelSpace(s) with live layers, backgrounds, and a per-view **Packets Processed** counter.

The bottom of the Simulator carries two rows of tabs. The upper row is the **PixelSpace selector**: an **All** tab (every PixelSpace together) followed by one tab per PixelSpace, named for that PixelSpace (for example **PS-1**), each appearing automatically as PixelSpaces are added. The lower row holds the **view tabs** — **Simulator** · **OpMon Configuration** · **Tile Configuration** — which share this pane region. The **View Stack** dropdown at the top mirrors the PixelSpace selector.

7.2.1 Add content to a PixelSpace

Add content by dragging from the Sources, Treatments, or Backgrounds panes onto the canvas:

- **A source** — a regular drag goes **through a Mix Effect**; a **Ctrl-drag** goes **directly as a layer** (Section 7.1.3).
- **A treatment** — applies to the layer under the drop.
- **A background** — applies to the PixelSpace (Section 7.5).

You can also right-click empty PixelSpace and choose **Add Layer** ► to pick from a submenu of layers available to add. The layer comes back with the content it last had, or empty. To put a source on an empty layer, drag the source onto it; the drop rule for an existing layer is in Section 7.1.3.

7.2.2 Size and position a layer

Select a layer to expose its geometry in the Properties panel **Key Frame** section. A **keyframe** holds the layer's size, position, and attribute state at a point in time (Section 5.3).

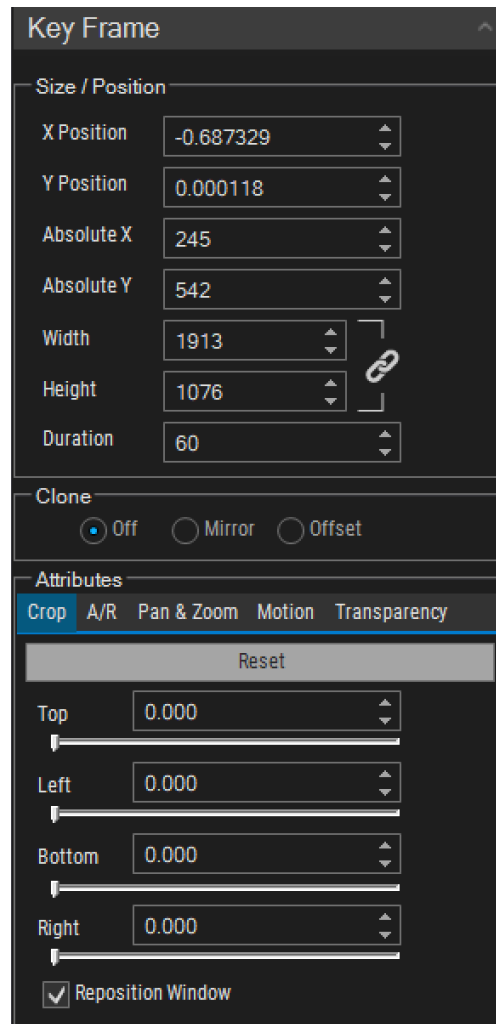


Figure 7-4. The Key Frame properties section — normalized/absolute position, size, duration, and Clone.

Position is stored **normalized** (–1.0 to +1.0) and resolved to pixels against the PixelSpace. The Key Frame section shows both at once:

- **X Position / Y Position** — normalized coordinates.
- **Absolute X / Absolute Y** — the same position in pixels (for example, a normalized X of –0.606775 resolves to an Absolute X of 262).
- **Width / Height** — with an aspect-lock link between them.
- **Duration** — the keyframe's transition time.

Note: Normalized position is why the same keyframe behaves consistently across PixelSpaces of different sizes (Section 5.3). Type into the normalized fields for portable placement, or into the Absolute fields when you need an exact pixel.

Note: A source dropped on the canvas defaults to a set percentage of the PixelSpace height (**80%** by default), aspect ratio preserved. Change it at **Tools > Options > Simulator > Initial Layer Height %**.

Super-scale — how wide a layer can go. How far a layer can be scaled depends on **which physical input feeds it**. Drag a source from **input port 1 or 2 of any Frame** onto the canvas and the resulting layer is **super-scaled**: it can be stretched to a width of **32,768 px** — the full width of a Frame group's maximum canvas (Section 3.4). Layers fed from input ports 3 and 4 do not scale that far.

This is a property of the input, not of the layer, so it is decided at patch time: if a source has to stretch across the whole canvas — a full-width backplate, a wide scenic wash — **patch it to input 1 or 2** when you cable the Frame. Re-patching later means rebuilding the layer.

You can also manipulate a layer directly on the canvas:

Gesture	Action
Click a layer	Select it.
Ctrl-click	Add/remove a layer from the multi-selection.
Ctrl + digit	Toggle that layer in or out of the selection.
Drag a layer	Move it.
Arrow keys	Nudge the selected layer (Shift = large step).
+ / -	Resize the selection larger / smaller (Shift = large step).
Drag corner handles	Resize the layer.

7.2.3 Clone, mirror, and offset a layer

The Key Frame section's **Clone** group replicates a layer's content. It has three option buttons:

- **Off** — no cloning.
- **Mirror** — adds a copy at the mirror-image position on the other side of the PixelSpace's vertical center line (Figures 7-5 and 7-6). A layer placed at the center has its copy directly on top of it.
- **Offset** — up to **four** offset copies. Selecting **Offset** shows the rows **1:** to **4:**, each a value with a slider (Figure 7-7). Each value moves one copy to the right of the layer, in the same units as **X Position: 1.000** is half the PixelSpace width. A copy whose value is **0.000** is not drawn.

The Simulator draws each copy with a **green** title bar (Figure 7-8).

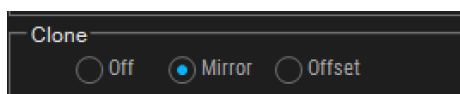


Figure 7-5. The Clone group with Mirror selected.



Figure 7-6. Mirror — the layer (left) and its mirrored copy (right, green title bar) on either side of the PixelSpace's center line.

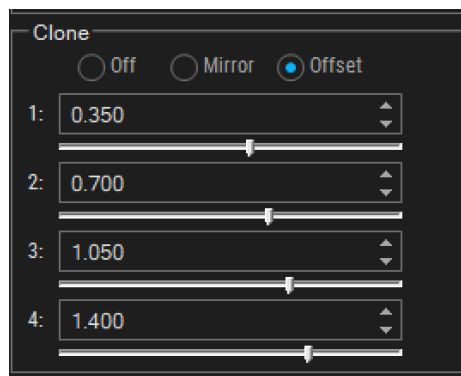


Figure 7-7. The Clone group with Offset selected — one value and slider per copy.

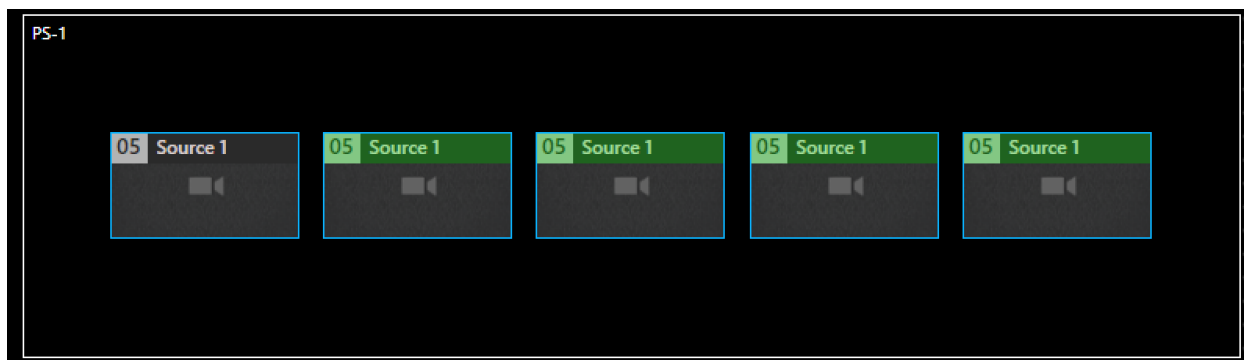


Figure 7-8. Offset with the values in Figure 7-7 — the layer (left) and its four copies.

Prerequisite: Clone acts on a layer that already carries content — drag an input source onto the canvas to create the layer first (Section 7.2.1). A clone duplicates that layer's content rather than adding a new feed, so the cloned copies cost no extra inputs, but the layer being cloned still consumes the input that feeds it.

7.2.4 Apply layer attributes — crop, aspect ratio, pan & zoom, motion, transparency

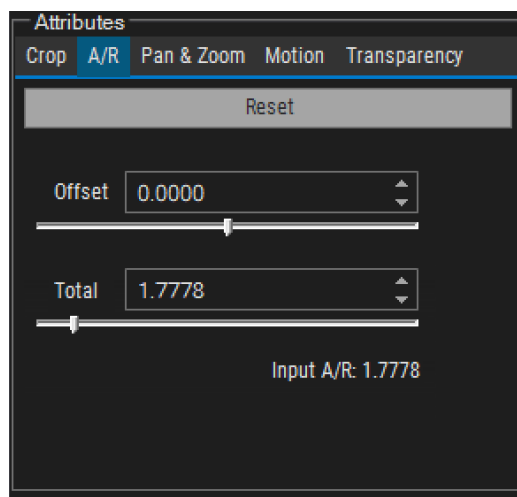


Figure 7-9. The layer attribute tabs with A/R selected — Crop, A/R, Pan & Zoom, Motion, and Transparency.

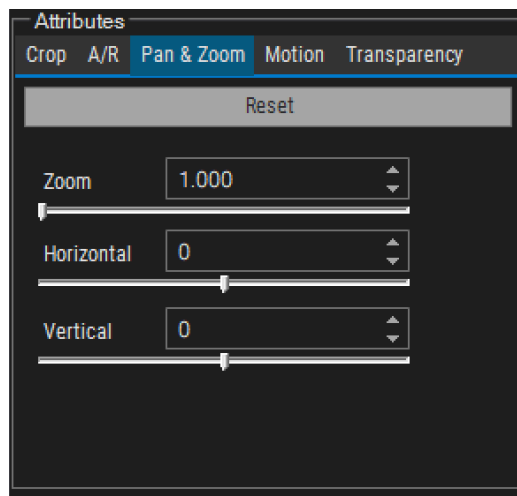


Figure 7-10. Pan & Zoom.

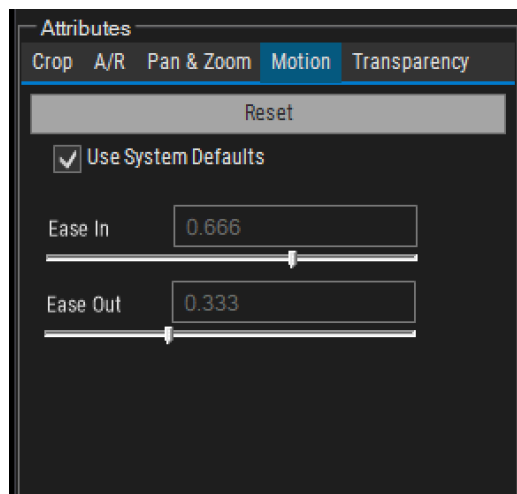


Figure 7-11. Motion (Ease In / Ease Out).

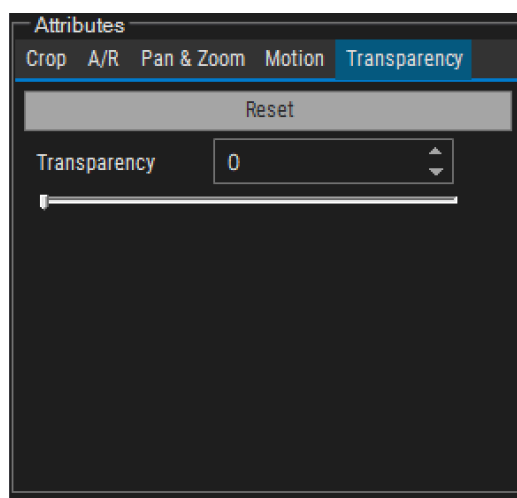


Figure 7-12. Transparency.

The Key Frame section groups the per-keyframe attributes into tabs:

Tab	Controls	Use
Crop	Top, Left, Bottom, Right, Reposition Window , Reset	Trim the layer's edges. Each value is a fraction of the picture from 0 to 1 — 0.250 on Top trims a quarter of the height. With Reposition Window on, a Top crop shrinks the layer's window equally from its top and bottom edges.
A/R	Offset, Total, <i>Input A/R</i> read-out, Reset	Correct or set aspect ratio.
Pan & Zoom	Zoom, Horizontal, Vertical, Reset	Zoom into and pan within the content.
Motion	Use System Defaults (checkbox), Ease In, Ease Out, Reset	Shape the transition's acceleration.
Transparency	Transparency, Reset	Set the layer's opacity.

Note: Attribute defaults — Crop 0 with **Reposition Window** on; A/R Offset 0 with **Total** initialized to the input's aspect ratio (the *Input A/R* read-out shows it, e.g. 1.7778 for 16:9); Pan & Zoom **Zoom 1.0**; Motion **Ease In 0.666 / Ease Out 0.333** (the system defaults); **Transparency 0 = fully opaque** (higher = more transparent).

Note: On the **Motion** tab, the Ease In / Ease Out sliders are greyed out while **Use System Defaults** is checked. Uncheck it to set custom easing.

Border and drop-shadow styling live in a separate **Border / Shadow** section of the Properties panel (Border Settings and Shadow Settings, the latter with its own **Enable** checkbox).

7.2.5 Align layers with the Simulator toolbar

The Simulator toolbar holds **28 layer-arrangement buttons**. Each one acts on the layers currently selected on the Program or Preview canvas; hover a button to see its name. To select several layers, Ctrl-click each one.



Figure 7-13. The Simulator alignment toolbar.

Some buttons work on a single layer; the others need two or more selected layers, and do nothing with only one. Where a button lines layers up with one another, the **first layer you select** is the reference: it stays where it is and the others move to match it.

Swap

Icon	Button	What it does
	Swap Windows	Swaps the position and size of two layers. Select exactly two layers.

Align — select two or more layers.

Icon	Button	What it does
	Align Left	Lines up the left edges of the selected layers with the first layer selected.
	Align Center	Lines up the horizontal centers with the first layer selected.
	Align Right	Lines up the right edges with the first layer selected.
	Align Top	Lines up the top edges with the first layer selected.
	Align Middle	Lines up the vertical centers with the first layer selected.
	Align Bottom	Lines up the bottom edges with the first layer selected.

Snap — works on one layer or several.

Icon	Button	What it does
	Snap Right	Moves each selected layer against the right edge of its PixelSpace.
	Snap Left	Moves each selected layer against the left edge of its PixelSpace.
	Snap Top	Moves each selected layer against the top edge of its PixelSpace.
	Snap Bottom	Moves each selected layer against the bottom edge of its PixelSpace.

Size

Icon	Button	What it does
	Size To Display Width	Resizes each selected layer to the width of its PixelSpace (up to the layer's maximum size) and centers it. Works on one layer.
	Size To Display Height	Resizes each selected layer to the height of its PixelSpace and centers it vertically. Works on one layer.
	Make Same Width	Gives the selected layers the width of the first layer selected. Select two or more.

Icon	Button	What it does
	Make Same Height	Gives the selected layers the height of the first layer selected. Select two or more.

Horizontal spacing — select two or more layers.

Icon	Button	What it does
	Make Horizontal Equal	Spaces the selected layers evenly left to right.
	Horizontal Increment	Widens the horizontal gaps between the selected layers by one step.
	Horizontal Decrement	Narrows the horizontal gaps by one step.
	Remove Horizontal Spacing	Closes the horizontal gaps so the layers sit edge to edge beside the first layer selected.

Vertical spacing — select two or more layers.

Icon	Button	What it does
	Make Vertically Equal	Places the selected layers at equal vertical steps, starting from the first layer selected.
	Vertical Increment	Widens the vertical gaps by one step.
	Vertical Decrement	Narrows the vertical gaps by one step.
	Remove Vertical Spacing	Closes the vertical gaps so the layers sit edge to edge below the first layer selected.

Stack and Center

Icon	Button	What it does
	Stack Horizontal	Makes the selected layers the size of the first layer selected and places them side by side in a row. Select two or more.
	Stack Vertical	Makes the selected layers the size of the first layer selected and stacks them in a column. Select two or more.

Icon	Button	What it does
	Center Horizontal	Centers the selection left to right in its PixelSpace, keeping the layers' spacing. Works on one layer.
	Center Vertical	Centers the selection top to bottom in its PixelSpace, keeping the layers' spacing. Works on one layer.
	Stack Grid	Opens a grid picker (Figure 7-14). Point to the number of columns and rows you want and click: the selected layers are arranged in that grid, starting from the first layer selected and filling each row left to right. Select two or more.

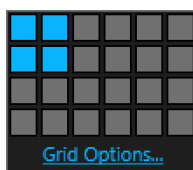


Figure 7-14. The Stack Grid picker, with a 2 × 2 grid highlighted.

Grid Options..., at the bottom of the picker, opens the **Grid Alignment Options** window (Figure 7-15). Its settings apply to the next Stack Grid:

Option	Values	Effect
Vertical Size (A/R Offset)	None (default), SmallestAR , LargestAR , FitToPixelSpace	How the layers' aspect-ratio offset is set when they are arranged.
Horizontal Size	FirstLayer (default), SmallestLayer , LargestLayer , FitToPixelSpace	The width every arranged layer is given. With FirstLayer each layer takes the width of the first layer selected; with SmallestLayer , the width of the smallest selected layer.

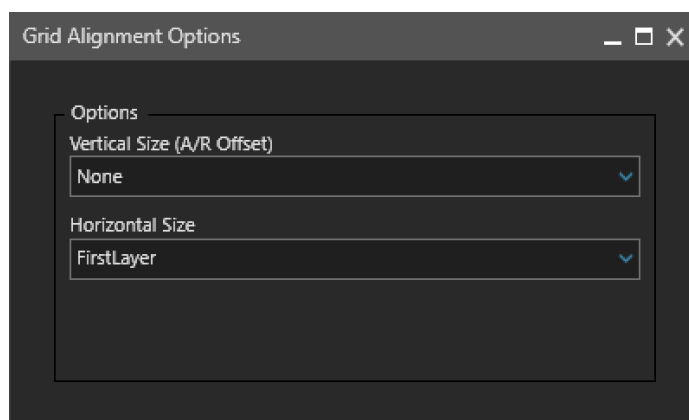


Figure 7-15. The Grid Alignment Options window.

7.2.6 Layer context menu and keyboard shortcuts

Right-clicking in the canvas opens a context menu, and **which menu you get depends on what you click**. Right-clicking also selects: empty space selects the PixelSpace, a layer selects that layer.

Right-clicking empty canvas — the PixelSpace menu. This is the shorter of the two, and it is where the background controls live:

Item	Action
Add Layer ▶	Submenu listing Layer 1 ... Layer n — adds an empty layer without dragging a source.
Remove All Layers	Clear the layers in this view.
Next Background	Step to the next background source.
Current Background	Re-apply the current background source.
Mix Background	Mix between the PixelSpace's background groups — this is the operator-side of the two-group rule in Section 6.4.1.3.
Toggle Background Freeze	Freeze or unfreeze the background.

⚠ **Remove All Layers clears only the view you right-clicked in.** Program and Preview are separate buses of the same PixelSpace, so clearing Program leaves Preview's layers in place. Repeat the command in the other view to clear both.

Right-clicking a layer — the layer menu.

Item	Action
Add Layer ▶	Submenu of available layers to add.
Remove Selected (R)	Remove the selected layer(s).
Remove All Layers	Clear the layers in this view (see the note above).
Transition (T)	Transition the selected layer(s).
Colocate (C)	Colocate the selected layers.
Copy KeyFrame (Ctrl+C)	Copy the layer's keyframe.
Paste KeyFrame (Ctrl+V)	Paste a keyframe (enabled when one is copied).
Move to PixelSpace ▶	Submenu — move the layer to another PixelSpace.
Treatments ▶	Submenu — Clear Treatment, Save As Treatment , then the list of saved treatments to apply to the layer.

The same actions, plus selection and edit commands, are bound to keyboard shortcuts on the canvas:

Key	Action
Ctrl + 0–9	Toggle that layer in the selection.
T	Transition selected layers.
R / Delete	Remove selected layers.
C	Colocate selected layers.
Ctrl + C / Ctrl + V	Copy / Paste KeyFrame.
Ctrl + A / Ctrl + D	Select all / Deselect all layers.
Ctrl + Z / Ctrl + Y	Undo / Redo.
Arrow keys	Nudge selected layers (Shift = large step).
+ / –	Resize selected layers larger / smaller (Shift = large step).

Note: Two Simulator controls affect how scripts and Command Keys reach air. **Live Updates Enabled** is a toggle whose label shows its state (**Live Updates Disabled**, with a padlock, when off) — but switching it off does **not** keep a Command Key recall off Program; to stage a look safely use **Preview Only Scripting Mode** (Section 8.2.5).

Disconnect Layers releases the layers from script / Command-Key control, stopping all running scripts and clearing each layer's script association so the layers become freely controllable again. Command Keys are covered in Chapter 8 and scripting in Chapter 9.

7.2.7 Borders and shadows

A border outlines a layer; a drop shadow lifts it off the background. Both are per-layer keyframe attributes, edited in the **Border / Shadow** section of the layer Properties panel. The section is collapsed by default — select a layer, then click the **Border / Shadow** header chevron to expand it. (For the rest of the layer Properties panel, see Section 7.2.4; for treatments that store a border/shadow look, see Section 7.3.)

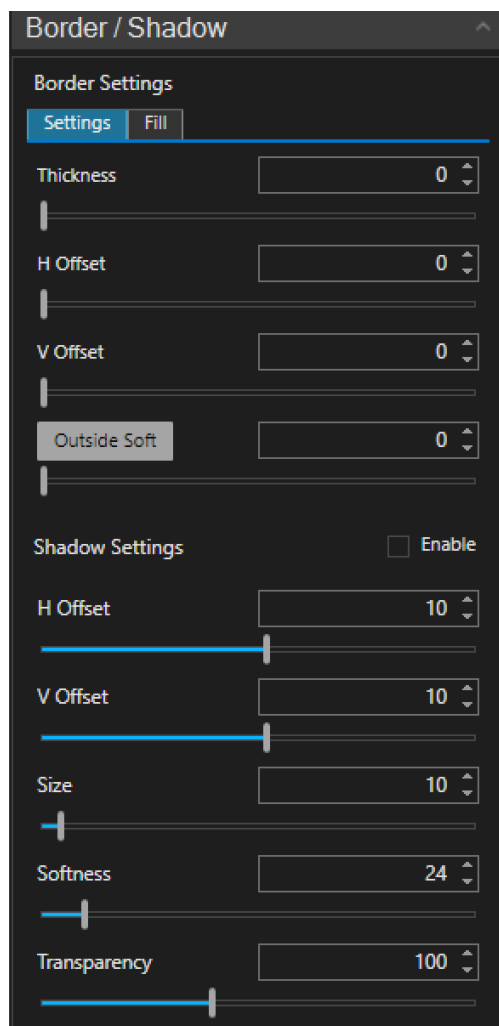


Figure 7-16. The Border / Shadow properties section — Border Settings and Shadow Settings.

The section has two sub-groups: **Border Settings** and **Shadow Settings**.

Every number in the section can be set three ways: type it in the box, click the box's ▲ / ▼ arrows (steps of 1), or drag the slider under the box. A typed value outside a field's range is changed to the nearest limit.

The Simulator draws the border's thickness and color and the drop shadow, so you can check the look on the canvas (Figure 7-18). It does not draw the border's **H Offset** or **V Offset**, and while **Outside Soft** is above 0 it draws neither the border nor the shadow for that layer.

7.2.7.1 Border Settings

Border Settings is a two-tab control: **Settings** and **Fill**.

The **Settings** tab sizes and places the border:

Control	Type	Range	What it does
Thickness	Slider and number box	0–100	Border width. On the canvas the border is drawn just inside the layer's edge.

Control	Type	Range	What it does
H Offset	Slider and number box	0–100	Horizontal offset of the border. Not shown in the Simulator.
V Offset	Slider and number box	0–100	Vertical offset of the border. Not shown in the Simulator.
Outside Soft	Button, slider and number box	0–100	Softness of the border's outer edge. Click the Outside Soft button to choose which edges are softened: Top , Left , Right and Bottom , all ticked by default.

The **Fill** tab sets the border color. A new layer's border is black (R, G and B all 0).

