Viz Engine Administrator Guide

Version 5.6



Viz Engine

Copyright ©2026 Vizrt. All rights reserved.

No part of this software, documentation or publication may be reproduced, transcribed, stored in a retrieval system, translated into any language, computer language, or transmitted in any form or by any means, electronically, mechanically, magnetically, optically, chemically, photocopied, manually, or otherwise, without prior written permission from Vizrt. Vizrt specifically retains title to all Vizrt software. This software is supplied under a license agreement and may only be installed, used or copied in accordance to that agreement.

Disclaimer

Vizrt provides this publication “as is” without warranty of any kind, either expressed or implied. This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document.

Vizrt’s policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time. Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Antivirus Considerations

Vizrt advises customers to use an AV solution that allows for custom exclusions and granular performance tuning to prevent unnecessary interference with our products. If interference is encountered:

- **Real-Time Scanning:** Keep it enabled, but exclude any performance-sensitive operations involving Vizrt-specific folders, files, and processes. For example:
  - C:\Program Files\[Product Name]
  - C:\ProgramData\[Product Name]
  - Any custom directory where [Product Name] stores data, and any specific process related to [Product Name].
- **Risk Acknowledgment:** Excluding certain folders/processes may improve performance, but also create an attack vector.
- **Scan Scheduling:** Run full system scans during off-peak hours.
- **False Positives:** If behavior-based detection flags a false positive, mark that executable as a trusted application.

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Created on

2026/11/19

Contents

1 Introduction to Viz Engine 16
1.1 Related Documents 16
1.2 Feedback and Suggestions 16
1.3 Communicating with Viz Engine 17
1.3.1 1. Introduction 18
1.3.2 2. Connecting to Viz Engine 18
1.3.3 3. How Commands Are Structured 22
1.3.4 4. Sending Commands and Reading Responses 24
1.3.5 5. Addressing Containers in the Scene Tree 25
1.3.6 6. Getting Started — Your First Commands 26
1.3.7 7. Practical Examples 28
1.3.8 8. Working with Multiple Engine Instances 31
1.3.9 9. Error Handling Best Practices 31
1.3.10 10. Quick Reference — Most Useful Commands 32
1.3.11 11. Common Mistakes to Avoid 35
1.3.12 12. Further Reading 35
1.4 Property Path Guide 36
1.4.1 Scene Tree Structure 36
1.4.2 TCP Command Path Syntax 36
1.4.3 Common TCP Property Paths 37
1.4.4 VizScript Property Access 38
1.4.5 DataPool Path Syntax 40
1.4.6 Cross Reference: Same property, Three Methods 40
1.4.7 Naming Rules and Escaping 41
2 New in Viz Engine 42
2.1 Gaussian Splatting - Reality, Rendered in Real Time 42
2.2 Introducing Maps - Every story happens somewhere 43
2.3 Get started - Introducing Template Library 43
2.4 New Logicmaker 44
3 WIBU Licensing System 45
3.1 Important Pre-installation Information 45
3.2 Key Features and Workflow of the Wibu Licensing System 46
3.3 Limitations and Known Issues 47

3.4 Basic Set-up 47
3.5 Configuration Settings 49
3.5.1 Why Do We Need to Configure the Licenses? 49
3.5.2 Configuring the License System in the Viz Engine Configuration UI 49
3.5.3 Configuration Entries in the Viz Engine Configuration File 49
3.6 WIBU License System 55
3.6.1 Summary of Key Concepts in the Licensing System 55
3.6.2 WIBU License Share Status 55
3.6.3 WIBU Core License Summary 56
3.6.4 Core Licenses and Their Effect 58
3.6.5 Additional Licenses 62
3.7 Notes 66
3.7.1 Colors used in License Overview 66
3.7.2 Starting Viz Engine with MezzIP OUT and/or 1xMezzIP IN 68
3.7.3 Permanent Licenses 68
3.8 Setup Graphic Hub Journal-based E-Mail Notifications 68
3.9 Setup Redundancy for a WIBU License Server 69
3.9.1 Set Up Redundancy for a WIBU License Server 69
3.9.2 Test License Server Failover 69
3.9.3 Recover First License Server 69
3.9.4 Notes 66
4 Security Advisor 71
4.1 Recommendations 71
5 Software Configuration 72
5.1 Prerequisites 73
5.1.1 Supported Operating Systems 73
5.1.2 Virtual Environments 75
5.1.3 Environment Settings 75
5.1.4 Hardware and BIOS Settings 76
5.1.5 Power Management Settings 76
5.1.6 User Rights 76
5.1.7 Secure Boot 77
5.1.8 Setting Permissions to Run Viz Engine and Viz Artist 77
5.1.9 Anti-Virus Software 78
5.1.10 Additional Prerequisites 79
5.2 Viz Artist and Viz Engine Installation 80
5.2.1 Installing Viz Artist and Engine 80

5.2.2 Upgrading from a Previous Installation 82
5.2.3 To Change or Reinstall an Existing Installation 83
5.2.4 Silent Installation of Viz Artist and Engine 83
5.2.5 Installation of a Specific Component 83
5.2.6 Unattended Installation of Viz Artist and Engine 84
5.2.7 To Identify Installed Version 87
5.3 Viz Engine Folders 89
5.3.1 Installation Folders 89
5.3.2 Data Folders 89
5.4 Ports and Connections 90
5.4.1 Port Numbers 90
5.4.2 Multiplexing Ports 97
5.4.3 Reserve Ports 98
5.5 Viz Log Files 99
5.5.1 Viz Render Log 99
5.5.2 Viz Trace Log 99
5.5.3 Viz Gui Log 99
5.5.4 Viz Shaders Log 100
5.5.5 Viz Console Log 100
5.5.6 Viz GUI Connection Log 100
5.5.7 Create Log Files with Log and Clog Commands 100
5.6 Logfiles Configuration 101
5.6.1 Location of Log Configuration 101
5.6.2 Concept of Log Configuration 101
5.6.3 Built-In Log Config 107
5.6.4 Examples 110
5.7 Supported Software 113
5.7.1 Viz Engine Software 113
5.7.2 Preview Server 113
5.7.3 Viz Artist Software 114
5.8 Import and Export Formats 115
5.8.1 EXIF 115
5.9 Viz Engine REST Interface 116
5.10 User Account Control 118
5.11 NDI 119
5.11.1 NDI Input 119
5.11.2 NDI Output 120

5.11.3 NDI in Software I/O Only Mode 120
5.11.4 SDK Versions 120
5.11.5 NDI HX Output 121
5.12 Dual Channel Mode 125
5.12.1 To Configure Dual Channel 125
5.13 Trio Box CG Mode 128
5.13.1 To Configure Trio Box CG 128
5.14 Dolby E Support 131
5.14.1 Dolby E Features 131
5.14.2 Dolby E Configuration 132
5.15 EVS Video Server Control 133
5.15.1 Setup Requirements 133
5.15.2 RS422 and XtenDD35 Configuration 133
5.15.3 RS422 Pin-out for the Connector Cable 134
5.15.4 RS422 Controller Set Up Examples 134
5.15.5 Bluestorm LP PCI Card Configuration 134
5.15.6 ExSys EX-1303 USB to RS422 Connector Configuration 134
5.16 Integration with Viz One 136
5.16.1 Configure Viz Engine 136
5.16.2 Install Transfer and Monitor Services on Viz Engine 136
5.16.3 To Configure Viz One 138
5.16.4 Configure Local Preview of Video Files 138
5.17 Matrox Stream 139
5.17.1 Matrox License 140
5.17.2 Example Matrox License Configuration 141
5.17.3 Output Mode 141
5.17.4 Configuring a Stream 142
5.17.5 Example Sending MPEG-TS over RTP and Receiving with Viz Engine on localhost 143
5.17.6 Example Receiving MPEG-TS over UDP on localhost 143
5.17.7 Example Receiving SRT on localhost 144
5.17.8 Example Receiving RTSP on localhost 144
5.17.9 Example Receiving RTMP on localhost 145
5.17.10 More Advanced Example Using ffmpeg 145
5.17.11 Receiving Different Input Types 146
5.17.12 Limitations 146
5.17.13 Known Issues and Troubleshooting 146
5.18 VML Clip Player 147

5.18.1 Feature Comparison between Clip Players 147
5.18.2 Mixed Resolution Mode in Matrox and VML Clip Player 149
5.18.3 Non-broadcast Clip Support in Matrox and VML Clip Player 154
5.18.4 VML Clip Player as a Fallback Clip Player 155
5.18.5 Clip Player AutoFormat 156
5.18.6 VML Clip Player with NVDEC 156
6 Hardware Related Information 158
6.1 Handling and Installing Cards 159
6.2 Graphics Boards 160
6.2.1 NVIDIA Driver Configuration 161
6.2.2 Supported GPUs and Driver History 167
6.2.3 Virtualized GPUs 170
6.2.4 Working with Synchronous Output 175
6.2.5 Working with Two or More GPUs 177
6.3 Supported Systems 179
6.3.1 HP Z4 G5 181
6.3.2 HP Z4 Rack G5 185
6.3.3 HP Z8 G5 Fury 187
6.3.4 Lenovo P3 Ultra / P3 Ultra Gen 2 191
6.3.5 Lenovo P620 193
6.3.6 Lenovo SR655 V3 196
6.3.7 HP Z4 G6i 198
6.3.8 HP Z8 Fury G6i 201
6.4 Video Boards 204
6.4.1 SMPTE ST 2110 Configuration 205
6.4.2 Matrox Hardware 220
6.4.3 BlueFish444 268
6.4.4 Aja Hardware 280
7 Starting Viz Engine 286
7.1 To Start Viz Engine 286
7.2 To Add a Viz Engine Startup Option 286
7.2.1 Viz Configuration 286
7.2.2 To Migrate a Previous Viz Configuration 287
7.2.3 Deleting Shader Cache 287
7.3 Viz Console 288
7.3.1 Autocompletion Feature 288
7.3.2 Issuing External Commands to Viz Engine via Console 289

7.3.3 Internal Commands 291
7.4 Viz Command Line Options 293
7.4.1 Systems with Two or More GPUs 295
8 Configuring Viz 297
8.1 Working with Viz Configuration 298
8.1.1 To Start Viz Configuration 298
8.2 Modify Viz Configuration 298
8.2.1 To Save the Current Configuration 298
8.2.2 To Reset the Viz Config File 298
8.2.3 To Restart Viz Configuration 299
8.3 Installed Configuration Profiles 299
8.3.1 To Load a Pre-Installed Configuration Profile 299
8.3.2 To Save a Custom Profile 299
8.3.3 To Load a Custom Configuration Profile 299
8.4 Audio Settings 301
8.4.1 Various Tab 301
8.4.2 Channels Tab 302
8.4.3 Manual Audio Configuration 304
8.5 Authentication 306
8.5.1 Authentication Properties 306
8.6 Camera 309
8.6.1 Camera Properties 310
8.7 Clock Formats 314
8.8 Communication 315
8.8.1 Global Properties 315
8.8.2 Shared Memory Properties 318
8.8.3 VDCP Properties 319
8.9 Database 321
8.9.1 Global Properties 321
8.9.2 Failover Properties 323
8.9.3 Deploy 323
8.10 Font Text Options 325
8.10.1 Classic 325
8.10.2 Viz Engine 326
8.11 Global Input 327
8.11.1 To Synchronize Multiple Viz Engines 327
8.12 Image File Name Convert 328

8.13 Import 3D Options....329
8.14 Local Settings....330
8.14.1 Select Multiple Directories....331
8.14.2 SAM SDC01, SDC02, and SDC03 Protocols....331
8.15 Maps....333
8.15.1 Viz Maps....333
8.15.2 Viz World....334
8.15.3 To Add the VizWorld.ini File....335
8.16 Matrox....336
8.16.1 General Properties....336
8.16.2 M264 Encoder/Decoder Boards....337
8.17 Memory Management....339
8.18 Path Aliases....341
8.18.1 To Add a Path....341
8.19 Plug-ins....343
8.19.1 Plugin Directories....343
8.19.2 Plug-in Panel List....343
8.20 Render Options....346
8.21 Scene Default Values....350
8.21.1 Media Asset Configuration....351
8.22 Unreal Engine....353
8.22.1 Start/Close Unreal Engine using Viz Engine in On Air Mode....354
8.23 User Interface....355
8.23.1 Various....355
8.23.2 Tree Colors....357
8.23.3 Shortcuts....359
8.24 Video Board....362
8.24.1 Video Board Properties....362
8.25 Video Input: Clip Input....365
8.25.1 Clip Input Properties....365
8.26 Video Input: Live Input....367
8.26.1 Live Input Properties: Type SDI....368
8.26.2 Live Input Properties: Type IP....369
8.26.3 Live Input Properties: Type STREAM....370
8.26.4 Live Input Properties: Type NDI....371
8.26.5 Live Input Properties: Type SHM - SMURF....371
8.26.6 Mapping of Legacy StreamIn Channels....371

8.27 Video Output 372
8.27.1 Video Output Properties 372
8.27.2 Output Format / Colorimetry / Bits Per Channel 372
8.27.3 Video Output Editor 374
8.27.4 Video Output Channel 375
8.28 Video Output: Clip Output 383
8.28.1 To Configure Clip Output 383
8.28.2 Clip-out Channel Placeholders 384
8.28.3 Sample Commands 384
8.29 Video Output: SHM Output 387
8.30 Viz License Configuration 388
8.31 Viz One 390
8.31.1 Viz One Properties 391
9 On Air Mode 394
9.1 Director Control Pane 395
9.2 Control Buttons 396
9.3 Performance 398
9.3.1 Performance Monitor 398
9.3.2 Renderer Overlays 399
9.4 On Air Information Panel 404
9.4.1 Basic Tab 404
9.4.2 Clients Tab 405
9.5 License Information 406
9.6 Tracking Timing Monitor 407
9.6.1 What is it? 407
9.6.2 Opening the monitor 407
9.6.3 Camera Status window 407
9.6.4 Reading the timing graph 408
9.6.5 Saving the data for support 410
9.6.6 Tips 410
9.7 Camera Stats Overlay 411
9.7.1 1. What it is for 411
9.7.2 2. Quick start 411
9.7.3 3. Turning the overlay on 412
9.7.4 4. The window at a glance 412
9.7.5 5. Choosing which values to plot 413
9.7.6 6. Working with the graphs 413

9.7.7 7. Recording the history 414
9.7.8 8. Saving a recording (CSV) 414
10 Video IO Configuration and Features 415
10.1 Basic IO Configuration 416
10.1.1 General Information 416
10.1.2 Clip Input Channels 417
10.1.3 Live Input Channels 417
10.2 High Dynamic Range 419
10.2.1 How to Configure HDR 419
10.2.2 NDI Additional Information 424
10.2.3 S-Log3 Support in Viz Engine 424
10.3 Mixed Mode Video Support 425
10.3.1 Source: PAL or NTSC 425
10.3.2 Source: 720p 426
10.3.3 Source: 1080i 427
10.4 Frame Accurate Output 430
10.4.1 FAO Prerequisites 431
10.4.2 Commands 432
10.4.3 General Purpose IO Commands 433
10.4.4 Timed Command Execution 437
10.5 Shared Usage of Input Channels 440
10.5.1 Configuration 440
10.5.2 Important 440
10.5.3 Sharing Inputs on X.mio5 ST 2110 Boards 440
10.6 Dynamic Channel Allocation 441
10.6.1 Limitations 441
10.7 Matrox Supported Codecs 442
10.8 DVE Performance 443
10.8.1 General Information 443
10.8.2 X.mio5 443
10.8.3 X.mio3 444
10.9 Watchdog 446
10.9.1 Mechanical Bypass 446
10.9.2 Hardware Bypass 446
10.9.3 Transition from Watchdog to Video 446
10.10 NDI Output from GFX Channel 447
10.10.1 Configuration 447

10.10.2 Operation 447
10.11 Clip Player Pro Features 449
10.11.1 H.265 449
10.11.2 HAP 449
10.11.3 GPU Hardware Acceleration / NVDEC 449
10.12 Extraction of VANC Data 451
10.12.1 Types of VANC Data 451
10.12.2 Configuration 451
10.12.3 Shared Memory Locations 451
10.13 Multiple SDI/IP Outputs 452
10.13.1 Prerequisites 452
10.13.2 Types of Output Signal 452
10.14 Clipout Channel Usage 454
10.14.1 Introduction 454
10.14.2 Available File Types 454
10.14.3 Video, Audio, VBI Settings 456
10.14.4 Clipout Control 472
10.14.5 Clipout Feedback Channel 475
10.14.6 Clipout Examples 478
11 SMPTE 2110 Information 487
11.1 General Information and Introduction 488
11.1.1 General Description 488
11.1.2 Standards 489
11.1.3 Definition of Terms 492
11.2 Configuring a Static Network 495
11.2.1 Static TCP/IP Address Assignment 495
11.2.2 Static Multicast Address Assignment 495
11.3 Working with NMOS 497
11.3.1 What is NMOS 497
11.3.2 A Typical NMOS Environment 497
11.3.3 How to Configure NMOS on Viz Engine Systems 497
11.4 Troubleshooting SMPTE ST2110 and NMOS 499
11.4.1 Known Issues 499
11.4.2 SMPTE ST 2110 499
11.4.3 NMOS 502
11.5 Explicit Usage of the Second SFP Pair on Matrox X.mio5 Q25 and DSX LE5 Q25 503
11.5.1 Examples 503

12 Audio in Viz Engine....504
12.1 Overview....505
12.1.1 Channels....505
12.1.2 Matrox Routing....506
12.1.3 Audio Clip Formats....506
12.1.4 Sound Card Output....506
12.1.5 Audio Input for Emergency Alerts....507
12.2 Device Recognition and Selection....508
12.2.1 Device Recognition....509
12.2.2 Device Selection....509
12.3 Timing Behavior and Delay Settings....510
12.3.1 Timing....510
12.3.2 Channel Device and Channel Track Settings....510
12.4 Audio Plug-in....514
12.5 Clip Formats....515
12.6 Speaker Names....516
12.7 Emergency Alert System....517
12.7.1 To Specify Output Channels for EAS....517
12.7.2 EAS Commands....517
12.8 Shared Memory Integration of Channel Volume Levels....518
12.9 Dolby E Configuration....521
12.9.1 Enable DolbyE....521
12.9.2 Prepare for DolbyE Decoding....521
12.9.3 Prepare for DolbyE Encoding....522
12.9.4 Decode Plus Encode....522
12.9.5 Pass-through Mode....522
12.10 Channel Setup and Clip Channel Routing....524
12.10.1 To Test Audio Channel Setup....524
12.11 Matrox Audio....525
12.11.1 To Enable Matrox Audio....525
13 Viz Engine Shared Memory....526
13.1 Viz Engine SHM Synchronization....527
13.1.1 TCP and UDP Synchronization....528
13.1.2 External Control Synchronization....529
13.1.3 Command Synchronization....529
13.2 Viz Engine SHM External Data Input....531

13.2.1 SHM over TCP 532
13.2.2 SHM over UDP 533
13.2.3 Plug-in API 534
13.2.4 REST Interface 535
13.3 Viz Engine Internal Data Interactive Scene 537
13.3.1 Example 537
13.4 Viz Engine SHM Snapshot 538
14 Video Wall Configuration 539
14.1 Hardware Requirements and Recommendations 540
14.1.1 Minimum Hardware Configuration for Video Walls 540
14.1.2 Recommended Configuration for Video Walls 540
14.1.3 UHD Configuration with X.mio5 Boards 541
14.1.4 IP-based Video Walls 541
14.2 Performance Considerations 542
14.2.1 Hardware Considerations 542
14.2.2 Clip Playback 542
14.2.3 Scene Design Considerations 543
14.3 Troubleshooting Video Wall Configurations 544
14.3.1 Performance Issues 544
14.3.2 Steps to Recover from Severe NVIDIA Mosaic Driver Related Issues 544
14.3.3 Experiencing BSOD or System Freeze while Setting up Mosaic 545
14.3.4 Only Some Displays of the Video Wall display an Image 545
14.3.5 Missing NVIDIA Control Panel Settings 545
14.3.6 NVIDIA Control Panel Crashes 545
14.3.7 Mosaic Configuration Not Supported Error 546
14.3.8 G-Sync Status LEDs or Topology Reports Indicate a Synchronization Issue 546
14.3.9 Poor Performance when Using GFX Channels as DVE 547
14.3.10 Blending Issues Using Classic Scenes in Superchannels with 16-bits per Channel Rendering 547
14.3.11 Jittering on HP Z840 547
14.3.12 Other Synchronization Issues 547
14.4 HDR Window Configuration 548
14.4.1 Introduction 548
14.4.2 Prerequisites 548
14.4.3 Configuration 548
14.4.4 References 549
14.5 Video Wall Setup Instructions 550
14.5.1 Pre-Requirements for All Setups 551

14.5.2 Configure the NVIDIA Driver for Video Wall 552
14.5.3 Order of Steps to Set Up NVIDIA Mosaic 552
14.5.4 NVIDIA Quadro Sync 553
14.5.5 Configuration Using Datapath Devices 554
14.5.6 NVIDIA Mosaic Configuration for 1080i50 559
14.5.7 NVIDIA Mosaic Configuration for 1080i60M 566
14.5.8 Custom Resolution for 59.94 Hz Refresh Rate 570
14.5.9 Viz Engine Video Wall Configuration Settings 575
14.5.10 Mosaic + 1 Setup 578
14.5.11 Large Canvas Output 583
15 Vizrt Viz Engine Monitor 586
15.1 Installation 586
15.2 Configuration 586
15.2.1 Viz Engine Configuration 586
15.2.2 OpenTelemetry Endpoints 586
15.2.3 Viz Engine Monitor Configuration 586
15.2.4 Viz Component Log Configuration 587
15.3 Usage 588
15.3.1 Install Service 588
15.3.2 Uninstall Service 589
15.3.3 Start Installed Service 589
15.3.4 Stop Installed Service 589
15.4 Troubleshooting 589
15.5 Logged Information 590
15.5.1 Shared Memory Information Logs 590
15.5.2 Shared Memory Information Metrics 590
15.6 Complete Engine Configuration File 591
15.6.1 Parameters for Viz Component Log Configuration 600

1 Introduction to Viz Engine

This Administrator Guide gives details on the configuration and installation of Viz Engine. It also explains settings available through its configuration user interface.

The term Viz is used for the programs installed and run on the computer. This is used as a general reference for all modes of the program:

- Viz Artist (see the Viz Artist User Guide)
- Viz Engine
- Viz Configuration

What mode of program that can run is determined from the hardware dongle that is attached to the actual machine (the different modes and the hardware dongle are detailed in this User Guide).

1.1 Related Documents

- **Viz Artist User Guide:** Contains information on how to install Viz Engine and create graphics scenes in Viz Artist.
- **Viz Artist Script Reference:** Contains information on how to create scripts for a scene. This is installed locally at *C:\Program Files\Vizrt\VizEngine\Documentation\viz-script-doc-webhelp*.
- **Viz Engine Plug-in SDK Reference:** Contains information on how to create a customized Viz plug-in. This is installed locally at *C:\Program Files\Vizrt\VizEngine\Documentation\viz-pluginsdk-doc-webhelp*.
- **Viz Engine Command Reference:** Contains information about Viz Engine Commands. This is installed locally at *C:\Program Files\Vizrt\VizEngine\Documentation\viz-cmd-doc-webhelp*

1.2 Feedback and Suggestions

We encourage suggestions and feedback about our products and documentation. To give feedback and/or suggestions, please contact your local Vizrt customer support team at www.vizrt.com.

Copyright © 2026 Vizrt

1.3 Communicating with Viz Engine

- 1. Introduction
  - What You Can Do via Commands
- 2. Connecting to Viz Engine
  - The TCP Socket
  - How to Connect
  - Command Syntax
  - Important Connection Rules
- 3. How Commands Are Structured
  - The Command Format
  - The Asterisk Delimiter
  - Common Verbs
  - Top-Level Objects
- 4. Sending Commands and Reading Responses
  - The Request-Response Cycle
  - Response Types
  - Error Detection
  - Liveness Check
- 5. Addressing Containers in the Scene Tree
  - Three Ways to Address a Container
  - Navigating Nested Containers
  - Accessing Properties on a Container
  - Key Container Properties
- 6. Getting Started — Your First Commands
  - Step 1: Check That the Engine Is Ready
  - Step 2: Load a Scene
  - Step 3: Modify Scene Content
  - Step 4: Trigger an Animation
  - Step 5: Read Back Values
  - Step 6: Animate Out and Unload
- 7. Practical Examples
  - Example 1: Complete Lower Third Workflow
  - Example 2: Reading and Modifying Transformations
  - Example 3: Show/Hide Containers
  - Example 4: Changing Colors
  - Example 5: Working with Multiple Directors
  - Example 6: Taking Snapshots
  - Example 7: Monitoring Engine Performance
  - Example 8: Shared Memory (Real-Time Data Injection)
  - Example 9: Batch Mode (Multiple Commands, One Response)
  - Example 10: Flicker-Free Multi-Command Updates
- 8. Working with Multiple Engine Instances
- 9. Error Handling Best Practices

Copyright © 2026 Vizrt

    - Always Check for ERROR in Responses
    - Always Gate on Renderer Readiness
- 10. Quick Reference — Most Useful Commands
    - Engine Status
    - Scene Loading
    - Scene Content
    - Animations
    - Plugin Parameters
    - Shared Memory
    - Cleanup
- 11. Common Mistakes to Avoid
- 12. Further Reading

1.3.1 1. Introduction

Viz Engine is a real-time graphics rendering engine used in broadcast television. Every control application — whether it's Viz Trio preparing a lower third, Viz Pilot filling a template, a newsroom automation system, or your own custom tool — communicates with Viz Engine by sending **text commands over a TCP socket**.

This guide teaches you how to connect to Viz Engine, how commands are structured, and how to send and receive them. By the end, you'll be able to load scenes, modify text and graphics, trigger animations, and monitor the engine — all from code.

What You Can Do via Commands

- Load and unload scenes on the renderer
- Modify text, images, materials, and transformations in a live scene
- Start, stop, and control animations (directors)
- Read and write shared memory for real-time data injection
- Take snapshots of the rendered output
- Monitor engine performance (FPS, GPU memory, render time)
- Manage the engine lifecycle (restart, shutdown, configuration)

1.3.2 2. Connecting to Viz Engine

The TCP Socket

Viz Engine listens for commands on a **TCP socket**. The connection parameters are:

Parameter	Value
Protocol	Raw TCP (no TLS, no HTTP framing)
Default Port	6100

Copyright © 2026 Vizrt

Encoding	Plain ASCII text
Message Separator / Terminator	Null character (\0)
Max Connections	255 simultaneous (configurable)

How to Connect

Open a TCP socket to the engine's hostname or IP address on port 6100. Once connected, you can immediately start sending commands.

Command Syntax

`CMD_ID` `BLANK_CHARACTER` `CMD` `NULL_BYTE` [...]

The `CMD_ID` is used to identify the command and uses the same ID for the answer, except when the CMD_ID is `-1`, where no answer is sent back. It is a string and may be composed of a short text and some number, or -1 to instruct the engine not to send an answer.

General Example

`MYID123 MAIN VERSION\0`

will send back this answer

`MYID123 Version: 5.5.2.40039`

Python example:

```python
import socket
# Connect to Viz Engine
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(("viz-engine-host", 6100))
# Send a command (must end with \0)
sock.sendall(b"MYID123 MAIN VERSION\0")

# Read the response (also newline-terminated)
response = sock.recv(4096).decode("ascii").strip()
print(f"Engine version: {response}")
sock.close()
```

PowerShell example:

first, we create a small helper function to read until we receive the null byte:

```
function Read-UntilNul {
    param(
        [Parameter(Mandatory)]
        [System.IO.Stream]$Stream,
```

Copyright © 2026 Vizrt

```
        [System.Text.Encoding]$Encoding = [System.Text.Encoding]::UTF8,

        [int]$MaxBytes = 1048576
    )

    $bytes = New-Object System.Collections.Generic.List[byte]

    while ($true) {
        $b = $Stream.ReadByte()

        if ($b -lt 0) {
            throw "Connection closed before NUL terminator was received."
        }

        if ($b -eq 0) {
            break
        }

        $bytes.Add([byte]$b)

        if ($bytes.Count -ge $MaxBytes) {
            throw "Message exceeded MaxBytes ($MaxBytes) without NUL terminator."
        }
    }

    return $Encoding.GetString($bytes.ToArray())
}
```

Then we can use the following snippet

```
$client = New-Object System.Net.Sockets.TcpClient("localhost", 6100)
$stream = $client.GetStream()
$utf8NoBom = [System.Text.UTF8Encoding]::new($false)
$writer = New-Object System.IO.StreamWriter($stream, $utf8NoBom, 1024, $true)
$reader = New-Object System.IO.StreamReader($stream, [System.Text.Encoding]::UTF8,
$false, 1024, $true )

# Send command
$writer.Write("MYID789 MAIN VERSION" + [char]0)
$writer.Flush()

$response = Read-UntilNul -Stream $stream -Encoding ([System.Text.Encoding]::UTF8)
Write-Host "Engine version: $response"

$writer.Dispose()
$stream.Dispose()
$client.Close()
$client.Dispose()
```

This would produce e.g

Copyright © 2026 Vizrt

Engine version: MYID789 Version: 5.5.2.40039

C# example:

```
using System;
using System.Net.Sockets;
using System.Text;

class Program
{
    static void Main(string[] args)
    {
        string host = args.Length > 0 ? args[0] : "viz-engine-host";
        int port = 6100;
        Console.WriteLine($"Connecting to {host}:{port}...");
        try
        {
            using var client = new TcpClient(host, port);
            client.ReceiveTimeout = 5000;
            var stream = client.GetStream();
            Console.WriteLine("Connected. Sending command...");
            // Send command immediately — VizEngine expects null-terminated commands
            byte[] cmd = Encoding.ASCII.GetBytes("MYID456 MAIN VERSION\0");
            stream.Write(cmd, 0, cmd.Length);
            // Read all responses until nothing more arrives
            while (true)
            {
                string response = ReadNullTerminated(stream);
                if (response.Length == 0) break;
                Console.WriteLine($"Response: {response}");
                // Stop after our tagged response
                if (response.StartsWith("MYID456")) break;
            }
        }
        catch (SocketException ex)
        {
            Console.WriteLine($"Connection failed: {ex.Message}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }

    static string ReadNullTerminated(NetworkStream stream)
    {
        var buffer = new System.Collections.Generic.List<byte>(256);
        int b;
        while ((b = stream.ReadByte()) > 0) // 0 = null terminator, -1 = closed
            buffer.Add((byte)b);
        return Encoding.ASCII.GetString(buffer.ToArray());
    }
}
```

Copyright © 2026 Vizrt

Important Connection Rules

- **Always terminate commands with a null byte** (\0). Without it, the engine buffers the input and never processes the command.
- **Keep connections open** for the duration of your session. Opening and closing connections for every command adds latency and wastes connection slots.
- **Close connections when done**. The engine has a maximum of 255 simultaneous connections. Leaked connections eventually block all clients.
- **One response per command**. Every command you send produces exactly one response line. Read one line after each send.

1.3.3 3. How Commands Are Structured

The Command Format

Every command follows this pattern: `OBJECT*PATH*PROPERTY VERB [arguments]`

There are three parts separated by spaces:

Part	What it does	Example
Object Path	Identifies *what* you're targeting	MAIN_SCENE*TREE*$Title*GEOM*TEXT
Verb	Identifies *what operation* to perform	SET, GET, INVOKE
Arguments	The *data* for the operation (optional)	"Hello World"

The Asterisk Delimiter

The object path uses **asterisks** (*) as delimiters to navigate through a hierarchy of objects. Think of it like a file path, but with * instead of /.

```
MAIN_SCENE * TREE * $Title * TRANSFORMATION * POSITION * X
     |      |     |       |         |     |
     |      |     |       |         |     └── The X coordinate
     |      |     |       |         └── Position property
     |      |     |       └── Transformation of the container
     |      |     └── Container named "Title" ($ = by name)
     |      └── The scene tree (hierarchy of containers)
     └── The currently loaded main scene
```

Always Use Asterisks: The path delimiter is always *. Do **not** use dots (.) or slashes (/) — these will not work and will return an error

Common Verbs

Verb	Purpose	Example
GET	Read the current value of a property	*POSITION*X GET
SET	Write a new value to a property	*POSITION*X SET 200.0
INVOKE	Execute an action or trigger a function	*FUNCTION*Toggle*switch INVOKE
SET_OBJECT	Load or assign a scene/object	RENDERER SET_OBJECT SCENE*path
START	Start an animation	RENDERER*STAGE START DEFAULT
CONTINUE	Continue a paused animation	*DIRECTOR*name CONTINUE
SHOW	Run a director to its end	*DIRECTOR*name SHOW
ADD	Add a child element	MAIN_SCENE*TREE ADD TOP
DELETE	Remove an element	MAIN_SCENE*TREE*$old DELETE

Top-Level Objects

Commands always start with one of these top-level objects:

Object	What it represents
RENDERER	The active renderer — load scenes, take snapshots, control layers
MAIN_SCENE	The currently loaded scene — its tree, stage, shared memory
SCENE	The scene pool — load, check, save, close scenes in memory
MAIN	Engine-level controls — version, restart, shutdown
VIZ_COMMUNICATION	Distributed shared memory (cross-engine data)

Copyright © 2026 Vizrt

GLOBAL	System-wide shared memory (local to this engine)
CONFIGURATION	Runtime configuration read/write
DATABASE	Graphic Hub connection management

1.3.4 4. Sending Commands and Reading Responses

The Request-Response Cycle

Communication is **synchronous** — you send one command, and the engine sends back one response. The response is always a single line of text terminated by a newline.

```
    You send:     VERSION\n
    Engine replies: 5.5.0.12345\n

    You send:     MAIN_SCENE*TREE*$Title*GEOM*TEXT GET\n
    Engine replies: Breaking News\n

    You send:     MAIN_SCENE*TREE*$Title*GEOM*TEXT SET "Hello World"\n
    Engine replies: \n    (empty = success, no value to return)
```

Response Types

Situation	What the response looks like
GET with a value	The value as a text string, e.g. "150.0"
SET success	Empty response (just a newline)
Error	A string containing the word ERROR, e.g. "ERROR: Unknown container"

Error Detection

Viz Engine does **not** return numeric error codes or JSON. To detect an error, check whether the response **contains the substring `ERROR`**:

```
    def send_command(sock, command):
```

Copyright © 2026 Vizrt

```
        sock.sendall((command + "\n").encode("ascii"))
        response = sock.recv(4096).decode("ascii").strip()
        if "ERROR" in response:
            raise RuntimeError(f"Viz Engine error: {response}")
        return response
```

Liveness Check

To verify the engine is alive and accepting commands, send: `IS_RENDERER_READY`

If you receive a 1, the engine is responsive. If the call times out, the engine is unresponsive or unreachable.

1.3.5 5. Addressing Containers in the Scene Tree

A Viz Engine scene is a tree of **containers**. Each container can hold geometry, text, images, plugins, and transformations. To target a container in a command, you navigate the tree using the MAIN_SCENE*TREE prefix.

Three Ways to Address a Container

Prefix	Method	Example	When to use
$	**By name**	MAIN_SCENE*TREE*$Title	Most common — uses the container's name
#	**By object ID**	MAIN_SCENE*TREE*#1234	When you know the numeric ID
/	**By index** (1-based)	MAIN_SCENE*TREE*/1/2/3	When addressing by tree position

Navigating Nested Containers

Containers can be nested. Chain the names to reach deep into the tree:

```
MAIN_SCENE*TREE*$root*$child*$grandchild
```

This addresses the container named "grandchild" which is a child of "child", which is a child of "root".

Accessing Properties on a Container

Once you've addressed a container, append the property you want to access:

```
MAIN_SCENE*TREE*$Title*TRANSFORMATION*POSITION*X GET
MAIN_SCENE*TREE*$Title*GEOM*TEXT SET "Breaking News"
MAIN_SCENE*TREE*$Title*MATERIAL*DIFFUSE_COLOR SET 1.0 0.0 0.0 1.0
MAIN_SCENE*TREE*$Logo*ACTIVE SET 1
```

Copyright © 2026 Vizrt

Key Container Properties

Property Path	What it controls
*TRANSFORMATION*POSITION*X/Y/Z	3D position
*TRANSFORMATION*ROTATION*X/Y/Z	3D rotation (degrees)
*TRANSFORMATION*SCALING	Scale (can set X Y Z as a triple)
*GEOM*TEXT	Text content (on text geometry)
*MATERIAL*DIFFUSE_COLOR	Material color (R G B A, each 0.0–1.0) (Classic Pipeline)
*MATERIAL*ALPHA	Material transparency (0.0–1.0) (Classic Pipeline)
*ACTIVE	Visibility (1 = visible, 0 = hidden)
*TEXTURE*IMAGE	Assigned texture/image (Classic Pipeline)
*FUNCTION*PluginName*param	Plugin parameter
*SCRIPT*INSTANCE*VarName	VizScript variable

1.3.6 6. Getting Started — Your First Commands

Here is a step-by-step sequence to get you started. Each command is shown with what it does and what response to expect.

Step 1: Check That the Engine Is Ready

```
VERSION
    → "5.5.0.12345"

IS_RENDERER_READY
    → "1"                    (1 = ready, 0 = not ready)
```

Always Check Readiness: IS_RENDERER_READY should return "1" before you send any scene or rendering commands. After engine startup or a scene load, the renderer may need a moment to initialize.

Copyright © 2026 Vizrt

Step 2: Load a Scene

```
RENDERER SET_OBJECT SCENE*Scenes/MyLowerThird
    → ""                    (empty = success)
```

This loads the scene located at Scenes/MyLowerThird in the Graphic Hub database and puts it "on air" on the main renderer layer.

Step 3: Modify Scene Content

```
MAIN_SCENE*TREE*$Headline*GEOM*TEXT SET "Breaking News"
    → ""

MAIN_SCENE*TREE*$Subline*GEOM*TEXT SET "Live from the studio"
    → ""
```

Step 4: Trigger an Animation

```
RENDERER*STAGE*DIRECTOR*Main START
    → ""
```

This starts the director named "Main" in the scene's stage, which typically animates the lower third in.

Step 5: Read Back Values

```
MAIN_SCENE*TREE*$Headline*GEOM*TEXT GET
    → "Breaking News"

MAIN_SCENE*TREE*$Headline*TRANSFORMATION*POSITION*X GET
    → "0.0"

RENDERER*STAGE*DIRECTOR*Main*STATE GET
    → "STOPPED"              (animation finished)
```

Step 6: Animate Out and Unload

```
RENDERER*STAGE*DIRECTOR*Main CONTINUE REVERSE
    → ""
```

Copyright © 2026 Vizrt

```
RENDERER SET_OBJECT
    → ""                    (empty argument = unload current scene)
```

1.3.7 7. Practical Examples

Example 1: Complete Lower Third Workflow

This is the most common broadcast workflow — load a template, fill it with data, animate it in, update it, then animate it out.

```
-- 1. Load the template
RENDERER SET_OBJECT SCENE*Templates/LowerThird

-- 2. Fill text fields
MAIN_SCENE*TREE*$Name*FUNCTION*ControlText*input SET "John Smith"
MAIN_SCENE*TREE*$Title*FUNCTION*ControlText*input SET "Senior Correspondent"

-- 3. Animate in
RENDERER*STAGE*DIRECTOR*Main START

-- 4. (Wait for on-air duration...)

-- 5. Update text while on air
MAIN_SCENE*TREE*$Title*FUNCTION*ControlText*input SET "Live from London"

-- 6. Animate out
RENDERER*STAGE*DIRECTOR*Main SHOW
```

Example 2: Reading and Modifying Transformations

```
-- Read current position
MAIN_SCENE*TREE*$Logo*TRANSFORMATION*POSITION*X GET
→ "100.0"

MAIN_SCENE*TREE*$Logo*TRANSFORMATION*POSITION*Y GET
→ "50.0"

-- Move the logo
MAIN_SCENE*TREE*$Logo*TRANSFORMATION*POSITION*X SET -200.0
MAIN_SCENE*TREE*$Logo*TRANSFORMATION*POSITION*Y SET 300.0

-- Rotate 45 degrees around Z axis
MAIN_SCENE*TREE*$Logo*TRANSFORMATION*ROTATION*Z SET 45.0

-- Scale to double size
MAIN_SCENE*TREE*$Logo*TRANSFORMATION*SCALING SET 2.0 2.0 2.0
```

Copyright © 2026 Vizrt

Example 3: Show/Hide Containers

```
-- Hide a container (and all its children)
MAIN_SCENE*TREE*$SponsorLogo*ACTIVE SET 0

-- Show it again
MAIN_SCENE*TREE*$SponsorLogo*ACTIVE SET 1
```

Example 4: Changing Colors

```
-- Set material to red (R=1, G=0, B=0, A=1)
MAIN_SCENE*TREE*$Bar*MATERIAL*DIFFUSE_COLOR SET 1.0 0.0 0.0 1.0

-- Make it semi-transparent
MAIN_SCENE*TREE*$Bar*MATERIAL*ALPHA SET 0.5
```

Example 5: Working with Multiple Directors

Scenes often have multiple directors for different animation segments:

```
-- Check director state
RENDERER*STAGE*DIRECTOR*InAnimation*STATE GET
→ "STOPPED"

-- Start the "in" animation
RENDERER*STAGE*DIRECTOR*InAnimation START

-- Later, start the "out" animation
RENDERER*STAGE*DIRECTOR*OutAnimation START

-- Or use SHOW to jump to the end instantly
RENDERER*STAGE*DIRECTOR*InAnimation SHOW
```

Example 6: Taking Snapshots

```
-- Capture the current render to a file
RENDERER SNAPSHOT "C:\Snapshots\frame001.png" RGBA 1920 1080
```

Copyright © 2026 Vizrt

Example 7: Monitoring Engine Performance

```
RENDERER*CURRENT_FPS GET
→ "50.0"

RENDERER*RENDER_TIME GET
→ "4.2"                    (milliseconds per frame)

MAIN*GPU_MEMORY GET
→ "2048"                   (MB)

MAIN*RAM_USAGE GET
→ "4096"                   (MB)
```

Example 8: Shared Memory (Real-Time Data Injection)

Shared memory lets external systems inject data that running scenes can pick up automatically:

```
-- Set a shared memory value (via command interface)
VIZ_COMMUNICATION*MAP SET_ELEMENT "score_home" "3"
VIZ_COMMUNICATION*MAP SET_ELEMENT "score_away" "1"

-- Read back
VIZ_COMMUNICATION*MAP GET_ELEMENT "score_home"
→ "3"

-- Clear all shared memory
VIZ_COMMUNICATION*MAP CLEAR
```

High-Volume Data: For high-frequency data updates (for example, live sports telemetry), use the dedicated **SHM TCP port (** `3001` **)** or **SHM UDP port (** `3000` **)** instead of the command interface. Commands sent via port `6100` block the render queue and can cause frame drops.

Example 9: Batch Mode (Multiple Commands, One Response)

When you need to execute several commands atomically:

```
MYID BATCH START
-1 MAIN_SCENE*TREE*$Score_Home*GEOM*TEXT SET "3"
-1 MAIN_SCENE*TREE*$Score_Away*GEOM*TEXT SET "1"
-1 MAIN_SCENE*TREE*$Timer*GEOM*TEXT SET "87:32"
-1 BATCH STOP
```

Copyright © 2026 Vizrt

```
    → ""    (single response for the entire batch with MYID as answer ID)
```

All commands inside BATCH BEGIN / BATCH END execute sequentially in the engine, but you only receive one response at the end. This reduces network round-trips and ensures all updates appear in the same frame. It is important to send the consecutive lines after `BATCH START` with an ID of -1.

Example 10: Flicker-Free Multi-Command Updates

To update multiple properties without the audience seeing intermediate states:

```
    -- Pause rendering updates
    RENDERER*UPDATE SET 0

    -- Make all your changes
    MAIN_SCENE*TREE*$Name*GEOM*TEXT SET "Jane Doe"
    MAIN_SCENE*TREE*$Title*GEOM*TEXT SET "Weather Presenter"
    MAIN_SCENE*TREE*$Photo*TEXTURE*IMAGE SET IMAGE*Photos/jane

    -- Resume rendering – all changes appear in one frame
    RENDERER*UPDATE SET 1
```

1.3.8 8. Working with Multiple Engine Instances

When multiple Viz Engine instances run on the same machine, each instance offsets its ports:
`Port = Base Port + (Instance Number - 1)`

Instance	Command Port	REST Port	Start Command
1 (default)	6100	61000	viz.exe -u1
2	6101	61001	viz.exe -u2
3	6102	61002	viz.exe -u3

To target instance 3, connect to port `6102` instead of `6100`.

1.3.9 9. Error Handling Best Practices

Always Check for ERROR in Responses

```
response = send_command("MAIN_SCENE*TREE*$NonExistent*POSITION GET")
    # response might be: "ERROR: Unknown container"
```

Copyright © 2026 Vizrt

The Engine Will Not Crash on Bad Input

Viz Engine is designed to handle invalid commands gracefully:

What you send	What happens
Wrong argument type	Silently ignored or returns ERROR
Missing required arguments	Returns ERROR message
Extra arguments	Extra arguments are ignored
Invalid object path	Returns "Unknown container" error
Out-of-range number	Clamped to valid range
Empty command	Returns empty or ERROR

Always Gate on Renderer Readiness

After engine startup or loading a scene, always wait for the renderer to be ready:

```
while True:
    response = send_command("MAIN IS_RENDERER_READY")
    if response == "1":
        break
    time.sleep(0.1)
```

Clean Up After Yourself

```
-- Close the scene from memory
SCENE*Scenes/MyLowerThird CLOSE

-- Release GPU resources (textures, geometry)
POOLS CLEANUP
```

Without POOLS CLEANUP, textures and geometry from unloaded scenes remain in GPU memory.