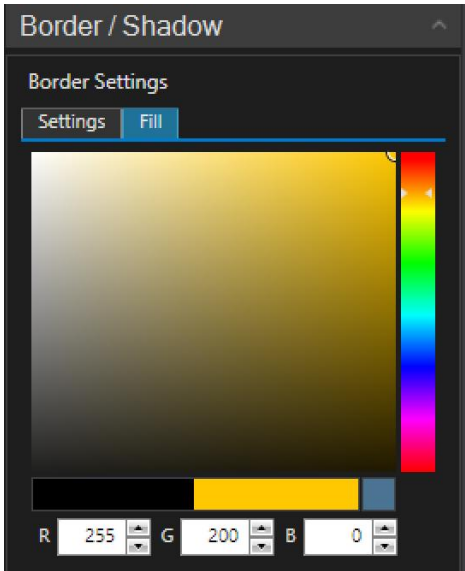


Figure 7-17. The Border Settings Fill tab.

Control	Type	What it does
Color square	Color picker	Click in the square to set the color's saturation and brightness.
Hue strip	Vertical slider (right of the square)	Click in the strip to set the hue.
Initial color	Left half of the color bar	Shows the color the editor started with. Click it to go back to that color.
Selected color	Right half of the color bar	Shows the current border color.
Previous color	Small square right of the color bar	Shows a color chosen earlier. Clicking it does not apply it.

Control	Type	What it does
R / G / B	Number boxes with ▲ / ▼ arrows	Exact red, green and blue values, 0–255. The square and hue strip move to match.

Hover the color bar or the small square to see its name.

7.2.7.2 Shadow Settings

Shadow Settings adds a drop shadow behind the layer:

Control	Type	Range	What it does
Enable	Checkbox	—	Turns the shadow on. Off by default. The other shadow values can be changed while it is off.
H Offset	Slider and number box	–255 to 255	Moves the shadow sideways; negative values move it left.
V Offset	Slider and number box	–255 to 255	Moves the shadow up or down; negative values move it up.
Size	Slider and number box	0–255	Enlarges the shadow; at high values it spreads past the layer's edges.
Softness	Slider and number box	0–255	Blurs the shadow's edge; 0 gives a hard edge.
Transparency	Slider and number box	0–255	0 gives a solid shadow; higher values let more of the picture behind show through.

The shadow is black; the panel has no shadow-color setting.

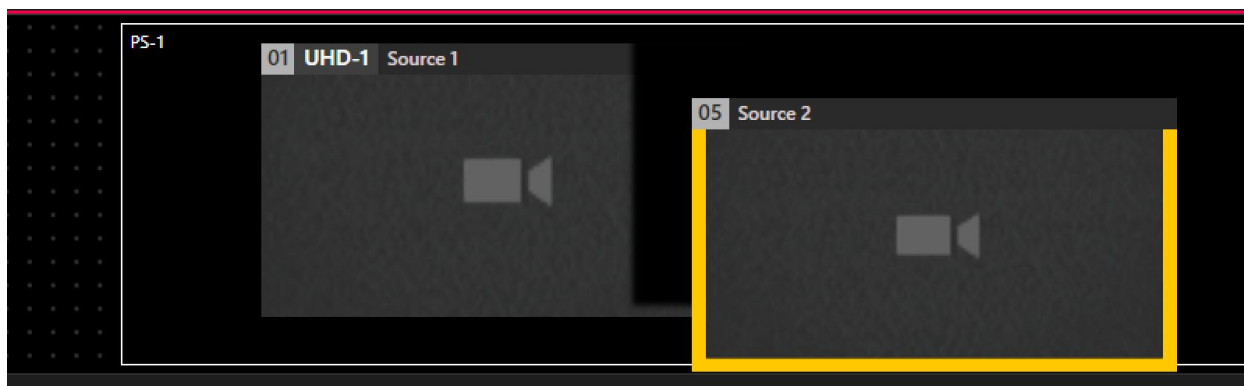


Figure 7-18. Layer 05 with a yellow border (Thickness 60) and a drop shadow (H and V Offset –255, Softness 40) falling across layer 01 on the Program canvas.

Note: Shadow defaults when enabled — H Offset 10, V Offset 10, Size 10, Softness 24, Transparency 100.

Tip: Border and shadow are part of a layer's keyframe, so they animate between keyframes like size and position, and a treatment can carry them as a reusable look (Section 7.3).

7.3 Using treatments

A **treatment** is a reusable layer "look" — a saved keyframe preset that you apply to one or more layers. Its power is that it is **selective**: a treatment can carry any subset of a layer's attribute categories, so applying it changes only the categories it holds. You can build a treatment that sets a border and shadow without disturbing a layer's size or position, for example.



Figure 7-19. The Treatments pane — saved layer "looks," tagged by the attribute categories they carry.

Treatments are managed in the **Treatments** pane, the right-hand tab of the left dock (**Sources • Backgrounds • Treatments**).

7.3.1 Attribute categories

Each treatment can selectively include any of these categories. The pane tags each treatment cell with the abbreviations for the categories it carries:

Category	Tag
Size	SZ
Position	PS
Clone	CL
Crop	CR
Border	BO
Shadow	SH
AR Offset	AR
Pan / Zoom	ZM
Duration	DN
Transparency	TP

A treatment cell shows the treatment **name** plus a parenthesized summary of its active categories — for example (SZ, PS, BO), listed in the order of the table above. The summary reads **(None)** if the treatment carries no attributes. The default treatment is **prefixed (DEFAULT)**, which sits alongside the category summary rather than replacing it — a default treatment carrying nothing reads (DEFAULT) (None).

Tally dots. A treatment cell carries the same **red** and **amber** dots as a source cell (Section 7.1.1), and here they mean the treatment's attributes **match the layer currently on that bus** — red for Program, amber for Preview. Change the Program layer so it no longer matches and the red dot clears while the amber one stays.

7.3.2 Creating and learning a treatment

A treatment is created by **learning** a layer's current keyframe — Studio captures that layer's present geometry, border, shadow, and other attributes into a new treatment.

Prerequisite: A layer is selected with the look you want to capture.

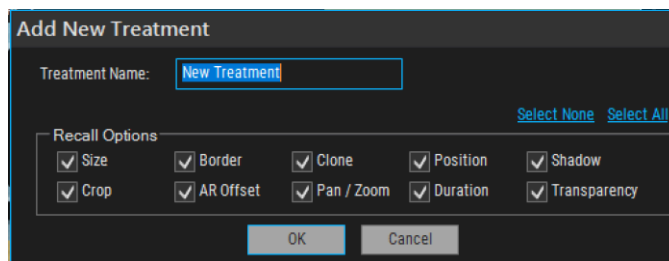


Figure 7-20. The Add New Treatment dialog — name the treatment and choose which Recall Options it captures.

1. Right-click in the Treatments pane and choose **Add New Treatment**.
2. The **Add New Treatment** dialog opens. Type a **Treatment Name** — it starts as *New Treatment* — and set the **Recall Options** the treatment should carry. **All ten are already checked when the dialog opens.** **Select All** and **Select None** set every checkbox at once.
3. Click **OK**. Studio creates the treatment by learning the selected layer's current keyframe.

You can rename a treatment later in its Properties panel (Section 7.3.5), but the name and Recall Options are both set here at creation time.

The treatment now holds the captured look and can be applied to any layer.

Note: To refresh an existing treatment from a layer, select both and choose **Re-Learn Treatment**. This updates the treatment with the layer's current keyframe.

7.3.3 Applying a treatment

Prerequisite: A layer is selected (or you drag onto a specific layer).

Apply a treatment any of three ways:

- **Double-click** the treatment to apply it to the selected layer(s).
- **Drag** the treatment onto a layer to apply it to that layer.
- Select the treatment, then right-click and choose **Apply Treatment** (enabled when a layer and a single treatment are selected).

Only the categories the treatment carries are changed on the target layer; all other attributes are left as they are.

7.3.4 Default treatment, files, and the context menu

Set as Default marks a treatment as the **default** — the look applied to every newly created layer. The item toggles (a checkmark shows) and the cell summary is prefixed **(DEFAULT)**.

Treatments are saved to and loaded from .treatment files. A single treatment exports as a "Favorite"; multiple treatments export as a package.

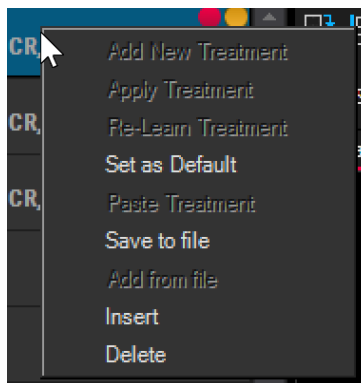


Figure 7-21. The Treatments context menu.

The full Treatments context menu:

Item	Action	Enabled when
Add New Treatment	Create a treatment by learning the selected layer's keyframe.	A layer is selected.
Apply Treatment	Apply the selected treatment to the selected layer.	Layer + single treatment selected.
Re-Learn Treatment	Update the treatment with the selected layer's current keyframe.	Layer + single treatment selected.
Set as Default	Mark / unmark as the default treatment (toggles).	Single treatment selected.
Paste Treatment	Paste a keyframe from the clipboard as a treatment.	The clipboard holds a keyframe.
Save to file	Export the selected treatment(s) to a .treatment file.	A treatment is selected.
Add from file	Import treatment(s) from a .treatment file.	Clicked on an empty slot.
Insert / Delete	Insert a slot / delete the treatment.	Base grid commands.

Note: Copying a treatment auto-names it <name> (Copy N).

7.3.5 Treatment properties and Recall Options

Selecting a treatment opens its Properties panel, headed *Treatment (ID: n)*. It has two parts.

The **Name** field renames the treatment.

The **Recall Options** group controls **which attribute categories the treatment applies** when recalled. It offers **Select None** / **Select All** convenience links, then one checkbox per category:

Checkbox label	Applies	Tag
Size	Layer size	SZ
Position	Layer position	PS
Border Settings	Border	BO
Shadow Settings	Shadow	SH
Clone Settings	Clone	CL
Crop Settings	Crop	CR
Aspect Ratio Offset	Aspect-ratio offset	AR

Checkbox label	Applies	Tag
Pan and Zoom	Pan & zoom	ZM
Duration	Transition duration	DN
Transparency	Transparency	TP

Checking an option **includes** that category when the treatment is applied; unchecking **excludes** it. **Select None / Select All** toggle every checkbox at once. The pane cell summary reflects whichever options are checked here — this is the mechanism behind a treatment that changes only some aspects of a layer's look.

Tip: The same attribute categories appear as live values in the **Key Frame** and **Border / Shadow** sections of a layer's properties (Sections 7.2.2–7.2.4, 7.2.7). Recall Options simply selects which of them a treatment carries.

7.4 Using Mix Effects

A **Mix Effect (ME)** is a two-input compositor — a **Top** content slot, a **Bottom** content slot, and a cross-fader between them — that occupies a single layer on the canvas (Section 5.4, the "Single Layer Mixer"). The transition happens inside the ME resource, so a full A-to-B transition costs one layer. The **Mix Effects** panel sits across the bottom of the window; each ME is a column with a header over a **Top** content row and a **Bottom** content row. When an ME is in use, its header carries a badge with the name of the PixelSpace it is on — for example **PS-1**, **ALL PGM**, or **Main** if the PixelSpace has been renamed.

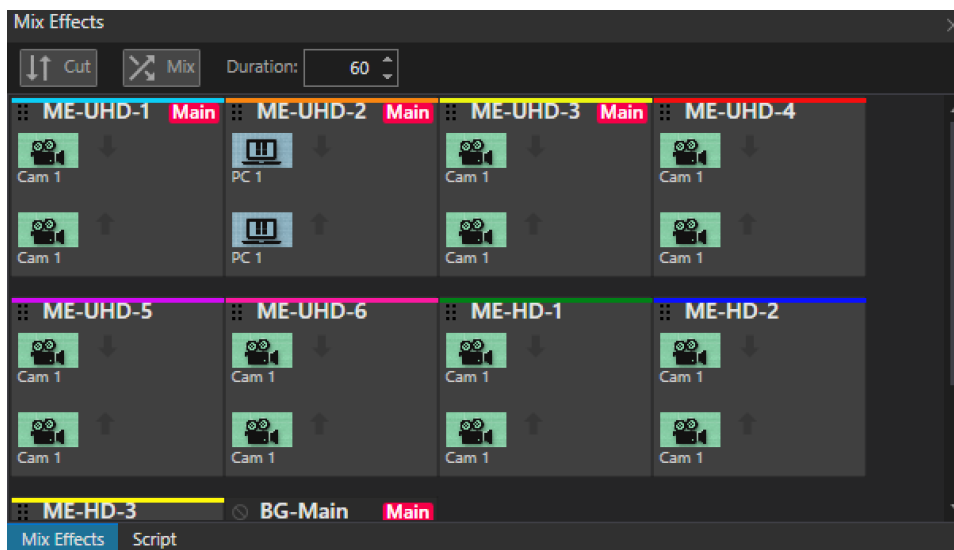


Figure 7-22. The Mix Effects panel — each ME is a column with Top and Bottom content rows. The badge on an ME's header names the PixelSpace it is on — here Main.

7.4.1 Understand Top → Program and Bottom → Preview

When an ME feeds a Program-bus layer, its **Top** input lands on **Program** (on-air) and its **Bottom** input lands on **Preview** (staging). This is the working operator model:

- **Top → Program** — the on-air content.
- **Bottom → Preview** — the staged content.

Note: "Top" and "Bottom" are the ME's two inputs; "Program" and "Preview" are the layer's two rails. They are different things that line up in normal use (Section 5.4). A Cut or Mix swaps which input is on top, which swaps Program and Preview.

7.4.2 Assign sources to Top and Bottom

Drag a source, background, or even another ME (Mix Effects can nest) onto an ME's **Top** or **Bottom** row to set that slot's content.

Put an ME on a layer. Drag the ME by the dotted grip at the left of its header:

- **Onto a PixelSpace on the canvas** — the ME goes onto the next free layer of that PixelSpace. Program shows the ME's **Top** and Preview shows its **Bottom**.
- **Onto a PixelSpace row in System Patch, or a PixelSpace cell in System Layout** (for example **PS-1**) — the same, on that PixelSpace's Program. Onto a Preview row or cell (for example **PS-1 - PVW**), it goes on that PixelSpace's Preview.
- **Onto a layer row in System Patch, or a layer cell in System Layout** — the ME is assigned to that layer.

Notice: Changing the **Top** content of an ME that is live on a *different* Program layer is blocked, so you don't accidentally alter on-air output elsewhere. The **Bottom** slot can also be locked in some cases — for example, a Background ME allows it only when its PixelSpace has more than one background source (Section 7.5).

7.4.3 Cut vs. Mix, and Duration

An ME transitions between its two inputs two ways, both reached from the panel toolbar (top-left):

Control	Function
Cut	An instant swap of the selected ME(s)' Top / Bottom.
Mix	A timed linear cross-fade over Duration .
Duration	Transition length in Frames (default 60).

Both buttons act on the **selected** ME column(s): select one or more MEs, then press Cut or Mix.

Note: A **Cut is literally a one-frame Mix** — the same linear cross-fade, jumped straight to the end. Each press of Mix or Cut **toggles** direction: if the Bottom input is hidden, the press fades it on; if it is visible, the press fades it off (transitioning back).

You can also trigger a transition from a Function Key or an external API command. All three paths funnel into the same underlying mix. The command-key and external-control tokens (the **MESR / METR / MELA / MELS** API tokens, with **Cut = METR <id> D1**) are covered in Section 10.4.

7.4.4 Worked example — transition between two sources

Prerequisite: ME-UHD-1 on Layer 1, feeding the **PS-1** PixelSpace. (A regular drag of any source onto the PixelSpace sets this state up — the source is applied through an available Normal ME, which lands on the next free layer; Section 7.1.3.)

1. Drag **Cam 1** onto the ME's **Top** row.
2. Drag **PC 1** onto the ME's **Bottom** row.

3. Observe the Simulator: **Program (PS-1)** shows Cam 1 (the Top) and **Preview (PS-1 - PVW)** shows PC 1 (the Bottom).
4. Select the ME and press **Mix** (or **Cut**).

Expected result: Program and Preview swap — PC 1 comes on-air and Cam 1 moves to Preview. A Mix dissolves over the Duration; a Cut swaps instantly. Pressing again transitions back.

7.4.5 Mix Effect types — Normal, Aux, and Background

Mix Effects come in three kinds, all built on the same model. Right-click a Normal or Aux ME to **Rename...** it or switch its type — on a Normal ME the item reads **Switch to 'Aux' Mix Effect...**; on an Aux ME it switches the ME back to Normal. A Background ME's menu has **Rename...**, **Freeze (Top)**, and **Freeze (Bottom)** instead (Section 7.4.6). The same toggle is on the ME's **Mode** dropdown in the Properties panel.

Selecting an ME opens its Properties panel, headed *Mix Effect (ID: n)*:

Field		What it does	
Name		The ME's name.	
Resolution		Read-only — the ME's resolution class (for example <i>UHD</i>).	
Mode		Normal or Aux .	
Custom Button Color		When checked, the ME is marked with the chosen Color : its name tag on the layer in the Simulator and in System Patch, and a bar across its tile in the Mix Effects panel.	
Color		Greyed until Custom Button Color is checked. Click ... to pick the colour in the Color dialog ; a value typed into the box is not applied.	

ME type	Purpose	Type-toggle	Freeze?
Normal	Standard program/preview mixing. Allocated in <i>MixEffects</i> mode.	Normal ↔ Aux	No
Aux	Backs unscaled-aux / tiled / OpMon output assignment. Allocated in <i>Layers</i> mode. Same underlying class as Normal.	Aux ↔ Normal	No
Background	The per-PixelSpace background mixer (Section 7.5). Bound to a PixelSpace; accepts only background content.	Locked to <i>Background</i>	Yes

Note: Normal and Aux are the same underlying ME class — the difference is intent and allocation, and the Mode toggle flips between them. A **Background** ME is a different subclass: it cannot be dragged or type-toggled, and selecting it does not open a Mix Effect Properties panel.

7.4.6 Freeze an ME

Freeze holds the current frame on an input — for example, to avoid artifacts while content changes underneath. Right-click a **Background** ME and choose **Freeze (Top)** or **Freeze (Bottom)**.

Note: Freeze is available **only on Background MEs**, and only when that bus's content is a live background source. The freeze items do not appear on Normal or Aux MEs. You cannot freeze a still — a still is already static — nor unfreeze one.

Tip: Mix Effects are **global resources** — any input on any Frame can feed any ME, and one ME can drive **several layers at once**, so a single Cut or Mix switches every screen it feeds, in sync. In a script, if the same ME is given different sources within one Command Key, Studio "**picks a winner**" to resolve the conflict (Chapter 9).

7.5 Working with backgrounds

A **background** is full-canvas content that sits behind every layer on a PixelSpace — "Layer Zero" (Section 5.5). The headline resource fact: a background consumes **inputs, not layers**. The layer allocator explicitly skips any Frame used as a background, so a background draws down the input budget without spending a layer (Section 5.9).

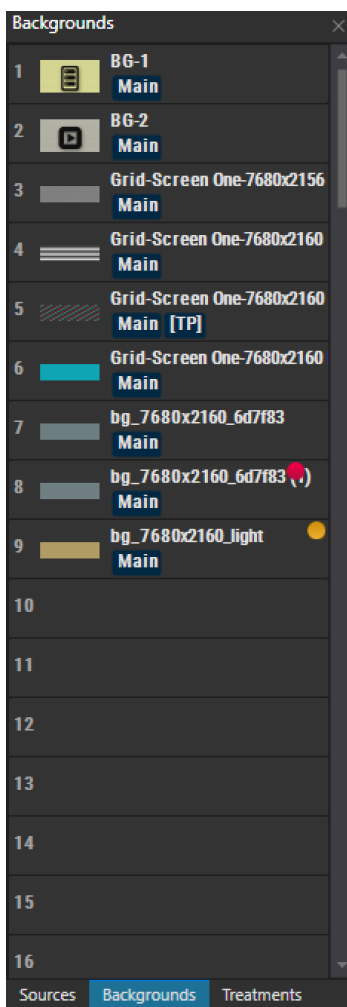


Figure 7-23. The Backgrounds pane — Video, Still, and Test Pattern content.

Backgrounds live in the **Backgrounds** pane — the middle left-dock tab (**Sources • Backgrounds • Treatments**). A background is **per PixelSpace**: one built for a PixelSpace applies only there, shown through that PixelSpace's background Mix Effect (the *BG-...* tile in the Mix Effects panel).

7.5.1 Understand the three content types and the background group

A background-content item is one of three kinds:

Type	What it is
Video	A live background built from one or more inputs — the background group .
Still	A still image, scaled to the PixelSpace. Added with Still is Test Pattern unticked (Section 7.5.5).
Test Pattern	A still image, scaled to the PixelSpace, added with Still is Test Pattern ticked. Marked with a [TP] badge. Shown directly on the PixelSpace's outputs (Section 7.5.7).

Reading a cell. A populated cell shows the background's **name** and a badge naming its **PixelSpace**. The cell does **not** spell the type out in words: a test pattern is marked with a [TP] badge beside the PixelSpace badge.

A **background group** (a Video background) is an **HCount × VCount grid of inputs**, tiled to fill the PixelSpace. Studio computes the grid from the PixelSpace size: each grid cell claims one background input. A PixelSpace wider or taller than a single input needs several inputs combined into the group — for example, a 7680 × 2160 PixelSpace fed by two 3840 × 2160 inputs is a 2 × 1 group consuming two inputs (live: *BG-1 = In-1A + In-1B*). Dual-link / paired connectors reserve two physical channels per cell (Section 5.7).

Tip: Background stills load and dissolve **instantly** from the Frame's memory bank (no legacy "chunking" delay), and several 4K inputs can be grouped as **one logical background** for an ultra-wide wall. Backgrounds are **Frame-Group-local** — they must be physically connected to the Frame Group that displays them and cannot cross the inter-Frame link (Mix Effects, by contrast, are global).

7.5.2 Crop a background input

Cropping is **per input**. Select a background to open its **Background Source** properties, click an input tile in the source-selection grid (for example *In-1A · In-1B*), then set that input's crop. The same panel also shows the background's **Name, Thumbnail / Browse...**, its **PixelSpace** (read-only), and an **Input Configuration** section with **Override** and **Levels** tabs.

Field	Notes
Crop → Top • Left • Right • Bottom	The crop applied to the selected input. Bounded 0...65535.
X Position / Y Position	The input's placement within the PixelSpace.
Width / Height	Read-only — computed.

Background Source

Name: BG-1

Thumbnail:

PixelSpace: PS-1 (ID: 0)

In-1A In-1B

Input: In-1A

Crop

Top:

Left: Right:

Bottom:

X Position:

Y Position:

Width:

Height:

Figure 7-24. Background Source properties for BG-1 with In-1A selected. A Left crop of 200 reduces its Width from 3840 to 3640.

The crop rule is why Width and Height are read-only: $\text{Width} = \text{InputRes} - (\text{Left} + \text{Right})$ and $\text{Height} = \text{InputRes} - (\text{Top} + \text{Bottom})$, each clamped to at least 0. **Crop drives the size** — you do not type Width or Height.

Cropping here works the same way as in Configuration Manager (*Cropping a tile by hand*, Section 6.4.1.3): the cropped tile keeps its X / Y Position, and the tile to its right moves left by the crop amount — with a Left crop of 200 on In-1A, In-1B moves from X 3840 to 3640 — and moves back when you reduce the crop. There is no Apply step: the neighboring tile's X Position updates as soon as you enter the value.

Several behaviors follow automatically:

- **Auto edge-crop.** Inputs at the group's right or bottom edge are auto-cropped so the group does not overrun the PixelSpace.
- **Clip to PixelSpace.** Display rectangles are clamped into the PixelSpace bounds.

Caution: Do not crop a tile down to zero width. The panel accepts it — Width shows 0 — but when you reduce the crop again, the tile to its right does not move back: it stays over the first tile, and it can no longer be selected in the grid.

Notice: Changing an input-size value while **stills exist** on that PixelSpace prompts a warning that the background stills will require a **server restart** to reload properly — the stills are baked into the input frame buffers.

7.5.3 Mix or Cut backgrounds — the two-group rule

Each PixelSpace has a background Mix Effect (the *BG-...* tile) with a Top and a Bottom bus. It Cuts and Mixes exactly like any ME (Section 7.4), with **Top** → **Program** and **Bottom** → **Preview** — but it needs **two background groups in the same PixelSpace**.

Note: With a single background group, the **bottom bus cannot be set** — there is nothing to transition to — so both Mix and Cut are inert. The UI refuses to drop content on the bottom bus, and the background Mix is a no-op unless both buses have content. Add a **second group** to the same PixelSpace to enable transitions.