1.3.10 10. Quick Reference — Most Useful Commands

Engine Status

Command	Returns

Copyright © 2026 Vizrt

MAIN VERSION	Engine version string
MAIN IS_RENDERER_READY	"1" if ready, "0" if not
MAIN IS_ON_AIR GET	"1" if on air
RENDERER*CURRENT_FPS GET	Current FPS

Scene Loading

Command	What it does
RENDERER SET_OBJECT SCENE*path	Load scene on air
RENDERER SET_OBJECT	Unload current scene
SCENE*path EXIST	Check if scene exists in database
SCENE*path LOAD	Pre-load into memory (not on air)
SCENE*path CLOSE	Release from memory

Scene Content

Command	What it does
MAIN_SCENE*TREE*$name*GEOM*TEXT SET "text"	Set text content
MAIN_SCENE*TREE*$name*GEOM*TEXT GET	Read text content
MAIN_SCENE*TREE*$name*ACTIVE SET 0/1	Hide/show container
MAIN_SCENE*TREE*$name*TRANSFORMATION*POSITION*X SET val	Set position
MAIN_SCENE*TREE*$name*MATERIAL*DIFFUSE_COLOR SET R G B A	Set color

Animations

Command	What it does
RENDERER*STAGE START DEFAULT	Start default director

Copyright © 2026 Vizrt

RENDERER*STAGE*DIRECTOR*name START	Start named director
RENDERER*STAGE*DIRECTOR*name SHOW	Jump to director end
RENDERER*STAGE*DIRECTOR*name CONTINUE	Resume paused director
RENDERER*STAGE*DIRECTOR*name*STATE GET	Get state: RUNNING, STOPPED, PAUSED

Plugin Parameters

Command	What it does
*FUNCTION*ControlText*input SET "text"	Set ControlText value
*FUNCTION*ControlImage*source SET "IMAGE*path"	Set ControlImage
*FUNCTION*Toggle*switch INVOKE	Toggle visibility
*FUNCTION*PluginName*param GET	Read any plugin param

Shared Memory

Command	What it does
VIZ_COMMUNICATION*MAP SET_ELEMENT "key" "val"	Set distributed SHM
VIZ_COMMUNICATION*MAP GET_ELEMENT "key"	Read distributed SHM
MAIN_SCENE*MAP SET_ELEMENT "key" "val"	Set scene-local SHM
GLOBAL*MAP SET_ELEMENT "key" "val"	Set system-wide SHM

Cleanup

Command	What it does
POOLS CLEANUP	Release all unused resources
POOLS*TEXTURE CLEANUP	Release unused textures only

Copyright © 2026 Vizrt

1.3.11 11. Common Mistakes to Avoid

Mistake	Correct Approach
Using dots as delimiter: MAIN_SCENE.TREE.$Title	Use asterisks: MAIN_SCENE*TREE*$Title
Forgetting the $ prefix: MAIN_SCENE*TREE*Title	Always prefix names: MAIN_SCENE*TREE*$Title
Forgetting the newline at end of command	Every command must end with \n
Connecting to port 6100 for instance 3	Use port 6102 (6100 + 3 - 1)
Not closing TCP connections	Always close — 255 max; leaks block everyone
Expecting JSON error responses	Check for ERROR substring in plain text
Not calling POOLS CLEANUP after unloading scenes	GPU memory accumulates without cleanup
Not checking IS_RENDERER_READY	Always wait for readiness after startup/load
Sending high-volume data via port 6100	Use SHM ports (3001 TCP / 3000 UDP) instead

1.3.12 12. Further Reading

- **Commands.xml self-documentation**: Run viz.exe --doc to generate a complete XML file of every available command with argument types and ranges. Output: *C:\ProgramData\vizrt\VizEngine\commands.xml*
- **REST API**: Viz Engine also exposes an HTTP API on port 61000 for stateless control.
- **WebSocket**: A WebSocket interface is available on port 6900 for bidirectional web-based control.
- **Shared Memory ports**: For high-frequency data, connect to TCP port 3001 (reliable) or UDP port 3000 (fast, unreliable) using the pipe-delimited format: key|value\0

Copyright © 2026 Vizrt

1.4 Property Path Guide

Viz Engine exposes scene properties through **various access methods** (such as TCP commands, VizScript, or DataPool). Each has its own path syntax addressing the same underlying data. This chapter explains how paths are constructed, how to translate between the three systems, and how to resolve plug-in parameters.

1.4.1 Scene Tree Structure

```
SCENE (root)
   └──── TREE (scene root container)
         ├──── $Background
         │    ├──── $BG_Image
         │    └──── $BG_Color
         ├──── $Content
         │    ├──── $Header
         │    │    └──── $HeaderText
         │    └──── $Body
         │       ├──── $Row1
         │       └──── $Row2
         └──── $Foreground
                └──── $Logo
```

Containers are addressed by **name** (prefixed with `$` in TCP, plain in VizScript) or by **index** (`#n` , 1-based).

1.4.2 TCP Command Path Syntax

The universal pattern:

```
MAIN_SCENE*TREE*$<ContainerName>*<PROPERTY> <ACTION> [<VALUE>]
```

The `*` character is the path delimiter. Segments can be combined freely:

Component	Syntax	Description
Scene reference	`MAIN_SCENE`	Currently loaded main scene (underscore, not `*`)
Tree root	`TREE`	Root of the container hierarchy
Container by name	`$ContainerName`	Name lookup (case-sensitive)
Container by index	`#1` , `#2`	Child index, 1-based

Copyright © 2026 Vizrt

Component	Syntax	Description
Property	`*POSITION`	Property node access
Sub-property	`*POSITION*X`	Nested property

1.4.3 Common TCP Property Paths

What	TCP Path Suffix	Example Value
Position XYZ	`*POSITION`	`SET 100.0 50.0 0.0`
Position X only	`*POSITION*X`	`SET 100.0`
Rotation	`*ROTATION`	`SET 0.0 0.0 45.0`
Scaling	`*SCALING`	`SET 1.0 2.0 1.0`
Visibility	`*ACTIVE`	`SET ON` / `SET OFF`
Alpha	`*MATERIAL*ALPHA`	`SET 0.5`
Diffuse color	`*MATERIAL*DIFFUSE`	`SET 1.0 0.0 0.0 1.0`
Text content	`*GEOM*TEXT`	`SET "Hello"`
Plugin parameter	`*FUNCTION*<Plugin>*<Param>`	`SET 42`

Examples

```
' By name — nested path through hierarchy
MAIN_SCENE*TREE*$Content*$Header*$HeaderText*GEOM*TEXT SET "Breaking News"

' By index — 1st child of TREE, then 2nd child, then 1st child
MAIN_SCENE*TREE*#1*#2*#1*GEOM*TEXT SET "Title"

' Mixed — name and index combined
```

Copyright © 2026 Vizrt

```
MAIN_SCENE*TREE*$Content*#1*$HeaderText*GEOM*TEXT SET "Mixed path"

' Read a property
MAIN_SCENE*TREE*$Header*POSITION GET
MAIN_SCENE*TREE*$Header*ACTIVE GET
```

⚠ **Note: MAIN_SCENE** uses underscore. Always write `MAIN_SCENE` (with underscore), never `MAIN*SCENE` (with asterisk). The asterisk is the path delimiter and breaks the command.

1.4.4 VizScript Property Access

VizScript uses **dot notation** on object references. Container names are *plain strings* without the `$` prefix.

Finding Containers

```
' Scene-wide search by name (depth-first, returns FIRST match)
dim c as Container
c = Scene.FindContainer("HeaderText")

' Relative: search children of a specific container
dim child as Container
child = this.FindSubContainer("Label")

' Tree navigation
dim parent as Container
parent = this.ParentContainer

dim first as Container
first = this.FirstSubContainer

dim next as Container
next = this.NextContainer

' Scene root
dim root as Container
root = Scene.RootContainer
```

FindContainer vs FindSubContainer `Scene.FindContainer("X")` searches the *entire* scene tree (depth-first). `this.FindSubContainer("X")` searches only *descendants* of the current container. Use `FindSubContainer` when you know the relative position, it is faster and avoids name collisions.

Property Dot-chains

```
' Transform properties
```

Copyright © 2026 Vizrt

```
c.Position.X = 100.0
c.Position.Y = -50.0
c.Rotation.Z = 45.0
c.Scaling.X = 2.0

' Visibility
c.Active = True

' Material (see chapter 7 for full coverage)
c.Material.Color = col        ' CColor – diffuse
c.Material.Alpha = 0.5        ' Double – transparency
c.Material.Specular.Red = 1.0
c.Material.Emission.Green = 0.1
c.Material.Shininess = 0.8

' Geometry / text
c.Geometry.Text = "Breaking News"
```

Plug-in Parameters from VizScript

```
' Access a function plugin by name
dim fn as PluginInstance
fn = c.GetFunctionPluginInstance("ControlText")

' Read / write parameters by name
fn.SetParameterString("value", "Hello World")
dim val as String
val = fn.GetParameterString("value")

' Access geometry plugin
dim gp as PluginInstance
gp = c.GetGeometryPluginInstance()
gp.SetParameterDouble("width", 200.0)

' Access another container's SCRIPT plugin
dim sp as ScriptPluginInstance
sp = this.FindSubContainer("F1").ScriptPluginInstance
sp.SetParameterDouble("topTransition", 0)
```

Batch Plug-in Updates

```
' Collect plugin instances into an array for bulk operations
dim arrows as Array[PluginInstance]
arrows.Push(this.FindSubContainer("ArrowFill").GetGeometryPluginInstance())
arrows.Push(this.FindSubContainer("Cylinder").GetGeometryPluginInstance())

dim i as Integer
for i = 0 to arrows.Size - 1
```

Copyright © 2026 Vizrt

```
    arrows[i].SetParameterDouble("Point1X", 10.0)
    arrows[i].SetParameterDouble("Point1Y", 20.0)
next
```

1.4.5 DataPool Path Syntax

DataPool uses **dot-separated** paths. Container names appear *without* the `$` prefix:

```
DataPool.Scene.<ContainerName>.<Property>
```

DataPool Path	Equivalent TCP Path
`Scene.Header.Position.X`	`MAIN_SCENE*TREE*$Header*POSITION*X`
`Scene.Header.Active`	`MAIN_SCENE*TREE*$Header*ACTIVE`
`Scene.HeaderText.Geom.Text`	`MAIN_SCENE*TREE*$HeaderText*GEOM*TEXT`
`Scene.Logo.Material.Alpha`	`MAIN_SCENE*TREE*$Logo*MATERIAL*ALPHA`
`Scene.MyCtrl.Function.Ctrl.Value`	`MAIN_SCENE*TREE*$MyCtrl*FUNCTION*Ctrl*VALUE`

1.4.6 Cross Reference: Same property, Three Methods

These three code blocks do *exactly* the same thing: Set the X position of a container named "Header" to 100.

TCP

```
MAIN_SCENE*TREE*$Header*POSITION*X SET 100.0
```

VizScript

```
dim c as Container
c = Scene.FindContainer("Header")
c.Position.X = 100.0
```

Copyright © 2026 Vizrt

DataPool

```
DataPool.Scene.Header.Position.X = 100.0
```

More cross-references

Operation	TCP Suffix	VizScript	DataPool
Set text	`*GEOM*TEXT SET "val"`	`c.Geometry.Text = "val"`	`Scene.X.Geom.Text`
Visibility	`*ACTIVE SET ON`	`c.Active = True`	`Scene.X.Active = 1`
Alpha	`*MATERIAL*ALPHA SET 0.5`	`c.Material.Alpha = 0.5`	`Scene.X.Material.Alpha`
Plug-in param	`*FUNCTION*P*key SET val`	`fn.SetParameterString("key","val")`	`Scene.X.Function.P.key`

1.4.7 Naming Rules and Escaping

- Container names should use **alphanumeric characters and underscores** only.
- The `*` character is **reserved** as the TCP path delimiter, never use it in container names.
- The `$` prefix is required **only in TCP** paths, VizScript and DataPool use plain names.
- Spaces in names are supported but may break TCP commands, avoid them.
- `FindContainer` returns the **first match** (depth-first). Use unique names to avoid ambiguity.
- Always guard `FindSubContainer` results: `If c.VizId > 0 Then`.

Copyright © 2026 Vizrt

2 New in Viz Engine

These are some of the highlights, alongside a vast number of smaller improvements listed in the release notes.

2.1 Gaussian Splatting - Reality, Rendered in Real Time



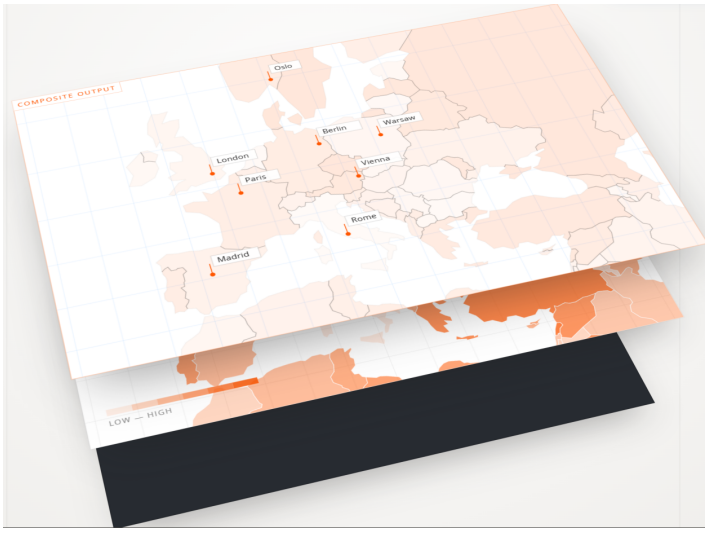
Photorealism used to be a trade-off: weeks of modelling, or "close enough." Gaussian Splatting in Viz Engine removes it. Instead of rebuilding a location, you capture it.

In **virtual studios**, set extensions and backdrops stop looking rendered and start looking photographed — holding their perspective and parallax under a moving tracked camera. In **augmented reality**, the payoff is the reverse: AR elements finally share the visual language of the real footage beside them, and the illusion holds.

Capture instead of model. Photoreal by origin. Real time, on air. Capture once, reuse everywhere.

Copyright © 2026 Vizrt

2.2 Introducing Maps - Every story happens somewhere



Maps in Viz Engine put the whole planet inside your graphics — live, navigable, and styled like everything else you put to air. Not a still someone ordered this morning. Not a browser bolted onto a texture. A real map, rendered in the scene, that you can fly.

Pick a place and go. The camera dives from orbit to a rooftop in one unbroken move, your graphics stay pinned to the ground beneath it, and the whole thing lights and composites with the rest of your design because it *is* part of your design.

2.3 Get started - Introducing Template Library



Viz Artist now includes a **Template Library**, available from the Templates tab of the Welcome Screen.

Browse a growing collection of scenes built by Vizrt, download what you need, and use it as a starting point for your own graphics or as a reference for how specific features work.

Copyright © 2026 Vizrt

2.4 New Logicmaker



Build, preview and check complex Transition Logic pages right inside Viz Artist, in a fast native panel connected directly to the engine.

Layers and states at your fingertips. Click a state to take it animated,
A live state graph. Takes run along the arrows for the real transition duration.
Scenes at a glance. Thumbnails, layer filters and search, with warnings on scenes whose Control Object doesn't match.
Guided setup. A wizard builds the master and leaf scenes from scratch for you.
Health checks and safer edits. Checks explain each problem in plain language and assist in fixing issues

Viz Engine	**Viz Artist**	Integrations
• Realtime Maps rendering • Gaussian Splatting Rendering • Gradient Editor • Security Upgrades	• More modern User Interface • Welcome Screen and Template Library • New Logicmaker • Pools Inspector • Workspace Manager • Quick Look • New Treeview Search	• Unreal Engine 5.7 and 5.8 support • NDI HX support

Copyright © 2026 Vizrt

3 WIBU Licensing System

This chapter describes management and usage of the licensing system based on CodeMeter from WIBU Systems.

This section contains the following information:

- Important Pre-installation Information
- Key Features and Workflow of the Wibu Licensing System
- Limitations and Known Issues
- Basic Set-up
- Configuration Settings
  - Why Do We Need to Configure the Licenses?
  - Configuring the License System in the Viz Engine Configuration UI
  - Configuration Entries in the Viz Engine Configuration File
    - Licensing Unreal Blade Core
    - WIBU Additional Licenses for Engine Core Licenses
    - WIBU Additional Licenses for Preview Core Licenses
    - WIBU Additional Licenses for Frameserver Core Licenses
    - WIBU Additional Licenses for Viz Libero Core License
    - WIBU Additional Licenses for Viz Arena Core License
    - WIBU Additional Licenses for Viz Eclipse Core License
- WIBU License System
  - Summary of Key Concepts in the Licensing System
  - WIBU License Share Status
  - WIBU Core License Summary
  - Core Licenses and Their Effect
  - Additional Licenses
- Notes
  - Colors used in License Overview
  - Starting Viz Engine with MezzIP OUT and/or 1xMezzIP IN
  - Permanent Licenses
- Setup Graphic Hub Journal-based E-Mail Notifications
- Setup Redundancy for a WIBU License Server
  - Set Up Redundancy for a WIBU License Server
  - Test License Server Failover
  - Recover First License Server
  - Notes

3.1 Important Pre-installation Information

The WIBU licensing system requires the installation of the CodeMeter Runtime Software.
This needs to be done separately and is no longer part of the bundle installer.

Copyright © 2026 Vizrt

Tested WIBU Runtime	Viz Engine Version
9.10	5.6
8.40a	5.5
8.30a	5.4
8.20	5.3.1, 5.3.2
8.10a	5.3.0
7.60a	5.1, 5.2
7.40	4.4.1, 5.0
7.21a	4.3
7.20	4.1
6.80	3.12.1
6.60a	3.10.0 - 3.12.0

When the license should be retrieved from a dedicated license server, it must be configured in the Vizrt Licensing Service (see **Client Configuration** page in the **License Server Setup and Administration** section of the Vizrt Licensing Administrator Guide) or the CodeMeter WebAdmin.

Information: There is an auto discovery if no license server is configured in the server search list of CodeMeter.

Information: On network disconnect and reconnect, it may happen that a license is checked out twice. In this case, it must be released manually on the CodeMeter service on the license server or the license server can be restarted.

Note: Prior to upgrading any version it is highly recommended to create a backup of the Viz Engine configuration files located in: *%PROGRAMDATA%\vizrt\VizEngine.*

3.2 Key Features and Workflow of the Wibu Licensing System

- Dongle-less operation on clients with monitoring and logging capabilities.
- Grace periods for allocated licenses to avoid immediate expiration on network or service interruptions.

Copyright © 2026 Vizrt

    - In Viz Engine 4.0.0/3.14.1 or newer license warnings and errors are logged to the Graphic Hub journal (see here how to enable E-Mails).
        - Logs license warnings (expiring within the next two weeks) to the Graphic Hub journal as dedicated log messages with level 820.
        - Logs license errors (any grace state or if expired) to the Graphic Hub journal as dedicated log messages with level 920.
- Configurable WIBU license container location (any local container, any network container or a dedicated container).
- Configurable licenses to consume.
- Viz Artist/Viz Engine startup was changed and no startup helpers are needed anymore (*.cmd* files, started executables).
- When a core license cannot be allocated on startup Viz Engine starts in configuration mode.
- When an additional license fails to allocate on startup an error is logged and the required feature keeps disabled on startup.

3.3 Limitations and Known Issues

- All additional licenses consumable to a core license must come from the same container as the core license.
- ⚠ Matrox DSX.Core installer corrupts an existing CodeMeter Runtime installation and causes a crash on Viz Engine startup.
    - Uninstall any CodeMeter Runtime prior to installing Matrox DSX.Core. Install DSX.Core and afterwards install Viz Engine and the newer CodeMeter Runtime.
- For a preview in TRIO and Pilot as Engine Core, a DVI OUT license is needed.
- Configuration of licenses requires to start Viz Engine in configuration mode with the configuration UI.
- In Viz Engine 3.12.1 and newer, the configuration UI works without a valid license and can be configured through the configuration UI.
    - In Viz Engine 3.12.0 and older, the configuration UI only works with a valid license, which depends on the configuration. If no proper license is available then a manual edit of the corresponding configuration entries is required. See Configuration Entries in the Viz Engine Configuration File.
    - By default, the WIBU licensing system with Engine Core license (ENG_ENG_CORE) is used when no configuration exists.
- File-based local WIBU licenses on server operating systems (terminal server) cannot be used via remote desktop connections. For example, if you have installed Windows 10 the access via remote desktop connection works but not, for example, with a Windows Server 2016 system.
- Viz Engine Startup may take long when network communication has high latency or bandwidth is limited. Workaround for now is to use a license server on a local network.
- Security groups respective Firewalls on Cloud providers might need to be configured to allow UDP traffic on port `22350`, to reach and consume licenses from a license server.

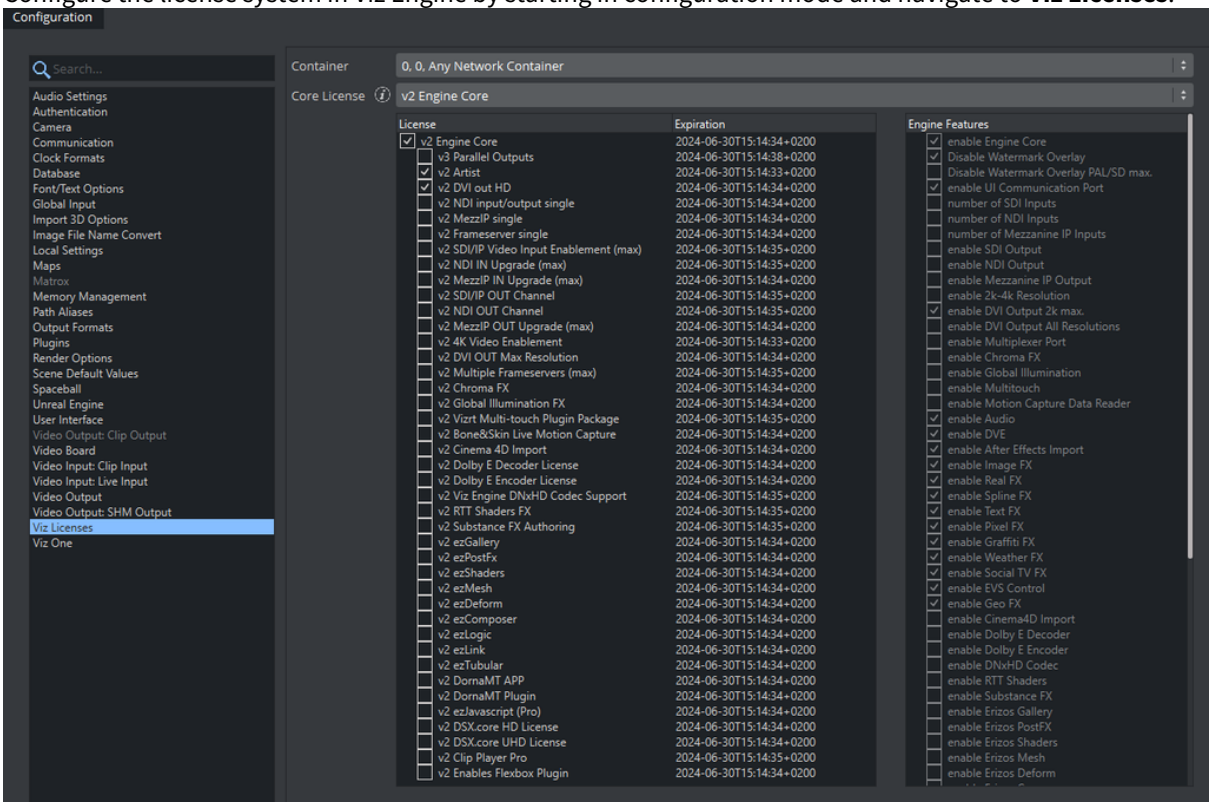
3.4 Basic Set-up

To set up Viz Artist/Viz Engine licensing with WIBU:

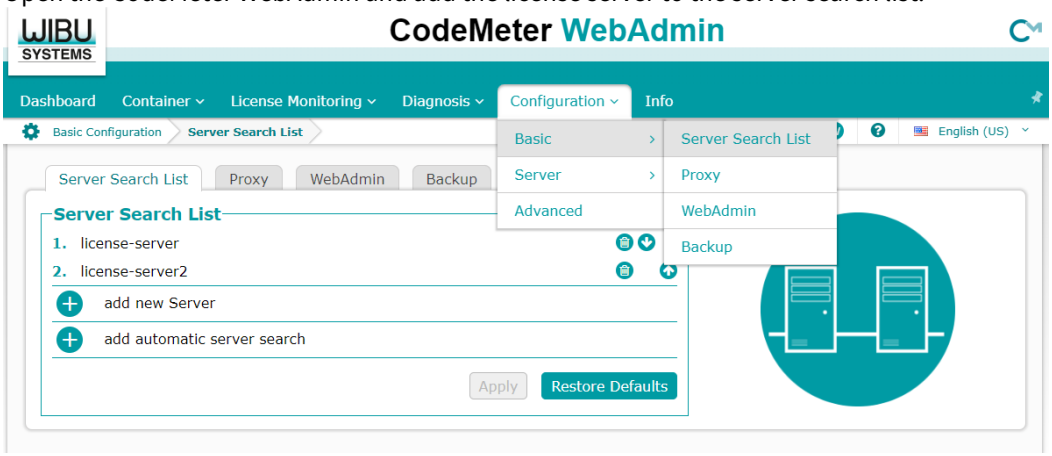
1. Install Viz Artist/Viz Engine with the bundle installer (see Viz Artist and Viz Engine Installation).