On startup the background ME auto-populates **Top = the first background source** for the PixelSpace and **Bottom = the second**; if there is no second source, Bottom stays empty. With two groups carrying distinct images, pressing **Cut** swaps Program and Preview instantly (one frame) and pressing **Mix** dissolves them over the Duration.

Tip: You can also drive the background straight from the canvas. **Right-click an empty area of a PixelSpace** (which selects the PixelSpace) to open its background menu: **Next Background**, **Current Background**, **Mix Background**, and **Toggle Background Freeze** — the same transitions as the BG Mix Effect, plus a freeze toggle, without leaving the Simulator.

7.5.4 Substitute a still for a background

A **still** can sit on a background bus exactly like a Video source. It substitutes for a source by **reusing that source's inputs and frame buffers**, clipped by the same per-input crop, and it obeys the same PixelSpace, crop, and two-group Mix/Cut rules. A test-pattern still works differently; see Section 7.5.7.

Note: A still needs an available backing source. If none is free, Studio reports *"No available backing source for still..."*. A source cannot claim inputs a still is using on the other bus. A still is inherently frozen and cannot be unfrozen.

7.5.5 Add a background image

Prerequisite: A PixelSpace that already has a background source. Otherwise Studio reports *"No PixelSpaces with background sources are available."*

To add a background still:

1. In the Backgrounds pane, right-click the PixelSpace's **background** item (for example **BG-1**) and choose **Add Background Image**. The **Add Background Still** dialog opens with that background's PixelSpace selected and **Still is Test Pattern** unticked.
2. Set the **Still Image File** with **Browse...**. A thumbnail preview and an image-size label appear.
3. Click **Save** (enabled only with a valid PixelSpace and file).
4. If the background is on Program, Studio asks **Freeze Background Source?** so the image loads without artifacts on screen. Click **Yes**.

Expected result: Studio scales the image to the PixelSpace resolution, loads it, and adds it to the Backgrounds pane with a thumbnail, its name, and a PixelSpace badge.

Note: To add the image as a test-pattern still instead, tick **Still is Test Pattern** before you click **Save**. Right-clicking an **empty** slot opens the same dialog with the box already ticked; choose the **PixelSpace** from the *Select Pixel Space* dropdown. Untick the box if you want a plain still.

The Backgrounds pane context menu also offers **Insert** (insert a register slot) and **Delete**. **Delete** removes a Still or Test Pattern (or an empty slot) only — a **Video** background cannot be deleted from this menu.

7.5.6 How many background stills fit

Each Frame has memory for **16** background stills. A still uses one of those 16 for **each background input** its PixelSpace uses on that Frame. PixelSpaces whose background inputs are on the same Frame **share** that Frame's 16. A second background on the same PixelSpace (for example BG-2, for Mix and Cut) covers the same area and uses no more.

All background inputs on one Frame:

PixelSpaces	Background inputs per PixelSpace	Stills (total)
1 × 15360 × 2160	4	4
2 × 7680 × 2160	2	8, shared between the two PixelSpaces
4 × 3840 × 2160	1	16, shared between the four PixelSpaces

Background inputs on two Frames. Each Frame adds its own 16. A PixelSpace whose inputs are split across both Frames uses one still's worth on each Frame at the same time.

PixelSpaces	Background inputs (Frame 1 / Frame 2)	Stills (total)
1 × 7680 × 2160	1 / 1	16
2 × 7680 × 2160	PS-1: 2 / 0 · PS-2: 0 / 2	16 (8 each)
4 × 3840 × 2160	Two PixelSpaces on each Frame	32 (16 per pair)

When a Frame's still memory is full, **Add Background Image** reports *"Maximum background still limit reached. Remove an existing still and then try again."* Delete a still to free its space. Test-pattern stills use separate memory and do not count toward these limits (Section 7.5.7).

7.5.7 Test-pattern stills

A test-pattern still is an image added with **Still is Test Pattern** ticked (Section 7.5.5). It belongs to the one PixelSpace you added it to, and it is not played through the background Mix Effect: it is shown directly on the outputs that display that PixelSpace. Click it in the Backgrounds pane to see its **Background Still** properties — **Name**, **PixelSpace**, **Is Test Pattern: Yes**, and a thumbnail.

To show a test-pattern still:

1. Set the Simulator's **View Stack** so that the test pattern's PixelSpace is shown on its own (not the combined *ALL PGM / ALL PVW* view).
2. Drag the [TP] item from the Backgrounds pane onto that PixelSpace on the canvas.

To remove it: in **System Patch**, right-click the PixelSpace and choose **Clear TestPattern** (Section 6.10.1).

How many test-pattern stills fit. Each Frame has memory for **48** output images, and a few are already in use for the outputs' own default test patterns. A test-pattern still uses one of them for **each output** that shows its PixelSpace on that Frame. PixelSpaces whose outputs are on the same Frame share that Frame's memory. On the reference system:

PixelSpace	Outputs showing it (one Frame)	Test-pattern stills
3840 × 2160	1	45
7680 × 2160	2	22

This memory is separate from the 16 background stills per Frame (Section 7.5.6): a full background-still memory does not stop you adding test-pattern stills, and the reverse. When it is full, **Add Background Image** reports the same message, *"Maximum background still limit reached. Remove an existing still and then try again."*

7.6 Managing PixelSpaces

To open a PixelSpace's properties, do any of the following:

- Click an empty area of the PixelSpace on the **Program** or **Preview** canvas (an area not covered by a layer).
- Click the PixelSpace in **System Patch**.
- Click the PixelSpace in **System Layout**.

The **Properties** panel opens, headed **PixelSpace (ID: n)**, with the PixelSpace's name, origin and size, **Titling**, and the **Test Pattern Generator**. **Titling is not supported in Spyder Studio 6.1.3**; leave its settings at their defaults (Section 6.8.2). Every field is described in Section 6.8; System Patch and System Layout are described in Section 6.10.

7.7 Managing outputs

Output configuration has two halves. The structural half — how many outputs, their roles (Normal, Aux, OpMon, Tiled), and the canvas mapping — is set in configuration; see Sections 6.4.1.4 and 6.5; OpMon and Tiled outputs, and their presets, are in Sections 6.6 and 6.7, and System Patch and System Layout in Section 6.10. This section covers the **live** side: adjusting one output's format, color, timing, and test pattern on a running system.

Output (ID: 12)

General ▾

Name
Out-4E

Connector
SDI

SDI Mode
Level A ▾

Resolution: 3840x2160 @ 59.94 Hz

Frame Locked ● Load Standard

Mode ▾

Normal ▾

HStart
0 < >

VStart
0 < >

Color Configuration

Color Space
BT.2020 ▾

Chroma
4:2:2 ▾

Color Range
Limited ▾

HDR
SDR ▾

Bit Depth
10-Bit ▾

Video Options ▴

Test Pattern Generator ▴

Advanced ▴

Figure 7-25. Output properties for an output in use — Out-4E, an SDI output in Normal mode. The four groups are General, Video Options, Test Pattern Generator and Advanced.

Select a physical **output** in **System Patch** or **System Layout**. The **Output properties** view opens in the Properties panel, headed *Output (ID: n)*. The panel has **four collapsible groups, in this order: General, Video Options, Test Pattern Generator, Advanced**.

⚠ **Test Pattern Generator appears only on an output that is in use.** Select an output whose Mode is **Unused** and the panel shows just General, Video Options and Advanced. Give the output a role and the group appears. (An Unused output still shows its own default test pattern on a connected display; see Section 6.5.) The **General** group varies with the port as well: an SDI output adds **SDI Mode**, and an output in use adds **HDR**.

Note: Selecting several outputs at once shows *N Outputs Selected*, and every edit applies to all of them. A blank or duplicate **Name** raises a validation popup.

7.7.1 General

The General group holds identity, format, and color.

Field	Example	Notes
Name	<i>Out-2A</i>	User-definable; must be unique and non-blank.
Connector	<i>HDMI</i>	Connector type. Varies by port (for example, DisplayPort on other ports).
Resolution	<i>3840×2160 @ 59.94 Hz</i>	Read-only. Change it via Load Standard .
Frame Locked	•	Genlock status indicator.
Load Standard	—	Button: load a standard video format.
SDI Mode	<i>Level A</i>	SDI outputs only — absent on HDMI and DisplayPort outputs.
Mode	<i>Normal</i>	Normal, Unscaled Aux, OpMon, Tiled, or Unused — the choices depend on the port (see below).
Background Color	R / G / B spinners plus a swatch	Appears only when Mode is OpMon .
HStart / VStart	0 / 0	Horizontal / vertical start (spinners). Hidden while Mode is Tiled .

Which **Mode** options an output offers comes from its card, as reported by hardware discovery (Section 6.5). As a rule of thumb:

Mode	Offered on
Normal · Unscaled Aux · Unused	all four outputs
Tiled	outputs 3 and 4
OpMon	output 4 only

A port that does not support a mode simply does not list it. These are the same role names the configuration wizards use (Section 6.5).

Note: **Load Standard** is greyed out while Mode is **Unused** — set a working mode before loading a video format.

The **Color Configuration** subgroup sets the output's color encoding:

Field	Example
Color Space	RGB
Chroma	4:4:4

Field	Example
Color Range	Full
Bit Depth	8-Bit

Note — HDR. On **SDI Frames**, Color Configuration carries an **HDR** field (for example *SDR*), between Color Range and Bit Depth. On HDMI and DisplayPort Frames that field is absent, and HDR is reached through the Advanced grid instead (Section 7.7.4).

7.7.2 Video Options

Video Options is a two-tab control: **Levels** and **Timing**.

The **Levels** tab carries **Gamma** (value plus slider). Gamma is the only level supported in Spyder Studio 6.1.3.

The **Timing** tab controls custom output timing:

Field	Notes
Enable custom timing	Checkbox, on by default. It is the same setting as UseAlternateOutputSync in the Advanced grid (Section 7.7.4).
Vertical Offset / Horizontal Offset	Greyed until <i>Enable custom timing</i> is checked.

With **Enable custom timing** checked, **Vertical Offset** and **Horizontal Offset** become editable. The full set of timing parameters is exposed in the Advanced grid (Section 7.7.4).

7.7.3 Test Pattern Generator

This generator places a test pattern directly on the **physical output** — useful for confirming cabling or identifying a port before content is up. (To fill a whole PixelSpace instead, use the PixelSpace test pattern generator; see Section 6.8.)

Field	Live value
Pattern → Category	<i>Color Bars</i>
Pattern → Type	<i>Color Bars, Grayscale Lower</i>
Options	Outline, Grid, Center O, Center X (toggles; <i>Animation</i> appears with Center X).
Background Color	R / G / B spinners plus swatch.
Foreground Color	R / G / B spinners plus swatch.
Preview	A preview of the selected pattern.
Clear / Apply	Clear the pattern / apply it to the output.

Note: The output generator has only **Clear** and **Apply**, and so does the PixelSpace generator (Section 6.8). Neither offers a *Save To Still List* button, and there is no *Test Pattern Active* field on either.

7.7.4 Advanced

The Advanced group is a raw property grid with a $\equiv$ / **A-Z** sort toggle (categorized versus alphabetical). It exposes **every** output-model property by its code name — the complete low-level view.

This is where the **detailed timing**, **HDR**, and **UseAlternateOutputSync** live:

Property	Example		Property	Example
BitDepth	8-Bit		HDR	SDR
Chroma	4:4:4		Mode	Normal
ColorRange	Full		RefreshRate	59.94 Hz
ColorSpace	RGB		TimingHorizontalOffset	0
Connector	HDMI		TimingVerticalOffset	0
FrameLock	☒		UseAlternateOutputSync	☒
Gamma	1.00		VActive	2,160
HActive	3,840		VFrontPorch	8.00
HFrontPorch	176		VSyncWidth	10.00
HSyncWidth	88		VTotat	2,250
HTotal	4,400			

Note: **UseAlternateOutputSync** is enabled by default on every output type — HDMI, DisplayPort, SDI and SFP — for any configuration created in 6.1.3 or later. A configuration created before 6.1.3 keeps whatever setting it already had; to pick up the new default on those outputs, create and apply a new configuration.

Tip: Use the A-Z toggle to find a property fast by name, or the categorized view to read related timing values together.

7.8 Using routers

Routers feed sources into Spyder and, where an external matrix sits in front, let Spyder pull any external source to any input. Spyder always has one **internal crosspoint** per Frame, shown as *Spyder Internal (Frame n)* — and you add external routers via **Tools ▶ Routers ▶ Add New Router**.

Two tools operate routers, and the distinction matters:

- **Router Patch** — the *static* wiring map: how routers cascade into one another and into Spyder.

- **Router XY** — the *live* X-Y crosspoint control: take an input to an output, now.

7.8.1 Router Patch (static wiring)

Router Patch defines how each output of a router is connected — to a Spyder layer directly, to an input on a downstream router (a cascade), or left unconnected. It is **closed by default**; open it from **View ▶ Router Patch**, where it docks along the bottom of the workspace, sharing a pane group with **System Patch** and the **Alert Viewer**.

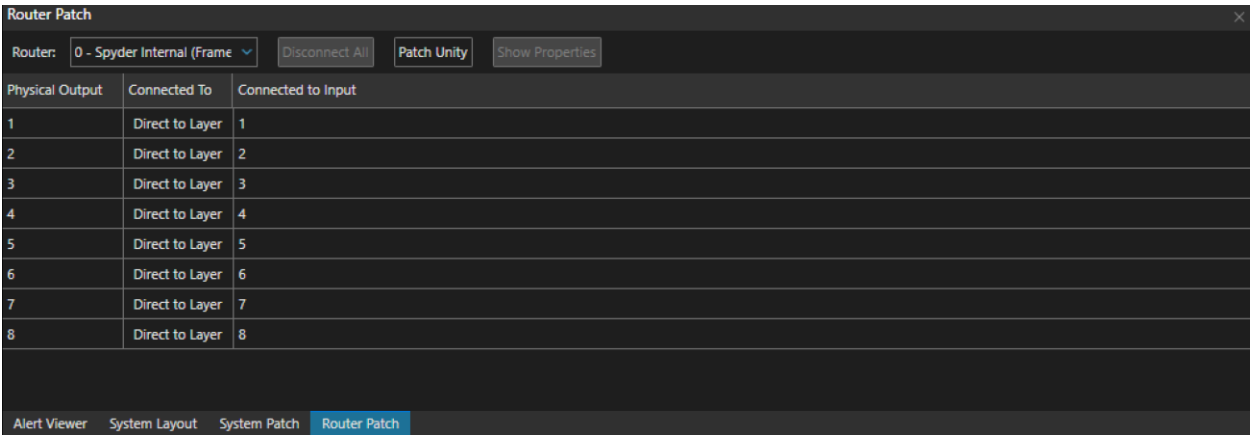


Figure 7-26. Router Patch — the static output-to-input wiring grid.

The toolbar:

Control	Function
Router (dropdown)	Choose which router to view/edit. It lists one internal crosspoint per Frame — 0 - <i>Spyder Internal (Frame 0)</i> , 1 - <i>Spyder Internal (Frame 1)</i> , and so on — followed by any external routers you have added.
Disconnect All	Clears all output patches on the selected router. Disabled for the internal crosspoint.
Patch Unity	Opens a router-selection dialog and builds a default 1:1 ("unity") patch from this router into a chosen downstream router.
Show Properties	Opens the selected router's properties in the Properties panel. Available for external routers; disabled for the internal crosspoint.

The grid has one row per physical output of the selected router:

Column	Meaning
Physical Output	The output number on the selected router (1, 2, 3 ...).
Connected To	Where that output is routed (dropdown — see below).
Connected to Input	The input number on the downstream router this output feeds. Entered 1-based in the UI.

The **Connected To** options are:

Option	Meaning
Not Connected	Output is unpatched.
Direct to Layer	The output feeds a Spyder layer/input directly — the normal end of the chain, and the default for every row.
<i>(another router, by name)</i>	Cascade — this output feeds that downstream router's input (set the number in <i>Connected to Input</i>). One entry appears per <i>other</i> router, so this only shows up once you have added a second router.

Editing a row and committing pushes the change to the router configuration; clearing a row removes that output's patch.

Note: When an **internal crosspoint** is selected the grid is **read-only** — you cannot re-patch it here. Clicking a **Connected To** cell opens no dropdown at all, **Disconnect All** and **Show Properties** are greyed out, and each physical output maps **Direct to Layer** at the matching input (the default 1:1 input-to-layer mapping). This is why, on a Spyder-S with no external router, sources map straight to inputs and layers. Select an external router instead and the same grid becomes editable — the **Connected To** cells open and those two buttons come on.

Tip: Use **Patch Unity** to lay down a straight 1:1 chain into a downstream router, then adjust individual rows. Enter input numbers as printed on the physical router — they are 1-based in the grid even though stored 0-based internally.

7.8.2 Router XY (live crosspoint)

Router XY is the live X-Y crosspoint panel — route any input to any output, in real time. It is **hidden by default**; enable it with **View ▶ Router XY**.

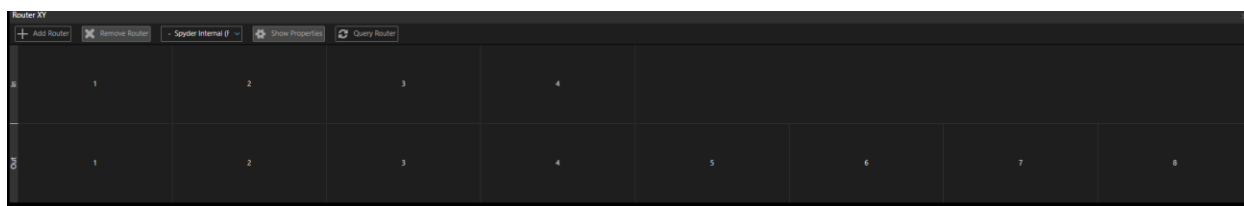


Figure 7-27. Router XY — the live X-Y crosspoint panel.

Control	Purpose
Router (dropdown)	Pick which router to operate. Like Router Patch's, it lists one internal crosspoint per Frame — <i>Spyder Internal (Frame 0)</i> through <i>(Frame 3)</i> on a four-Frame system — followed by any external routers, and opens on the first of them. ⚠ The number is zero-based , as it is in the source editor (Section 7.1.2): <i>(Frame 0)</i> is Frame 1 .
Out strip	One button per router output — the destination (the "Y" axis).
In strip	One button per router input — the source (the "X" axis).
Toolbar	Add Router, Remove Router, Show Properties, Query Router. With an internal crosspoint selected only Add

Control	Purpose
	Router and Query Router are available — Remove Router and Show Properties are greyed out, since an internal crosspoint is neither removable nor editable.

Operate it destination-first, the classic X-Y workflow:

Prerequisite: The correct router is selected in the **Router** dropdown.

1. **Select an output** (destination). The panel highlights the input that currently feeds it, reading live state — so you can see what is routed where.
2. **Select an input** (source). The crosspoint is **taken immediately** — the input is routed to the selected output live.

The display stays in sync with the system: if a route changes elsewhere — a script, a salvo, or external control — the highlighted input for the selected output updates automatically.

The toolbar commands:

Command	Action
Add Router	Opens the Add New Router dialog, adds it, and selects it.
Remove Router	Removes the selected router (confirmation prompt). Disabled for the internal crosspoint.
Show Properties	Shows the selected router's properties in the Properties panel. Disabled for the internal crosspoint.
Query Router	Polls the physical router for its current output-to-input state and re-syncs the display with the hardware.

Tip: Use **Query Router** to reconcile the panel with an external router that may have been changed from its own front panel or another controller. **Router Salvo** function keys fire a saved set of these crosspoints at once; Router XY is the manual, one-at-a-time equivalent (see Section 8.4).

7.8.3 The Add New Router dialog

Tools ▸ **Routers** ▸ **Add New Router** defines an **external router** so Spyder can control it. The dialog is also reachable from **Router XY**'s own **Add Router** button (Section 7.8.2).

Field	Purpose
Name	The router's name, as it will appear in the Router dropdowns. Defaults to <i>Virtual</i> .
Router Type	The router model / protocol to drive. Defaults to <i>Virtual</i> .
Connector	The router's connector type. Defaults to HDMI .
Inputs × Outputs	The number of router inputs and outputs, which together size the crosspoint. Defaults to 32 × 32 , and the Router Patch grid then shows one row per output.

Field	Purpose
Patch To	Initial downstream target, applied to every row of the new router's patch. On a four-Frame system with no external routers it lists the four internal crosspoints , numbered 0 to 3 (<i>0 - Spyder Internal (Frame 0)</i> and so on), and defaults to 0 - Spyder Internal (Frame 0) . There is no Direct to Layer choice here .
Ok / Cancel	Create the router, or abandon.

After creation the router appears in **Router Patch** and **Router XY**, and can be fired in sets via Router Salvo function keys (Section 8.4.3).

With sources, layers, Mix Effects, backgrounds, outputs, treatments, and routers all running, the remaining chapters cover repeatability and integration: saving looks as Command Keys and running one-touch Function Keys (Chapter 8), building and running scripts (Chapter 9), and driving the system from external controllers (Chapter 10).

8. Command Keys and Function Keys

A **Command Key** is a saved look you can recall onto the canvas on demand — a stored arrangement of layers, sources, positions, and treatments that the system reapplies in a single action. Command Keys turn a built composition into a repeatable cue, so an operator can move between known states during a show instead of rebuilding each one by hand. A **Function Key** is the single-action counterpart: it performs one action on the layers you have selected.

This chapter covers both key panels. Command Keys and several Function Key types emit the same string-command tokens used by the external control API; this chapter covers how you build and fire them in Studio, and Chapter 10 documents the API surface those tokens reach. The concepts behind layers, Mix Effects, and the Program/Preview rails are established in Chapter 5; live show operation is in Chapter 7; the Script panel, where a key's script is shown and edited, is in Chapter 9.

8.1 Keys and scripts

Every Command Key is stored as a **script** — an ordered list of cues, each cue carrying the content, layers, and keyframes that define a moment of the show. The Script panel (Chapter 9) shows and edits that script; it can also be used on its own, with no key involved.

The three tools divide along one axis — how much state each recalls:

Tool	Recalls	Triggered by
Command Key	a full saved snapshot (a script)	Preview or Program button
Function Key	one parameterized action on selected layers	a single execute button
Script panel	the live working timeline (Script ID -1)	Space, double-click, triggers, or clock

Selecting a Command Key loads its script into the Script panel: the panel header shows the key's Script ID and the grid shows its elements. Selecting a key does not recall it. Changes you make to those elements are saved in the key and reach the canvas the next time you recall it (Section 9.4).

Note: Command Keys and Function Keys serve different jobs. A **Command Key** recalls a full snapshot. A **Function Key** performs a single parameterized action on whatever layers are selected (Section 8.4).

8.2 Command Keys

A **Command Key** is a saved snapshot of system state — which sources are on which layers, where those layers sit, their size and geometry, and the treatments applied to them. Each key is a **script** that, when recalled, instantly sets up that look. Command Keys share a pane group with **Function Keys**, docked in the column to the right of the Simulator.

Notice — Basic Presets was removed in 6.1.3. Earlier Spyder systems offered a separate **Basic Presets** feature. **It is not available on Spyder-S:** the interface was removed from Spyder Studio in 6.1.3, and there is nothing to see or manage in the application. Use Command Keys instead. One caution for anyone upgrading from an earlier Spyder: the old external-control commands were never removed from the command processor, so a controller that still sends

them **can change what is on air** without anything appearing in Spyder Studio to explain it, and nothing stored that way can be deleted. Audit any such controller before going on air — see Section 10.3.14.

8.2.1 Layout and pages

The Command Keys panel has three parts:

- A **toolbar**: **Disconnect**, **Cancel Triggers**, **Preview Only Scripting Mode** (see Section 8.2.5).
- **Page tabs** down the left edge. Command keys are organized into **pages** (for example *1 - Page 1*). Right-click a tab for **Add Page**, **Rename Page**, and **Remove Page**. **Add Page** prompts for a name (default *New Page*) and the new tab is numbered in sequence — *2 - New Page* — with its own empty key grid. **Rename Page** opens the same prompt with the tab's current name already in the field; the tab's number stays the same, so *1 - Page 1* renamed to *Ops Page* becomes *1 - Ops Page*. **Cancel** on either prompt leaves the pages unchanged.
- A **grid** of command-key cells, empty until you learn keys.