Copyright © 2026 Vizrt

2. Install the WiBu CodeMeter runtime.
3. Configure CodeMeter with the Vizrt Licensing Service or the CodeMeter WebAdmin (can be opened from the CodeMeter Control Center).
4. Configure the license system in Viz Engine by starting in configuration mode and navigate to **Viz Licenses**.



a. When using any WIBU network container (license server):
   i. Open the CodeMeter WebAdmin and add the license server to the server search list:



   ii. In the configuration UI under **Viz Licenses** set **Container** to `0,0 Any Network Container` or in the Viz Engine configuration file set `License_Containers = 0,0,1`.

iii. Configure the core license and any additional licenses. In the configuration UI under **Viz Licenses** configure **Core License** and **Additional Licenses** or in the Viz Engine configuration file configure `License_Core` and `Additional_Licenses`.

b. When using any local WIBU container (dongle, file):

i. In the configuration UI under **Viz Licenses**, set **License Location** to `0,0 Any Local Container` or in the Viz Engine configuration file set `License_Containers = 0,0,0`

ii. Configure the core license and any additional licenses. In the configuration UI under **Viz Licenses**, configure **Core License** and **Additional Licenses**or in the Viz Engine configuration file configure `License_Core` and `Additional_Licenses`.

c. When using a dedicated WIBU container (with full container identifier):

i. In the configuration UI under **Viz Licenses**, set **License Location** to a specific container (for example, `130,3910158968 Network license`) or in the Viz Engine configuration file set `License_Containers = 13 0,3910158968,1`

ii. Configure the core license and any additional licenses. In the configuration UI under **Viz Licenses**, configure **Core License** and **Additional Licenses** or in the Viz Engine configuration file configure `License_Core` and `Additional_Licenses`.

5. Start Viz Engine.

3.5 Configuration Settings

3.5.1 Why Do We Need to Configure the Licenses?

- Prevent license theft by instances on other nodes or instances on the same node. Licenses are in one container and have a count.
- Several licenses may enable the same feature. There is no one to one mapping of application feature to a single license possible. For example, with multiple instances on one node (Trio One Box), it is important which instance consumes which license for proper operation (first needs SDI OUT, SDI IN and 2nd DVI HD).
- A license may enable multiple internal application features.

3.5.2 Configuring the License System in the Viz Engine Configuration UI

See Viz License Configuration.

> **Note:** By default, the WIBU licensing system with Engine Core license is used when no configuration exists. When no such license is available the configuration file must be edited. See Configuration Entries in the Viz Engine Configuration File.

3.5.3 Configuration Entries in the Viz Engine Configuration File

Entries changed in 4.0.0 and later: `Legacy_Licensing` (removed), `License_Location` (removed, was replaced by `License_Containers`)

Copyright © 2026 Vizrt

- **License_Containers:** WIBU license source. Determines where WIBU should search for license containers and allocate licenses from.
  - Can be set to one of the following values:
    - `0,0,0` - Any Local container. Searches exclusively for containers located on the same PC or allocated to the same VM (for example, dongle, file).
    - `0,0,1` - Any Network container. License is to be sought in the network (LAN, WAN), CodeMeter License Server activated as network server or CmWAN.
    - `X,Y,1` - A dedicated container, where `X` is the container mask (for example, 130) and `Y` is the container serial (for example, 3910158968)
  - Default: `0,0,1`
- **License_Core:** WIBU core license, which determines also the available additional licenses.
  - For WIBU license details, see WIBU Licenses and their Effect.
  - Can be set to one of the following values:
    - Engine Core
    - Preview Core
    - Engine Core VER (New Render Pipeline only)
    - Preview Core VER (New Render Pipeline only)
    - Frameserver Core
    - Free Viz Artist
  - and for OEM (see Note on Shared Licenses)
    - Viz Libero Core
    - Viz Arena Camera Tracking Engine
    - Viz Eclipse Core
- **Additional_Licenses:** WIBU additional licenses that are additionally consumed and determined by the core license.
  - For WIBU license details, see WIBU Licenses and their Effect.

Note: When a license is configured it is consumed, even when it has no effect. If licenses are configured that are not allowed, an error is logged on startup.

Licensing Unreal Blade Core

Configuration details can be found in **Integration with Unreal Engine** under **Third Party Applications and Files** in the Viz Artist User Guide.

WIBU Additional Licenses for Engine Core Licenses

License	Description
Viz Artist	Enables Viz Artist User Interface
DVI OUT HD	Enables Render Preview in OnAir mode

Copyright © 2026 Vizrt

License	Description
NDI input/output single	Enables NDI in and Out channel for one instance
MezzIP single	Enables Mezzanine in/out channel for one instance
Frameserver single	Enables Multiplexer communication ports for one instance
SDI/IP Video Input Enablement (max)	Enables all SDI/IP input channels as well as NDI inputs
NDI IN Upgrade (max)	Enables all NDI input channels
MezzIP IN Upgrade (max)	Enables all mezzanine IP inputs
SDI/IP OUT Channel	Enables SDI/IP output as well as NDI output
NDI OUT Channel	Enables NDI output
MezzIP OUT Upgrade (max)	Enables mezzanine IP output
4K Video Enablement	Allows rendering in UHD
8K Video Enablement	Allows rendering up to UHD2
DVI OUT Max Resolution	Enables Fullscreen output (Videowall)
Multiple Frameservers (max)	Enables Multiplexer communication ports for all instances
PrecisionKeyer	Enables Chroma Keyer
Global Illumination FX	Enables Global Illumination Editing
Vizrt Multi-touch Plug-in Package	Enables Multitouch plug-ins
Cinema 4D Import	Allows import of C4D files
Dolby E Decoder License	Allows processing of Dolby E audio
Dolby E Encoder License	Allows output of Dolby E audio
Viz Engine DNxHD Codec Support	Allows playback/encoding of DNxHD compressed video

Copyright © 2026 Vizrt

License	Description
RTT Shaders FX [1]	Enables RTT shader package
Substance FX Authoring	Allows editing of Substance Materials
ezGallery [1]	Enables Erizos plug-in
ezPostFx [1]	Enables Erizos plug-in
ezShaders [1]	Enables Erizos plug-in
ezMesh [1]	Enables Erizos plug-in
ezDeform [1]	Enables Erizos plug-in
ezComposer [1]	Enables Erizos plug-in
ezLogic [1]	Enables Erizos plug-in
ezLink [1]	Enables Erizos plug-in
ezTubular [1]	Enables Erizos plug-in
ezJavascript (Pro) [1]	Enables Erizos plug-in
DornaMT APP [1]	Enables Dorna Multitouch Application
DornaMT Plug-in [1]	Enables Dora Multitouch plug-in
DSX Core HD license	Enables DSX Core functionality up to HD resolution
DSX Core UHD License	Enables DSX Core functionality up to UHD resolution
ClipPlayer Pro	Enables playback of licensable Video Codecs like h265 or HAP
Enables Flexbox plug-in	Enables Flexbox plug-in for Adaptive Scenedesign
Parallel Outputs	Enables multiple outputs

Copyright © 2026 Vizrt

License	Description
Object Tracker	Enables Object Tracker
Shared Memory Input	Enables Shared Memory Inputs
Shared Memory Output	Enables Shared Memory Outputs
Gaussian Splatting	Enable Gaussian Splat rendering

(1) Items are not available in the Engine VER core.

WIBU Additional Licenses for Preview Core Licenses

License	Description
Viz Artist Preview	Enables Viz Artist User Interface
Cinema 4D Import	Allows import of C4D files
Dolby E Decoder License	Allows processing of Dolby E audio
Dolby E Encoder License	Allows output of Dolby E audio
Viz Engine DNxHD Codec Support	Allows playback/encoding of DNxHD compressed video
RTT Shaders FX (1)	Enables RTT shader package
ezGallery (1)	Enables Erizos plug-in
ezPostFx (1)	Enables Erizos plug-in
ezShaders (1)	Enables Erizos plug-in
ezMesh (1)	Enables Erizos plug-in
ezDeform (1)	Enables Erizos plug-in
ezComposer (1)	Enables Erizos plug-in
ezLogic (1)	Enables Erizos plug-in

Copyright © 2026 Vizrt

License	Description
ezLink [1]	Enables Erizos plug-in
ezTubular [1]	Enables Erizos plug-in
ezJavascript (Pro) [1]	Enables Erizos plug-in
DornaMT APP [1]	Enables Dorna Multitouch Application
DornaMT Plug-in [1]	Enables Dora Multitouch plug-in
DSX Core HD license	Enables DSX Core functionality up to HD resolution
DSX Core UHD License	Enables DSX Core functionality up to UHD resolution
ClipPlayer Pro	Enables playback of licensable Video Codecs like h265 or HAP
Parallel Outputs	Enables multiple outputs
Object Tracker	Enables Object Tracker

[1] Items are not available in the Engine VER core

WIBU Additional Licenses for Frameserver Core Licenses

License	Description
Frameserver single	Enable Frameserver functionality
4K Enablement	Allows rendering in UHD
Multiple Frameservers (max)	Enables Multiplexer communication ports for all instances

WIBU Additional Licenses for Viz Libero Core License

- Viz Engine DNxHD Codec Support

WIBU Additional Licenses for Viz Arena Core License

- Viz Engine DNxHD Codec Support

Copyright © 2026 Vizrt

WIBU Additional Licenses for Viz Eclipse Core License

- Viz Engine DNxHD Codec Support

3.6 WIBU License System

3.6.1 Summary of Key Concepts in the Licensing System

- Core- and additional licenses exist.
- OUT based channel counting for NDI/SDI/IP
- IN and OUT channels are resolution independent up to 3G HD
  - <=2K size of render output
- Preview channels (preview/clean feed OUT) are co-enabled with every OUT channel license.
- MezzIP applies to all compressed streams:
  - Included: RTP/UDP/WebRTC
  - Excluded (requires SDI/IP licenses): SMTPE ST2022-06, ST2022-07, ST2110.
- Multiple licenses may enable the same application feature!
- Licenses are consumed by multiple processes and license share mode influences possible consumption (see WIBU License Share Status).

3.6.2 WIBU License Share Status

Refers to a WIBU license and how its usage is counted when it is allocated/consumed.

- **Exclusive:** Allocation can only be performed once on a node (=machine).
- **User Limit:** Each allocation is counted, independent of instance and node.
- **Shared:** Each allocation is counted once per node. Multiple allocations on the same node allocate only a single license.
- **No User Limit:** Only checked for existence, no counting is performed.

License count reduction when consumed on a node in an instance						
License share status	**Node 1**			**Node 2**		
	Instance 1	**Instance 2**	**Instance 3**	**Instance 1**	**Instance 2**	**Instance 3**
Exclusive	-1	fails	fails	-1	fails	fails
User Limit	-1	-1	-1	-1	-1	-1
Shared	-1	0	0	-1	0	0

Copyright © 2026 Vizrt

License count reduction when consumed on a node in an instance						
License share status	**Node 1**			**Node 2**		
	Instance 1	**Instance 2**	**Instance 3**	**Instance 1**	**Instance 2**	**Instance 3**
No User Limit	0	0	0	0	0	0

3.6.3 WIBU Core License Summary

The following table represents which features are included in one of the Core features ✓
and what additional feature might need to be configured ⊕
or are not available ✗

License	Engine (VER) Core	Preview (VER) Core	Frameserver Core	Viz Artist for Learners (Free Artist)	Viz Arena Viz Libero Viz Eclipse
Artist	⊕	⊕ Artist Preview	✗	✓	✗
DVI Out HD (<=2K)	⊕	✓ (Watermark on Resolution >=PAL)	✗	✓ Watermark	✓
DVI Out Max Resolution (Videowall)	⊕	✓ (Watermark)	✗	✓ (Watermark)	✓
NDI input/output single NDI in Upgrade NDI Out Channel	⊕	✗	✗	✗	✗
MezzIP single MezzIP IN Upgrade MezzIP Out Upgrade	⊕	✗	✗	✗	✗

Copyright © 2026 Vizrt

Frameserver single Multiple Frameservers	⊕	✗	✓	✗	✗
SDI/IP Video Input Enablement	⊕	✓	✗	✗	✓
SDI/IP Output Channel	⊕	✓ (Watermark)	✗	✗	✓
NDI Input/Output single	⊕	✓ (Watermark)	✗	✗	✓
NDI IN Upgrade	⊕	✓	✗	✗	✓
NDI Out Channel	⊕	✓ (Watermark)	✗	✗	✗
4K Enablement	⊕	✓ (Watermark on Resolution >=PAL)	⊕	✓ (Watermark)	✓
8K Video Enablement	⊕	✓ (Watermark on Resolution >=PAL)	✗	✓ (Watermark)	✗
8K+ Video Enablement	⊕	✓ (Watermark on Resolution >=PAL)	✗		
Chroma Keyer	⊕	✓	✗	✗	✓
Global Illumination	✓	✓	✗	✗	✗
Multitouch Plug-ins	⊕	✗	✗	✗	✗
Bone&Skin Live Motion Capture	⊕	✓	✗	✗	✗
Cinema 4D Import	⊕	⊕	✗	✗	✗

Copyright © 2026 Vizrt

Dolby E Encoding	✚	✚	✖	✖	✖
Dolby E Decoding	✚	✚	✖	✖	✖
DNxHD Codec Support	✚	✚	✖	✖	✚
RTT Shaders [1]	✚	✚	✖	✖	✖
Substance FX Authoring	✔	✔	✖	✖	✖
Erizos Plug-ins [1]	✚	✚	✖	✖	✖
Dorna Plug-ins [1]	✚	✚	✖	✖	✖
DSX Core HD license	✚	✚	✖	✖	✖
DSX Core UHD license	✚	✚	✖	✖	✖
ClipPlayer Pro	✚	✚	✖	✖	✖
Flexbox plug-in	✚	✔	✔	✖	✖
Parallel Outputs	✚	✔	✖	✖	✖
Object Tracker	✚	✚	✖	✖	✖
Shared Memory Input	✚	✔	✖	✖	✔
Shared Memory Output	✚	✔	✖	✖	✔
Gaussian Splatting	✚	✔	✖	✖	✖

[1] Items are not available in the Engine VER core

3.6.4 Core Licenses and Their Effect

A WIBU license is a dedicated license (application ID) in the WIBU license system.

Copyright © 2026 Vizrt

Core Licenses	Core licenses are used as prerequisite for other licenses and define the basic feature set.	
Engine Core **Engine Core VER** enables the same functionality, but disables Classic pipeline	Shared	Basic core license to be able to start Viz Engine. Included functionality: • Viz Engine Render pipeline and Classic Pipeline for up to 2K resolution. • Clips / Clip Player • No limits on Clip Player (incl. broadcast format clip playback, of course proper Matrox Codecs required) • Standard set of Plug-ins (Image FX, Real FX, Spline FX, Text FX, Pixel FX, Graffiti FX, SocialTV, EVS and Maps Plugins) • Basic Viz Engine functionality including Audio, Video, After Effects Import,Global Illumination and Substance FX • Postrendering up to 2K

Preview Core
Preview Core VER enables the same functionality, but disables Classic pipeline
	User Limit	Usage 1: Trio / other client applications requiring local preview
		Usage 2: Standalone preview machine for DVI, Mezzanine, NDI or SDI output always with watermark
		Optional (if licensed): Artist (together with ENG_PRV_ARTIST)
		Includes rendering in all resolutions (Watermark on all OUTs at any time and all resolutions).
		• Inputs including
		  • SDI/IP
		  • NDI
		  • Mezzanine IP
		  • Shared Memory
		• Outputs enabled (no extra licenses required)
		  • SDI/IP Output
		  • NDI Output
		  • Mezzanine IP Output
		  • Shared Memory
		  • Parallel Outputs
		⚠ **Note:** No watermark on DVI OUT (local preview) and renderer snapshots when the resolution is <= 720x576 pixels (PAL, SD). Output system must be set to PAL/SD.
		⚠ **Note:** Cannot be used as Frameserver.

Copyright © 2026 Vizrt

Core Licenses	Core licenses are used as prerequisite for other licenses and define the basic feature set.	
Engine Frameserver Core	Shared	Basic core license to use Viz Engine as Frameserver (NLE) only. Needs to have a Frameserver single license enabled. Disables all watermarks. Can be upgraded to 4K resolution. ⚠ **Note:** The following communication ports are disabled with this license: • Default communication Port (6100) • GFX and Superchannel ports • Websocket command extension • Shared Memory Ports • REST Interface

Free Viz Artist / Artist for Learners	Exclusive	Free Viz Artist version core license.
		Equal to an ENG_ENG_CORE with 1 x DVI OUT (limited to 2K) and everything is watermarked, except scene editor (small window in bottom right).
		• Enabled:
		  • Artist
		• Restricted:
		  • Means no SDI/IP, NDI, RTP/UDP IN/OUT, WebRTC OUT.
		  • Watermark on all OUTs including Postrender and Clip Out.
		  • Not combinable with any other licenses
		  • No Support for Classic scenes.

Copyright © 2026 Vizrt

Core Licenses	Core licenses are used as prerequisite for other licenses and define the basic feature set.	
Libero Core	User Counted	OEM Core for Viz Libero Enabled: • Engine Core • Disable Watermark Overlay • SDI Output • NDI Output • 2k-4k Resolution • DVI Output 2k max. • DVI Output All Resolutions • Precision Keyer • unlimited SDI Inputs • unlimited NDI Inputs • Shared Memory Inputs • Shared Memory Outputs Optional: • DNxHD Codecs
Viz Arena Core	User Counted	OEM Core for Viz Arena Enabled: • Engine Core • Disable Watermark Overlay • SDI Output • NDI Output • 2k-4k Resolution • DVI Output 2k max. • DVI Output All Resolutions • Precision Keyer • unlimited SDI Inputs • unlimited NDI Inputs • Shared Memory Inputs • Shared Memory Outputs • Clip Playback • Post Renderer Optional: • DNxHD Codecs

Copyright © 2026 Vizrt

Core Licenses	**Core licenses are used as prerequisite for other licenses and define the basic feature set.**	
Viz Libero Camera Tracking Core	User Counted	OEM Core for Viz Arena Enabled: • Engine Core • Disable Watermark Overlay • SDI Output • NDI Output • 2k-4k Resolution • DVI Output 2k max. • DVI Output All Resolutions • Precision Keyer • unlimited SDI Inputs • unlimited NDI Inputs • Shared Memory Inputs • Shared Memory Outputs • Clip Playback • Post Renderer Optional: • DNxHD Codecs

3.6.5 Additional Licenses

Viz Artist Licenses		
Artist	Shared	Enables the Artist Design User Interface. This is required to create new scenes.
Artist Preview	Shared	Enables the Artist with Preview Core.
Optional Input Licenses		
SDI/IP Video Input Enablement (max)	Shared	Enables unlimited SDI or IP video inputs for all engine instances. ⚠ **Note:** Disables DSX Core HD and UHD licenses.
NDI input/output single	User Limit	Enables NDI input and NDI output for one engine instance.

Copyright © 2026 Vizrt

Viz Artist Licenses		
NDI IN Upgrade (max)	Shared	Upgrades NDI input for all engine instances.
MezzIP single	User Limit	Enables optional mezzanine in, and output like RTSP, RTP for one engine instance.
MezzIP IN Upgrade (max)	Shared	Additional RTP/UDP streams for all engine instances.
Shared Memory INPUT		Allows reading of Shared Memory Streams.
Optional Output Licenses		
SDI/IP OUT Channel	User Limit	Enables Video Output for SDI/IP hardware for **one engine instance**. Fill+Key is considered to be one channel only (even though requiring two physical outs). Parallel Output channels (preview/clean/matte etc...) are co-enabled with every SDI/IP OUT channel license item being purchased.
NDI OUT Channel	User Limit	Additional NDI streams output for **one engine instance**
MezzIP OUT Upgrade (max)	Shared	Additional RTP/UDP streams for all engine instances.
Parallel Outputs	User Limit	Enables multiple outputs from one engine instance. This can be an additional Key Preview, Matte scene, GFX channel output etc. using NDI or SDI/IP. ⚠ **Note:** An additional NDI license is not required for GFX channel NDI output.
Shared Memory OUTPUT		Allows sending of Shared Memory Streams.
Frame Server Licenses		
Frameserver single	User Limit	Enables Frame Server functionality for NLE plug-ins with this license. This enables the multiplexer ports (`50007` - `50009`) for one engine instance. Previewport `50010` is not affected.

Copyright © 2026 Vizrt

Viz Artist Licenses		
Multiple Frameservers (max)	Shared	Upgrades additional engine instances with Frame Server functionality.
Output Resolution Licenses		
DVI Out HD	User Limit	Enables OnScreen rendering up to 2K resolution. DVI OUT is required for local preview in external applications (for example, Viz Trio, Pilot Template Wizard, Pilot Director) or to enable the OnAir Preview. When configuration is changed at runtime and violates the license restriction, the resolution is limited to the allowed resolution (for example, 2K width/height). When resolution is configured higher than 2K the startup fails.
DVI OUT Max Resolution	Shared	Enables unlimited DVI output size, mainly needed for Videowall setups. ⚠ **Note:** No DVI OUT size restriction anymore.
4K Video Enablement	Shared	Enables output up to (and including) 4K (in any direction) on all output devices This affects SDI/IP, NDI, Postrendering, Snapshots, Clipout, WebRTC,.... ⚠ **Note:** Allows OUT of <=4K in both dimensions.
8K Video Enablement	Shared	Enables output up to (and including) 8K (in any direction) on all output devices This affects SDI/IP, NDI, Postrendering, Snapshots, Clipout, WebRTC,.... ⚠ **Note:** Allows OUT of <=8K in both dimensions.
8K+ Video Enablement	Shared	Enables output more than 8K (in any direction) on all output devices This affects SDI/IP, NDI, Postrendering, Snapshots, Clipout, WebRTC,.... ⚠ **Note:** Allows OUT of >=8K in both dimensions.
Clip and Codec Licenses		

Copyright © 2026 Vizrt

Viz Artist Licenses		
ClipPlayer Pro license	Shared	Allows playback of additional Clip codecs like h265 or HAP with VML player.
DSX Core HD license DSX Core UHD license	Shared	Enables the use of the Matrox Software only DSX Core solution. DSX Core also allows to use RTP/SRTSP streaming and ClipOut channels. Matrox Topology Utils need to be installed. ⚠ **Note:** Disables SDI/IP input licenses.
DNxHD Codec Support	Shared	Allows handling clips in DNxHD format on all Matrox devices.
Additional Viz Engine Licenses		
Dolby E Decoder License Dolby E Encoder License	Shared Shared	Allows to process Dolby E inputs and Dolby E outputs. A pass through of Dolby E signals does not require any of these licenses.
Cinema 4D Import	Shared	Allows to import Cinema 4D (C4D) files and to use the Cinema 4D LiveLink feature.
Precision Keyer	Shared	Enables the Chroma Keyer Feature.
Global Illumination FX	Shared	No longer requires a license in Viz Engine 5.4 or higher.
Substance FX Authoring	Shared	No longer requires a license in Viz Engine 5.4 or higher.
Bone&Skin Live Motion Capture	User Limit	Allows to capture live motion data in real-time.
Object Tracker		Enables the object tracker functionality of VizArc.
AI Keyer Greenscreen-less		Enables the use of AI Keyer.

Copyright © 2026 Vizrt

Viz Artist Licenses		
Gaussian Splatting	Shared	Enables rendering of Gaussian Splat assets. Import of .ply and .sog files, placing the object and editing the scene are not restricted. Without the license the scene opens, the splat is not rendered and a message is logged.
Plug-in Licenses		
Flexbox plug-in	User Counted	Allows to use advanced flexbox layouting capabilities for adaptive graphics use cases.
Vizrt Multi-touch Plug-in Package	Shared	Enables the use of Vizrt Multitouch plugins. Note, these plugins are disabled by default.
RTT Shaders FX	Shared	Enables the full functionality of RTT shaders bundled with Viz Engine Classic pipeline. Without license, some options are disabled.
• ezGallery • ezPostFx • ezShaders • ezMesh • ezDeform • ezComposer • ezLogic • ezLink • ezTubular • ezJavascript (Pro)	Shared	3rd party Erizos plugins for Classic Pipeline.
• DornaMT APP • DornaMT Plug-in	Exclusive	3rd party Dorna MultiTouch plugins for Classic Pipeline.

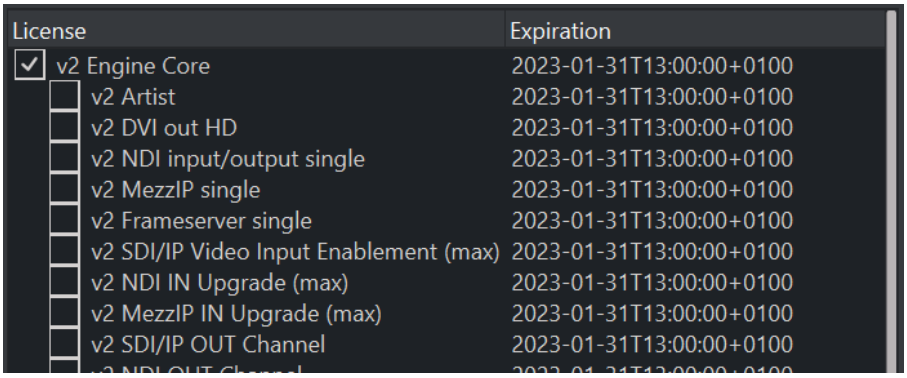
3.7 Notes

3.7.1 Colors used in License Overview

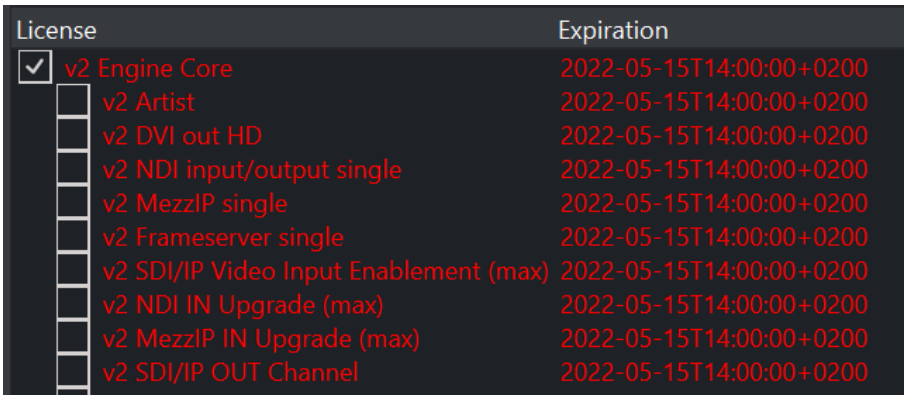
The following color schema is used in Viz Engine to display the status of the license:

Correct License (white)

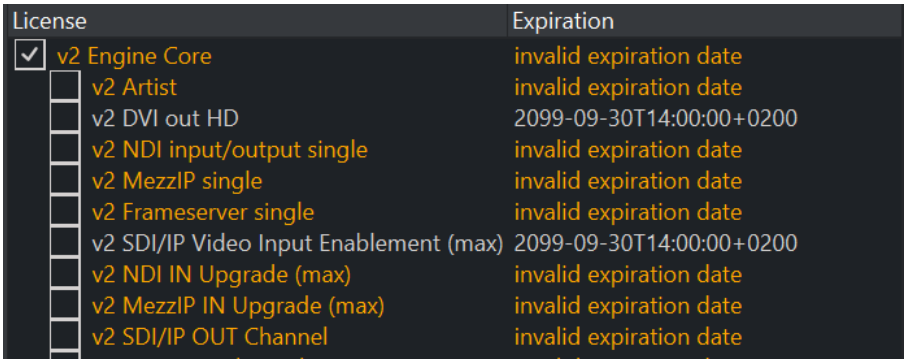
Copyright © 2026 Vizrt



Expired License (red)



Mixed "Permanent" Licenses

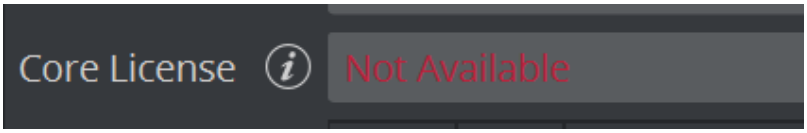


Note: A permanent license needs to have a end date set. If no end date is found, the license is invalid and should be renewed.

These licenses are marked as orange.

Core not available

Copyright © 2026 Vizrt



A previously configured Core license which is not available anymore (for example, by switching dongles) is labeled *Not Available* and shown in red.

3.7.2 Starting Viz Engine with MezzIP OUT and/or 1xMezzIP IN

When running Viz Engine with RTP/UDP IN/OUT or Service Host with WebRTC OUT the following licenses need to be available in the WIBU license system:

- OUTPUT: MezzIP Single or MezzIP Out upgrade (max)
- INPUT: MezzIP In Upgrade(max)
- See Note on Shared Licenses.

MezzIP Possible OUTs	Engine Effect	Engine Licenses Consumed
1 RTP/UDP IN (2K), 1 RTP/UDP OUT (2K)	Instance 1: 1x RTP/UDP IN (2K), 1x RTP/UDP OUT (2K)	1 ENG_ENG_CORE, ENG_CF
1+ RTP/UDP IN (2K),	1+ RTP/UDP IN (2K) (one or multiple instances)	1(+) ENG_ENG_CORE, ENG_IN_MEZZIP_MAX
1 RTP/UDP IN (2K), 1+ RTP/UDP OUT (2K)	1 RTP/UDP IN (2K)	ENG_ENG_CORE, ENG_CF, ENG_OUT_MEZZIP_MAX
1+ RTP/UDP IN (2K), 1+ RTP/UDP OUT (2K)	1+ RTP/UDP IN (2K) 1+ RTP/UDP OUT (2K)	ENG_ENG_CORE, ENG_IN_MEZZIP_MAX, ENG_OUT_MEZZIP_MAX
4K OUT	4K resolution on any OUT	ENG_ING_4KVIDEO

3.7.3 Permanent Licenses

Permanent Licenses need to have a end date set (typically 99 years). Any permanent license without an end-date is listed as "invalid date" and should be renewed by Vizrt. If this is the case, please contact Vizrt Support.

3.8 Setup Graphic Hub Journal-based E-Mail Notifications

- Configure e-mail sending in Graphic Hub Terminal Options E-Mail settings.
- Configure e-mail addresses and notification options for specific alerts in Graphic Hub Manager Alert settings.

Copyright © 2026 Vizrt

- Viz Engine directly logs to the Journal of the connected Graphic Hub and triggers e-mail messages for the following alert levels:
  - Level 820: Viz Engine WARNING: License, Warnings reported by Viz Engine related to licensing.
  - Level 920: Viz Engine ERROR: License, Errors reported by Viz Engine related to licensing.

3.9 Setup Redundancy for a WIBU License Server

Setting up WIBU license server redundancy is described below. See **License Server Redundancy** in the Vizrt Licensing Administrator Guide for additional details.

> **Note:** The grace period for acquired licenses is 72 hours if the license is lost to recover possible network issues or the license server.

3.9.1 Set Up Redundancy for a WIBU License Server

1. Make sure two hosts (A & B) have *VizrtLicensingInstaller.exe* installed and select **use as network license server** during installation. See **License Server Activation** in the Vizrt Licensing Administrator Guide.
2. Connect one WIBU dongle to 'A' and import a key with licenses.
3. From WIBU web admin UI of 'A', make sure the container exist and all the added features are listed.
4. On the client configure 'A' as first license server, and 'B' as second one (in the search list of the local CodeMeter web admin).

3.9.2 Test License Server Failover

1. Set the license configuration in Viz Engine.
2. Start Viz Engine.
3. From CodeMeter web admin UI of 'A', verify that the licenses are allocated and used by the client.
4. Disconnect the network cable of 'A' or shut down the machine.
5. Wait until the Viz Engine(s) starts writing 'License Server not found..' and the grace status for the configured features is 'grace_state=2/VL_GRACE_GRACE, '.
6. Move the WIBU dongle from 'A' to 'B'.
7. Viz Engine recovers and re-obtains the license from 'B'.

3.9.3 Recover First License Server

1. Recover license server ‘A’.
2. Disconnected the failover license server from network or shut it down.
3. Waited until the Viz Engine(s) enter grace period.
4. Moved WIBU dongle back to the original license server ‘A’.
5. Viz Engine recovers and re-obtains the license from 'A'.

3.9.4 Notes

- The license must be in the same container to recover. A second license server with a different container does not work.
- Not started Viz Engines which are configured the same, can start and obtain the license from 'B'.

Copyright © 2026 Vizrt

- An arbitrary number of servers can be added to the server search list of CodeMeter, even when Viz Engine is running. It is possible to add a failover server after the original license server failed.

See Also

- Vizrt Licensing Administrator Guide
- Graphic Hub Administrator Guide
- **Third Party Applications and Files > Integration with Unreal Engine** in the Viz Artist User Guide.

Copyright © 2026 Vizrt

4 Security Advisor

4.1 Recommendations

Vizrt's Ecosystem allows flexible integration with external services and communication with third party applications. Viz Engine offers an open API to allow any control application to integrate with. For security reasons, these installations need to be secured and one might need to disable certain services or block network communications to Viz Engine.

In addition, there are more recommendations to prevent someone in the network to misuse Viz Engine.

After installation of the software, please proceed with the following advisories:

- Change the default User password. See Prerequisites.
- Do not run Viz Engine with Administrator rights.
- Disable Remote Access for WIBU Codemeter Webadmin and use SSL (**Configuration > Basic > Webadmin**).
- Remove any unneeded services.
    - If FSmon and ArdFTP are not needed, do not install them.
    - Enable Viz Engine REST interface only if needed.
- The recommended NVIDIA driver should be used; however, new vulnerabilities can have been made public after the release of this software and it might be necessary to upgrade the NVIDIA driver (see https://www.nvidia.com/en-us/security/ for details).
- Do not expose any shares on the network.
- The amount of simultaneous network connections can be limited by setting *max_tcp_connections* in the Viz Configuration file.
- Viz Engine allows to block certain commands (setting *blacklisted_commands* in the Viz Configuration file).
    - For example, to prevent execution of Action Task keyframes, modification of Configuration File options and the replacement and execution of script parts, the following blocks the required commands.

```
blacklisted_commands = CONFIGURATION RESET;PLUGIN COMPILE;SOURCE_CODE
SET;MODE SET TASK
```

See Also

- Ports and Connections

Copyright © 2026 Vizrt

5 Software Configuration

This section details the prerequisites and supported options for the Viz Artist/Viz Engine installation, and procedures on how and where to install Viz Artist/Engine.

This section contains information on the following topics:

- Prerequisites
- Viz Artist and Viz Engine Installation
- Viz Engine Folders
- Ports and Connections
- Viz Log Files
- Logfiles Configuration
- Supported Software
- Import and Export Formats
- Viz Engine REST Interface
- User Account Control
- NDI
- Dual Channel Mode
- Trio Box CG Mode
- Dolby E Support
- EVS Video Server Control
- Integration with Viz One
- Matrox Stream
- VML Clip Player

Copyright © 2026 Vizrt

5.1 Prerequisites

- Supported Operating Systems
- Virtual Environments
- Environment Settings
- Hardware and BIOS Settings
- Power Management Settings
- User Rights
- Secure Boot
- Setting Permissions to Run Viz Engine and Viz Artist
- Anti-Virus Software
- Additional Prerequisites

5.1.1 Supported Operating Systems

Viz Artist and Engine has been tested to run on the following platforms:

Viz Artist/Engine version	Operating System
5.5.0,5.6.0	• Windows Server 2025 (4) • Windows Server 2022 (1) • Windows 10 LTSC 2021 Edition (21H2) • Windows 11 LTSC 2024 Edition
5.4.0, 5.4.1	• Windows Server 2025 (4) • Windows Server 2022 (1) • Windows Server 2019 (1) • Windows 10 LTSC 1809 Edition (3) • Windows 10 LTSC 2021 Edition (21H2) • Windows 11 LTSC 2024 Edition
5.3.1, 5.3.2	• Windows Server 2022 (1) • Windows Server 2019 (1) • Windows 10 LTSC 1809 Edition (3) • Windows 10 LTSC 2021 Edition (21H2) • Windows 11 LTSC 2024 Edition
5.3	• Windows Server 2022 (1) • Windows Server 2019 (1) • Windows 10 LTSC 1809 Edition (3) • Windows 10 LTSC 2021 Edition (21H2) • Windows 11 (2)

Copyright © 2026 Vizrt

Viz Artist/Engine version	Operating System
5.2	• Windows Server 2022 (1) • Windows Server 2019 (1) • Windows 10 LTSC 1809 Edition (3) • Windows 10 LTSC 2021 Edition (21H2) • Windows 11 (2)
5.1	• Windows Server 2022 (1) • Windows Server 2019 (1) • Windows 10 LTSC 1809 Edition (3) • Windows 10 LTSC 2021 Edition (21H2) • Windows 11 (2)
5.0	• Windows Server 2022 (1) • Windows Server 2019 (1) • Windows Server 2016 (1) • Windows 10 LTSC 1809 Edition (3) • Windows 10 LTSC 2021 Edition (21H2) • Windows 11 (2)
4.4.x	• Windows Server 2019 • Windows Server 2016 • Windows 10 LTSC 1809 Edition • Windows 10 LTSC 2021 Edition (21H2)
4.3	• Windows Server 2019 • Windows Server 2016 • Windows 10 LTSC 1809 Edition
4.0, 4.1, 4.2	• Windows Server 2016 • Windows 10 LTSC 1809 Edition
3.14	• Windows Server 2008 R2/SP1 (64-bit) • Windows Server 2016 • Windows Server 2012 R2 • Windows 7 (64-bit) • Windows 10 (64-bit) 1809 • Windows 10 IoT/LTSC Edition

Copyright © 2026 Vizrt

Viz Artist/Engine version	Operating System

[1] Server OS are used for VM and cloud environments only.
[2] Windows 11 LTSC versions have not been fully tested and certified yet.
[3] Unreal Engine requires a newer Windows 10 version than 1809. UE Integration was successfully tested with 21H2

[4] Only Viz Engine 5.4.1 and newer was tested with Windows Server 2025

For optimal performance, use the pre-installed Windows image from Vizrt. You can obtain the Windows image files from your local support office.

Note: Only English Operating System(s) are supported.

5.1.2 Virtual Environments

The following virtual environments have been tested and certified to host Viz Engines:

	Viz Engine Render Pipeline	Viz Classic Render Pipeline
Amazon AWS cloud	✓	✓
Microsoft Azure	✓	✓
Proxmox 9.2.2 (12q instance)	✓	✓
VMWare ESxi (6.0, 6.5,7.02, vSphere 8.0.2)	✓	✓
nutanix/fra.me	not tested	✓
AliBaba Cloud	not tested	✓

Note: Backup and Restore on Azure systems is currently not supported.

5.1.3 Environment Settings

The following prerequisites apply on all platforms. Applying the changes may require local administrator access rights, new or changing group policy entries, or modifying services. Contact your local IT manager for further instructions.

Perform all Windows Updates, except hardware driver updates. This is especially important for drivers related to NVIDIA and Matrox hardware, and Codemeter dongle drivers. In addition to this;

- Turn off windows sounds.

Copyright © 2026 Vizrt

- Turn off AutoPlay.
- Turn off Windows Media Player Network Sharing Service.
- Disable Windows Defender.

Set **Visual Effects** to **Adjust for best performance**. This is set in the Performance Options window; right click the Start button and select **System**, then select **Advanced system settings** and click the **Settings** button in the **Performance** section.

5.1.4 Hardware and BIOS Settings

The following considerations must be made regarding hardware:

- There must only be one active network card.
- Hardware must be installed, and BIOS configured, as suggested for the machine model.

5.1.5 Power Management Settings

Power management and hibernation mode must set to `Off`. Execute `powercfg -h off` from the command line to remove *hiberfil.sys* from the hard disk.

In addition to this, set the following under Power Options:

- Never turn off display.
- Never turn off hard disks.
- Disable USB selective suspend setting.
- Set Power button action to **Do nothing**. This prevents accidental shutdown in case someone presses the power button by mistake.

5.1.6 User Rights

The user must have special rights to run Viz Artist and Viz Engine. This can be achieved by assigning local administrator rights to the user, or by explicitly granting the required privileges. See Running Viz Engine and Viz Artist without administrator rights below for further details.

Any hardware solution provided by Vizrt is certified for use with Viz Engine. These come with a predefined default User that has administrator rights on the machine. The default administrator account is as follows:

User name	Password	Account Type
Admin	vizrt	Computer Administrator

IMPORTANT! Make sure that you change this password after initial installation!

IMPORTANT! Running the Engine as Administrator results in limitations in Viz Artist. For example, it is not possible to import files using drag & drop into Viz Artist.

Copyright © 2026 Vizrt

5.1.7 Secure Boot

Matrox drivers require a special version (labeled *_EV.exe*) for installing on secure boot enabled systems. If your driver fails to install, please disable secure boot in the BIOS settings.

To learn more about the different user account types, refer to the Windows operating system documentation, or visit www.microsoft.com.

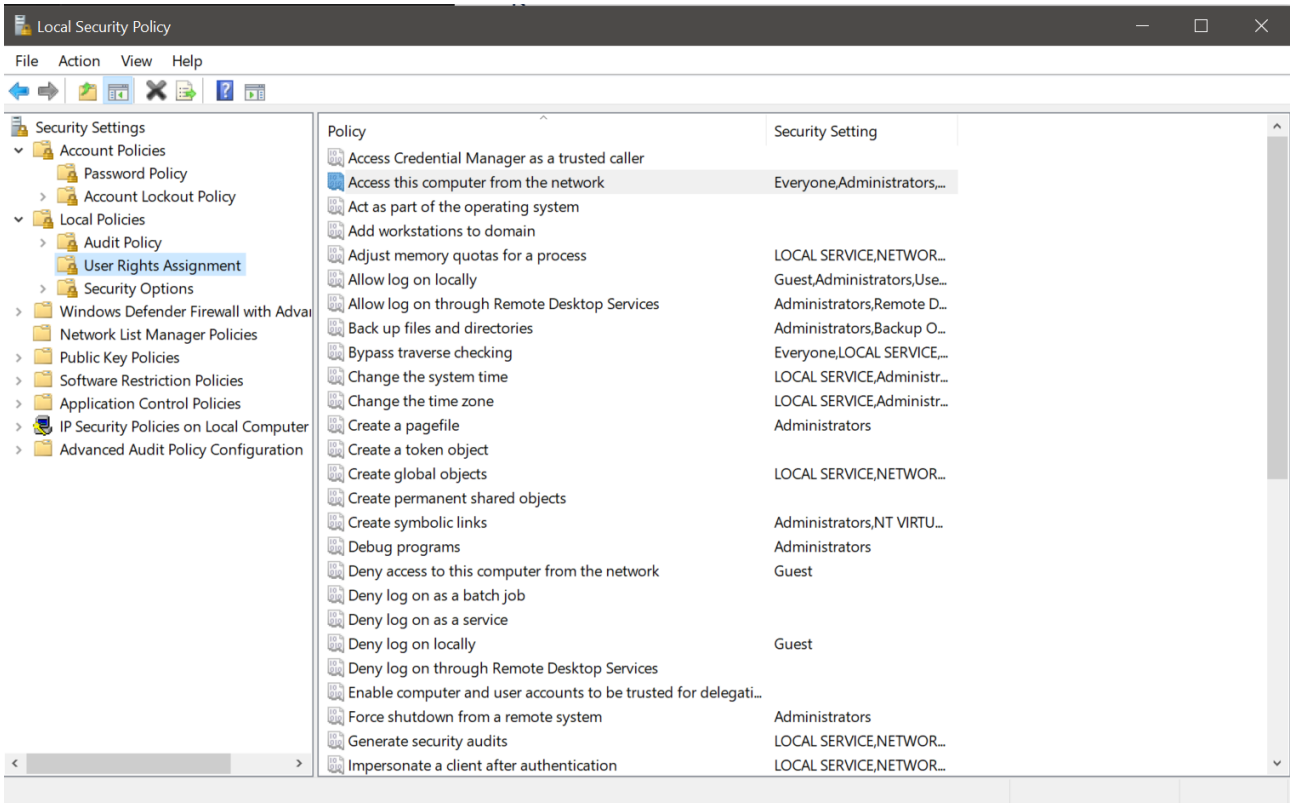
5.1.8 Setting Permissions to Run Viz Engine and Viz Artist

Since version 5.x, there is no need for administrator privileges to run Viz Engine. However, some operations still require special permission that usually are not given to a standard user:

Privilege	Description	Necessary if	Comment
`SeIncreaseBasePriorityPrivilege`	Increases scheduling priority	Matrox installed or engine-mode (`-n`) DataReader plug-in (for example, to access Excel files in use)	Loading content from external applications like using DataReader with Microsoft Excel without closing the source Excel file would need this token configured.
`SeCreateGlobalPrivilege`	Creates global objects	VDCP enabled or scroller plug-in	Scroller plug-in usage is not detected on startup.
`SeCreatePagefilePrivilege`	Creates a pagefile	Scroller plug-in	Scroller plug-in usage is not detected on startup.

These settings need to be changed in the local security policy:

Copyright © 2026 Vizrt



IMPORTANT! You need to restart Windows after setting these permissions.

IMPORTANT! Please contact your local IT manager for further information on how this is handled within the organization.

5.1.9 Anti-Virus Software

Anti-Virus software, including end-point protection, can cause various problems such as time-out and performance issues on the Graphic Hub database and other Vizrt machines, as every file is checked. To avoid these problems, make sure to exclude the *VizGH.exe* process and the underlying Graphic Hub data directory (default *D:\VizGHData* or *E:|VizGHData*) in the scan/real-time settings of the Anti-Virus software. Regarding Graphic Hub, more anti-virus-related information can be found in the Graphic Hub Administrator Guide.