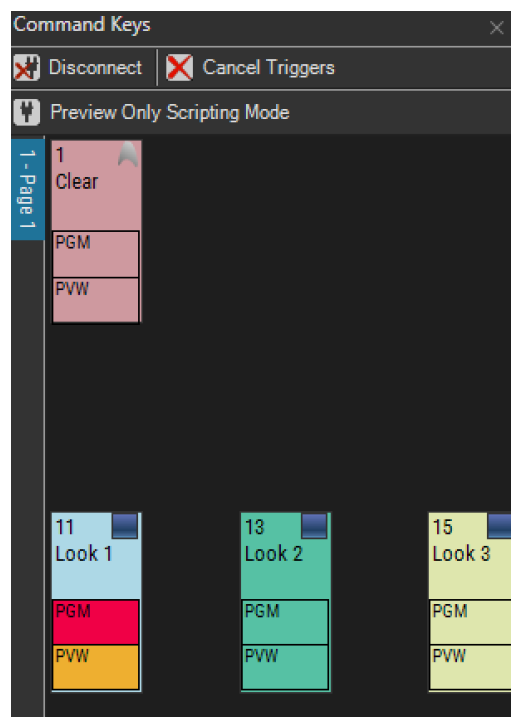


Figure 8-1. The Command Keys panel — toolbar, page tabs, and the key grid.

Tip: Command Keys are recalled far more often than they are built. Organize pages around the running order — a page per act, per segment, or per look family — so the keys an operator needs during a show stay one click away.

8.2.2 Recalling a command key

Each command-key cell has **two buttons**:

Button	Action
PGM	recalls the key's script to Program (cue 1)

Button	Action
PVW	recalls the key's script to Preview (cue 0)

This is the classic **preview-then-take** workflow: load a look to Preview, check it, then take it to Program. Selecting a key shows its properties. Drag a key to rearrange it.

"Learning" a key captures that state; recalling it reapplies the whole look in one action. Because the snapshot is stored as a script rather than a fixed image, a key can be recalled to either rail — to **Preview**, so the look can be checked before it goes out, or directly to **Program**.

Once a key has been recalled, its buttons act as tally: **PGM** lights **red** while the key's look is on Program and **PVW** lights **amber** while it is on Preview. Change what is on a bus and that button goes dark.

What a recall does to the rest of the canvas depends on the key's type:

Key type	Behavior on recall
Absolute	Reapplies the captured values exactly — layers land where they were when the key was learned. Layers that are not in the key are taken off.
Relative	Reapplies the captured values to the layers in the key and leaves layers that are not in the key as they are.

A new key takes its type from **Create as Relative** in **Tools ▶ Options ▶ Command Key Grid** (Section 4.9). To change a key's type later, use **Make Absolute** or **Make Relative** on its right-click menu (Section 8.2.4).

On an ordinary key the tile's corner icon shows which type it is — a blue panel when relative, a grey arch when absolute.

Caution: With **Live Updates Enabled** on (Section 9.8.1), a recalled key stays linked to its layers. Repositioning one of those layers saves the new position into the key, so the next recall brings back the changed look, not the one you learned. To adjust layers without changing the key, click **Disconnect Layers** (Section 9.8.2) first.

8.2.3 Creating a command key

A command key is created by capturing — "**learning**" — the current system state into a new key's script, or by adding a blank key to build up by hand.

To create a command key:

Prerequisite: The look you want to capture is set up on Program and/or Preview (unless you are adding an empty key).

1. Right-click an empty cell in the grid.
2. Choose a create option from the context menu.
3. Enter the key's **name** when prompted (default *untitled*) and click **OK**. **Cancel** abandons the key — nothing is created.

The new key appears in that cell, showing its number, its name, and its **PGM** and **PVW** buttons. The create menu, in order:

Item	Action
Add Empty	create an empty command key to build up manually
Add From Screens	learn both the Program and Preview state
Add From Preview	learn the current Preview state
Add From Program	learn the current Program state
Cut / Copy / Paste	Cut and Copy work only on an existing key. Paste is available on an empty cell once a key has been cut or copied; it opens the name prompt with the copied key's name filled in
Add Multi Key	create a multi key — a command key that holds a list of other command keys

Multi keys. **Add Multi Key** prompts for a name like the other create options, and the new key sits in the grid with the same **PGM** and **PVW** buttons — but its tile icon shows two stacked panels instead of one, so you can tell a multi key from an ordinary key at a glance. Double-click the tile to edit it: the Properties panel gains a **Command Key List** section with an **Add Command Key** button. The list starts with one row already filled in. Each click of **Add Command Key** adds another row, with a **Delete** button to remove it again. Each row is a dropdown of the project's ordinary command keys — multi keys, including this one, are not listed.

8.2.4 The command-key context menu

Right-click an existing key for key-management actions:

Item	Action
Cut / Copy	put the key on the clipboard
Paste	greyed out until a key has been cut or copied
Delete	remove the command key — it goes immediately, with no confirmation prompt
Rename	rename the key in place — the tile turns into an edit box with the current name selected. Type the new name and press Enter ; Esc cancels. (Creating a key uses a Command Key Name dialog instead.)
Make Absolute / Make Relative	switch the key between absolute and relative (Section 8.2.2). Only the one that applies is listed, so the label tells you the key's current state.
Build Slave Layers / Clear Slave Layers	add the key's slave layers to its script, or take them out again. Both act immediately — there is no prompt and no confirmation message. See Section 9.7.

Command-key properties. Selecting a key fills the Properties panel with a **Command Key** section, headed with the key's ID:

Field	Purpose
Name	the key's name, the same text the tile shows
Type	Absolute or Relative — the same setting as Make Absolute / Make Relative on the context menu
Custom Button Color	tick to colour this key's tile yourself instead of using the default
Color	the tile colour, as an R, G, B value with a ... button that opens a colour picker

Below it, a **Clock Source** section holds a **SourceMaster Port** setting — <None> or a port (*Port 1, Port 2, ...*), defaulting to *Port 1*. **Server** ▶ **Global Parameters** has a matching **SourceMaster** group, holding the SourceMaster **Master IP** (Section 10.2).

8.2.5 Command Keys toolbar

Button	Function
Disconnect	disconnect layers from scripts so they become manually controllable — the same action as the Simulator's Disconnect Layers (Section 9.8.2)
Cancel Triggers	cancel pending cue triggers
Preview Only Scripting Mode	toggle — when on, script recalls affect Preview only (mirrors the Script panel's Enable preview only scripting , Section 9.5)

8.3 Startup behavior

Spyder provides a configurable **startup recall**, set in **Server** ▶ **Global Parameters** ▶ **Startup Options**:

Option	Default	Effect
Load Backgrounds	On	Loads the configuration's backgrounds at startup.
Load Command Key	<None>	Optionally recalls the selected Command Key when the system starts. <None> means no Command Key is fired.

Option	Default	Effect
Cue ID	0	The cue recalled with the startup Command Key: 0 is the key's Preview cue and 1 its Program cue, matching the 0 Preview and 1 Program rows in the Script panel (Section 9.1).

Changes to Startup Options take effect only when you click **Apply Settings Now** at the top of the panel; **Cancel Changes** discards them.

Note: By default **Load Command Key** is <None>, so **no Command Key is recalled automatically** — a configuration loads its saved layouts and stored data, and Program stays in its loaded state. To make the system come up on a specific look, assign a Command Key (and Cue ID) under Startup Options. This is the one startup-recall mechanism; there is no separate Tools ▸ Options toggle for it, and the command-key code path does not auto-fire on load.

Note: A related behavior is documented elsewhere: **Preview Only Scripting Mode** keeps a recall on **Preview** instead of Program (Sections 8.2.5 and 9.8.1). That governs where a recall lands, not what happens at startup.

8.4 Function Keys

A **Function Key** is a single-action soft button. Where a Command Key recalls a whole saved snapshot, each Function Key performs **one specific action on the layers you currently have selected** — assign a source, nudge a position, fire a router salvo, trigger a Mix Effect transition. Function Keys share a pane group with Command Keys.

8.4.1 Layout

The Function Keys panel has **no toolbar** — just pages and the key grid.

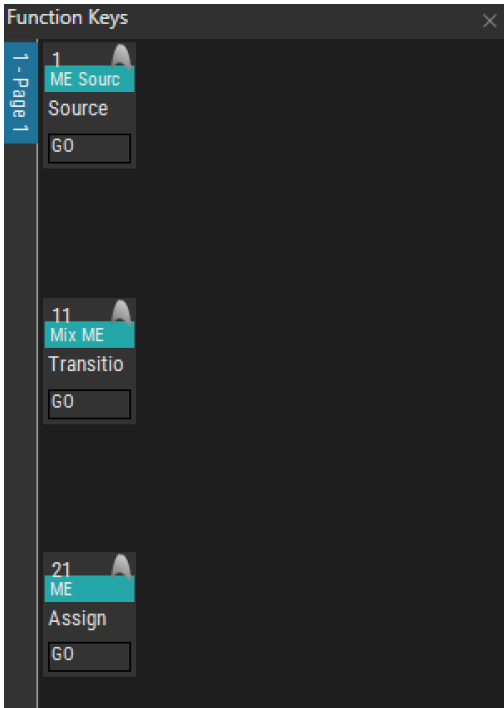


Figure 8-2. The Function Keys panel.

- **Page tabs** down the left edge, with **Add Page**, **Rename Page**, and **Remove Page** on right-click, as in Command Keys.
- A **grid** whose dimensions come from the **Function Key Grid** size in **Tools > Options** (default **5 × 8**, minimum **2 × 2** — Section 4.9). The grid is empty until you add keys.

8.4.2 Running a function key

Each function-key cell shows its number, a teal band naming the key type (for example *Offset* or *Mix ME*), the key's name, and a single **execute button**, labelled **GO**. **Left-click it** to run the key. The action applies to **whatever layers are currently selected** in the Simulator.

- There is **no separate Preview/Program button** — a function key is a live action, not a preview-then-take snapshot.
- **Ctrl-click** a key to select it *without* running it, for drag or rearrange.
- Drag a key to another cell to **move** it. **Ctrl-drag** copies it instead: a **Function Key Name** prompt opens with the original key's name filled in, and the copy is not created until you click **OK** — **Cancel** leaves the grid unchanged. Drag across pages to move between pages. The **Delete** key removes selected keys.

Notice — moving a key changes its external-control ID. A function key's register ID is derived from where the key sits: **(page – 1) × 1000 + (cell – 1)**. A key in page 1 cell 6 is register **5**; drag that same key to page 2 cell 10 and it becomes register **1009**. Moving a key one cell along rennumbers it by one; moving it to the next page shifts it by a thousand. Rearranging the grid will therefore break any external controller that recalls function keys by ID until the controller is updated (Chapter 10).

Tip: Most keys act on the layer selection at execution time. Select layers first, then press the key.

8.4.3 Creating a function key

To create a function key:

Prerequisite: The layers the key should act on can be selected now or at run time; the hardware or config must support the key type (see gating below).

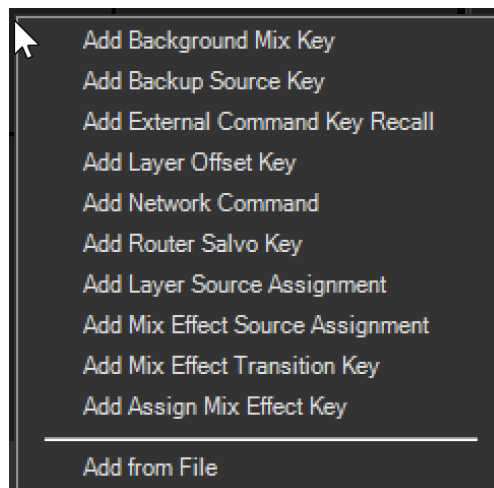


Figure 8-3. The function-key create menu — right-click an empty cell for the "Add ... Key" list.

1. Right-click an empty cell. A menu of "Add ... Key" types appears.
2. Pick a key type.

3. Enter the key's **name** when prompted (default "New Key").

The new key is created in that cell and its properties open in the **Properties** panel. For a Layer Offset key those are **Position Offset Options** (Vertical Offset, Horizontal Offset, Duration) and **Absolute / Relative**, which also carries the per-layer target list. Both sections start **collapsed** — expand them to reach the values. A new Layer Offset key starts as **Absolute**. Choosing **Relative** greys the layer list out, because a relative key acts on the selection at run time.

The available key types:

Item	What it does
Add Background Mix Key	mix/transition a background on a PixelSpace
Add Backup Source Key	switch a layer to a pre-set backup source (failover)
Add External Command Key Recall	recall a Command Key / script by reference
Add Layer Offset Key	apply a position/size offset to the selected layers, as absolute values or relative offsets
Add Network Command	send an external/network command string to another device
Add Router Salvo Key	fire a router salvo (a set of router crosspoints at once)
Add Layer Source Assignment	assign a chosen source to the selected layer(s)
Add Mix Effect Source Assignment	assign a source to a Mix Effect bus
Add Mix Effect Transition Key	trigger a Mix Effect transition (take/cut)
Add Assign Mix Effect Key	assign/load a Mix Effect
Add from File	load a previously saved function key from a function-key file

8.4.4 Why the create menu varies

The create menu is **context-sensitive**. Items appear only when the hardware and configuration support them:

- **Mix Effect** keys (Source Assignment, Transition, Assign) — shown only when **at least one Mix Effect exists**.
- **Add Background Mix Key** — shown only when the Frame has the **BackgroundBuffer** capability *or* a background Mix Effect exists.

Note: These key types are not available in this build, regardless of hardware: *Add Layer Replace Key*, *Add Layer Freeze Key*, *Add Device Freeze Key*, *Add Select Output Aux*, and *Add Direct Aux Preset*.

8.4.5 The function-key context menu

Right-click a populated cell for key-management actions:

Item	Action
Make Absolute / Make Relative	toggle absolute vs. relative values — shown only for keys that support relative (for example, Layer Offset); the label reflects the current state, and so does the tile's corner icon — a grey arch when absolute, a blue panel when relative
Cut / Copy / Paste	clipboard operations (Paste enabled only when a key has been copied)
Delete	remove the key
Rename	rename the key (inline edit)
Save to file	export the selected key(s) to a function-key file (re-import with Add from File)

Multi-select supports Cut, Copy, Delete, and Save to file across several keys at once.

Tip: Exporting keys to file and re-importing them with **Add from File** moves looks between configurations. Function-key registers can also be auto-mapped via **Tools ▶ Repair Data List ▶ Function Keys**.

9. Scripts

Spyder Studio stores and replays looks as **scripts**. A script is an ordered list of cues, each cue carrying the content, layers, and keyframes that define a moment of the show. The **Script** panel is where scripts are built, edited, and run. Every Command Key (Chapter 8) is stored as a script, so the same panel shows and edits a key's script; it can also be used on its own, with no key involved.

The concepts behind layers, Mix Effects, and the Program/Preview rails are established in Chapter 5; live show operation is in Chapter 7.

9.1 The Script panel

The **Script** panel is Spyder's cue-based timeline and sequence editor. A script is an ordered list of **cues** (steps). On the grid you place **elements** — blocks that say *what content plays on which layers across which cues* — then run the script cue by cue (preview → take) or let it advance on triggers or a clock. The panel docks along the bottom of the workspace, as the **Script** tab beside **Mix Effects**.

The script shown in the panel is the **view script, Script ID -1** — the live working timeline. Recalling or selecting a saved script loads it into the panel instead, and the header then shows that script's own ID until the layers are released (for example by **Disconnect Layers**, Section 9.8.2). Command Keys (Chapter 8) are saved scripts; this panel is where the live/scratch script is edited and run, and where a key's script is shown and edited (Section 9.4).

9.1.1 Grid layout

The grid is cues against layers:

Axis	Holds	In the UI
Rows = cues	the sequence steps (cue 0, 1, 2 ...)	down the left column
Columns = layers	the system layers	across the top row
Cell (0,0)	script info — Script ID, Clock Source , and a timecode	top-left

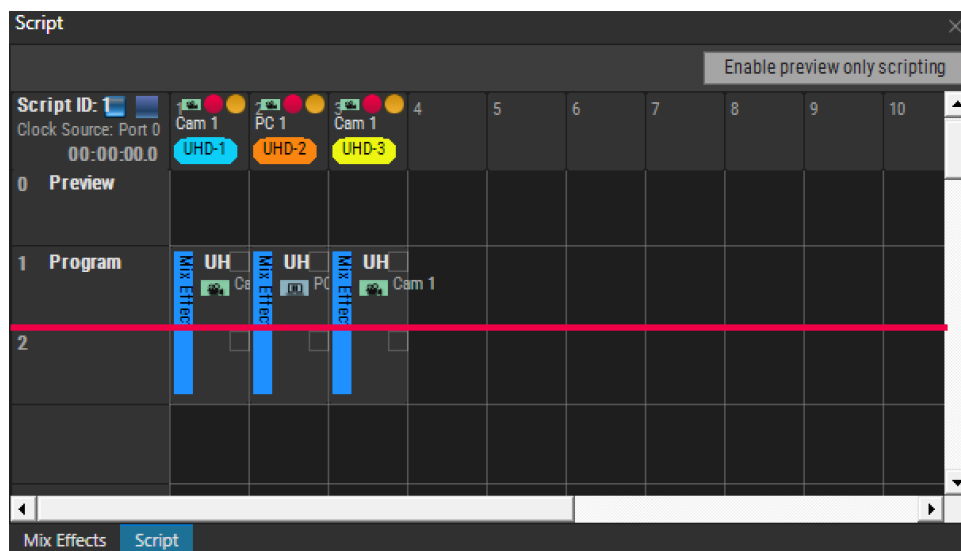


Figure 9-1. The Script panel grid — cues (rows) against layers (columns).

- **Top row (layer headers):** each shows that layer's live content (for example, *Cam 1 / UHD-1*). A red dot marks content on **Program**; yellow marks **Preview**.
- **Left column (cue headers):** the cue numbers, plus the **execution indicator lines** — **Program** (red) and **Preview** (yellow) mark where each currently sits in the cue list.

9.2 Element types

Each block on the grid is an **element**:

Element	What it is
Source	a source on a layer (with keyframes)
Mixer	a mix of multiple contents on a layer (cross-points between sources)
Mix Effect	a Mix-Effect-driven element (auto-created in Mix-Effects layer mode)
Input Array	a layer fed from an array of inputs
Off	turns a layer off at that cue (Section 9.3.4)

An element occupies a rectangle of **cues (height) × layers (width)** and can carry **keyframes** at its cues. Each keyframe holds the layer's look at that cue; when the script reaches the cue, the layer animates to it (Section 9.3.5).

Tip: **Off** elements are how you script a layer **clearing** at a given cue.

9.3 Building and editing a script

9.3.1 Adding elements

Build a script by dragging content from the Sources or Treatments lists onto a grid cell:

Drag	Result
Source	creates/sets a Source element (2 cues). In Mix-Effects layer mode it auto-converts to a Mix Effect element; Ctrl-drag forces a direct layer element. Dropping on an existing element replaces its content.
Treatment	applies the treatment to the element's keyframe at the drop cue.

To add an **Off** element (Section 9.3.4), right-click an empty cell and choose **Add Off Element**. It is placed in the cell you right-clicked, one cue high.

9.3.2 How long an element plays

An element's layer comes on at the element's first cue and goes off at its last cue. An element that spans cues 1–2 is on air at cue 1 and goes off at cue 2; one that spans cues 1–3 stays on through cue 2 and goes off at cue 3.

- **Resize an element:** drag its **bottom edge** to span more or fewer cues. The pointer changes to an up-down arrow when it is over the edge; start the drag from there. An element cannot be made shorter than 2 cues; an Off element is one cue high (Section 9.3.4).
- Dragging an element's top edge moves the whole element instead of resizing it (Section 9.3.3).

9.3.3 Moving and copying elements

- **Move / copy:** drag an element to a new cell — a different layer, a different cue, or both (**Ctrl** = copy).
- Dragging up or down changes the cue at which the element plays; dragging left or right moves it to another layer. Off elements move the same way.
- A move onto cells that another element already occupies is refused: the element stays where it was, and no message appears.

9.3.4 Off elements

An **Off** element is a one-cue black block that takes its layer off at that cue.

Why use one: a **relative** Command Key leaves layers that are not in the key as they are (Section 8.2.2), so leaving a layer out of the key does not take it off. To take that layer off, put an Off element on it at the cue where it should go off.

Selecting an Off element fills the Properties panel with a **Script Off Element** section:

Field	Default
Override global settings	Checkbox, unticked.

Field	Default
Off Duration	60

To widen an Off element across neighbouring layers, drag its **right edge**.

To take one layer off with a relative key:

Prerequisite: The layers are on Program, and the Command Key is **Relative** (Section 8.2.2).

1. Right-click an empty Command Key cell, choose **Add Empty**, and name the key (Section 8.2.3).
2. Click the key's tile to show its script in the Script panel (Section 9.4).
3. Right-click the layer you want to take off at cue **1** and choose **Add Off Element** (Figure 9-2).
4. Click the key's **PGM** button. That layer goes off; the other layers stay on Program.

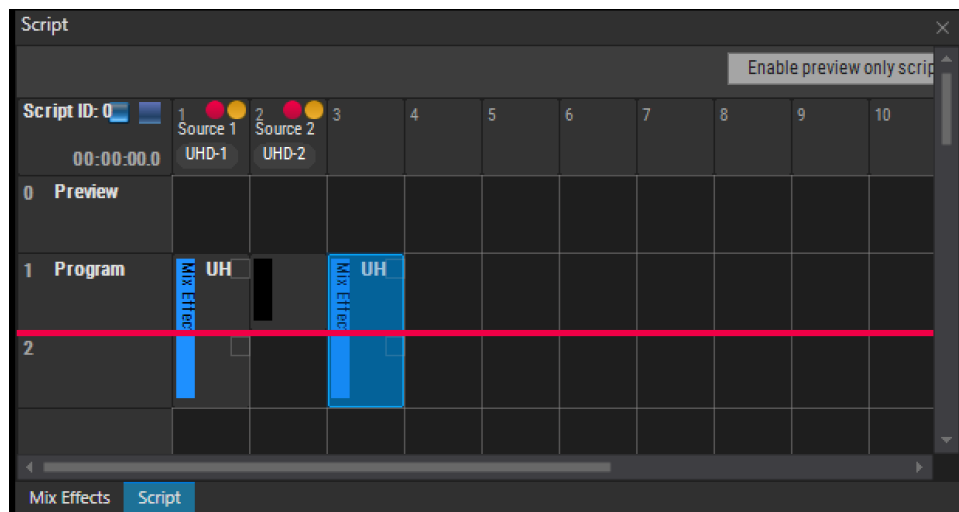


Figure 9-2. A script with Mix Effect elements on layers 1 and 3 (cues 1–2) and an Off element on layer 2 at cue 1, between them.

9.3.5 Keyframes, deleting, and undo

- **Add or remove a keyframe:** right-click the element at the cue you want and choose **Add Keyframe** or **Remove Keyframe** (Section 9.6). Keyframes are not dragged. To add one to every element at a cue at once, right-click that **cue header** and choose **Add Keyframes**.

To change a layer partway through an element — for example, move or resize it at a later cue:

1. Right-click the element at the cue where the change should happen and choose **Add Keyframe**.
2. Run the script to that cue (double-click the cue).
3. With **Live Updates Enabled**, move, resize, or restyle the layer. The change is saved to that keyframe.
4. Run the earlier cue, then this cue. The layer animates to the new look over the keyframe's **Duration** (Section 7.2.2).

Note: If the cue has no keyframe, an adjustment made at that cue is saved to the element's previous keyframe instead, which changes the earlier cue's look.

- **Delete** removes selected elements or cues; **Insert** inserts a cue.

- **Undo / Redo:** **Ctrl+Z** / **Ctrl+Y** (per-script undo stack). You can also right-click an empty cell and choose **Undo** or **Redo**.

Convert an element's type in place from the element context menu (Section 9.6): **Convert To** ▶ **Mixer** • **Source** • **Mix Effect** • **Input Array**. Only valid conversions for that content and hardware are shown — for example, **Source** → **Mixer** to add a second content.

9.4 A Command Key's script

Every Command Key (Chapter 8) is stored as a script, and the Script panel is where you see and change it.

To open a key's script: click the key's tile (not its **PGM** or **PVW** button). The panel header shows the key's **Script ID** and the grid shows its elements. Selecting a key does not recall it. A key learned with **Add From Program** has its elements starting at cue **1**, the Program cue; cue **0**, the Preview cue, is empty.

Editing a key's script: add or change elements as in Section 9.3 — for example, drag a source onto a cell. The change is saved in the key but does not reach the canvas until you recall the key, for example with its **PGM** button.

Keeping a Script-panel look as a key: run the script so the look is on Program (Section 9.5), then learn it with **Add From Program** (Section 8.2.3). The new key's script holds the same elements.