These are the general Viz Engine recommendations for anti-virus scan exceptions:

- Services
  - Fsmon
  - Mediaftp
  - VizrtLicensing
  - CodeMeter.exe
- Processes
  - Fsmon
  - Mediaftp

Copyright © 2026 Vizrt

    - Viz.exe
- Folders
    - **Installation directory:** *C:\Program Files\Vizrt\VizEngine*
    - **Program data:** *C:\ProgramData\vizrt\VizEngine*
    - Configured clip data directory

Also, full system scans should not be done while Viz Engine is On Air.

If any Anti-Virus software is used without the above settings, optimal performance cannot be guaranteed, nor the long-term stability of Vizrt products.

There are hundreds of Anti-Virus software packages on the market. We do not recommend the use of any specific Anti-Virus software package or version, or give any recommendations on how to setup any Anti-Virus software suites in relation to Vizrt software and machines.

5.1.10 Additional Prerequisites

- DirectX 9 runtime is required if Global Illumination is used.

See Also

- Viz Engine Folders
- Supported Software

Copyright © 2026 Vizrt

5.2 Viz Artist and Viz Engine Installation

Use the Viz Artist Bundle Installer to install both Viz Artist and Viz Engine. The bundle installer contains Viz Artist, Viz Engine and all required dependencies.

This section covers the following topics:

- Installing Viz Artist and Engine
- Upgrading from a Previous Installation
- To Change or Reinstall an Existing Installation
- Silent Installation of Viz Artist and Engine
- Installation of a Specific Component
- Unattended Installation of Viz Artist and Engine
    - Additional Parameters
    - Installed Components
    - Example
- To Identify Installed Version

The software package is available in 64-bit, where the architecture is **x64**. By common convention, x64 signifies the 64-bit version.

The file name of the bundle installer indicates the architecture and software version in the following pattern: *Name-Architecture-Major.Minor.Maintenance.Build*.

Example: The file *VizArtistBundle-x64-4.3.0.99999.exe* is the bundle installer for Viz Artist and Viz Engine version 4.3.0, for 64-bit architecture platforms.

5.2.1 Installing Viz Artist and Engine

Installation of Viz Artist and Engine requires elevated permissions on the destination computer. Make sure to log on to the computer with either an administrator account or a user account with elevated rights, before installing the software. See Running Viz Engine and Viz Artist without administrator rights for further details. If you install Viz Artist on a non-supported operating system platform, you get a warning message.

Before installing the software, make sure to:

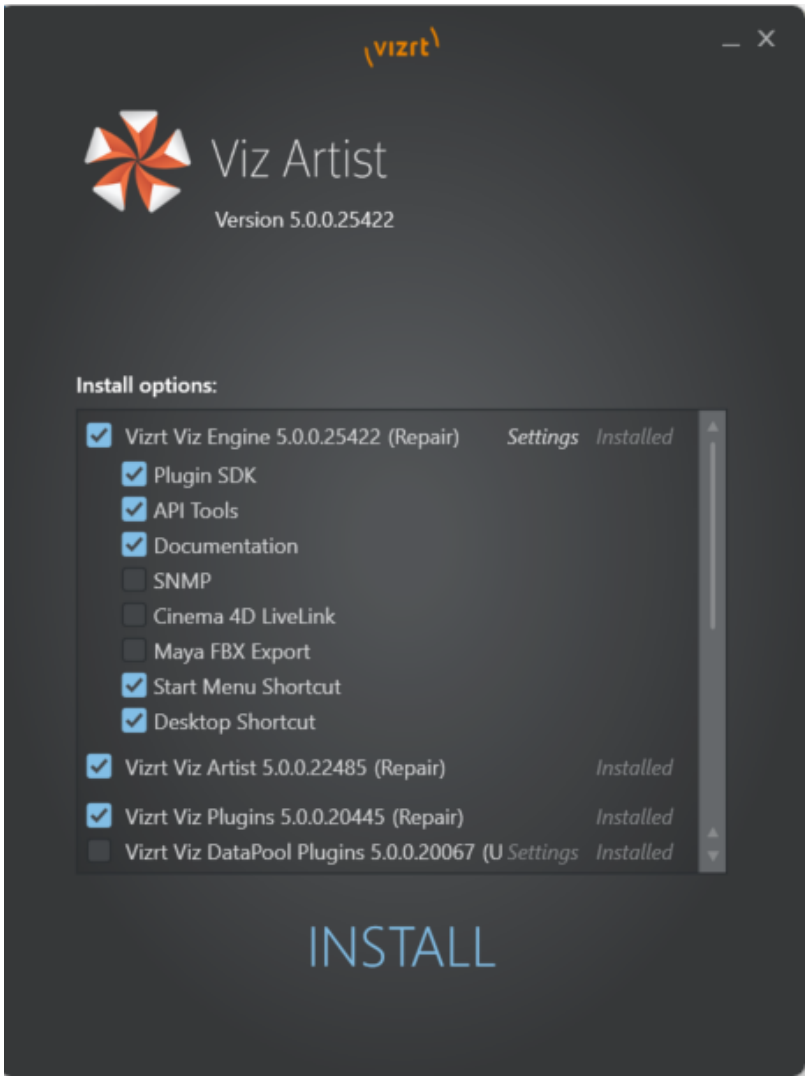
- Check that your computer meets the Prerequisites for hardware and system configuration.
- Decide which of the Viz Artist/Viz Engine Platforms is suitable for your installation.
- If upgrading, familiarize yourself with the information in the Upgrading from a previous installation section.

Tip: Download the Viz Artist and Engine installer from Vizrt's FTP server.

Info: Both **Mediaftp** and **Fsmon** have individual *.msi* installers rather than being included in the Viz Engine *.msi* starting with Viz Engine version 4.0. They are included in the Viz Artist Bundle installer.

1. Run the installer. The bundle installer provides an overview of all required dependencies that need to be installed in addition to the main application. Already installed dependencies are listed as **Installed**. You can collapse the selection list by clicking **Settings**.

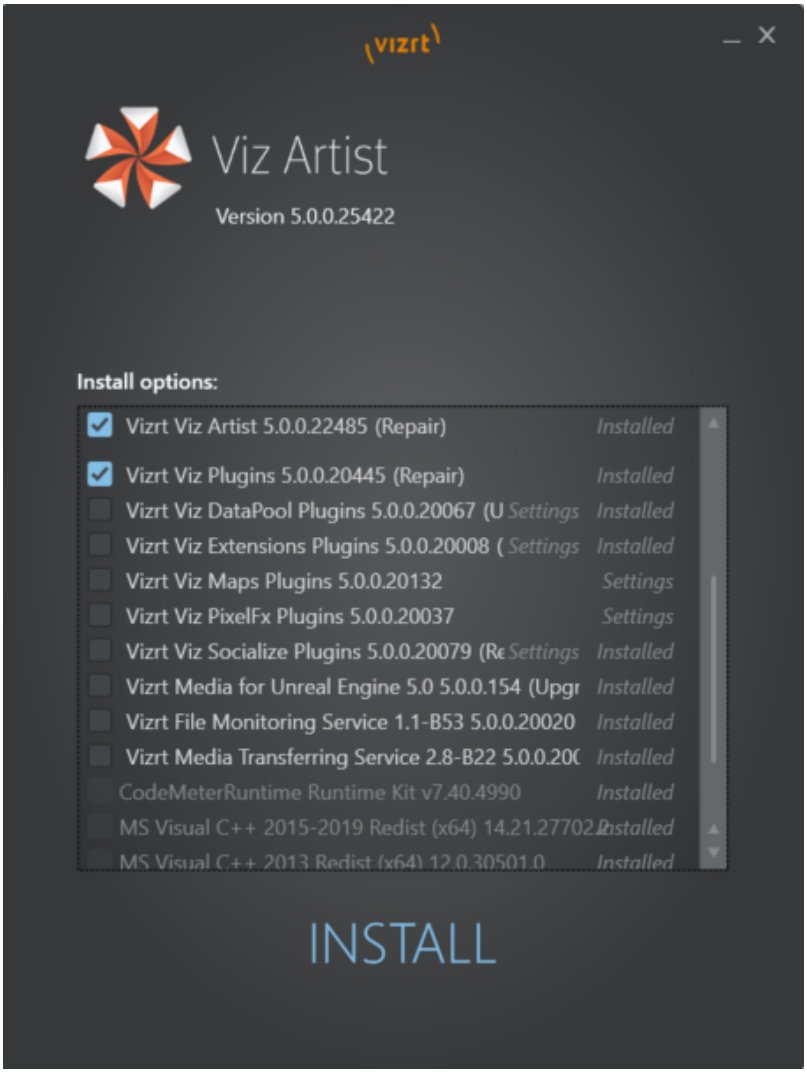
Copyright © 2026 Vizrt



Note: The bundle installer selects most features by default. Check the feature list and add or remove features as required.

2. Scroll further down to select a custom installation location, if required.

Copyright © 2026 Vizrt



3. Click **INSTALL** to complete the installation wizard. The various dependencies are installed as required.
4. Click **Finish**.

If something interrupts the installation, click **Show log** to view the installation log files.

5.2.2 Upgrading from a Previous Installation

When you upgrade an existing installation to a new version of Viz Artist and Engine, observe the following:

- Create a backup file of your existing configuration files (*.cfg*), especially if you are upgrading from a 3.x version. The default data directory for Viz Engine 3 is *%ProgramData%\vizrt\Viz3*. The default Viz Engine data directory is *%ProgramData%\vizrt\VizEngine*.
- If Viz Engine 3 was installed, follow-up versions since Viz Engine 4 print the error message *An incompatible version of Viz Engine is installed.* In this case, Viz Engine 3 must be uninstalled, as it is not possible to have both versions installed on a system.
- If upgrading an existing installation with the Viz Artist*.msi* installation file, you see this message: *"Viz Artist 32-bit is already installed. Remove the existing installation first, then restart the installer."* Open **Uninstall a**

Copyright © 2026 Vizrt

program from the Control Panel, select Viz Artist and remove the existing installation. Then run the *.msi* file to install Viz Artist/Engine.

- Installing Viz Artist/Viz Engine using the bundle installer upgrades older installations. Any older version is removed, except versions prior to Viz Artist and Engine *3.6*. This is true even if you opt to install Viz Artist/Viz Engine in a different directory.
- The installer suggests installing to the previous installation folder. You may change it. The recommended installation directory is *C:\Program Files\vizrt\VizEngine*.
- The installer pre-selects all previously installed features. You may change them.
- The installer does not support downgrading. Remove the currently installed version to install an earlier version of Viz Artist and Viz Engine.
- Upgrading does not change any modified or newly added files of the old installation. For example, Configuration files, Log files or additional files like customer plug-ins remain unchanged in their original folder.
- Since Viz version 3.7.1, *viz.exe* no longer checks the *%Program Files%* folder for Configuration files (*.cfg*) or Lens files (*.lcb*). These files must be located in *%ProgramData%\vizrt\vizEngine*.
    - If the previous Viz Artist/Viz Engine was installed in the default folder, the Configuration and Lens files are automatically copied during upgrade from the default installation folder to *%ProgramData%\vizrt\vizEngine*.
    - If the previous Viz Artist/Viz Engine was installed in a custom folder, the Configuration and Lens files needs to be copied manually from the old installation folder to *%ProgramData%\vizrt\vizEngine*.

Note: Matrox DSX.Core installer corrupts an existing CodeMeter Runtime installation and causes a crash on Viz Engine startup. Uninstall any CodeMeter Runtime, prior to installing Matrox DSX.Core. Install DSX.Core and afterwards install Viz Engine and the newer CodeMeter Runtime.

Information: If the upgraded version comes with a new CodeMeter version, make sure that all applications installed using CodeMeter on the same host, are still running.

5.2.3 To Change or Reinstall an Existing Installation

Use the Viz Artist bundle installer to change or reinstall an existing installation. The installer pre-selects any already installed features. Check the boxes for the components that you want to add or remove, then click **INSTALL**. This reinstalls the complete software suite, repairing any damaged or accidentally deleted files.

5.2.4 Silent Installation of Viz Artist and Engine

To install Viz Engine bundle with all its default settings call `VizArtistBundle-x64-5.x.x.x.exe -s` as administrator.

5.2.5 Installation of a Specific Component

Warning: When using *msiexec*, always use command line. Powershell does not work in all cases.

To install a specific *.msi* from the bundle, perform following steps:

Copyright © 2026 Vizrt

1. Open a command line and call *VizArtistBundle-x64-5.x.x.x.exe --dump*. This extracts all bundles *.msi* installers into a subfolder called **"VizArtistBundle - 5.x.x.x"**.
2. Enter `msiexec` on the command line and press enter to see other options. Common options are:
   - **/i:** Installs or configures a product. Package: Specifies the name of the Windows Installer package file. ProductCode: Specifies the globally unique identifier (GUID) of the Windows Installer package.
   - **/x:** Removes a product. Package: Name of the Windows Installer package file. ProductCode: Globally unique identifier (GUID) of the Windows Installer package.
   - **/l, /log:** Enables logging. Make sure to specify the log file name.

Example: `msiexec /i VizArtist.4.3.0.5.msi /norestart /passive INSTALLDIR_ARTIST="C:\tmp" -lv installation.log`

Warning: Be aware, certain plug-ins for various integrations need to be copied manually to their respective locations (Unreal, Cinema4D, After Effects, ...).

5.2.6 Unattended Installation of Viz Artist and Engine

To install Viz Artist in an unattended mode (the window of the Viz Artist bundle installer is displayed but without allowing user interaction except to cancel the installation progress) with default parameters, the following command should be used in a command prompt started as administrator:

```
“[Absolute path to the Viz Artist Bundle installer] --silent”
```

Additional Parameters

Parameter name	Default Value	All possible values
Viz Artist (--artist)		
INSTALLDIR_ARTIST	%\Program Files% \vizrt\VizArtist	*
INSTALL_SHORTCUTS_DESKTOP	1	0, 1
INSTALL_SHORTCUTS_STARTMENU	1	0, 1
Viz Engine (--engine)		
INSTALLDIR_ENGINE	%\Program Files% \vizrt\VizEngine	*
INSTALL_SHORTCUTS_DESKTOP	1	0, 1

Copyright © 2026 Vizrt

INSTALL_SHORTCUTS_STARTMENU	1	0, 1
Viz Fsmon (--fsmon)		
INSTALLDIR_FSMON	%\Program Files% \vizrt\Fsmon	*
Viz Mediaftp (--mediaftp)		
INSTALLDIR_MEDIAFTP	%\Program Files% \vizrt\Mediaftp	*

Installed Components

Component name	Value	Default	Setable via Bundle parameters
Viz Artist (--artist)			
Documentation	CM_C_Documentation	Installed	Yes
Viz Artist	CM_C_VizArtist	Installed	Yes
Viz Engine (--engine)			
API Tools	CM_C_API.Tools	Installed	Yes
Cinema 4D LiveLink	CM_C_Cinema.4D.LiveLink	Not installed	Yes
Documentation	CM_C_Documentation	Installed	Yes
Maya FBX Export	CM_C_Maya.FBX.Export	Not installed	Yes
Plugin SDK	CM_C_Plugin.SDK	Installed	Yes
Viz Engine	CM_C_VizEngine	Installed	Yes
Viz Fsmon (--fsmon)			
Fsmon Service	CM_C_Fsmon.Service	Not installed	No

Copyright © 2026 Vizrt

Viz Mediaftp (--mediaftp)			
MediaFtp Service	CM_C_Mediaftp.Service	Not installed	No
Viz Plugins - Basic (--plugins_basic)			
Configs	CM_C_Configs	Installed	Yes
Documentation	CM_C_Documentation	Installed	Yes
VizPlugins	CM_C_VizPlugins	Installed	Yes
Viz Plugins - DataPool (--plugins_datapool)			
Configs	CM_C_Configs	Not installed	Yes
Documentation	CM_C_Documentation	Not installed	Yes
VizPlugins	CM_C_VizPlugins	Not installed	Yes
Viz Plugins - PixelFx (--plugins_pixelfs)			
VizPlugins	CM_C_VizPlugins	Not installed	Yes
Documentation	CM_C_Documentation	Not installed	Yes
Viz Plugins - Socialize (--plugins_socialize)			
Configs	CM_C_Configs	Not installed	Yes
Documentation	CM_C_Documentation	Not installed	Yes
Examples	CM_C_Examples	Not installed	Yes

Copyright © 2026 Vizrt

VizPlugins	CM_C_VizPlugins	Not installed	Yes
Vizrt Media UE x.yy (--vizrtmedia_uexyy)			
Vizrt Media UEx.yy	CM_C_Vizrt.Media.UEx.yy	Not installed	Yes

Note: Documentation does no longer install the full documentation (PDF & Webhelp). These are available online only.

Example

Description	Command
• Install Viz Engine, and basic plug-ins in path *C:\Temp\Engine* • Install Viz Artist in path *C:\Temp\Artist'*	VizArtistBundle-x64-4.2.0.184.exe --silent --artist="INSTALLDIR_ARTIST='C:\Temp\Artist'" --engine="INSTALLDIR_ENGINE='C:\Temp\Engine'"
• Install Viz Engine, and basic plugins in path *C:\Program Files\vizrt\VizEngine*, Viz Artist in path *C:\Program Files\vizrt\VizArtist* • Install additional plug-in packages for DataPool, PixelFX, Socialize in path *C:\Program Files\vizrt\VizEngine*	VizArtistBundle-x64-4.2.0.184.exe --silent --engine="INSTALLDIR_ENGINE='C:\Program Files\vizrt\VizEngine'" --plugins_datapool="INSTALL_ME=true" --plugins_pixelfx="INSTALL_ME=true" --plugins_socialize="INSTALL_ME=true" --

5.2.7 To Identify Installed Version

To check the installed version of Viz Artist and Viz Engine, click on the **Viz Artist Information**  button:

In the Information window, the version numbers show as:

Copyright © 2026 Vizrt



Click on the version numbers to copy them to the clipboard.

5.3 Viz Engine Folders

This section details the location of the Viz Engine default installation and data folders.

5.3.1 Installation Folders

The default installation folder is *C:\Program Files\Vizrt\VizEngine*.

In this Administrator Guide, any reference to the Viz Engine installation folder, for example *C:\Program Files\Vizrt\VizEngine*, is replaced with the text *<viz install folder>*.

Note: Viz Config files that already exist from a previous installation are automatically copied on install time if the installation folder did not change. If Viz Engine is installed in a different installation folder then Viz Config files must be manually copied from the old installation folder to *<viz data folder>.*

5.3.2 Data Folders

Files which are created or modified by Viz Engine are located at *%ProgramData%\Vizrt\VizEngine*, which usually resolves to *C:\ProgramData\Vizrt\VizEngine*. This folder is referenced as *<viz data folder>* throughout this Administrator Guide, and contains, for example:

- Viz Config files
- Script plug-ins
- Crash dump files

Temporary files are located at: *%TMP%\Vizrt\VizEngine*, which usually resolves to *C:\Users\<user name>\AppData\Local\Temp\Vizrt\VizEngine*. This folder is referenced as *<viz temp folder>* throughout this User Guide.

Script Plug-ins are located in the *<viz data folder>\ScriptPlugins* subfolder.

Note: Script Plug-in files that already exist from a previous installation are automatically copied on install time if the installation folder did not change. If Viz Engine is installed in a different installation folder then Script Plug-in files must be manually copied from the old installation folder to *<viz data folder>\ScriptPlugins*.

Copyright © 2026 Vizrt

5.4 Ports and Connections

This section contains information on the following topics:

- Port Numbers
  - Viz Engine Ports
  - Other Integrations
- Multiplexing Ports
- Reserve Ports

5.4.1 Port Numbers

The table below lists all default server and listening port numbers used. When the firewall is well configured, it can be kept enabled. Client-side firewalls may cause issues when the Viz Artist/Viz Engine version is below 3.8.2 or the Graphic Hub version is below 3.0.0, since the communication protocol used was two-way and requires the firewall on the client side to be open for the Viz Artist/Viz Engine processes. However, it is not required to disable the firewall completely.

Viz Engine Ports

Service	Port(s)	Descriptions and comments
MVCP and Xlator control port for video servers. VDCP ports	`5250` TCP	⚠ **Note:** This port is only necessary in combination with the video server extension (Service: AVCP). These ports are configurable in Config File: • `MVCPServer_Port` • `vdcp_controller[n]_port`
Viz World Maps Editor	`1337` TCP	Websocket communication with Viz World Maps Client.
Tracking Hub communication	`3000` UDP	Used by Tracking Hub to communicate with Viz Engine. These ports are configurable in Config File: • `trackinghub_port`

Copyright © 2026 Vizrt

Service	Port(s)	Descriptions and comments
Control Ports	`6100` TCP `6700` TCP `6800` TCP `55000-55031` TCP `56000-56015` TCP	Communication ports used by Control Applications (like Media Sequencer) to connect to a Viz Engine program and/or preview channel. Viz Engine's default program and preview port is `6100`. In a single channel configuration where both program and preview output is on the same machine, the default preview port is set to `6800` to separate the program and preview channels. In a dual channel configuration, the default program ports are `6100` and `6800` for channel one and channel two, respectively. In a dual channel configuration, when used for stereo production, the default program ports are `6700` and `6800` for channel one (left eye) and channel two (right eye), respectively. For controlling Graphics Channels and/or Super Channels, ports `55000-55031` and `56000-56007` are used (can be changed in the config). These ports are configurable via Config GUI.
Viz Engine Unreal Integration	`6102` UDP `6104` UDP `6103` UDP	Communication ports from Viz Engine to Unreal Engine. These ports are configurable via Config GUI.
Viz Artist	`6998, ...` TCP	Used to communicate with Viz Artist. Every instance opens two ports, starting with `6998` and `6999`. ⚠ **Note:** These ports can not be changed!
Shared Memory Master Port	`freely configureable`	These ports are configurable in Config File: • `smm_master_eng_port`
Command Feedback Port	`7476` UDP	Port to send feedback of commands. This port is configurable in Config File: • `command_feedback_port`

Copyright © 2026 Vizrt

Service	Port(s)	Descriptions and comments
Viz Engine REST service	`freely configureable` TCP	To control Viz Engine via REST commands. This port can be configed via Config GUI.
Extension Plugin Commands via Websockets	`6900` TCP	To allow to control Viz Engine via a Websocket. This port can be changed manually in the JSON file *C:\ProgramData\vizrt\VizEngine\extensions\CommandOverWebsocket.json*.
Multitouch communication TUIO	`3333` UDP	To connect to a TUIO multitouch service. This port is configurable in Config File: • `tuio_serv`
Graphic Hub	`19396-19398`	Ports in use when connecting to different Graphic Hub components. Since Graphic Hub version 3.0.0 and Viz Artist/Viz Engine version 3.8.2, a limited number of open network ports is required. These do not require any port exemption rules on the client side firewall configuration. However, for earlier versions, all ports must be open on the client side.
MV Network Service	`50000` TCP	responsible for NMOS requests. It might be necessary to manually add this exception to an existing firewall. netsh advfirewall firewall add rule name="Matrox NMOS API" protocol=TCP dir=in localport=50000 action=allow program="C:\ProgramFiles\Matrox DSX-TopologyUtils\System64\mvNetworkService.exe"
Multiplexer ports	`50007 - 50009` TCP	Multiplexing Ports that enable Viz Engine to work on other scenes in sessions that are used for preview purposes.
	`50007` TCP	**MUX Isolated port:** All connections to this port get their own session.
	`50008` TCP	**MUX Shared port:** All connections from one single host shares one session.