Multi keys: selecting a multi key shows a script built from the keys in its **Command Key List** (Section 8.2.3), under a negative Script ID (for example **-1002**).

Notice — Disconnect Layers replaces Script ID -1. Script ID -1 is the panel's own working script, and changes made there are not saved to any key. After a Command Key has been recalled, **Disconnect Layers** (Section 9.8.2) returns the panel to Script ID -1 and fills it with a copy of that key's script, replacing whatever Script ID -1 held — anything built in Script ID -1 before then is lost.

9.5 Running a script, keyframes, and triggers

Action	Effect
Space	run the view script
Double-click a cue (left column)	run to that cue , triggers disabled — preview/cue a specific step
Execution line	the red (Program) / yellow (Preview) indicator tracks the current cue as the script runs

A cue is a *step*, not a fixed time — it advances manually (Space, double-click) or by **trigger**.

Cue properties. Click a cue header and the Properties panel fills with a **Cue Data** section: a **Name** field for the cue, and four collapsible groups.

Group	Contents
Jump Options	Type — None or Direct — and, once Direct is chosen, the target Cue number. A cue with a jump shows a jump icon and GOTO: n in its header.

Group	Contents
Trigger	Type — None , Wait , WaitX , Time or TimeOfDay — and a Time field that stays greyed out until the type needs one. TimeOfDay turns the Time field into a clock-time picker and adds Days Of Week checkboxes (S M T W T F S). Set Time and the cue header shows a clock icon and the timecode.
Function Keys	Function Key 1 , a dropdown of the project's function keys listed as <i>ID - name</i> , so a cue can fire a function key as it runs. Clear Function Keys empties the list.
Play Items	Not supported in Spyder Studio 6.1.3 — leave it at its default. It holds a Clear Play Items link, a PlayItem 1 dropdown (default <None>), and a Command dropdown (Cue , greyed out while PlayItem 1 is <None>).

The **Cue Data** heading carries the icon of whichever trigger type the cue is set to — a red *stop* sign for **None** — so you can see a cue's trigger without opening the group.

Enable preview only scripting. The button at the top-right of the panel toggles preview-only scripting. When on, the panel **border turns yellow**, the message strip along the top of the panel reads *Preview only scripting mode is enabled*, and the button itself changes to **Disable preview only scripting** — so the mode is readable three ways without hunting for it. Script recalls then affect **Preview only**, the same mode as the Command Keys toolbar's **Preview Only Scripting Mode** (Section 8.2.5), which turns yellow as well. The execution line in the grid also turns from red to yellow while the mode is on.

9.6 Script panel context menus

Right-click in the Script panel to open a menu for the place you click: an empty cell, an element, a cue header (the cue number and name in the left column), or a layer header (the layer number and source at the top of a column).

Empty cell

Menu item	What it does
Add Off Element	Places an Off element in that cell (Section 9.3.4).
Undo / Redo	Undoes or redoes the last change to the script.

Element

Menu item	What it does
Disable Element / Enable Element	A disabled element is shown greyed out and is skipped when the script runs, so its layer is left as it is. Enable Element turns it back on.
Delete Element	Removes the element from the script.
Add Keyframe / Remove Keyframe	Adds or removes a keyframe on the element at the cue you right-clicked. A keyframe shows as a small gold key.

Menu item	What it does
Remove Source	On Mix Effect and Mixer elements. Removes a source that was dropped onto a Mix Effect element at that cue. Greyed out when there is none.
Convert To Source · Convert To Mixer · Convert To Mix Effect · Convert To Input Array	Changes the element to another type (Section 9.2). Only the conversions that are possible for that element are listed. A Mixer or Input Array also uses the layer to the right, so those two are offered only when that layer is free.

Cue header

Menu item	What it does
Remove Jump	Removes the cue's jump (the jump icon and GOTO: n in the header). Greyed out when the cue has no jump. Jumps are set in the cue's Cue Data (Section 9.5).
Insert Cue	Inserts a new, empty cue at that position. That cue and the cues below it, with their elements, move down one.
Remove Cue	Removes the cue. The cues below it move up one.
Add Keyframes	Adds a keyframe to every element at that cue.

Layer header

Menu item	What it does
Always Relative	When ticked, an Absolute Command Key leaves this layer on even if the key's script does not use it. (An Absolute key otherwise takes such layers off; Section 8.2.2.) The layer header shows an icon while it is ticked.
Lock / Unlock	Lock shows a padlock on the layer header and on the layer in the Simulator, and the layer can no longer be selected on the canvas. Unlock removes the lock.

9.7 Slave layers (Discreet mode)

In **Discreet** Layer Mode (Section 5.9.2), a layer is displayed only within its own Frame group. To let a layer span another Frame group, give it a slave layer in that group. Slave layers are set per script element, in the element's properties.

To add slave layers manually:

1. Build the look on **Program** — for example, drag a source onto the canvas.
2. Right-click an empty cell in the Command Keys panel and choose **Add From Program**. Name the key in the **Command Key Name** dialog and click **OK**.
3. In the **Script** panel, click the key's element to show its properties.

4. In **Properties** ▶ **Slave Layers**, tick a slave layer in each Frame group you want the layer to reach. The list does not include the element's own layer.
5. Recall the key with its **PGM** button, so it reloads with its slave layers.

To add slave layers automatically: right-click the command key and choose **Build Slave Layers**. Spyder goes through every element in the key's script and gives each one a slave layer in every Frame group that its layer touches in any keyframe, or in the transition between keyframes.

What you see: the slave layers appear connected to the PixelSpace in the **System Patch**. The Simulator shows only the primary layer while **Hide Slave Layers** is ticked (**Tools** ▶ **Options** ▶ **Simulator**, Section 4.9); untick it to see the slave layers drawn as well. Moving the primary layer, or any other keyframe change to it, is applied automatically to its slave layers.

To clear slave layers: untick them in the element's **Slave Layers** list, or right-click the command key and choose **Clear Slave Layers** to clear the slave layers from every element in its script. A slave layer still on the canvas can then be removed with right-click ▶ **Remove Selected (R)**.

Tip: Build one or more **template** command keys with the slave layers you need. To make a new key, copy the template, remove the elements you don't need, recall it, and adjust the keyframes — changes save automatically to the active command key. This saves re-creating slave layers for each key and keeps the same slave layers allocated consistently across keys.

9.8 Live control behavior

Two Simulator controls govern how scripts and Command Keys interact with what is on air. Both live in the control row below the Simulator toolbar; full Simulator operation is in Chapter 7. Understanding them prevents the two most common scripting surprises — recalls that unexpectedly hit Program, and layers that no longer respond to a script.

9.8.1 Live Updates Enabled

Live Updates Enabled is a toggle in the Simulator's control row. Clicking it switches the label between **Live Updates Enabled** and **Live Updates Disabled** (shown with a padlock icon), so the current state is always readable on the button itself.

Notice — turning Live Updates off does not protect Program. Recalling a Command Key to Program with Live Updates **disabled** still applies that recall to Program: the layer changes on the frame exactly as it does with Live Updates enabled. Do not rely on this toggle to stage a look safely while on air.

To stage a recall without disturbing Program, use Preview Only Scripting Mode (Section 8.2.5). With that mode on, the panel border turns yellow and the same recall leaves Program untouched and lands on **Preview** instead.

9.8.2 Disconnect Layers

Disconnect Layers releases the layers from script and Command-Key control. It stops all running scripts and clears each layer's association with the script that was driving it, so the layers become freely and manually controllable again.

Use it when a layer is "stuck" under a script or Command Key and you need to take direct manual control. The Script panel's **Script ID** returns to **-1** — the live view script — which is how you confirm the layers have been released. Script playback will not drive those layers again until a fresh recall reconnects them.

Note: Disconnect Layers also temporarily suspends live updates until the next recall. While that is in effect, the **Disconnect Layers** button and the Command Keys toolbar's **Disconnect** button both stay highlighted; the highlight clears when the next Command Key is recalled. The same action is available from the Command Keys toolbar's **Disconnect** button (Section 8.2.5).

10. External Control & API

Spyder Studio exposes a text command API for automation controllers, show-control systems, and custom integrations. Send a short token-first string — `METR 1 D60` — and the server executes the same action a live operator would trigger from the GUI. This chapter is the developer reference for that API: the command model, how a controller connects, the full registered command set, and the Mix Effect tokens used most in live production.

This is the legacy Spyder (Vista Systems) string-command API carried forward onto the Spyder-S (E4K-BBC) hardware. The model, terms, and resource rules it drives are defined in Chapter 5; this chapter does not redefine them. If a token manipulates layers, Mix Effects, or backgrounds, the meaning of those objects is in Sections 5.3, 5.4, and 5.5.

Note: The same command tokens drive in-app **command keys** and **scripts** (Chapters 8 and 9). A command key is a saved snapshot built from these same operations; a script is a sequence of them. If you are building looks inside the application rather than driving it from outside, read Chapters 8 and 9 — the token vocabulary is identical, only the trigger differs.

10.1 The external-control model

External control is handled by a single command processor. Spyder-S 6.1.3 registers **111 command tokens**, of which **100 are supported for Spyder-S operation**. Section 10.3 documents them, together with the legacy tokens a controller carried over from an earlier Spyder system may still send.

A small number of commands carried over from earlier Spyder systems are not supported on Spyder-S. Most are still present in 6.1.3 and accept a command rather than rejecting it, so a controller can appear to succeed even though the feature is unavailable; only OCS and OCU are absent and return result code 2. They are listed in Section 10.3.14 so that a controller written for a legacy Spyder can be identified and updated.

Each command is a **space-delimited ASCII string**. The processor reads it like an API call:

1. Split on the first space; the leading word is the **token** (command name), upper-cased before lookup. Tokens are matched case-insensitively.
2. Look the token up in the handler table. Unknown token → reply 2 (Header error).
3. The handler re-splits the full string on spaces to read its arguments, validates argument count and values, and executes.

A command therefore has the shape:

```
TOKEN arg1 arg2 ... argN
```

Arguments are positional and space-separated. Up to 100 tokens are parsed per command string.

Command-format notation. Christie's official *Spyder-S API Commands v6.1.3* guide ("Communicating with Spyder") documents commands using a consistent notation, which this chapter follows:

- Required parameters are enclosed in angle brackets — `<Required Argument>`.
- Optional parameters are enclosed in square brackets — `[Optional Argument]`.
- Spaces separate the command name and all arguments.

So a command signature reads `Command Name <Required Arguments> [Optional Arguments]`.

Command considerations (official). Two rules from the official guide govern argument construction:

- **Internal layer IDs start at 2.** Layer IDs **0 and 1 are reserved** for the system's two background layers (the "Layer Zero" background concept of Chapter 5; see Section 5.5), so the first real operator layer is internally index **2**. Layer-addressing commands expect this **internal** index — not the operator's 1-based UI number (see Section 10.1.2, where the convention is verified command by command against hardware).
- **Encode spaces inside string arguments.** Because the space is the argument delimiter, any value that itself contains spaces — a source name, a register name — is URL-style encoded: replace each literal space with the three-character ASCII string %20. A source named Camera 1 is passed as Camera%201. The server decodes %20 back to a space on input, and encodes spaces the same way in its responses.

10.1.1 Replies and result codes

A response is returned for **every** command sent to the Spyder system. The first response argument is **always the error (result) code**, which tells the caller whether the command was processed; for query commands the requested data follows it, space-delimited. If multiple values are returned in one response, the delimiter is an ASCII space and any value containing a space is encoded as %20.

This is the official result-code table from Christie's *Spyder-S API Commands v6.1.3* guide ("Communicating with Spyder"). It applies to every command's <Result Code> response:

Code	Name	Description (official)
0	Success	The command was successfully processed.
1	Empty	The data requested is not available.
2	Header	An invalid command was specified.
3	Argument count	The command is missing the required minimum number of arguments.
4	Argument value	One or more arguments of the command were invalid.
5	Execution	An error occurred while processing the command. For details, check the Alert Viewer in Spyder Studio.
6	Checksum	Reserved.

So a successful action with no return data replies 0. A query replies 0 followed by its fields. An error replies the bare error code.

Note (code 7 — continuation). The official guide stops at code 6 (Checksum, Reserved). The 6.1.3 processor defines one additional code beyond the published set — 7 **SuccessWithContinuation** — used when a query reply is too long for a single message: the first chunk is returned prefixed 7 <id> ... and the remainder is retrieved with **RCM** (see below). This code does not appear in the official table; treat 0–6 as the documented wire contract and 7 as an implementation detail of long-reply chunking.

Note: A reply of 0/Success means the command was *accepted and dispatched* — not that the downstream hardware confirmed the result. A few handlers accept arguments the Spyder-S hardware does not act on (Section 10.3); they still reply 0. Treat Success as "the server took the command," not "the pixel changed."

Notice — some commands cannot be undone by sending them again. Two verified cases:

- **TRN** does not toggle. Issuing the same transition a second time does **not** return the layer to its previous bus — the layer leaves both Program and Preview. To restore it, re-establish the layer's content explicitly.
- **OCF** does not fully reverse. It rewrites blanking (HTotal, VTOTAL, front porches, sync widths) and BitDepth alongside the arguments you name. Re-issuing it with the original refresh rate restores RefreshRate only; the other fields stay where the first call left them. Restore them individually with OSP.

Read state back before assuming a corrective command worked.

Long replies (result code 7). When a query's response exceeds the channel's maximum message length, the server returns the first chunk prefixed 7 <id> ... and buffers the remainder under <id>. Retrieve the next chunk with **RCM** <id>, which returns it and frees the buffer. Repeat until the response is complete. This is the real behavior of **RCM** ("Request Continuation Message") — it is *not* an image-capture command, contrary to older documentation.

10.1.2 Layer ID convention

Layer IDs **0 and 1 are reserved** for the system's two background layers (the "Layer Zero" concept of Section 5.5), so the first operator layer — the one the Simulator labels **01** — is internally index **2**.

Supply the internal index. Commands that address a layer expect the internal index, which is one higher than the layer number shown in Spyder Studio:

Command	UI number (1)	Internal index (2)
MELA	code 0 — no effect	applied
KPS	code 0 — no effect	applied
KSZ	code 5 (Execution)	applied
RLS	—	returns UI layer 01's source
TRN	—	index 5 transitions UI layer 04

KSZ 2048 3 4 was confirmed to resize UI layers **02 and 03** (internal 3 and 4) while leaving 04 and 05 untouched, so the multi-layer form indexes the same way.

Why the convention differs from older documentation. The server applies an internal *layer-ID offset* whose value is not fixed. In the 6.1.3 command processor, LayerIdOffset returns **−2 when UseLegacyApi is enabled and 0 when it is disabled**; handlers add that offset to the layer ID you supply. On 6.1.3 the offset resolves to **0**, so the ID you send is used unchanged and must therefore be the internal index. This was confirmed with **Advanced ▶ API ▶ UseLegacyApi** ticked in Global Parameters: layer **05** answered RLS 6, and RLS 8 returned Empty. Do not infer the offset from that checkbox — measure it. Documentation that tells you to send the operator's UI number describes the offset −2 case.

This means the correct layer numbering is a property of the server's configuration, not of the command. Verify it on the system you are integrating with rather than assuming: send a known command against a known layer and read the result back.

Notice — a wrong layer index does not reliably produce an error. MELA and KPS return result code **0 (Success)** when handed a layer index that does not exist, having done nothing. KPS returns code **5** for the same mistake. The error contract is not uniform, so a controller cannot treat Success as confirmation that the addressed layer was real. Read the value back (RLS, RLK, or KGP) when it matters.

To map a UI layer to its internal index, query RLS upward from 2: a populated layer replies code 0 with <sourceName> <sourceIndex>; an empty one replies code **1 (Empty)**.

Output IDs are also zero-based. An output is numbered from 0 in a command, one lower than the number Spyder Studio shows: the output Studio lists as **16** is output **15** on the wire. Confirm with OGP <id> — the reply carries the output's name (for example Out-4D), which is the reliable way to check you have addressed the right one.

10.2 Connection and transport

A controller opens a connection to the Spyder-S server, sends a command string, and reads the reply string. The processor is transport-neutral: the same parser serves both a byte stream (ProcessChars) and a whole-message call (ProcessCommand), so the command syntax is identical regardless of how bytes arrive.

Two framing styles are supported:

- **Network (message-framed):** the controller sends one command string per message and reads back one reply string. This is the path used for automation and for the bench tools below.
- **Serial (character-framed):** bytes accumulate until a carriage return (\r); backspace (0x08) edits the buffer, ESC clears it, and an empty \r returns a 0> prompt. Serial ports are enumerated as COM1–COM3. The serial protocol option also enables a checksum (result code 6).

The protocol family is selectable in the server (ExternalControlProtocol: Odetics, **StringCommands**, CricketRS, VDCP). The command set in this chapter is the **StringCommands** protocol.

Transport and port (official). Connectivity to the Spyder Frame is over an **RJ-45 (Ethernet)** connection. Per Christie's *Spyder-S API Commands v6.1.3* guide ("Communicating with Spyder"), to control the Frame you **send each command in a UDP packet to port 11116 on the Frame; no configuration is required.** The Frame listens on that **UDP control port, default 11116**, and replies to each datagram it receives there. A separate **UDP event port, default 11118**, carries the Frame's outbound event broadcasts. Both ports are configurable in the Frame configuration.

The 10-byte message header (required). Every UDP message sent to Spyder must be **preceded by a 10-byte header**. Bytes 0–5 are the ASCII characters of the string spyder; bytes 6–9 are four null bytes (0x00). The command string follows the header **immediately** — do **not** put an argument delimiter (space) between the header and the command.

Index	Byte
0	s
1	p
2	y
3	d
4	e
5	r
6	0x00
7	0x00

Index	Byte
8	0x00
9	0x00

So the on-wire payload of METR 1 D60 is the 10 header bytes 73 70 79 64 65 72 00 00 00 00 followed directly by the ASCII command text. The in-app **UDP Protocol Tester** (below) prepends this header for you; a custom controller must add it explicitly.

Note: This UDP control port (11116) is **distinct from the Studio remoting channel**, which is the **NewRemoting TCP marshallng port, default 11112** (MarshallngPort). Remoting is how the Spyder Studio client drives the Frame; it is not the external string-command interface. The Frame configuration also defines a TcpControlPort (default 11119), but in 6.1.3 the StringCommands processor is wired only to the UDP control port — drive external control over **UDP 11116**.

The in-app **UDP Protocol Tester** (Section 11.2.1) is the recommended way to probe commands before wiring them into an external controller: it reports the decoded result code for each command you send, so you can confirm the result-code mapping in Section 10.1.1 against your own commands.

Tip: Develop against the UDP Protocol Tester first. It shows the decoded result code by name (Success / ArgValue / ...), so you can confirm a command's syntax and target IDs interactively before committing it to controller logic.

Server ▶ Global Parameters. The Frame-level integration and automation settings live in **Server ▶ Global Parameters**, which — like **Connection Manager** — opens in the **Properties** panel rather than as a dialog. Its sections:

Section	Settings
Global Settings	Server Name , Sync Source / Sync Type , Switch Known Crosspoints , Temperature Display units.
Layer Provisioning	The Layer Mode for multi-Frame-group systems (Section 5.9.2).
Scripting Engine	Element defaults — Speed / Motion Transform and, when enabled, global Entrance / Exit Duration — with a Set Defaults link.
Startup Options	Load Backgrounds and the startup Command Key / Cue ID (Section 8.3).
SourceMaster / SR-208	The SR devices' Master IP / Slave IP addresses, each with a Set Defaults action.
Advanced	The API ▶ UseLegacyApi switch, which turns the external-control string-command API — the interface this chapter documents — on or off.

Edits are committed with **Apply Settings Now** or discarded with **Cancel Changes**, both at the top of the panel.

Note: If a Frame stops responding to string commands, confirm **Advanced ▶ API ▶ UseLegacyApi** is enabled — with it off, the external-control API in this chapter is disabled. Changing **UseLegacyApi** does not take effect straight away: **Apply Settings Now** opens a prompt saying the change needs a frame restart and offering to restart the remote frame now, which disconnects Spyder Studio.

Apply Settings

☒

 Apply Settings Now

☐

 Cancel Changes

Global Settings

Name:

GPPL0515

Sync Source:

Freerun

Sync Type:

Black Burst

☒

 Switch Known Crosspoints

Temperature Display:

Celsius

Layer Provisioning

In systems with multiple frame groups, Spyder can optimize layer provisioning to optimize for specific use cases.

Layer Mode:

Global Linking

 The number of layers available is based on the frame group with the fewest layers, however all layers are permanently linked to one layer in each frame group.

Scripting Engine

[Set Defaults](#)

Speed Transform:

Bezier

Motion Transform:

Hermite

☐

 Enable element's global settings

Entrance Duration:

60

Exit Duration:

60

Startup Options

☒ Load Backgrounds

Load Command Key:

<None>

Cue ID:

0

SourceMaster

⌵

SR-208

⌵

Advanced

⌵

Figure 10-1. Server ▶ Global Parameters — Apply Settings, Global Settings, Layer Provisioning, Scripting Engine, and Startup Options, with the SourceMaster, SR-208, and Advanced groups collapsed beneath.

10.3 Command reference

The table below lists the registered command set — **108 of the 111 tokens** in 6.1.3, grouped by function; the three omitted tokens are itemized under **Coverage** at the end of this section. Each token and command name is the one the Frame itself registers. Argument signatures were verified against the command processor shipped in 6.1.3 and, where the effect is observable in the application, confirmed in Spyder Studio; for the commands central to live operation (the Mix Effect tokens) see the worked examples in Section 10.4.

Hardware-support legend (as supported by the E4K-BBC hardware layer):

- **OK** — functional on Spyder-S.
- **GATED** — functional only when the target output is in the required mode (OpMon / Multiviewer / Tiled), else returns an arg-error.
- **PART** — partially implemented; some arguments are accepted but not acted on.
- **STUB** — legacy Spyder X-series feature the Spyder-S hardware does not implement; accepted but a no-op (correctly absent from Christie's published doc).

10.3.1 Mix Effects

Token	Name / meaning	Argument signature	Notes
MESR	Mix Effect Source (Top/Bottom)	MESR <meId> <T B> <sourceName>	OK. Assigns a source to the Top or Bottom input. Source name is one token (%20-encode spaces). Exactly 4 fields.
METR	Mix Effect Transition	METR <id> [<id> ...] [<frames>]	OK. Timed mix of one or more MEs; <frames> is optional, default 60, case-insensitive, clamped to ≥1. Cut = METR <id> D1 — no separate cut token. See Section 10.4.
MELA	Mix Effect Assign to Layer(s)	MELA <meId> <layer> [<layer> ...]	OK. Routes an ME onto layers on the Program bus. Takes the internal layer index (Section 10.1.2) — a UI-numbered index returns code 0 and silently does nothing.
MELS	Mix Effect List	MELS	OK. Query. Returns count, then per ME: Id, name, Type, top content, bottom content. A PixelSpace with background groups appears as a Mix Effect of type Background holding the two groups.

10.3.2 Layers – content, transition, geometry

Token	Name / meaning	Argument signature	Notes
SRA	Source Apply to Layer(s)	SRA <sourceName> <layer> [<layer> ...] [R]	OK. The source is identified by name, not by ID — a numeric argument returns code 4. Source name is one token (%20-encode spaces). Internal layer index (Section 10.1.2). Where the target layer's content is a Mix Effect the source is applied to that ME's Top bus and appears on Program immediately, with no transition.
TRN	Transition Layer(s)	TRN <mix> <duration> <layer> [<layer> ...]	OK. <mix> accepts 0–3 (0=off, 1=on, 2=smart per-layer, 3=mix); <duration> in frames. Transitions <i>layers</i> , not MEs — distinct from METR. Internal layer index. Not reversible by re-issuing — see Section 10.1.1.
SWA	Swap Layers	SWA <layerA> <layerB>	OK.
FRZ	Freeze Layer(s)	—	Not supported on Spyder-S. Freeze applies to background Mix Effects only, not to layers (Section 7.4.6). The command is accepted but has no effect.
KSZ	Keyframe Size	KSZ <horzSize> <layer> [<layer> ...]	OK. The first argument is the layer's horizontal size in pixels .
KPS	Keyframe Position	KPS <mode> <x> <y> <layer> ...	OK. Four position modes: 0/1 = absolute pixels (1 also offsets by PixelSpace origin), 2 = set normalized, 3 = increment normalized.
LSP	Layer Size and Position	LSP <position> <horizontal> <vertical> <size> <layer> [<layer> ...]	OK. Sets position and size in one call; <horizontal> is the absolute X in pixels.
KBD	Keyframe Border Adjust	KBD <layer> <thickness> [<red> <green> <blue> [<hBevel> [<vBevel> [<insideSoftness>]]]]	OK. Layer FIRST — verified; the layer-last form returns code 4. Only the layer and thickness are required. The colour is set only when all three of red, green and blue are given. A negative thickness sets the border's outside softness to

Token	Name / meaning	Argument signature	Notes
			that amount instead of its thickness.
KSH	Keyframe Shadow Adjust	KSH <layer> <hOffset> <vOffset> <size> <transparency> <softness>	OK. Layer FIRST — verified. Side effect: sets ShadowIsEnabled to True.
KTP	Keyframe Transparency Adjust	KTP <value> [a r] <layer> ...	OK. <value> is 0–255. The optional second argument chooses how it is applied: a, or no letter, sets the transparency to <value>; any other letter (for example r) adds <value> to the current transparency.
CRP	Crop Adjustment	CRP <left> <right> <top> <bottom> <layer> [<layer> ...]	OK. △ The four crop values are normalised fractions (0–1), not pixels — CRP 0.5 0 0 0 removes the left half. Out-of-range values such as pixel counts are accepted with result code 0 and crop the layer out of sight ; the layer still exists and RLS still reports it. Clear a crop with CRP 0 0 0 0 <layer>.
ZPA	Zoom / Pan Adjust	ZPA <mode> <zoom> <panH> <panV> <layer>	OK. <zoom> is a multiplier (2 = 2×), not a fraction. Scales the content within the layer — the layer's own size on the canvas is unchanged, unlike KSZ/LSP.
LAC	Layer Alignment Control	LAC <effectId> <recallDuration> <layer> [<layer> ...]	OK.
LCC	Layer Clone Control	LCC <layer> <mode> [<offset>] <offsetType>	OK. Layer first, then mode — the reverse of the older published order.
LAP	Layer Assign PixelSpace	LAP <pixelSpace> <visible> <layer> ...	OK. Three arguments minimum — <visible> sits between the PixelSpace and the layer list: 1 shows the layer on its new PixelSpace, 0 assigns it while leaving it hidden. Trailing layer IDs may be repeated to move several layers at once.
AIR	Aspect Insensitive Resize	AIR <relative> <dimension> <size> <layer> [<layer> ...]	OK. Changes one dimension of a layer without changing the other, by adjusting its aspect-ratio offset. <relative>: 1 adds <size>

Token	Name / meaning	Argument signature	Notes
			pixels to the current size, 0 sets the size to <size> pixels. <dimension>: 0 = width, 1 = height. All arguments are numeric; the layer comes last . A layer that does not exist is skipped and the command still returns code 0.
ARO	Aspect Ratio Offset	ARO <type> <value> <layer> ...	OK. <type> is a LETTER, not a number: t (total), o (offset) or a (aspect).
KTL	Keyframe Learn Treatment	KTL <register> <layer> <name>	OK. Stores the layer's treatment into the register under <name> (%20-encode spaces); the register then appears in the Treatments pane. Three arguments — two returns code 3. The register number is the one shown in Spyder Studio's Treatments list; see the register-numbering note below.
KTR	Keyframe Recall Treatment	KTR <register> <layer> ...	OK. Does not apply the layer-ID offset — supply the internal zero-based index (Section 10.1.2).
ARL	Apply Register to Layer	ARL <registerType> <register> <layer> [<layer> ...] [R]	OK. <registerType> is the numeric type code — 0 Effect, 1 Playitem, 4 Command key/script, 5 Treatment, 6 Source, 7 Function key. The optional trailing R forces legacy replace-layer behaviour for Source-type registers instead of routing the source to the Mix Effect's Top bus; it is not a register-type selector. Requires the register to exist — an ID that does not exist returns code 4. The register argument is the raw ID RRL reports; see the register-numbering note below.
LCK	Learn Command Key	LCK <keyId> <name> <arg3> <arg4> <arg5>	OK. Optional 6th arg is read and discarded.
DCK	Delete Command Key	DCK <id> [r]	OK. Append the literal letter r to delete by register ID rather than script ID.

Notice — register numbering is not consistent between KTL and ARL. RRL <type> reports registers with **zero-based** IDs, while Spyder Studio's Treatments list numbers the same registers from **1**. The two commands do not agree on which to use: KTL 2 ... creates the register RRL reports as **ID 1** (Studio's slot 2), whereas ARL <type> 2 ... addresses **ID 2** (Studio's slot 3). A controller that learns a register with KTL n and then recalls it with ARL <type> n will address the **wrong register**. Read the IDs back with RRL <type> and use those values for ARL.

10.3.3 Sources, stills & inputs

Token	Name / meaning	Argument signature	Notes
RSN	Request Source Names	RSN	OK. Query.
RAS	Request Active Sources	RAS [<pixelSpace> ...]	OK. Query. Lists the sources on visible layers, on Program and on Preview. The reply is a count, then for each source its name and a bus value: 1 Program, 0 Preview, 2 both. With PixelSpace IDs given, only layers on those PixelSpaces are listed (each ID also includes its paired Program or Preview PixelSpace). A source on a layer that is not visible is not listed, though RLS still reports it.
DSL	Detailed Source List	DSL	OK. Query.
RLS	Request Layer Source	RLS <layer>	OK. Query. Returns the source name (result code 0) for an assigned layer; a defined-but-empty layer replies result code 1 (Empty) — the handler's internal -1 sentinel surfaces on the wire as Empty, not as a 0 -1 reply. Uses the internal 2-based layer index (see §10.1.2).
ICR	Input Config Recall	ICR <configId> <layer>	OK. 3rd token required by count check but ignored; configId < 0 is a silent no-op.
ICL	Input Config Learn	ICL <input> <layer>	OK.
SIP	Set Input Properties	SIP <input> <property> <value>	PART. Only the ColorSpace property is wired through.
ILA	Input Level Adjust	ILA <input> <gamma> <brightness> <contrast> <saturation>	PART. Gamma works; Brightness/Contrast/Hue/Saturation accepted but not written to hardware.

Token	Name / meaning	Argument signature	Notes
ILK	Input Luminance Key	—	Not supported on Spyder-S. Accepted by the command processor, but Spyder-S provides cut (linear) keying only.
ICK	Input Color Key	—	Not supported on Spyder-S. Accepted by the command processor, but Spyder-S provides cut (linear) keying only.
ISP	InputConfig Set Property	ISP <layer> <prop> <value> [<prop> <value> ...]	OK. Sets properties of the input configuration used by the source on that layer (internal layer index, Section 10.1.2). Requires ≥1 balanced property pair (even arg count).
IGP	InputConfig Get Properties	IGP <layer>	OK. Query. Takes a layer (internal layer index, Section 10.1.2) and returns every property of the input configuration used by the source on that layer — Connector, HActive, VActive, ColorSpace, KeyerMode, Id and related fields. There is no property filter: further arguments are ignored.
SLD	Still Load on Layer	—	Not supported on Spyder-S. Stills are a background feature; they cannot be loaded onto a layer (Section 7.1.4). The command is accepted but has no effect.
SCL	Still Clear on Layer	—	Not supported on Spyder-S. See SLD above.
RAR	Request AspectRatio	RAR <layer>	OK. Query; returns the aspect ratio in the response.

10.3.4 Backgrounds

Backgrounds transition as Mix Effects. A PixelSpace that has two background groups is exposed as a Mix Effect of type Background; MELS reports its ID and the two groups as its Top and Bottom inputs. Transition it with METR <backgroundMeId> exactly as you would any other Mix Effect — sending METR again transitions back.

Token	Name / meaning	Argument signature	Notes
BTR	Background Transition	BTR <pixelSpace> [D<frames>]	OK. Transitions the per-PixelSpace background ME (Section 5.5).
BLD	Background Load Image	BLD <pixelSpace> <arg2> <arg3>	OK. Three arguments — two returns code 3. Christie's guide names the second and third only as a <i>background selector</i> and does not give their values.

Freeze is not reachable from external control. Background freeze is available in Spyder Studio only — there is no command for it. FRZ and OFZ are legacy layer and output tokens, are not supported on Spyder-S, and have no effect (Sections 10.3.2 and 10.3.7); freeze does not apply to layers or outputs at all. See Section 7.4.6.

10.3.5 Keyframe / property get-set

Token	Name / meaning	Argument signature	Notes
KSP	KeyFrame Set Property	KSP <layer> <prop> <value> [<prop> <value> ...]	OK. ≥1 balanced pair.
KGP	KeyFrame Get Properties	KGP <layer>	OK. Query. Returns every keyframe property; there is no property filter, and further arguments are ignored.
RLK	Request Layer Keyframe	RLK <layer> [n]	OK. Query. Optional n (or normalize) re-bases the geometry of a layer on a Preview PixelSpace to its Program PixelSpace; it has no effect on a Program layer.

10.3.6 Scripts & registers

Token	Name / meaning	Argument signature	Notes
RSC	Recall Script Cue	RSC <scriptId> <cue> [idType]	OK.
ASC	Advance Script Cue	ASC [<steps>]	OK. Advances the running script by <steps> cues — it does not select

Token	Name / meaning	Argument signature	Notes
			a script. The argument is optional; a bare ASC is accepted. Verified on 6.1.3: ASC 1 moved the Script pane's playhead on by one cue with the Script ID unchanged. Δ A controller that sends this expecting to choose a script will instead advance what is on air.
SCR	Script Cue Request	SCR <scriptId> [<type>]	OK. Query. <type> optional, defaults to ScriptID.
SRS	Stop Running Scripts	SRS [<opType>]	OK. <opType> optional, defaults to D (disconnect).
RSCC	Request Script CueData Count	RSCC <scriptId> [R]	OK. Query. Returns the number of cues. Add a trailing R to give a command key's register ID instead of a script ID.
RSCD	Request Script CueData Details	RSCD <scriptId> <cueIndex> [R]	OK. Query. Add a trailing R to give a command key's register ID instead of a script ID.
RSEC	Request Script Element Count	RSEC <scriptId> [R]	OK. Query. Add a trailing R to give a command key's register ID instead of a script ID.
RRC	Request Register Count	RRC <type>	OK. Query.
RRL	Request Register List	RRL <type>	OK. Query.
RRD	Request Register Details	RRD <type> <id>	OK. Query. Treatment response emits 10 fields.
RPN	Request Page Names	RPN <registerType>	OK. Query. Only CmdKey and FuncKey register types return data.
RLC	Request Layer Count	RLC	OK. Query. Takes no arguments and returns the system's layer count, which includes the two background layers (IDs 0 and 1).
FKR	Function Key Recall	FKR <key> [layers...] [R F]	OK. ID-type token may appear anywhere in the trailing list.

10.3.7 Outputs

Token	Name / meaning	Argument signature	Notes
OCF	Output Config Format	OCF <output> <hActive> <vActive> <refreshRate> <interlaced> [<timing>]	OK. Five arguments required , with an optional sixth (timing); four returns code 3. <interlaced> is 1 for interlaced. Rewrites blanking and bit depth as well as the named fields; not self-reversing , see Section 10.1.1.
OCM	Output Config Mode	OCM <output> <mode> [args]	OK. <mode> is a NAME, not a number: normal, unused, scaled, opmon, sourcemon, aux, unscaledaux, multiviewer. Not every mode is available on every port — opmon requires an output whose hardware supports it, and returns an arg-error on one that does not. Confirm a port's current mode with OGP, whose reply carries a Mode field.
OCC	Output Config Connection	OCC <output> <arg2> <arg3>	OK. Three arguments. Christie's guide does not name the second and third.
OSP	Output Set Property	OSP <output> <prop> <value> [<prop> <value> ...]	OK. ≥1 balanced pair.
OGP	Output Get Properties	OGP <output>	OK. Query. Returns every output property; there is no property filter. For an OpMon or Multi-Viewer output the reply also carries OsdThemeId and the Full, Halves and Quarters tile group definitions.
OFZ	Output Freeze	—	Not supported on Spyder-S. Freeze applies to background Mix Effects only, not to outputs (Section 7.4.6). The command is accepted but has no effect.
CSO	Clear Still on Output	CSO <output>	OK.
LSO	Load Still on Output	LSO <output> <still>	OK.
OCR	Output Config Rotation	—	Not supported on Spyder-S. Output rotation is not implemented, and

Token	Name / meaning	Argument signature	Notes
			there is no rotation control anywhere in Spyder Studio. The token is still registered, but every call is rejected with code 4 .
OCB	Output Config Blend	—	Not supported on Spyder-S. The token is still registered and its handler still exists, but every call is rejected with code 4 : a Spyder-S output configuration is not blend-capable, so the handler's lookup fails before it reaches any argument you supply. Edge blend is not reachable via the API.

10.3.8 Routers & device commands

Token	Name / meaning	Argument signature	Notes
RCR	Router Crosspoint Recall	RCR <router> [L P] <output> <input> [<output> <input> ...]	OK. Output first, then input. Repeat the output/input pair to switch several crosspoints in one command. An L or P after the router ID is accepted for older controllers and ignored.
QRC	Query Router Crosspoint(s)	QRC <router> [args]	OK. Query.
DMT	Device Mixer Transition	DMT <duration> <device> [<device> ...]	OK. Acts on the software device model (auto-generated layer-pair mixers; see Appendix B) — not an ME command. ⚠ Duration first, then the device(s) — the reverse of the older published order, and the same convention as DMB.
DMB	Device Mixer Bus Select	DMB <duration> <bus> <device> [<device> ...]	OK. Duration first, then the bus, then the device(s). <bus> is 0 Off, 1 Preview, 2 Program; <duration> is in frames. A device is a layer-pair mixer, so it occupies a pair of layers and moving it swaps that pair between the buses. Not an ME command.

10.3.9 PixelSpaces & layout

Token	Name / meaning	Argument signature	Notes
RPD	Request PixelSpace Definitions	RPD	OK. Query.
RPM	Request PixelSpace Mappings	RPM	OK. Query. Emits <PreviewID> <ProgramID> per pair; scale-ratio field hardcoded 1.
PPE	Parallel PixelSpace Expansion	PPE <durationMs>	Takes one argument, the transition duration in milliseconds. Applies a configuration change that has already been staged; with nothing staged to apply it returns an execution error.
SLR	Slide Layout Recall	SLR <pixelSpace> <clearLayers> <reservedLayerCount> [<reservedLayer> ...] [<slideLayoutEntry> ...]	OK. ⚠ No argument-count check: a malformed call returns result code 5, not 3.

10.3.10 Test patterns

Token	Name / meaning	Argument signature	Notes
TPL	Test Pattern Load	TPL <targetType> <id> <pattern>	OK. Three arguments — two returns code 3. <targetType> selects the kind of target (0 = output) and <id> the item within it, so an output is addressed as TPL 0 <output> <pattern>. Places the pattern on the physical output , so nothing changes in Spyder Studio. Clear it with TPC.
TPC	Test Pattern Clear	TPC <targetType> <id>	OK. Two arguments — one returns code 3. Same target convention as TPL. Target type 3 (input-clear) is a no-op.

10.3.11 System, status & hardware queries

Token	Name / meaning	Argument signature	Notes
SAV	Force Server Save	SAV	OK.

Token	Name / meaning	Argument signature	Notes
SDN	Restart Spyder Server	SDN <mode> [0]	OK. <mode>: 1 restarts the Spyder server, 2 restarts the system manager in place without saving, 3 reboots the Frame (embedded systems only; otherwise code 4). Modes 1 and 3 save first unless the optional 0 is added. No power-off path.
RHI	Request Hardware Information	RHI	OK. Query; returns serialized license/hardware signature.
RCM	Request Continuation Message	RCM <id>	OK. Retrieves the next buffered chunk of an over-long reply (Section 10.1.1). One integer arg.
RFI	Request Frame Info	RFI	OK. Query.
RIF	Request Image File	RIF <imageName> [<maxWidthOrHeight>]	OK. Query.
RSF	Request System FrameRate	RSF	OK. Query.
RCS	Request Connection Status	RCS <layer>	PART. Takes a layer (internal layer index, Section 10.1.2). Replies <layer> <connectorType> <status> for the input used by that layer; the status field is always 2 (Unknown) — the command does not poll the connector.
RPS	Request I/O Processor Status	RPS [<chars>]	PART. Always returns 0 <empty> — progress not implemented.

10.3.12 Multiviewer (capability-gated)

These require the target output to be in **OpMon / Multiviewer mode**; otherwise they return an arg-error.

Token	Name / meaning	Argument signature	Notes
MVCA	Multi-Viewer Clear All	MVCA <output>	GATED.
MVAS	Multi-Viewer Add Single	MVAS <output> <viewType> <contentId> <left> <top> <width> <height> [<sectionLeft> <sectionTop>]	GATED. Adds one tile. Seven arguments are required, all numeric; four more may be appended to show a cropped section of the content rather than the

Token	Name / meaning	Argument signature	Notes
		<code><sectionWidth></code> <code><sectionHeight>]</code>	whole of it. <code><viewType></code> selects what the tile shows — 0 Program PixelSpace, 1 Preview PixelSpace, 2 an input, 3 an output — and <code><contentId></code> is read according to that choice: a PixelSpace ID for 0/1 (-1 binds no PixelSpace), an input ID for 2, an output ID for 3. For 2 and 3 the content must live on the same frame as the target output. The addition is also checked against the tiles the output already carries, so a call can be refused on a fully populated output; use MVQ0 first to see what is there.
MVAC	Multi-Viewer Assign Content	MVAC <code><output></code> <code><tileIndex></code> <code><inputId></code>	GATED.
MVKF	Multi-Viewer Set Keyframe properties	MVKF <code><output></code> <code><tile></code> <code><left></code> <code><top></code> <code><width></code> <code><height></code> <code><borderThickness></code> <code>[<red></code> <code><green></code> <code><blue>]</code>	GATED. Sets a tile's geometry and border. The values are applied in the order given — position, then size, then border thickness, then border colour — so to change only the border thickness the position and size still have to be supplied.
MVST	Multi-Viewer Titling	MVST <code><output></code> <code><tileIndex></code> <code><titleMode></code>	GATED.
MVQ0	Multi-Viewer Query Output	MVQ0 <code><output></code>	OK. Query. Returns the tile map — per tile: index, type, X, Y, width, height, border width and border colour. Answers for an output in either Multi-Viewer or OpMon mode; use it to discover the tile layout before assigning content.
MVPL	Multi-Viewer Preset Learn	MVPL <code><output></code> <code><presetName></code>	OK (space-delimited).
MVPR	Multi-Viewer Preset Recall	MVPR <code><output></code> <code><presetType></code> <code>[<presetName>]</code>	OK. <code><presetType></code> : 0 OpMon, 1 SourceMon, 2 Custom. A Custom preset is recalled by its name, which is then required.

10.3.13 OpMon & Tiled output (capability-gated)

Token	Name / meaning	Argument signature	Notes
OMSL	OpMon Set Large Tile	OMSL <output> <startIndex> <sourceIds>	OK on an OpMon output. Output ID first.
OMSS	OpMon Set Small Tile	OMSS <output> <startIndex> <sourceIds>	OK on an OpMon output. Output ID first.
OMSD	OpMon Set Dimensions	OMSD <output> <row> <x> <y> <width> <height> <sourceIds>	GATED.
SRAT	Source Apply to Tile	SRAT <sourceId> <Q H F C> <output> <groupIndex> [D<frames>]	GATED (TiledMode).
SRAG	Source Apply to Tile Group	SRAG <output> <tileSize> <tileIndex> <sourceIdList...> <duration>	OK. TileSize F/H/Q/C; variable-length source list. Output and tile index are both required — fewer than four arguments returns code 3.
ATO	Apply Theme to Output Tile	ATO <output> <MV OM> <themeId>	GATED (Tiled or OpMon mode).

10.3.14 Hardware dumps & stubs

Token	Name / meaning	Argument signature	Notes
RHWC	Request Hardware Config Data Dump	RHWC <F H B>	GATED. Returns payload over MQTT , not the command channel. Every argument form tried returned code 4.
RHWS	Request Hardware Status Data Dump	RHWS <id>	GATED. Payload over MQTT. RHWS 0 0 accepted (code 0). One numeric argument; a letter returns code 4.
ESSC	Execute Special System Command	—	Not supported on Spyder-S. Not implemented.
COI	Capture Output Image	—	Not supported on Spyder-S. Image capture is not implemented on Spyder-S hardware.
CII	Capture Input Image	—	Not supported on Spyder-S. Image capture is not implemented on Spyder-S hardware.

Token	Name / meaning	Argument signature	Notes
CLI	Capture Layer Image	—	Not supported on Spyder-S. Image capture is not implemented on Spyder-S hardware.

Note: The four commands marked **Not supported** above come from earlier Spyder systems and are not supported on Spyder-S. They are still present in 6.1.3 and accept a command rather than rejecting it, so a controller carried over from an earlier system may appear to succeed. Confirm the result on the hardware rather than relying on the reply code.

Notice: Two other groups of legacy commands behave differently and are covered separately below. The Basic Preset commands (BPL, BPR, RBL) do not merely *appear* to succeed — they **work**, and BPR changes what is on Program. The stills-on-layer commands (SLD, SCL) are accepted but genuinely have no effect.

Coverage. Spyder-S 6.1.3 registers **111** command tokens. Sending every one of them to a live 6.1.3 system establishes what a controller can actually reach:

	Count
Tokens that answer (result code ≠ 2)	111 — every registered token
Tokens with no handler (result code 2)	2 — OCS and OCU, retired and no longer present
Of those that answer, accepted but inert	8 — CII CLI COI ESSC ICK ILK OCB OCR
Of those that answer, removed features whose tokens still respond	3 — BPL BPR RBL (Basic Presets — removed from the interface; see below)

SLD (Still Load on Layer) and SCL (Still Clear on Layer) are accepted by the command processor but have no effect on Spyder-S — stills are a background feature and cannot be placed on a layer. They are listed in Section 10.3.3 so that controllers carried over from earlier Spyder systems can be identified and updated.

Basic Presets — BPL, BPR and RBL. Basic Presets was a feature of earlier Spyder systems. **It was removed from Spyder Studio in 6.1.3 and is not available on Spyder-S.** These three tokens are listed here, and deliberately left out of the command tables above, so that a controller written for an earlier system can be checked for them and corrected. They are not commands to build against.

Notice — the removed Basic Preset commands still respond, and one of them changes Program. Unlike the inert tokens above, these were never taken out of the command processor. A controller that still sends BPR **will change what is on air**, and one that sends BPL will silently store state, with nothing shown in Spyder Studio to explain either. Nothing stored this way can be removed: Basic Presets are held outside the register types, so RRL and RRD cannot reach them, and DCK reports success without deleting anything. Audit any controller carried over from an earlier Spyder system and remove these three commands from it before going on air.

Christie's published *Spyder-S API Commands v6.1.3* document covers the **100** commands supported for Spyder-S operation, including the Mix Effect family (MESR, METR, MELA, MELS), and lists the retired and client-only tokens — among them **OCS** and **OCU**, which return code 2 — in its unsupported-commands appendix. Earlier editions of that document covered 72 commands and omitted 28 that work.

Note: Where a signature in this chapter differs from an earlier Spyder document, this chapter describes Spyder-S behaviour. The differences most likely to affect an existing controller are: **RCM** retrieves a continued reply rather than capturing an image; **KPS** provides four position modes; **METR** has no separate cut command — cut with a one-frame duration; **KBD** and **KSH** take the layer as the first argument; and **RCR** takes the output before the input.

10.4 Mix Effect control tokens

The Mix Effect tokens are the workhorses of live external control: they assign sources, route the ME onto layers, and trigger the on-air transition. They map directly onto the Single Layer Mixer model in Section 5.4 — an ME is a two-input compositor whose Top input lands on Program and whose Bottom input lands on Preview when the ME feeds a Program-bus layer. Cut/Mix swaps Top and Bottom.

Four tokens cover the full ME workflow:

Token	Action	Underlying call
MESR <meId> <T B> <sourceName>	Assign a source to the Top or Bottom input	SelectContent → ApplyContent
MELA <meId> <layer> [<layer> ...]	Route the ME onto layers on the Program bus	SelectMixEffect(Program, ...)
METR <id> [<id> ...] [D<frames>]	Transition one or more MEs over <frames> (default 60)	TransitionMixEffect(dur, ids)
MELS	List all MEs (query)	GetMixEffects()

10.4.1 Transitioning — Mix and Cut

METR triggers the cross-fade. The optional D<frames> token sets the duration in frames; if omitted it defaults to **60**. The D prefix is case-insensitive, and a duration of zero or less is clamped up to 1.