Copyright © 2026 Vizrt

Service	Port(s)	Descriptions and comments
	`50009` TCP	**MUX Fixed port:** Same as shared port except that allocated resources are never cleared from memory.
	`50010` TCP	**Still Preview port:** Enables a user to request a preview of the next scene to be put On Air while another scene is On Air.
	These ports are configurable via Config GUI.	
WebRTC Extension Plugin	`9092`	Can be configured in the config file: `WebrtcOut1.WebsocketPort` or via command: `CONFIGURATION*CHANNELS*WEBRTCOUT_0*PORT SET <port_number>`
OpenTelemetry	`4318` (configurable)	Endpoint for Open Telemetry Monitoring Service (Viz Engine Monitor).

Information: Depending on your hardware (Matrox IP) in use, the number of ports can be much higher.

Other Integrations

Listener	Port(s)	Descriptions and comments
Mediaftp	`21`	Used for video transfers from Viz One to Viz Engine.
Viz One	`22`	TCP and UDP for logging in to the Viz One operating system (Service: SSH).
Viz World Server	`1022` `1023`	`1022` (TCP) is a Viz World Server listener port for Viz World Client connections when Server Allocator is not in use or only has one Viz World Server running. `1023` (TCP) is a Viz World Server listener port for configuration tool connections to the first Viz World Server instance (as configurations are controlled by the first server instance). See also Vizrt Maps.

Copyright © 2026 Vizrt

Listener	Port (s)	Descriptions and comments
Viz One	`137` `139`	Used for SMB file sharing (Service: Netbios)
Viz One, Microsoft Bing and Imagery on Demand	`80` `8080`	Web interface and client software. SOAP port for communication with Viz One. For download of Microsoft Bing and Imagery on Demand images (Service: HTTP).
OpenSLP	`427`	Service Location Protocol (SLP) based discovery and search (TCP, UDP).
Viz One	`443` `445`	(Service: HTTPS) TCP and UDP used for SMB file sharing (Service: Microsoft-DS).
Viz One Delivery	`554`	Real-time Streaming Protocol (Service: TCP).
Oracle database	`1521`	For clients that connect to the Viz Pilot Database.
Viz One	`3080`	Low resolution video and index files (Service: lighttpd).
Viz Trio	`6200` `6210`	`6200` is used for controlling the Viz Trio client over a socket connection. `6210` is used by the Graphics Plug-in to establish a connection to Viz Trio.
Newsroom Component	`6220`	Used by the Graphics Plug-in to establish a connection to Viz Pilot's Newsroom client.

Copyright © 2026 Vizrt

Listener	Port (s)	Descriptions and comments
Graphics Plugin Editor	6230	Used by the Graphics Plug-in to establish a connection to the Graphics Plug-in Editor (on Mac).
Graphics Plugin Config	6240	Used by the Graphics Plug-in to establish a connection to the Graphics Plug-in Configuration tool (on Mac).
Viz Ticker Service	6300 6301	Viz Ticker handler in the Media Sequencer connects to port `6300` for feedback from Viz Ticker Service. Viz Ticker handler in the Media Sequencer connect to port `6301` when controlling Viz Ticker via a socket connection.
Viz Pilot	6484	Socket connection used for controlling Viz Pilot using macro commands.
Viz One	6555	Message bus port for communication with Viz One (Service: Message bus).
Preview License server	7452	For the Newsroom Component using an unlicensed Viz Engine for local preview with a connection to the Preview License server (is not the same as the Preview Server).
Viz Pilot Data Server	8177	Used to connect over HTTP with the REST interface.
Media Sequencer	8580 8594	For clients connecting to the Media Sequencer. `8580` is specifically used to connect over HTTP with the REST interface.
Viz One	8080	Used for sending key frames (Service: ardok).

Copyright © 2026 Vizrt

Listener	Port(s)	Descriptions and comments
Gateway	`10001` `10002` `10540` `10541`	For DB notification events. For Gateway controller clients. For MOS object updates. For MOS playlist updates.
Viz World Server	`10100` `10200`	`10100` (TCP) is a Server Allocator listener port for Viz World Client connections, and is only used for clients to get connection details about Viz World Server(s). The first client connection is always diverted to port `102`. In case of multiple server instances, port numbers are assigned according to a predefined schema (`10101`, `10102` for server instance two and three and so on). In case there is no Server Allocator, Viz World Server itself switches to port `102`. `10100` (UDP) is a Viz World Server listener port for Server Allocator communication. `10200` (UDP) is a Server Allocator listener port for Viz World Server communication. Both UDP ports are internal ports used between the servers. For more information, please see the Viz World Client and Server 11.1 User Guide and later.
Viz Pilot	`10640`	Used by Gateway to establish a connection to Viz Pilot to send and receive updates on MOS messages (for example, items and playlists).
Graphic Hub	`19396-19398`	Ports in use when connecting to different Graphic Hub components. Since Graphic Hub version 3.0.0 and Viz Artist/Viz Engine version 3.8.2, a limited number of open network ports is required. These do not require any port exemption rules on the client side firewall configuration. However, for earlier versions, all ports must be open on the client side.
Connection Broker	`21098`	Connection to the Connection Broker configuration interface (for example, http://localhost:21098/).

Copyright © 2026 Vizrt

Listener	Port (s)	Descriptions and comments
Preview Server	`54000`	Used to connect over HTTP with the REST interface.
Codemeter Webinterface	`22352`	To configure the Codemeter runtime via WebInterface.

5.4.2 Multiplexing Ports

Ports	Viz Engine
All other ports	
Still Preview Port	
MUX Isolated PortMUX Shared PortMUX Fixed Port	

The multiplexer functionality is an integral part of Viz Engine. When using Viz Engine a session management takes place internally, with one default session for the GUI and internal/external commands, and additional sessions created on-demand for the multiplexing ports or the preview port.

With multiplex ports, other than the MUX Still Preview port, the Viz Engine state is only switched when a command is received, which means a new session is created; hence, ten consecutive commands from a client only results in one state switch on the first command.

- The MUX Still Preview Port (`50010`) state is switched when a command is received and immediately switched back to the main session such that On Air rendering is not hindered in any way.
- The MUX Fixed Port (`50009`) is traditionally used by the old Viz Pilot Newsroom Client, and is the same as the MUX Shared Port, except that allocated resources are never cleared from memory. To avoid memory overload, it is recommended to clean up the Viz Engine regularly when this port is used.

Info: There is no automated cleanup of memory on this port.

Copyright © 2026 Vizrt

- The MUX Shared Port (`50008`) is a shared port where all connections from one single host shares one session. It is most often used by Viz Trio and the Newsroom Client to show preview frames.
- The MUX Isolated Port (`50007`) is an isolated port where all connections get their own session. It is used, for example in an NLE configuration, to deliver frames to the host NLE-system when rendering or scrubbing video clips with graphics. Using this port also suppresses bounding box commands.

All multiplexing ports are supported by all Viz Engine versions, but require a license.

5.4.3 Reserve Ports

It may happen that the operating system is allocating ports Viz Engine is configured to use, and therefore these ports can no longer be used by Viz Engine. To prevent Windows to allocated (for example, the Superchannel) ports, run the following command:

```
netsh interface ipv4 add excludedportrange protocol=tcp startport=55000
numberofports=32
```

This ensures the IP range `55000` to `55031` is no longer occupied by Windows.

Copyright © 2026 Vizrt

5.5 Viz Log Files

All Viz Artist/Engine log files are located in the *<viz data folder>*.

> **Note:** This is normally at *C:\ProgramData\Vizrt\VizEngine* Check the directory name with the command `echo %programdata%` from a Windows command prompt. This directory is by default hidden in Windows, so to navigate to this directory in Windows Explorer specify the explicit path.

Viz Artist/Engine can provide various log files as documented in the section below.

5.5.1 Viz Render Log

- **Name:** *VizRender_<timestamp>.log*
- **Purpose:** Provides information on current status of Viz Engine.

5.5.2 Viz Trace Log

- **Name:** *VizTrace_<timestamp>.vlog*
- **Purpose:** Provides command trace that facilitates playback for error reproduction, contains at most the last 500 commands.

5.5.3 Viz Gui Log

- **Name:** *VizGui.log*
- **Purpose:** Provides information on Viz GUI errors.
- **Log Description:** Each line in the log file has six components or entries, each separated by pipe (|). A typical log-line looks like:

```
Thu Jul 04 10:02:15 EST 2019|LM_ART|5420|Version: 4.0.1.42057|CONFIG|GPU1
```

The components for each log-line are:

- Date/time
- Type, one of:
    - LM_STARTUP (Regular startup)
    - LM_QUIT (Regular quit)
    - LM_QUIT_TIMEOUT (Timeout quit)
    - LM_QUIT_LOGIN (Login canceled)
    - LM_CFG (Restart with configuration)
    - LM_ENG_GUI (Restart engine with GUI)
    - LM_ENG (Restart engine without GUI)
    - LM_ART (Restart of Artist)
    - LM_ (Current mode restarted)
- Pid (Process id)

Copyright © 2026 Vizrt

- Viz version
- Mode:
    - CONFIG (Config Mode)
    - NOGUI (Engine Mode)
    - NORMAL (Artist Mode)
- Starting on GPU<x>, for example *GPU1* (Graphical Processing Unit number one)

5.5.4 Viz Shaders Log

- **Name:** *VizShaders.log*
- **Purpose:** Provides information on shader compilation.

5.5.5 Viz Console Log

- **Name:** *Viz_<timestamp>.log*
- **Purpose:** Logs console output to a file when the engine is started without console (`-C` option).

5.5.6 Viz GUI Connection Log

This Log is created if *Write GH Connection Log* is active (see Local Settings).

- **Name:** *VizGuiConnection<timestamp>.log*
- **Purpose:** Provides information on the Graphic Hub Manager database connection.

5.5.7 Create Log Files with Log and Clog Commands

The output of the Viz Engine can be redirected to a file using the command `log <filename>`, for example, *log C:\temp\my-engine-log.txt*. Note that the log file has no content until the Viz Engine in-memory buffers are flushed, meaning written to disk. The log memory buffers are flushed to disk either when the buffer are full or when Viz Engine quits. You can force the buffer to be written to the log file on disk by sending the command: `CONSOLE FLUSH`.

You can take an immediate snapshot of the Engine's current in-memory log with the command `clog`. A new log file is immediately written to *<viz data folder>\VizRender-ID.log* The ID in the filename is the GPU ID, making it easy to differentiate log files in a Dual Engine setup for example.

Both the **log** and **clog** commands can be executed by sending them to Viz or by entering them directly in the Engine Console window.

Click on the  (show commands) button to access the Console window.

Copyright © 2026 Vizrt

5.6 Logfiles Configuration

- Location of Log Configuration
- Concept of Log Configuration
  - Known Sinks
    - Parameters Common To All Sinks
    - daily_file_sink
    - rotating_file_sink
    - stdout_color_sink and stderr_color_sink
  - Known Formatters
    - Parameters Common To All Formatters
    - pattern_formatter
  - Logger
  - Known Loggers in the Viz Engine
- Built-In Log Config
- Examples
  - Log to File only
  - Only Warning and Errors on Console

5.6.1 Location of Log Configuration

Viz Engine Log Configuration is stored in *.json* files. Viz Engine is looking in its *ProgramData* folder for log configuration files. By default this folder is located at *%ProgramData%\vizrt\VizEngine* if not changed by command line argument. Viz Engine looks for a file called *viz_engine_%N_log_config.json*, where `%N` is replaced by the instance number, starting at 0 for the first instance. If that file is not found it tries *viz_engine_log_config.json* next. In case that also fails the last resort is the built-in log configuration.

This approach allows to have distinct log configuration for each Viz Engine instance or a common one for all instances, depending on the use case, one or the other might be preferred.

Order of Log Configuration Search:
- *${ProgramData}\viz_engine_%N_log_config.json*
- *${ProgramData}\viz_engine_log_config.json*
- built-in as fallback

5.6.2 Concept of Log Configuration

The Log Configuration consists of loggers who contain zero, one, or more sinks. A sink uses a formatter. The application knows loggers by its ID. A sink used by a logger determines the destination where a log record is sent to and a formatter is used to process the log record to bring it to the desired format expected by the sink.

Log configuration Concept Overview:
Formatters: [*formatter1* , formatter2, ...]
formatter1 : {id, *concrete_type*}, ...

Copyright © 2026 Vizrt

Loggers: [*logger1* , logger2, ...]
logger1 : {id, **sinks**: [*sink1* , sink2, ...]}, ...
sink1 : {id, *formatter1*, *concrete_type*}, ...

Warning: Only the formatters and sinks listed below are supported. Using other formatters or sinks result in undefined behavior!

Known Sinks

At the moment there are four sinks available. Two file sinks and two console sinks.

- **daily_file_sink:** Creates a log file that is daily rotated.
- **rotating_file_sink:** Creates a log file that is rotated when it exceeds a certain size.
- **stdout_color_sink:** Creates a colored console sink writing on stdout.
- **stderr_color_sink:** Creates a colored console sink writing on stderr.

Parameters Common To All Sinks

Some parameters are common to all concrete sinks and are described below.

Parameters Common To All Sinks:

```
" id " : "default_file_sink" ,
" formatter " : "engine_file_formatter" ,
" log_level " : "debug" ,
" deferred " : false ,
```

- **id:** Determines identifier for this sink. This ID is used by the application.
- **formatter:** References an id of a formatter.
- **log_level:** Applies a log level to this sink. Possible values are:
  - trace
  - debug
  - info
  - warning
  - error
  - critical
  - off
- **deferred:** Does not create this sink during configuration. It may later be instantiated by the application.

daily_file_sink

Creates a log file that is rotated daily. The concrete parameters are below.

Daily file sink: Concrete parameters:

Copyright © 2026 Vizrt

```
"daily_file_sink" : {
      " base_filename " : "logs/viz_engine_%N_default" ,
      " rotation_hour " : 9 ,
      " rotation_minute " : 3 ,
      " truncate " : false,
      " max_files " : 10
}
```

- **base_filename:** Determines the path and name of your log file. It may include a placeholder for instance as %N. Path can be absolute or relative. If path is relative, then it is always relative to the current directory. Creates filename in the form *basename.YYYY-MM-DD*.
- **rotation_hour:** Rotates at this hour. Range 0-23. Do not include a leading zero.
- **rotation_minute:** Rotates at this minute. Range 0-59. Do not include a leading zero.
- **truncate:** Empties the log file at start or reconfiguration of this sink, if set to *true*.
- **max_files:** Determines the maximum number of files to keep after rotating.

rotating_file_sink

Creates a log file that is rotated when it exceeds a certain size. The concrete parameters are below.

Rotating file sink: Concrete parameters:

```
"rotating_file_sink" : {
      " base_filename " : "logs/viz_engine_%N_shaders" ,
      " max_size_MB " : 1024 ,
      " max_files " : 8 ,
      " rotate_on_open " : false
}
```

- **base_filename:** Determines path and name of your log file. It may include a placeholder for instance as %N. Path can be absolute or relative. If path is relative, then it is always relative to the current directory.
- **max_size_MB:** Sets the maximum size in MB after which file gets rotated.
- **max_files:** Determines the maximum number files to keep after rotating.
- **rotate_on_open:** Rotates the log file at start or reconfiguration of this sink, if set to *true*.

stdout_color_sink and stderr_color_sink

Creates a colored console sink writing on stdout or stderr. Since they are identical they are gathered together here in the documentation. The concrete parameters are below.

stdxxx color sink: Concrete parameters:

```
"stdout_color_sink" : {
      " color_mode " : "automatic"
}
```

Copyright © 2026 Vizrt

```
"stderr_color_sink": {
     " color_mode ": "automatic"
}
```

- **color_mode:** Sets the color mode used by sinks with color support. Possible values are:
  - automatic
  - always
  - never

Known Formatters

At the moment there is one formatter known.

- **pattern_formatter:** A highly versatile text formatter.

Parameters Common To All Formatters

Parameters Common To all Formatters:

```
" id " : "engine_file_formatter" ,
```

- **id:** Sets the identifier for this formatter. This ID is used as a reference by a sink.

pattern_formatter

A highly versatile text formatter.

Pattern formatter: Concrete parameters:

```
"pattern_formatter" : {
    " pattern_string " : "[%Y-%m-%dT%H:%M:%S.%fZ] [%l][pid=%P][tid=%t]:
%a %v" ,
    " pattern_time_type " : "utc" ,
    " eol " : "default" ,
    " escape_nl " : true ,
}
```

- **pattern_string:** Describes how to format the log message. Placeholders in the string expand to.
  - `"%n"` logger name.
  - `"%l"` log level.
  - `"%L"` short log level.
  - `"%a"` weekday.
  - `"%A"` short weekday.
  - `"%b"` or `"%h"` month.

Copyright © 2026 Vizrt

- `"%B"` short month.
- `"%c"` datetime.
- `"%C"` year two digits.
- `"%Y"` year four digits.
- `"%D"` or `"%x"` datetime MM/DD/YY.
- `"%m"` month 1-12.
- `"%d"` day of month 1-31.
- `"%H"` hours 24.
- `"%I"` hours 12.
- `"%M"` minutes.
- `"%S"` seconds.
- `"%e"` milliseconds.
- `"%f"` microseconds.
- `"%F"` nanoseconds.
- `"%E"` seconds since the epoch.
- `"%p"` am/pm.
- `"%r"` 12 hour clock 02:55:02 pm.
- `"%R"` 24-hour HH:MM time.
- `"%T"` or `"%X"` ISO 8601 time format (HH:MM:SS).
- `"%z"` timezone.
- `"%t"` thread id.
- `"%P"` pid. see
- `"%v"` message text.
- `"%^"` color range start.
- `"%$"` color range end
- `"%@"` source location (filename:linenumber).
- `"%s"` short source filename - without directory name.
- `"%g"` full source filename.
- `"%#"` source line number.
- `"%!"` source funcname.
- `"%%"` "%" char.
- `"%u"` elapsed time since last log message in nanoseconds.
- `"%i"` elapsed time since last log message in microseconds.
- `"%o"` elapsed time since last log message in milliseconds.
- `"%O"` elapsed time since last log message in seconds.

Some placeholders may not be available and depend on the log component itself, such as source location, funcname, etc. This is not a defect, but intentional.

Copyright © 2026 Vizrt

- **pattern_time_type:** Determines how the time is presented. Possible values are:
    - local
    - utc
- **eol:** string to append at the end of the log message. Possible values are:
    - default ... the default value for end of line on your OS
    - LF ... line feed
    - CR ... carriage return
    - CRLF ... carriage return followed by line feed
    - "none" ... nothing will be appended
- **escape_nl:** Escapes CR and LF in the log message, if true.

Logger

Logger: Parameter

```
"logger": {
    "id": "engine_sync_console",
    "log_level": "info",
    "flush_level": "error",
    "async_mode": true,
    "register": true,
    "deferred": false,
    "dup_filter_max_skip_duration": 0,
    "sinks": [ ... ]
}
```

- **id:** Determines identifier for this logger. This ID is used by the application.
- **log_level:** Applies a log level to this logger. Possible values are:
    - trace
    - debug
    - info
    - warning
    - error
    - critical
    - off
- **flush_level:** Flushes this logger on all sinks when a log message with this level or higher is logged. Possible values are:
    - trace
    - debug
    - info
    - warning
    - error
    - critical
    - off
- **async_mode:** Operates the logger asynchronously.
- **register:** Registers the logger in the logger registry if true.

Copyright © 2026 Vizrt

- **deferred:** Does not create this logger by default. It is instantiated later by the application.
- **dup_filter_max_skip_duration:** Determines the time interval (in seconds) within which duplicate payloads are filtered out when greater than zero.
- **sinks:** Lists sinks for this logger. See below in the Built-In Log Config for details.

Known Loggers in the Viz Engine

The Viz Engine uses the following loggers:

Known Loggers to Viz Engine:
- engine_default
- engine_sync_console
- engine_shaders

- **engine_sync_console:** Answers to input read from the console is using the synchronous console logger.
- **engine_shaders:** Emits shader construction to this logger.
- **engine_default:** All other log messages end up here.

5.6.3 Built-In Log Config

Viz Engine Built-in Log Configuration

```
{
  "version": "1.0",
  "async_params": {
    "queue_size": 32768,
    "overflow_policy": "overrun_oldest",
    "number_of_workers": 1,
    "flush_interval": 10
  },
  "formatters": [
    {
      "id": "engine_file_formatter",
      "concrete": {
        "pattern_formatter": {
          "pattern_string": "[%Y-%m-%d %H:%M:%S.%fZ] [%l][pid=%P][tid=%t]: %v",
          "pattern_time_type": "utc",
          "eol": "default",
          "escape_nl": true,
        }
      }
    },
    {
      "id": "engine_console_formatter",
      "concrete": {
        "pattern_formatter": {
          "pattern_string": "%^[%Y-%m-%d %H:%M:%S.%e] [%L]:%$ %v",
          "pattern_time_type": "local",
```

```
        "eol": "default",
        "escape_nl": false,
      }
    }
  },
  {
    "id": "plain_console_formatter",
    "concrete": {
      "pattern_formatter": {
        "pattern_string": "%v",
        "pattern_time_type": "local",
        "eol": "default",
        "escape_nl": false
      }
    }
  }
],
"loggers": [
  {
    "logger": {
      "id": "engine_default",
      "log_level": "info",
      "flush_level": "error",
      "async_mode": true,
      "register": true,
      "deferred": false,
      "dup_filter_max_skip_duration": 0,
       "sinks": [
        {
          "id": "default_file_sink",
          "formatter": "engine_file_formatter",
          "log_level": "debug",
          "deferred": false,
          "concrete": {
            "rotating_file_sink": {
              "base_filename": "logs/viz_engine_%N_default",
              "max_size_MB": 1024,
              "max_files": 10,
              "rotate_on_open": false
            }
          }
        },
        {
          "id": "console_sink",
          "formatter": "engine_console_formatter",
          "log_level": "debug",
          "deferred": true,
          "concrete": {
            "stdout_color_sink": {
              "color_mode": "automatic"
            }
          }
        }
```

```
      ]
    }
  },
  {
    "logger": {
      "id": "engine_sync_console",
      "log_level": "info",
      "flush_level": "error",
      "async_mode": false,
      "register": false,
      "deferred": false,
      "dup_filter_max_skip_duration": 0,
      "sinks": [
        {
          "id": "sync_console_sink",
          "formatter": "plain_console_formatter",
          "log_level": "debug",
          "deferred": true,
          "concrete": {
            "stdout_color_sink": {
              "color_mode": "automatic"
            }
          }
        }
      ]
    }
  },
  {
    "logger": {
      "id": "engine_shaders",
      "log_level": "info",
      "flush_level": "info",
      "async_mode": true,
      "register": true,
      "deferred": true,
      "dup_filter_max_skip_duration": 0,
      "sinks": [
        {
          "id": "shader_file_sink",
          "formatter": "engine_file_formatter",
          "log_level": "info",
          "deferred": false,
          "concrete": {
            "rotating_file_sink": {
              "base_filename": "logs/viz_engine_%N_shaders",
              "max_size_MB": 1024,
              "max_files": 10,
              "rotate_on_open": false
            }
          }
        }
      ]
    }
```

```
    }
  ]
}
```

The first object contains the parameters for the default asynchronous factory. Followed by three formatters, engine_file_formatter, engine_console_formatter, and plain_console_formatter. The last one just emits the log message as it was received. No other decoration, such as timestamps are added. The former two add timestamps and the engine_console_formatter does colorization where possible.

On the loggers side we have engine_default, which has a log_level of info, is using the asynchronous factory and is created during the initial configuration. It uses two sinks, a rotating_file_sink logging to logs/viz_engine_%N_default, and stdout_color_sink to emit logs messages to the console. Next is engine_shaders, it uses a rotating_file_sink as well and the same formatter as engine_default. It is not created at configuration time but on demand. Next is engine_sync_console, it is created at configuration time, does not register itself and the sink is created deferred. Meaning the logger can be found but the sink is created on demand.