A **Mix** is any METR with a multi-frame duration:

```
METR 1 D60 → 0 (Success: ME 1 cross-fades over 60 frames)
```

A **Cut** is a one-frame Mix. There is **no dedicated cut token** — a cut is simply a transition with a duration of one frame. To cut over the API, set the duration to one frame:

```
METR 1 D1 → 0 (Success: ME 1 cuts instantly)
```

Note: The ME ID is **required** over external control — there is no bare METR D1 cut form. METR D1 with no ID is rejected with result code 3 (see Section 10.4.2). The GUI's Cut button acts on whichever ME is selected, but that selection context does not exist on the wire, so an API cut must always name its ME. Each METR **toggles** the ME — sending it again transitions back (Section 5.4).

You can transition several MEs together by listing their IDs before the optional duration:

```
METR 1 2 3 D30 → 0 (Success: MEs 1, 2, 3 mix together over 30 frames)
```

METR 1 D60 is the canonical "mix ME 1 over the default duration" command.

10.4.2 Argument validation and errors

METR requires at least one ME ID. Each token is either an integer ME ID or a trailing D<frames>; the D token, if present, must be **last**. Violations return a non-zero error code rather than executing:

```
METR → 3 (ArgCount: no ME ID given)
METR D60 → 3 (ArgCount: duration but no ME ID)
METR 1 Dxx → 4 (ArgValue: "xx" is not a frame count)
```

METR 1 D60 2 → 4 (ArgValue: D token is not last)

MESR is strict on shape — exactly four fields — and validates that the bus is T or B and that the source name resolves to a real source:

MESR 1 T Source%201 → 0 (Success: "Source 1" → ME 1 Top input)

MESR 1 X Source%201 → 4 (ArgValue: bus must be T or B)

MESR 1 T Bogus → 4 (ArgValue: no source named "Bogus")

Note: Changing the **Top** content of an ME that is live on a *different* Program layer is blocked by a safety guard, so an external command cannot accidentally alter on-air output elsewhere. A blocked assignment surfaces as an arg-error, not a silent change.

10.4.3 Reading ME state

MELS returns the full ME list with no arguments — count first, then five fields per ME: Id, name, Type (Normal / Aux / Background), top content, bottom content. Empty content fields are emitted as -, and names are space-encoded.

MELS → 0 2 1 ME-UHD-1 Normal Source%201 Source%204 2 BG-PS-1 Background BG-1 -

This is the read-back to confirm what a controller set: which source is on Top, which on Bottom, and the ME type. Pair it with MESR/MELA/METR to build a closed-loop controller.

Tip: Related tokens that are **not** Mix Effect commands: **TRN** transitions *layers* (Section 10.3.2), and **DMT/DMB** act on software *devices* (Section 10.3.8). Do not substitute them for METR — they drive different objects.

10.5 Companion integration

Spyder external control is also driven from a **Bitfocus Companion** module, which turns these string commands into Stream Deck buttons for operators.

At a high level: Companion holds the connection to the Spyder-S server — the same UDP control interface documented in Section 10.2 — and maps each button to a string command (a METR, MESR, MELA, a script recall, and so on). Because Companion emits the same tokens documented above, everything in Sections 10.1–10.4 applies — the module is a convenience layer over this API, not a separate protocol.

10.6 Console Simulator (UDP)

Tools ▶ Console Simulator (UDP) opens a separate window that **emulates an external control console** in software, talking to a server over the **console UDP protocol**. It lets you operate (or test) a Frame from a simulated console without the physical console hardware present.



Figure 10-2. The Console Simulator — a software emulation of an external control console.

Control	Purpose
Server IP	The server to connect the simulated console to.
Connect (toggle)	Connect or disconnect the simulated console.
Console layout / Segments	The on-screen console buttons.

Note: The Console Simulator speaks the console UDP protocol to a server, standing in for an external control console. It is distinct from the in-app **Devices** pane (Appendix B), which is a Studio software control surface.

11. Maintenance & Troubleshooting

This chapter covers reading the system's own diagnostics when something is wrong: the Alert Viewer's live log, the bench tools for testing external-control commands, a troubleshooting table for common symptoms, and hardware-level factory recovery.

Notice: Preventative maintenance is an important part of the continued and proper operation of the product. Failure to perform maintenance as required, and in accordance with the maintenance schedule specified by Christie, **voids the warranty**. For preventative maintenance schedules, refer to www.christiedigital.com.

11.1 Alerts and monitoring

The **Alert Viewer** is a log panel that lists system messages, warnings, and errors. It shares a pane group with **System Patch**, docked along the bottom of the workspace. The tab **highlights orange** when new, unviewed alerts arrive.

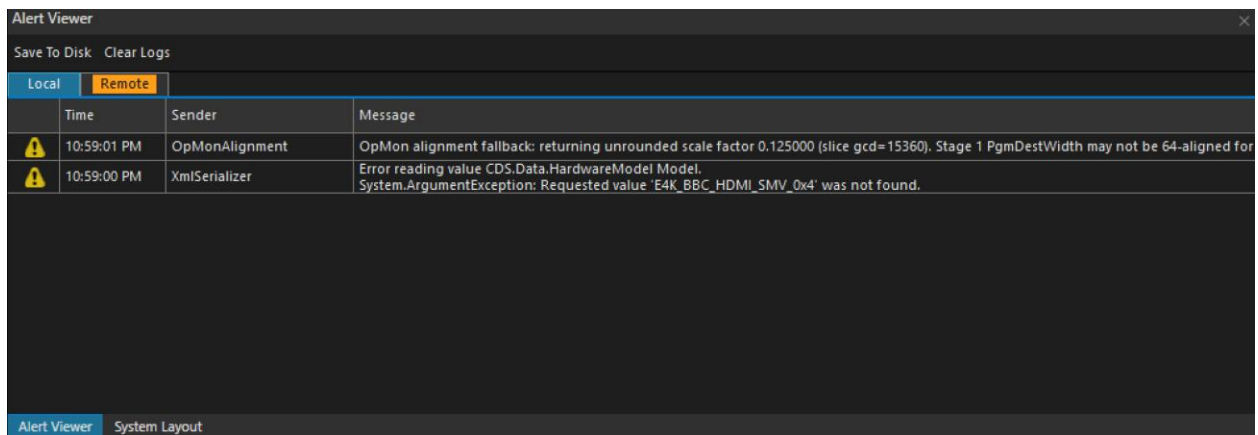


Figure 11-1. The Alert Viewer — Local/Remote provider tabs and the alert grid.

11.1.1 Reading alerts

Alerts are grouped by **provider**, shown as tabs:

- **Local** — the Spyder Studio app's own alerts and diagnostics.
- **Remote** — alerts reported by the connected Spyder server. This tab appears only once Studio is connected to a server (a physical Frame, or a local remoting server on the PC).

Note: Offline, only the **Local** tab is present. The **Remote** tab appears as soon as Studio connects to a server and disappears again when it disconnects.

A provider tab also turns **orange** when it holds alerts you have not viewed; the highlight clears when you select that tab.

The alert grid has four columns:

Column	Contents
(icon)	Severity indicator — for example a yellow warning icon.

Column	Contents
Time	Timestamp.
Sender	The component that raised the alert (for example <i>RemotedSystemManager</i> , <i>ServerConnectPanel</i> , <i>OpMonAlignment</i>).
Message	The alert text.

Double-click an alert to open it in the **Alert Details** window, which shows its severity, **Sender**, full **Message**, **Level**, and the time it was **Logged**. The up and down arrow buttons step to the previous and next alert — the selection in the alert grid follows — and **Copy** places the alert's type, source, date, time, and description on the clipboard. **Close** dismisses the window.

Note: The grid shows alert times in local time. The **Logged** time in Alert Details, the time copied by **Copy**, and the times in a saved log file (Section 11.1.2) are in **UTC**; the saved file marks them with a trailing Z.

11.1.2 Saving and clearing alerts

Button	Function
Save To Disk	Saves the alert log to a .txt file (default <i>Alert Log.txt</i> , in Documents). Writes a timestamped header, then each visible provider's alerts.
Clear Logs	Clears the selected provider's alert list.

The saved file starts with a timestamped header, followed by a section for each tab — **Local Alerts**, then **Remote Alerts**.

Right-clicking the alert list opens a menu. Its submenus open to the left:

Item	Function
Clear Logs	Same as the button.
Messages to Display ▸	25, 50, 100, 250, or 500 (default 500).
Message Priority Level ▸	Error, Warning (default), Information, Debug, or Trace — the lowest level of new alert that is added to the list. Alerts already in the list stay. For example, at Error , new warnings are not listed.

Tip: The related option **FlashAlertsTabOnEventLevel** (Tools ▸ Options ▸ **Advanced**) flashes the Alert Viewer tab when an alert at or above that level arrives and the panel is not visible. See Section 4.9.

11.2 Tools and diagnostics

The **Tools** menu holds a bench for testing the external-control command set, a set of project cleanup utilities, and the User Diagnostics tools.

Note: The UDP Protocol Tester and the maintenance utilities are **always available** in the Tools menu and do not require a remote-Frame connection to open. The UDP Protocol Tester does, however, need a reachable server (a real Frame, or the local remoting server) to actually do anything — point it at a server IP.

11.2.1 UDP Protocol Tester

Tools ▸ UDP Protocol Tester opens in the Properties panel. It **sends raw external-control string commands** to a Spyder server over **UDP** so you can test them — the bench for the external-control command set (Chapter 10).

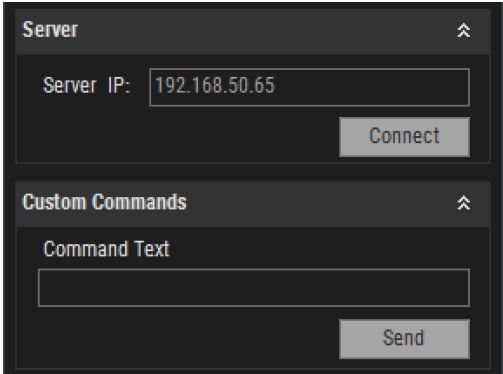


Figure 11-2. The UDP Protocol Tester.

Control	Purpose
Server IP	Target server address. It keeps the last server used — including across a restart and while disconnected — and shows the current remote server's IP when connected.
Connect	Open a UDP client to that server.
Command Text	Type a raw command string.
Send	Transmit the typed command over UDP.

Each **Send** opens a **Send Results** window that names the result — for example *Execution Result: Success*, or *Execution Result: Header* for an unrecognized command — and, when the server returns data, shows it as *Response from remote server:* Click **OK** to close it.

Use it to confirm a command string produces the expected result before wiring it into an automation controller.

11.2.2 Maintenance utilities

The simpler Tools actions — **Repair Data List**, **Remove Unused Input Configurations**, **Clean Command Key Scripts**, and **Frame Configs** — clean up and repair project data. Each is listed with its effect and availability in Appendix A.5.

11.2.3 User Diagnostics

Tools ▸ User Diagnostics opens the **User Diagnostics** window, which shows the status of the system's diagnostic checks.

Control	Purpose
System Status	The overall result. Until a check has run it reads NotRun , with the prompt <i>Click 'Refresh Now' to check status.</i>
Monitor Status	The list of checks, with a Name and Status for each.
Reset Error Counters	Link that resets the checks' error counters.
Refresh Now	Run the checks now.
Auto Refresh	Tick to refresh the checks automatically.
View/Save Report	Drop-down: Save as XML... , Detail Report , Test Definitions , and Create Service Bundle .
Details / Recommended Action / Advanced	Information tabs. When the window opens, Details describes Frame Lock Monitoring — the check that reports genlock errors.

Note: When Studio is connected to the Local Remoting Server rather than a Frame, **Refresh Now** reports *Update failed — Failed to update module info. Check alert log for more details.*

Tools ▸ RealTime Protection Settings opens the **User Diagnostics Automatic Protection** dialog, which controls real-time monitoring:

- **General — On (recommended)** enables real-time monitoring, so you are notified if any warning or error events occur; tick **Allow Exceptions to client monitored events** to ignore specific event types without disabling all monitoring (**Current Exceptions...** then opens the **Exceptions** tab). **Off (not recommended)** disables it, and the client can then no longer detect potential problems in the Spyder system.
- **Exceptions** — the monitored events that can be ignored individually, with a **Details** box for the selected event.

Commit with **Ok** or **Apply**, or close with **Cancel**. The choice is the **UserDiagnosticsMode** setting in **Tools ▸ Options ▸ Advanced** (Section 4.9.1): **On** is Enabled, **On with Allow Exceptions** ticked is EnabledWithExceptions.

11.3 Common problems

The table below lists common symptoms, with their likely cause and the corrective action. Cross-references point to the section that covers the fix in full.

Symptom	Possible cause	Solution
Most Server menu items are greyed out.	No remote server is connected. The Frame-acting items (Push/Pull, HotSync, etc.) require a live connection.	Normal when offline — offline configuration work does not need them. Connect to a Frame in Connection Manager (Section 4.5.1) when you need them.

Symptom	Possible cause	Solution
The Alert Viewer has no Remote tab ; only Local alerts appear.	Not connected to a server. The Remote provider tab appears only once connected.	Normal offline. Connect to a Frame (Section 4.5.1) to see remote alerts — or, for testing without hardware, start the Local Remoting Server (Section 4.5).
Studio refuses to connect: the Connection Manager button reads Unable to connect , and a local alert reads <i>"Preventing connection to server ... because it requires an update, and the following downstream frame(s) are unreachable ..."</i> .	The server needs a firmware update, but one or more downstream frames cannot be reached . Studio blocks the connection so that a partial update cannot be applied — connection <i>and</i> updates from this client are disabled while that is true.	Restore network connectivity to the frames named in the alert, then update the Frame (Section 4.4.2). This is not routine offline chatter; the named frames need attention.
After a Push, the Frame does not reflect the local configuration, and Studio reports <i>"Failed to restart remote system manager after pushing data to server."</i>	The data copied, but the Frame's system manager did not come back afterwards, so the pushed configuration was never loaded. Studio restarts it automatically as part of the push — there is nothing for you to accept.	Push again. If it fails a second time, reboot the Frame with Server ▶ Restart Remote Frame , reconnect, and confirm the Frame matches the local configuration (Section 4.8.2).
The New Password / Confirm Password fields are greyed out in the Security dialog.	Enable authentication requirement is off — the password fields stay disabled until it is on.	Turn the authentication toggle on, then set a password of 4+ characters (Section 4.6).
Security change rejected with <i>"The passwords you entered did not match."</i>	New Password and Confirm Password differ; the match is checked on Submit.	Re-enter the new password identically in both fields (Section 4.6).
An EDID cannot be edited, exported, or removed.	A factory EDID is selected. Factory EDIDs are read-only — their fields are greyed out and the Export and Remove buttons are disabled.	This is expected. Use Add Edid to create a custom EDID, which is fully editable and removable (Section 6.9).
UI rendering glitches in certain controls.	Hardware (GPU) rendering is causing the issue.	Set Allow Hardware Acceleration to Off in Tools ▶ Options (Section 4.9).
A backup source switch is too slow / has an unwanted transition.	ImmediateSwitchBackupSource is False, so the switch transitions.	Set it True in Tools ▶ Options ▶ Advanced for an instant switch (Sections 7.1.4, 4.9).
A change to Allow Hardware Acceleration has no effect.	The setting needs an application restart — the Options panel says so beside the checkbox: <i>"Note: An application restart is required for hardware acceleration changes to take effect"</i> .	Restart Spyder Studio for the change to take effect (Section 4.9).

11.4 Factory reset and recovery

A **factory reset / recovery** reflashes the Spyder-S Frame from recovery firmware on a USB-C drive, using the recessed reset button on the front panel. It is a hardware-level recovery — not a routine update — and is performed with a recovery image supplied by Christie Technical Support.

Notice: A factory recovery reflashes the device and overwrites its on-Frame software state. Perform it only with recovery firmware supplied by **Christie Technical Support**, and only when directed to do so. For normal software and firmware updates, use Spyder Studio (Section 4.5) — not this procedure.

11.4.1 Perform a factory reset

Prerequisite: The recovery firmware on a USB-C drive (provided by Christie Technical Support), an unfolded paperclip, and a monitor connected to the Frame's **DIAG HDMI** port.

1. **Unplug the power** on the Spyder-S device.
2. Connect the USB-C drive that contains the recovery firmware to the USB-C port labeled **Device**.
3. On the front panel, insert an unfolded paperclip into the **small pinhole under the nameplate**. You will feel a click when you reach the button.
4. **Hold the button in.** Power on the device and continue to hold for **five to seven seconds**. The light on the front of the device goes off and a **purple light** comes on; you can then release the button.
5. Follow the process on the monitor connected to the **DIAG HDMI** port. The length varies but takes approximately **10 minutes**.



Figure 11-3. Spyder-S front panel. The recessed factory-reset button is reached through the small pinhole under the nameplate.

Note: The recovery firmware is provided by Christie Technical Support and is separate from the normal software-update packages. Updating the device's firmware this way is **not recommended** for routine updates — use the Spyder Studio update flow instead (Section 4.5).

Note: When the installation completes, the system performs a **soft reboot** and the connected devices come back on, including the monitor on the DIAG HDMI port. You can also monitor progress on the **front panel**, which begins blinking when the process starts.

12. Appendices, Glossary & Contact

This chapter is the quick-reference back of the guide. It does not introduce new behavior — it condenses the menu/toolbar surface, the auxiliary panes, and the resource model into reference tables, then collects every domain term in a glossary. Each appendix points back to the body chapter that documents the feature in full; use those cross-references when a table entry needs context.

Appendix A — Menu & toolbar reference

The Spyder Studio menu bar carries six dropdown menus — **File**, **View**, **Layout**, **Server**, **Tools**, **Help** — plus a conditionally shown **Debug** menu and a direct-action **Save** button. The bar is assembled at runtime, so its contents depend on the connected hardware, the server connection state, and user options.

Availability legend: *Always* — present in every session · *Remote only* — enabled only when connected to a remote Spyder server (greyed out otherwise) · *Hardware-gated* — shown only for hardware that supports the feature · *Option-gated* — shown only when a user option is enabled.

A.1 File

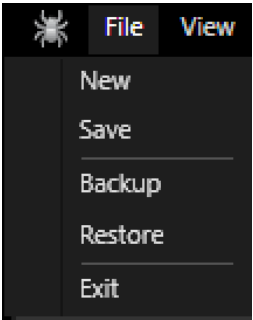


Figure 12-1. The File menu.

Item	Action	Availability
New	Opens the Configuration Manager (create/open a configuration)	Always
Save	Saves the current configuration and programming data	Always
Backup	Creates a backup file of the current data	Always
Restore	Restores data from a backup file	Always
Exit	Closes Spyder Studio	Always

A.2 View

Toggles the visibility of dockable panels; each item shows a checkmark when its panel is open. Eighteen panels can be toggled. In the factory layout, eleven are open: **Simulator, Sources, Backgrounds, Treatments, Mix Effects, Script, Command Keys, Function Keys, System Patch, Alert Viewer, Properties**. Seven are closed: **Devices, Dynamic Help, Router Patch, Router XY, System Layout, OpMon Configuration, Tile Configuration**. Auxiliary panes are summarized in Appendix B.

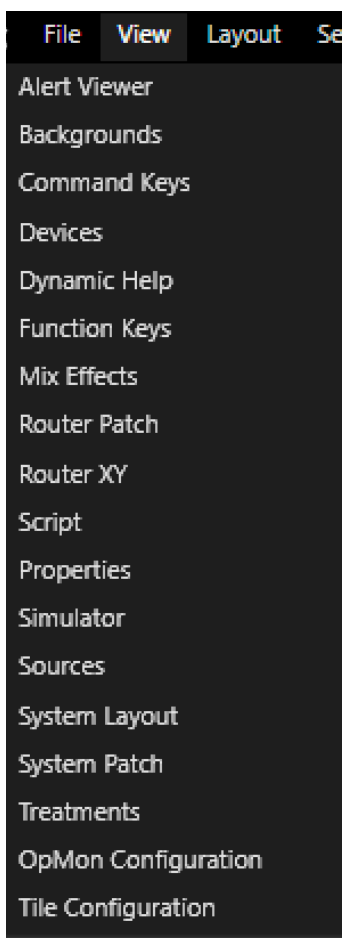


Figure 12-2. The View menu — toggling dockable panels.

A.3 Layout

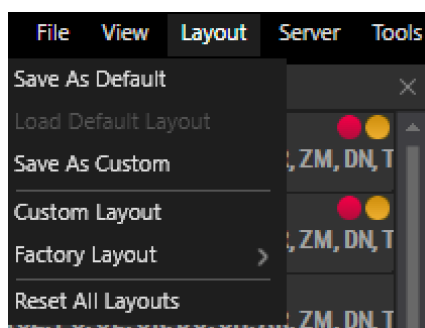


Figure 12-3. The Layout menu.

Item	Action	Availability
Save As Default	Saves the current arrangement as the default layout	Always
Load Default Layout	Restores the saved default layout	After Save As Default
Save As Custom	Saves the current arrangement as a named custom layout	Always
Custom Layout ▶	Submenu of saved custom layouts (load/rename/delete each); no submenu until a layout has been saved with Save As Custom	Always
Factory Layout ▶ Load Factory Layout	Restores the original factory arrangement	Always
Reset All Layouts	Clears all saved layouts back to factory state	Always

A.4 Server

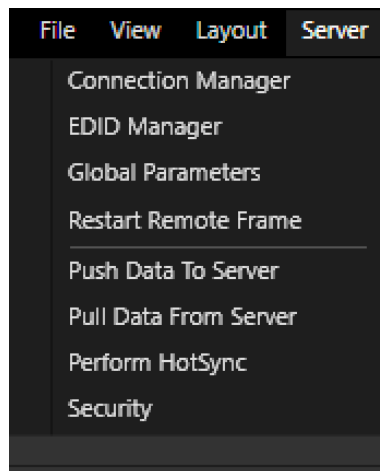


Figure 12-4. The Server menu.

Item	Action	Availability
Connection Manager	Open the server connection manager	Always
EDID Manager	Manage input EDIDs	Always
Global Parameters	Edit Frame/global settings	Always
Restart Remote Frame	Reboot the connected Frame (confirmation prompt)	Remote only
Push Data To Server	Copy local PC data → Spyder hardware (overwrites server data)	Remote only

Item	Action	Availability
Pull Data From Server	Copy Spyder hardware data → local PC	Remote only
Perform HotSync	Copy the primary frame's data to a backup Spyder system (overwrites the backup)	Remote only
Security	Security settings (Section 4.6)	Remote only

Notice: Push/Pull overwrite all configuration and programming data on the target. Back up first; see Section 4.8.

A.5 Tools

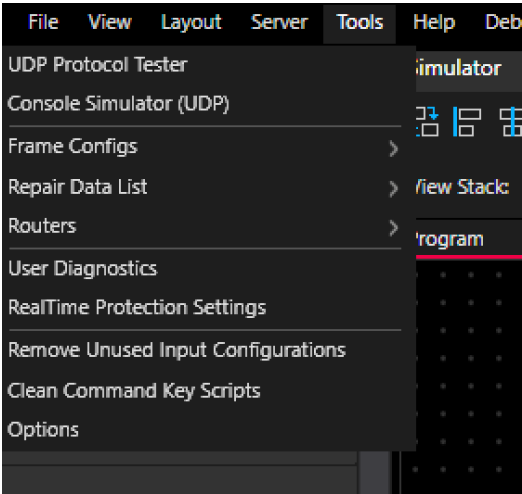


Figure 12-5. The Tools menu.

Item	Action	Availability
UDP Protocol Tester	Send/test external-control UDP commands	Always
Console Simulator (UDP)	Emulate an external control console over UDP	Always
Frame Configs ▶	Submenu, one entry per Frame (Frame <id>)	Always
Repair Data List ▶	Reset/auto-map data register lists (All Data Types, or per type)	Always
Routers ▶	Add New Router, plus one entry per configured router	Always
User Diagnostics	Open the user-diagnostics form	Remote only
RealTime Protection Settings	Configure real-time protection	Remote only

Item	Action	Availability
Remove Unused Input Configurations	Delete input configs not assigned to a source (confirmation)	Always
Clean Command Key Scripts	Remove inaccessible/unused scripts and cues (confirmation)	Always
Options	User options (Advanced ▶ User Diagnostics ▶ ShowDebugMenuItem shows/hides the Debug menu)	Always
Configure Expansion Link	Configure expansion link	Hardware-gated
IP Device Info	IP device information	Hardware-gated

A.6 Help

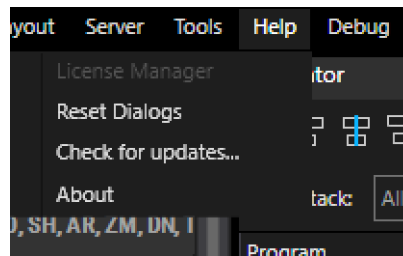


Figure 12-6. The Help menu.

Item	Action	Availability
License Manager	License request manager	Present but disabled in 6.1.3
Reset Dialogs	Clear all remembered "don't show again" dialog choices	Always
Check for updates...	Check for application updates	Always
About	Version/about dialog	Always

A.7 Debug (option-gated)

Hidden unless **Tools ▶ Options ▶ Advanced ▶ User Diagnostics ▶ ShowDebugMenuItem** is set to **True**.

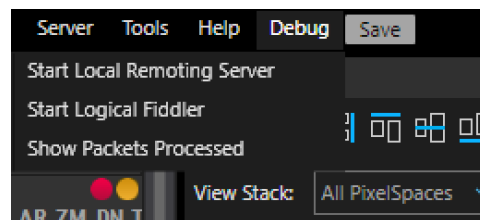


Figure 12-7. The Debug menu.

Item	Action
Start Local Remoting Server	Runs Studio's local engine as a network server so Studio can connect to it as a virtual Frame — for testing the connected path (Push/Pull, live apply) without hardware. Not required for offline configuration. See Section 4.5.
Start Logical Fiddler	Launch the internal hardware/control simulator
Show Packets Processed	Toggle the on-screen "Packets Processed" counter

A.8 Save (toolbar button)

A direct-action button on the menu bar — not a dropdown. Equivalent to **File ▶ Save**; commits the current configuration and programming data.

Appendix B — Auxiliary panes

These panes are hidden in the default layout. Enable each from the **View** menu. The primary operating panes (Simulator, Sources, Backgrounds, Treatments, Mix Effects, Script, Command/Function Keys, System Patch, Alert Viewer) are documented in Chapters 7–11.

B.1 Devices pane

A Studio software control surface that offers button-style device control over the layer model. It presents two 2×8 lit-button segments — **Device Selection** to **select a device**, and **Transition Controller** to **control it** — instead of a list. A device is a software grouping of one or two layers: **LYR** (single layer), **DLYR** (dual layer), or **MIXR** (mixer); devices are generated automatically from the system's layers. The control segment sends the selected device(s) to the **Off / Preview / Program** bus and, for mixers, runs **CUT** (instant) or **AUTO** (timed over Duration) transitions. **PROP** pushes the selected layer into the Properties panel for keyframe editing.

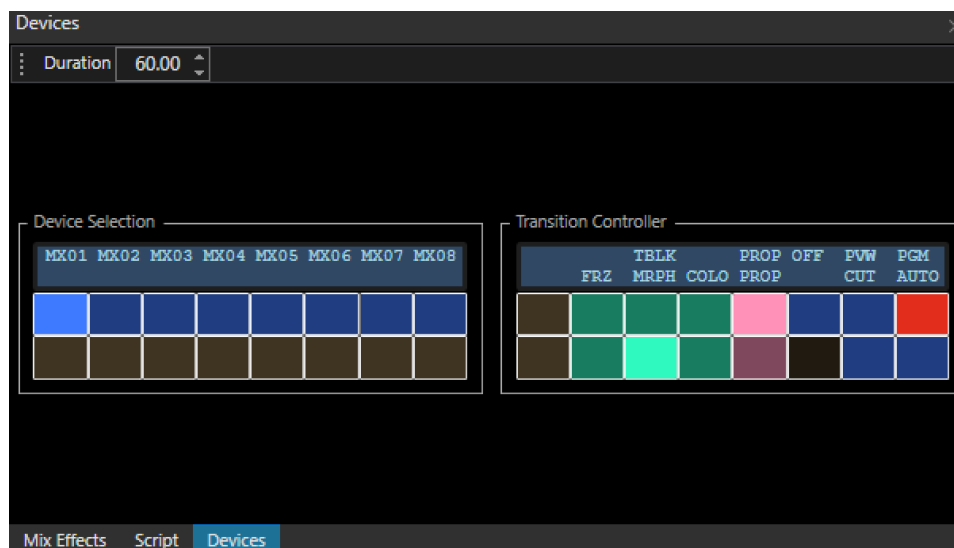


Figure 12-8. The Devices pane — button-style device control.

When to use it: when you prefer "pick a device, take it to air" button control over the Simulator's layer canvas. Enable with **View ▶ Devices**.

B.2 System Layout pane

A signal-flow grid of the whole system — Frame Group → Outputs → PixelSpaces → Preview → Layer. It is described with System Patch in Section 6.10.2. Enable with **View ▶ System Layout**.

B.3 Dynamic Help pane

A read-only, context-sensitive help reader. It shares the same selection signal as the Properties panel, so it automatically shows a short article about whatever object you currently have selected — a source, layer, treatment, PixelSpace, router, and so on. Each article has a title, intro, a "What To Do:" list, and a "Did You Know:" list. Selecting an object with no shipped article leaves the panel blank.

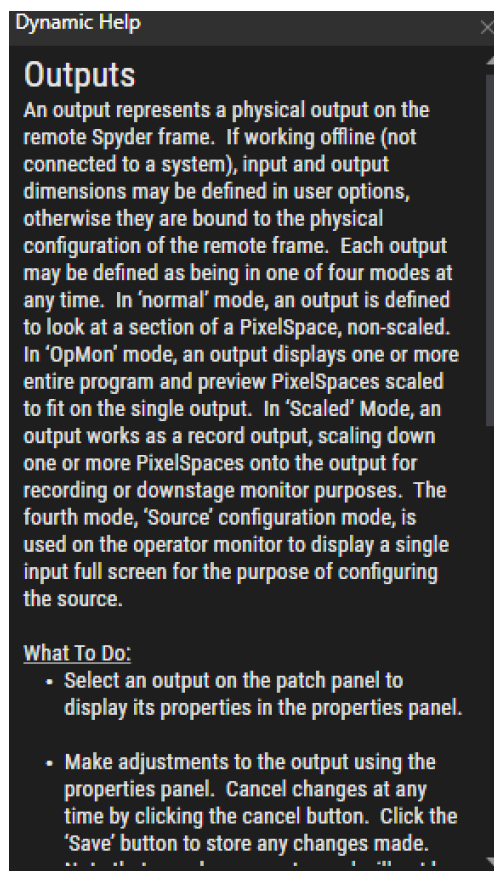


Figure 12-9. The Dynamic Help pane — context-sensitive help for the current selection.

When to use it: keep it open alongside Properties while learning a panel — Properties gives the editable fields, Dynamic Help explains what they do. Enable with **View ▶ Dynamic Help**.

Appendix C — Capacity & bandwidth

The full resource model is in Section 5.9 (the three budgets) and is enforced live by the Configuration Manager's allocation engine (Section 6.3.1.2 — the Resource Utilization panel). This appendix is the model in numbers.

C.1 The three budgets

Spyder-S allocates three budgets that can be exhausted independently:

Budget	Total	Spent by	Not spent by
Layers	Four UHD layers per Frame	Direct layers, Mix Effects, Aux content	Backgrounds (Layer Zero); clones (four per layer)
Inputs	Physical input channels per Frame	Sources <i>and</i> background groups; dual-link sources spend two each	—
Bus bandwidth	4.728 Gpps (whole system)	Every active layer, background input, Mix Effect, and output	—

The bus bandwidth ceiling is the hard limit: a configuration is valid only if total used bandwidth ≤ 4.728 Gpps. Exceeding it raises a footer warning ("*Total bus bandwidth exceeded by {x} Gpps*") and blocks Apply.

C.2 What draws against the bus

The bandwidth bar breaks total use into four segments. The quanta below are pacing-rounded (to 0.01846875 Gpps, 256 units per bus):

Segment	Approximate draw	Color
Canvas	The Frame-group raster, rounded to 960×540 tiles $\times$ frame rate	Red
UHD Mix Effect	$\approx 0.498 \text{ Gpps} \times (\text{frameRate} \div 60) \times \text{count}$	Orange
HD Mix Effect	$\approx 0.125 \text{ Gpps} \times (\text{frameRate} \div 60) \times \text{count}$	Blue
System / Utility	$\approx$ a quarter-tile per input, plus OpMon controllers	Yellow

Rate scaling: Mix Effect cost scales linearly with frame rate. A UHD ME at 60 Hz costs ≈ 0.498 Gpps; at 30 Hz it costs roughly half. This is why dropping the per-ME (or per-PixelSpace) frame rate frees bandwidth for more layers or more MEs.

C.3 Frame groups: canvas bandwidth vs. Mix Effects

Canvas bandwidth is set by the **single widest Frame group**, not the sum of all groups — each group's canvas is renewed independently. At 2160 high / 59.94 Hz a canvas raster costs $\approx$ **0.5 Gpps per UHD of width** ($8\times$ UHD ≈ 4.0 Gpps, $4\times$ UHD ≈ 2.0 , $2\times$ UHD ≈ 1.0). The leftover after canvas and system overhead is auto-provisioned to Mix Effects. So **splitting Frames into more, smaller groups lowers canvas bandwidth and frees the bus for Mix Effects**, while the **total canvas is the tiled sum of the per-group rectangles** (each $\leq 8\times 4K$) — so the surface can grow past one group's limit.

Worked example — one 11-Frame config, total canvas held at 8× UHD, regrouped (59.94 Hz, system/utility ≈ 0.7 Gpps):

Frame groups	Widest group	Canvas BW	UHD MEs (Auto Provision)
1 group	8× UHD (30,720 × 2,160)	≈ 4.0 Gpps	0
2 groups	4× UHD (15,360 × 2,160)	≈ 2.0 Gpps	4
4 groups	2× UHD (7,680 × 2,160)	≈ 1.0 Gpps	6

Enlarging the canvas works the same way: a **16× UHD** surface from **four 4× UHD groups** costs only the widest group (≈ 2.0 Gpps), leaving room for MEs — whereas the same 16× UHD as **two 8× UHD groups** costs ≈ 4.0 Gpps and leaves none.

System / utility is a separate, roughly fixed draw that scales with input count (≈ a quarter-tile per assigned input, plus OpMon/preview): about **0.4 Gpps at 24 inputs (6 Frames)**, **0.5 at 32 (8 Frames)**, and **1.0 at 64 (16 Frames)**. At full Frame count this floor measurably cuts into the bus available for canvas and Mix Effects.

C.4 Mix Effect caps and Auto Provision

Pool	Cap
UHD Mix Effects	12
HD Mix Effects	48

With **Auto Provision** on, Studio ignores the manual counts and packs as many UHD MEs as the leftover bus bandwidth allows (≤ 12), then fills the remainder with HD MEs (≤ 48). On the reference 4-Frame E4K-BBC rig this yields 6 UHD + 3 HD.

C.5 Worked reference — a four-Frame system

A 4-Frame E4K-BBC configuration (7680 × 2160 canvas at 59.94 Hz, one 2×1 PixelSpace fed by two background inputs):

- **Outputs** — each tile of a PixelSpace array consumes one physical output, so an H×V array uses H·V outputs. PS-1 (2×1) uses two DP outputs.
- **Inputs** — each background-source cell consumes one physical background input; BG-1 (2×1) uses two inputs. Dual-link cells consume two physical inputs each.
- **Bandwidth** — canvas raster + 6 UHD + 3 HD MEs + per-input system overhead fit inside 4.728 Gpps. Adding a 5th Frame pushes the total raster past the bus — *"exceeded by 0.037 Gpps"* — and Apply greys out.

Tip: When you run out of layers for a full-screen plate, move it to a **background** (Layer Zero). A background spends input channels, not layers — often the right trade for static scenic content (Section 5.9).

See Section 5.9 for the resource concepts and Section 6.3.1.2 for the live Resource Utilization panel and validation behavior.

Appendix D – Third-Party & Open-Source License Notices

Spyder Studio incorporates third-party open-source software components. By using the Software you also accept the additional terms and conditions of the third-party programs (see the Software License, Section 1.8). This release includes software distributed under the following licenses:

- **BSD 3-Clause License**
- **MIT License**
- **Mozilla Public License, Version 1.1**
- **Apache License, Version 2.0**
- **GNU Lesser General Public License, Version 2.1**
- **GNU General Public License, Version 2**

Note: The complete, verbatim text of each license above — including all required copyright notices, conditions, and disclaimers — is provided in the **Christie Digital License Information** document that accompanies this release. Refer to that document for the authoritative notices.

Appendix E – Supported formats

This appendix lists the video formats each connector type supports. Section E.1 summarises the maximums; Sections E.2 to E.4 list every supported format for layer inputs and outputs, and for background inputs. Frame types are introduced in Section 3.2.

Frame rates include the matching fractional rates: 60 includes 59.94, 30 includes 29.97, and 24 includes 23.98.

E.1 Summary by connector

Connector	Maximum resolution (layers)	Maximum background input	Colour formats
SDI 12G / SFP	4096 × 2160 at 60 Hz	4096 × 2160 at 60 Hz	RGB / YCbCr 4:4:4 and YCbCr 4:2:2, 10-bit (YCbCr 4:2:2 only at 4K, 50/60 Hz)
HDMI 2.0	4096 × 2160 at 60 Hz	7680 × 1080 at 60 Hz	RGB / YCbCr 4:4:4 8, 10-bit (8-bit only at 4K, 50/60 Hz); YCbCr 4:2:2 12-bit
DisplayPort 1.4	4096 × 2160 at 60 Hz	7680 × 2160 at 60 Hz	RGB / YCbCr 4:4:4 and YCbCr 4:2:2, 8, 10-bit

A background input supports every format listed for its connector in Sections E.2 to E.4, and wider formats up to the maximum above. A DisplayPort background input above 4096 × 2160 uses two physical inputs (Section 6.4.1.3).

E.2 SDI 12G and SFP formats

Note: The SFP Frame's interfaces can be SDI or HDMI 2.0, but its format support is the SDI set below in every case.

Resolution	Frame rates (Hz)	Colour formats
1280 × 720	25, 30, 50, 60	RGB / YCbCr 4:4:4 10-bit · YCbCr 4:2:2 10-bit
1920 × 1080	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 10-bit · YCbCr 4:2:2 10-bit
2048 × 1080	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 10-bit · YCbCr 4:2:2 10-bit
3840 × 2160	24, 25, 30	RGB / YCbCr 4:4:4 10-bit · YCbCr 4:2:2 10-bit
3840 × 2160	50, 60	YCbCr 4:2:2 10-bit
4096 × 2160	24, 25, 30	RGB / YCbCr 4:4:4 10-bit · YCbCr 4:2:2 10-bit
4096 × 2160	50, 60	YCbCr 4:2:2 10-bit

E.3 HDMI 2.0 formats

Background inputs support every format in this table; the last row is for background inputs only.

Resolution	Frame rates (Hz)	Colour formats
1024 × 768	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1280 × 720	25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1280 × 768	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1280 × 800	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1280 × 960	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1280 × 1024	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1360 × 768	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1366 × 768	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1400 × 1050	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit

Resolution	Frame rates (Hz)	Colour formats
1440 × 900	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1600 × 1200	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1680 × 720	50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1680 × 1050	50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1792 × 1344	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1856 × 1392	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1920 × 1080	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1920 × 1200	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1920 × 1440	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
1920 × 2160	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
2048 × 1080	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
2560 × 1080	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
2560 × 1600	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
3840 × 2160	24, 25, 30	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
3840 × 2160	50, 60	RGB / YCbCr 4:4:4 8-bit · YCbCr 4:2:2 12-bit
4096 × 2160	24, 25, 30	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit
4096 × 2160	50, 60	RGB / YCbCr 4:4:4 8-bit · YCbCr 4:2:2 12-bit
7680 × 1080 (background inputs only)	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 12-bit

E.4 DisplayPort 1.4 formats

Background inputs support every format in this table; the last row is for background inputs only.

Resolution	Frame rates (Hz)	Colour formats
1024 × 768	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1280 × 720	25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1280 × 768	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1280 × 800	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1280 × 960	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1280 × 1024	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1360 × 768	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1366 × 768	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1400 × 1050	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1440 × 900	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1600 × 1200	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1680 × 720	50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1680 × 1050	50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1792 × 1344	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1856 × 1392	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1920 × 1080	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1920 × 1200	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit

Resolution	Frame rates (Hz)	Colour formats
1920 × 1440	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
1920 × 2160	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
2048 × 1080	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
2560 × 1080	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
2560 × 1600	60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
3840 × 2160	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
4096 × 2160	24, 25, 30, 50, 60	RGB / YCbCr 4:4:4 8, 10-bit · YCbCr 4:2:2 8, 10-bit
7680 × 2160 (background inputs only)	60	RGB / YCbCr 4:4:4 8-bit · YCbCr 4:2:2 8, 10-bit

Glossary

Term	Definition
AOR (Area of Raster)	A Frame's rectangle (X, Y, Width, Height) on the shared canvas.
Auto Provision	Configuration mode that packs as many UHD then HD Mix Effects into the free bus bandwidth as the caps allow.
Aux	An auxiliary output role, driven by its own Mix Effect; <i>UnscaledAux</i> (flag A) is an unscaled Aux feed.
Background ("Layer Zero")	Full-canvas content behind every layer on a PixelSpace; consumes inputs, not layers.
Background group	An HCount × VCount grid of inputs tiled to fill a PixelSpace's background.
Bandwidth (bus)	Total internal pixel throughput; the fixed system ceiling is 4.728 Gpps.
BBC	The Spyder-S card family (E4K-BBC) that populates a Frame.
Canvas	The single coordinate space holding every PixelSpace in a configuration.

Term	Definition
Clone	A layer effect that duplicates the layer on the canvas — <i>Mirror</i> (a reflected copy) or <i>Offset</i> (up to four offset copies); set in the layer's Key Frame properties (Chapter 7).
Command Key	A saved snapshot of system state — layers, sources, positions, treatments — recalled from the Command Keys panel to restore a whole configured look at once. Created by "learning" the current state into a grid cell (Section 8.2).
Cut	An instant transition — a Mix with a one-frame duration.
Device	A software grouping of one or two layers (LYR, DLYR, or MIXR), generated automatically from the system's layers and controlled from the Devices pane.
DLYR	A dual layer — two independent layers (a top and a bottom) treated as one device.
DP (DisplayPort)	An input/output connector type; the E4K-BBC-DP card.
Dual-link	A paired connector that reserves two physical channels per logical input.
Duration	The transition time in frames (default 60).
E4K-BBC	The card model populating a Spyder-S Frame (DP/HDMI/SDI/SFP variants, 4 in / 4 out each).
EDID	The display-identification data Studio manages per input via the EDID Manager.
Frame	One Spyder-S hardware Frame, populated with an E4K-BBC card; Frame 0 is the master.
Frame group (renewal group)	A set of Frames sharing one canvas rectangle, headed by a renewal master.
Front panel	The Spyder-S Frame's information-only status display — shows the IP address and Christie logo; no operator buttons or touchscreen (Section 3.3); a recessed factory-reset button sits behind a pinhole (Section 11.4).
Function Key	A single-action soft button that performs one specific action on the layers currently selected — assign a source, nudge a position, fire a router salvo, trigger a Mix Effect transition. Contrast with a Command Key, which recalls a whole saved snapshot (Section 8.4).
Genlock	Reference-sync (Burst/Tri-Level, <i>BBTL</i>) in/out on the Frame that locks the system to an external timing reference (Chapter 3).

Term	Definition
Gpps	Giga-pixels per second — the unit of bus bandwidth (system ceiling 4.728).
HDMI	An input/output connector type; the E4K-BBC-HDMI card.
HotSync	The Server-menu action that synchronizes a backup Spyder system with the primary, overwriting the backup's data.
Keyframe	A layer's size, position, crop, aspect, pan/zoom, border/shadow, and transparency at an instant.
Layer	A single piece of content on a PixelSpace, controlled by keyframes; a finite hardware resource.
Local Remoting Server	An optional Debug tool that runs Studio's local engine as a network server, so Studio can connect to it as a virtual Frame — used to test the connected path (Push/Pull, live apply) without hardware. Not required for offline work.
Locked layer	A layer Spyder will not place content on, because it would be drawn behind the background — it sits on a Frame numbered lower than a background Frame in the same frame group. Shown with a padlock in Script and System Patch (Section 6.4.1.3).
LYR	A device that is a single layer.
ME (Mix Effect)	A two-input mixer (Top/Bottom) producing one transitionable layer — the "Single Layer Mixer."
Mix	A linear cross-fade between an ME's two inputs over the Duration.
MIXR	A device that is a mixer: two layers paired for transitions (top = on-air, bottom = next).
MQTT	The message channel some external-control data-dump commands return their payload over.
Normal (output)	Output flag N — a scaled program output for a PixelSpace.
OpMon	Output flag O — an output monitor / multiviewer tiling Program, Preview, and inputs.
PGM / Program	The on-air rail of a PixelSpace (red in the UI).
PixelSpace	A logical display region (a screen) with its own resolution, Program view, and Preview view.
Preset	A saved OpMon or Tiled output layout, recalled from the Presets dropdown in OpMon Configuration or Tile Configuration (Sections 6.6, 6.7). Tools ▸ Options ▸ Defaults also has a Preset list of setting profiles (Section 4.9).

Term	Definition
PVW / Preview	The staging rail of a PixelSpace (gold/yellow in the UI).
RCM	External-control continuation-retrieval command (returns long replies).
Renewal master	The Frame that starts a new Frame group.
Router	An external X-Y crosspoint device patched through Studio (Router Patch / Router XY).
Script	An ordered set of layer / Mix Effect commands built in the Script panel; a Command Key runs one script (Chapter 9).
SDI	An input/output connector type; the E4K-BBC-SDI card.
SFP	An input/output connector type; the E4K-BBC-SFP card.
Simulator	The primary operating pane — the Program/Preview canvas where layers, Mix Effects, and backgrounds are composed and arranged (Chapter 7).
Single Layer Mixer	The conceptual name for a Mix Effect — one layer that holds a full A-to-B transition.
Slave layer	A layer in another Frame group linked to a primary layer, so the primary layer can span that group. Global Linking provisions slave layers automatically; in Discreet mode you add them in a command key (Section 9.7).
Source	A named, placeable input signal.
Spyder Studio	The Windows client software that edits a local copy and drives the Spyder-S server.
Spyder-S	Christie's image-processing hardware (the Frame).
System Layout	The auxiliary pane showing the output → PixelSpace → layer wiring stack (Appendix B).
System Patch	The 5-column grid mapping layers to outputs.
Tiled	Output flag T — one output driving a grid of tiles (LED wall / tiled display).
Treatment	A saved set of layer look/geometry attributes that can be learned and recalled.
UHD	The full-resolution layer/ME class (vs. the lighter HD class).
UnscaledAux	Output flag A — an unscaled Aux feed (not generic Aux).
VI (Visual Image)	A composed picture on the canvas — what the audience sees.

Term	Definition
Virtual Mode	Applying a configuration locally while disconnected (<i>Apply to local PC</i>) — Studio builds it on the PC, with no server or Frame.

Contact Information

Spyder Studio and the Spyder-S are products of **Christie Digital Systems USA, Inc.**

Resource	Detail
Website	www.christiedigital.com
Products & downloads	https://www.christiedigital.com/products/image-processors/
Warranty information	https://www.christiedigital.com/help-center/warranties/
Technical documentation	www.christiedigital.com — on the product page, select the model and the Downloads tab
Corporate / legal notices	Christie Digital Systems USA, Inc., c/o Christie Digital Systems Canada, Inc., 809 Wellington Street North, Kitchener, Ontario, Canada N2G 4Y7
Manufacturing (ISO 9001:2015)	Kitchener, Ontario, Canada · Shenzhen, China

For the most current technical documentation and office contact information, visit www.christiedigital.com. For warranty terms, contact your Christie dealer or Christie.

Technical support. Technical support for Christie Enterprise products is available at:

Region	Telephone	Email
North & South America	+1-800-221-8025	Support.Americas@christiedigital.com
Europe, Middle East & Africa	+44 (0) 1189 778111	Support.EMEA@christiedigital.com
Asia-Pacific (general)	—	support.apac@christiedigital.com
Australia	+61 (0)7 3624 4888	tech-Australia@christiedigital.com
China	+86 10 6561 0240	tech-supportChina@christiedigital.com
India	+91 (80) 6708 9999	tech-India@christiedigital.com
Japan	+81 3 3599 7481	—
Singapore	+65 6877-8737	tech-Singapore@christiedigital.com
South Korea	+82 2 702 1601	tech-Korea@christiedigital.com

Region	Telephone	Email
Arizona (Americas)	+1-602-943-5700	techsupport-az@christiedigital.com

Regional office. Christie Digital Systems USA, Inc. — Phoenix office: 2001 W Melinda Ln, Phoenix, AZ 85027.