5.6.4 Examples

Log to File only

If one does not want any messages emitted onto the console window the console sink can just be omitted from the engine_default logger. Such as:

Example: Log to file only:

```
    {
        "logger" : {
            "id" : "engine_default" ,
            "log_level" : "info" ,
            "flush_level" : "error" ,
            "async_mode" : true ,
            "register" : true ,
            "deferred" : false ,
           "dup_filter_max_skip_duration": 0,
           "sinks" : [
               {
                   "id" : "default_file_sink" ,
                   "formatter" : "engine_file_formatter" ,
                   "log_level" : "debug" ,
                   "deferred" : false ,
                   "concrete" : {
                      "rotating_file_sink" : {
                         "base_filename" : "logs/viz_engine_%N_default" ,
                         "max_size_MB" : 1024 ,
```

```
                "max_files" : 10 ,
                "rotate_on_open" : false
            }
        }
    },
    {
        "id" : "console_sink" ,
        "formatter" : "engine_console_formatter" ,
        "log_level" : "debug" ,
        "deferred" : true ,
        "concrete" : {
            "stdout_color_sink" : {
                "color_mode" : "automatic"
            }
        }
    }
]
}
}
```

Only Warning and Errors on Console

Sometimes it may be beneficial if only warning and error appear in the console. For this just set the log_level in the console sink to warning.

Example: Only warning and error in console:

```
{
    "logger" : {
        "id" : "engine_default" ,
        "log_level" : "info" ,
        "flush_level" : "error" ,
        "async_mode" : true ,
        "register" : true ,
        "deferred" : false ,
      "dup_filter_max_skip_duration": 0,
      "sinks" : [
        {
            "id" : "default_file_sink" ,
```

```
            "formatter" : "engine_file_formatter" ,
            "log_level" : "debug" ,
            "deferred" : false ,
            "concrete" : {
                "rotating_file_sink" : {
                    "base_filename" : "logs/viz_engine_%N_default" ,
                    "max_size_MB" : 1024 ,
                    "max_files" : 10 ,
                    "rotate_on_open" : true
                }
            }
        },
        {
            "id" : "console_sink" ,
            "formatter" : "engine_console_formatter" ,
            "log_level" : "warning" ,
            "deferred" : true ,
            "concrete" : {
                "stdout_color_sink" : {
                    "color_mode" : "automatic"
                }
            }
        }
    ]
  }
}
```

5.7 Supported Software

5.7.1 Viz Engine Software

To run Viz Engine as a program or preview (optional) machine, the following software and configuration is needed:

Viz Engine Specifications

Category	Requirement
Software	• Installation of latest Viz Artist/Viz Engine package • Optional: (Additional plug-in packages) • Codemeter runtime for WIBU-based licensing
Hardware	Valid Codemeter license Supported GPU and videoboard
Executable(s)	*viz.exe*
Ports and Connections	• See Ports and Connections

Info: The Viz Engine Render Pipeline requires at least an NVIDIA Quattro P6000 GPU.

5.7.2 Preview Server

The Preview Server option is used in situations where Viz Engine is used to provide frames for snapshot or thumbnail generation. A typical use case would be to connect multiple Newsroom Components to a preview server.

The Preview Server must be installed on a separate Viz Engine machine with its own license.

Category	Requirement
Software	• Latest Preview Server package • Microsoft .NET Framework 4 • Latest Viz Artist/Viz Engine package • Codemeter runtime for WIBU-based licensing
Hardware	Codemeter license Supported GPU
Executable(s)	*PreviewServer.exe*

Copyright © 2026 Vizrt

Category	Requirement
Ports and Connections	`54000`: Used to connect over HTTP with the REST interface.
Network access	Uses the ZeroConf protocol to announce available services.

5.7.3 Viz Artist Software

The Viz Artist design machine should preferably have the same specifications as the Viz Engine playout renderer, especially if the designers need to test performance issues on demanding scenes.

Category	Requirement
Software	• Installation of latest Viz Artist/Viz Engine package • Optional: (Additional plug-in packages) • Codemeter runtime for WIBU-based licensing
Hardware	Valid Codemeter license Supported GPU and videoboard
Executable(s)	*vizgui.exe*
Ports and Connections	See Ports and Connections.

If designers are creating templates for Viz Pilot, it is recommended that Viz Pilot is installed on a separate machine for more accurate playout emulation on Viz Engine.

Copyright © 2026 Vizrt

5.8 Import and Export Formats

Format	Importable (with/without compression)	Exportable	Supported by Viz Classic Render Pipeline	Supported by Viz Engine Render Pipeline
BMP	✓	✓	✓	✓
PNG[1]	✓	✓	✓	✓
TGA	✓	✓	✓	✓
JPEG	✓	✓	✓	✓
HDR[2]	✓	✓	✓	✓
EXR	✓	✗	✓	✓
VBN[3]	✓	✗	✓	✓
TIF	✓	✓	✓	✓
PSD[4]	✓	✓	✓	✓
GIF	✓	✗	✓	✓

1. No support for 16-bit per channel PNG images.
2. For use with RTT plug-in or as Environment Maps in Viz Engine Render Pipeline.
3. Compressed images are not supported.
4. Layered PSDs are imported as separate images or as complete scenes

5.8.1 EXIF

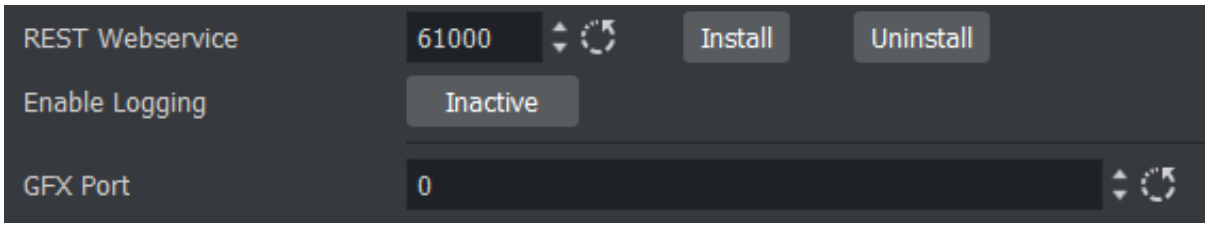
EXIF information is read when importing JPEG images. Images are automatically rotated based on the rotation information stored within the image.

Copyright © 2026 Vizrt

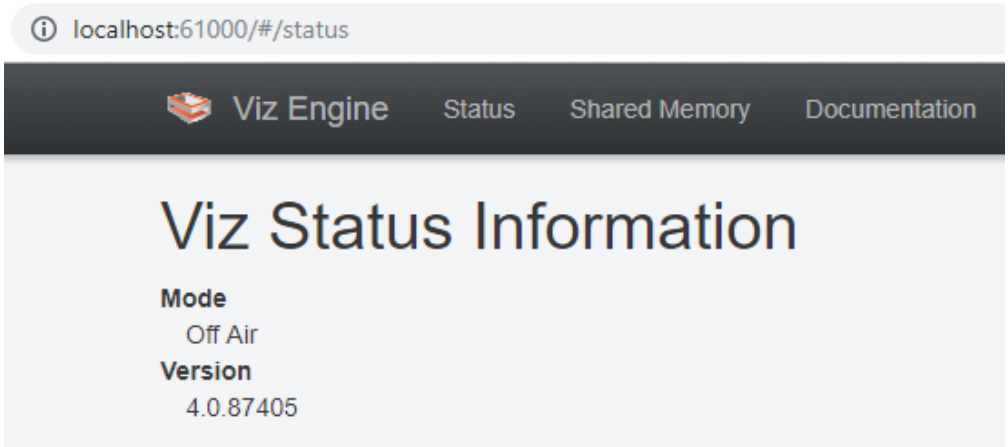
5.9 Viz Engine REST Interface

REST (short for Representational State Transfer) is an architectural paradigm for distributed systems, such as web services. Broadly speaking, REST relates to HTTP in the same sense that XML relates to HTML; a paradigm of the general concepts behind HTTP with some added restrictions. One such constraint is the statelessness of the communication; which HTTP violates via cookies. The most widely known application of REST, are web pages accessed via HTTP. Each web page is a unique resource, accessed via its URL (Uniform Resource Locator) which can be operated on (for example, `GET /books/dune.html`). This combination of operation and URL is called a message. In REST, a message has to be self-descriptive, meaning that all the information needed to process the message is required to be contained within the message.

The Viz Engine REST interface can be activated by enabling **Webservice** in the Global tab of the Communication section of Configuring Viz.



If User Account Control is active, press **Install** after setting the Port. The default port for the Viz Engine web service is `61000`. After restarting Viz, the REST interface can be accessed by navigating a web browser to *http://localhost:61000*. The landing page displays the status of the Engine, with information on which mode it is running in and its version number.



Once the REST interface has been activated, the complete documentation for the Viz Engine REST interface can be accessed by navigating to *http://localhost:61000/#/documentation*, or by clicking the **Documentation** link from the landing page.

Copyright © 2026 Vizrt

Information: Make sure your firewall is configured correctly. You need to add a new inbound rule in Firewall settings to enable TCP with the designated port *without specifying which program the rule applies to*.

OpenAPI description

The Viz Engine installer also deploys an OpenAPI description of the REST interface to *C:\Program Files\Vizrt\VizEngine\library\REST\openapi.json*. This single file describes all endpoints with their parameters and responses. It can be imported into API tools such as Postman, Insomnia or Swagger UI, or used to generate client code.

Copyright © 2026 Vizrt

5.10 User Account Control

Since version 3.7.1, Viz Artist/Viz Engine is UAC aware. This means Viz Artist/Viz Engine can run when UAC is enabled on the computer.

IMPORTANT! UAC configuration is the responsibility of the individual company's own IT policy.

When UAC is enabled, an additional confirmation prompt shows when these items are installed or removed:

- Viz Artist/Viz Engine
- Viz One Services (Mediaftp or fsmon)
- The web service for the Viz Engine REST Interface

When UAC settings are modified, a reboot is required. Viz Artist/Viz Engine and all plug-ins shipped with the Viz Artist installer are UAC aware. However, if other plug-ins are used, a warning message may be displayed during Viz Engine startup:

WARNING: The plug-in *<plugin-name>.vip* may not be UAC aware. Contact your plug-in vendor for an updated version.

This warning means that the plug-in may not work correctly if UAC is enabled. For example, the plug-in might attempt to write into the installation folder, which is not allowed anymore. Contact the plug-in vendor for information on the UAC awareness of the plug-in.

IMPORTANT! Even though the warning shows, the plug-in is still loaded on startup.

Most changes result from the UAC requirement that an application must not write into the installation folder.

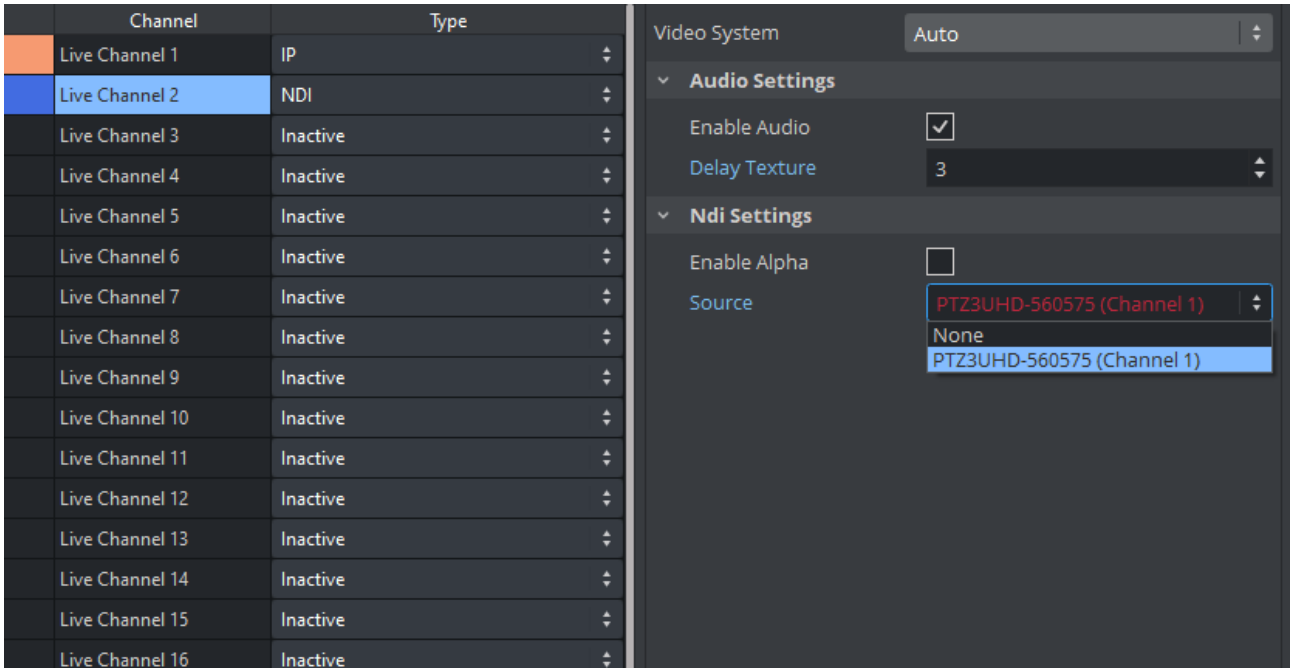
Copyright © 2026 Vizrt

5.11 NDI

The NDI protocol is built in natively into Viz Engine. It can be used in combination with a Matrox topology board or in pure Software I/O mode. Other video boards are not supported.

5.11.1 NDI Input

To receive NDI streams, set the type of a Live input channel to NDI and select an NDI source. This is done **Configuration > Video Input: Live Input**. The NDI source can also be changed with the NDI control plug-in. The NDI stream can be used as a regular live input.



If at least one NDI input is configured during startup, Viz Engine scans the network for NDI senders. All discovered sources are shown in the drop down menu.

> **Note:** If a source can not be discovered, install the NDI tools and manually add the source by using the NDI Access Manager.

The configured input video system is only used as the initial value. At runtime, Viz Engine dynamically adapts to changes in the resolution and frame rate of the NDI input stream.

NDI senders are generally not synced and run at slightly different frequencies. To compensate for these differences, Viz Engine skips or repeats NDI input frames as necessary to make playback as smooth as possible.

> **Note:** The maximum number of inputs is limited by the network capabilities and the resolution. Usually two 1080p inputs on a dedicated network are realistic.

If the alpha channel of the NDI input signal needs to be considered, *Enable Alpha* needs to be turned on.

Copyright © 2026 Vizrt

Input Drop Detection

To prevent the engines console to be flooded with messages about dropping frames if the incoming signals is unstable or does not match the output framerate, you can enable the drop detection in config file or with a command.

`ndi_input_drop_detection = 1/0` in section VIDEO en/disables the detection and protects console flooding. Alternatively you can send `CONFIGURATION*VIDEO*NDI_INPUT_DROP_DETECTION SET <value>`

5.11.2 NDI Output

The Viz Engine output registers itself as NDI stream if a valid license is available and configured and the output is enabled in the Viz Configuration File. The name of the NDI sender is *VizEngine-N*, where *N* is the instance number.

> **Information:** To use NDI output in combination with Matrox boards, it has to be explicitly enabled by setting `NdiOut1.Enable = 1` in section **CHANNELS_CONFIG** in your Viz Configuration File. In Software I/O Mode NDI (see below) this the NDI output is enabled by default and can be disabled by setting `NdiOut1.Enable = 0`.

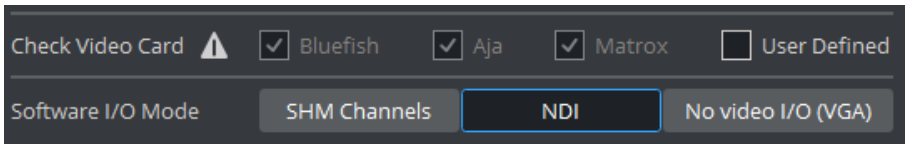
By default the NDI output of Viz Engine is a fill and key signal. To send fill only, set " `NdiOut1.ContainsAlpha = 0` " in the configuration file.

In Software I/O mode, the NDI output can also be sent as compressed NDI HX (H.264) to save network bandwidth. See NDI HX Output.

5.11.3 NDI in Software I/O Only Mode

To configure Viz Engine using NDI without any video board:

- Uncheck *User Defined* in Check Video Card.
- Set the Software I/O Mode to **NDI**.
- Live inputs can then be mapped to NDI sources in **Configuration > Video Input: Live Input.**



5.11.4 SDK Versions

The following table lists the SDK version Viz Engine is built with:

Copyright © 2026 Vizrt

Viz Engine	NDI SDK
5.6	6.3.2
5.5	6.2.1
5.4.1	6.2.1
5.4.0	6.2.0
5.3	6.0.1
5.2	5.6
5.1	5.5.3
5.0	5.5.1
4.3, 4.4	4.6
4.1, 4.2	4.1
4.0	3.8
3.14.3-3.14.5	4.1
3.14.1, 3.14.2	3.8

5.11.5 NDI HX Output

Starting with Viz Engine 5.6, the NDI program output can be sent as **NDI HX**: an H.264-compressed NDI stream encoded on the NVIDIA GPU (NVENC), instead of the standard full-bandwidth NDI stream (SpeedHQ). NDI HX needs a fraction of the network bandwidth, at the cost of some picture quality and latency. The NDI source keeps its name (*VizEngine-N*), so receivers do not need to be reconfigured.

When to Use NDI HX

NDI HX is the better choice when network bandwidth matters more than the last bit of picture quality:

- **Remote and distributed production:** sending graphics over WAN, VPN or Internet links (for example through NDI Bridge), or over Wi-Fi.
- **Cloud-hosted engines:** where outgoing network bandwidth is limited or billed by volume.
- **Many streams on one network:** several engines feeding a vision mixer, or one engine watched by many receivers (multiviewers, producer or confidence monitoring) on a 1 GbE network.
- **NDI HX-aware tools:** software mixers, recorders and streaming encoders that take H.264 NDI sources.

Copyright © 2026 Vizrt

- **Less CPU compression load:** the video compression runs on the GPU's dedicated video encoder instead of the CPU.

Keep the standard NDI output when you need the highest picture quality or critical keying (H.264 4:2:0 compression softens fine text, thin lines and key edges), the lowest latency, interlaced output, HDR, more than 8 bits, or fill and key in UHD.

Standard NDI and NDI HX Compared

	Standard NDI (default)	NDI HX
Video codec	SpeedHQ (intra-frame)	H.264 High profile, 8-bit 4:2:0, BT.709, constant bit rate
Bandwidth at 1080p50	more than 100 Mbps	about 14 Mbps by default (configurable), plus a preview stream of about 5 Mbps
Compression runs on	CPU	NVIDIA GPU (NVENC)
Formats	Progressive and interlaced, 8 and 16 bit, SDR and HLG	Progressive, 8-bit, SDR only
Fill and key	Yes	Up to 2048 lines (for example 1080p); UHD is sent fill only
Latency	Lowest	One frame more on the sender, plus decoding on the receiver
Audio	PCM	PCM (unchanged)

To Enable NDI HX

1. Configure Viz Engine for **Software I/O mode NDI.**
2. In section **CHANNELS_CONFIG** of the Viz Configuration File, set:

```
NdiOut1.HxEnable = 1
NdiOut1.HxTargetBitRateInBitsPerSec = 0
```

3. Restart Viz Engine.
4. Check the console for a line like: `NDI HX output enabled: 'VizEngine-1' 1920x1080 @ 50/1 H.264 (NVENC), 14200000 bps.` While the engine is on air, the **Video Encode** graph of the GPU in Windows Task Manager shows activity.

Copyright © 2026 Vizrt

Setting	Default	Description
`NdiOut1.HxEnable`	`0`	`1` sends the NDI output as NDI HX (H.264). `0` sends standard NDI.
`NdiOut1.HxTargetBitRateInBitsPerSec`	`0`	Target bit rate of the main stream in bits per second, for example `8000000` = 8 Mbps. `0` uses the bit rate the NDI SDK recommends for the output format (14.2 Mbps at 1080p50).

Note: These settings are available in the configuration file only, and changes need a restart. `NdiOut1.ContainsAlpha` works as for standard NDI. The codec, keyframe interval and preview stream are fixed.

What Viz Engine Sends

- **Main stream:** full output resolution and frame rate, constant bit rate, with a keyframe about every second. A receiver that connects mid-stream gets a keyframe immediately.
- **Preview stream:** 640 pixels wide (640x360 for 16:9) at the same frame rate, about 5 Mbps at 50 fps. Receivers in preview or low-bandwidth mode (multiviewers, Studio Monitor in low-bandwidth mode) use it. It is not sent when the output is 640 pixels wide or less.
- **Alpha:** with `NdiOut1.ContainsAlpha = 1` (the default), fill and key are sent as one H.264 stream with alpha. Alpha doubles the encoded picture height, and the H.264 encoder is limited to 4096 lines. So alpha is available up to 2048 lines; UHD outputs are sent fill only, with a warning in the console.
- **Audio, Viz Trio and TriCaster integration** work as with standard NDI.

Performance

Encoding adds work on the GPU and the CPU for every frame. On a Lenovo P620 with an NVIDIA RTX A4000, the maximum render rate at 1080p50 went from about 600 fps (standard NDI) to about 300 fps (NDI HX), and was about 160 fps at UHD50. Both are well above real time, but test heavy scenes with HX enabled.

Automatic Fallback and Limitations

If NDI HX cannot be used, Viz Engine sends standard NDI on the same source and writes a warning to the console. In these messages, "raw video path" means standard NDI.

Situation	Result	Console message
Interlaced format (including PAL/NTSC) or 16-bit output	Standard NDI	`NDI HX: only progressive 8-bit output is supported; using the raw video path.`

Copyright © 2026 Vizrt

Situation	Result	Console message
HDR (HLG) output	Standard NDI	`NDI HX: HDR (HLG) output is not supported by the H.264 path; using the raw video path.`
Odd width or height	Standard NDI	`NDI HX: <w>x<h> is not supported (H.264 4:2:0 needs even dimensions); ...`
No NVIDIA driver or NVENC, session limit reached, or resolution beyond the encoder's limits	Standard NDI	`NDI HX: NVENC unavailable (...); using the raw video path.`
Fill and key above 2048 lines (UHD)	NDI HX, fill only	`NDI HX: alpha (A264) is unavailable for this output; sending opaque H.264 instead.`
Preview stream cannot be created	NDI HX, main stream only	`NDI HX: proxy encoder unavailable; sending the main stream only.`
Encoder error while on air (for example a driver reset)	The frame is dropped and the encoder restarts, up to 3 times in a row. After that, the output switches to standard NDI on the same source without a restart.	`NDI HX: NVENC session rebuilt after encode failure (n/3); frame dropped.` / `NDI HX: NVENC failed permanently; falling back to raw NDI video.`

Further limitations:

- **NDI HX applies only to the NDI program output in Software I/O mode** (`NdiOut1`). NDI output together with a Matrox board, *NDI Output from GFX Channel*, and additional (parallel or Media Asset) NDI outputs always use standard NDI.
- H.264 only: no HEVC, no 10-bit, no HDR.

Copyright © 2026 Vizrt

5.12 Dual Channel Mode

This section details how to configure the Dual Channel platform of Viz Artist/Viz Engine.

Dual Channel is a video version with, typically, **two program** outputs (fill and key on two channels). To support two program outputs, this option requires two graphics cards. Once installed and configured, open the two Viz Engine consoles and add commands as required, or use an external application (for example, Viz Trio or Viz Pilot) to control the Viz Engine.

5.12.1 To Configure Dual Channel

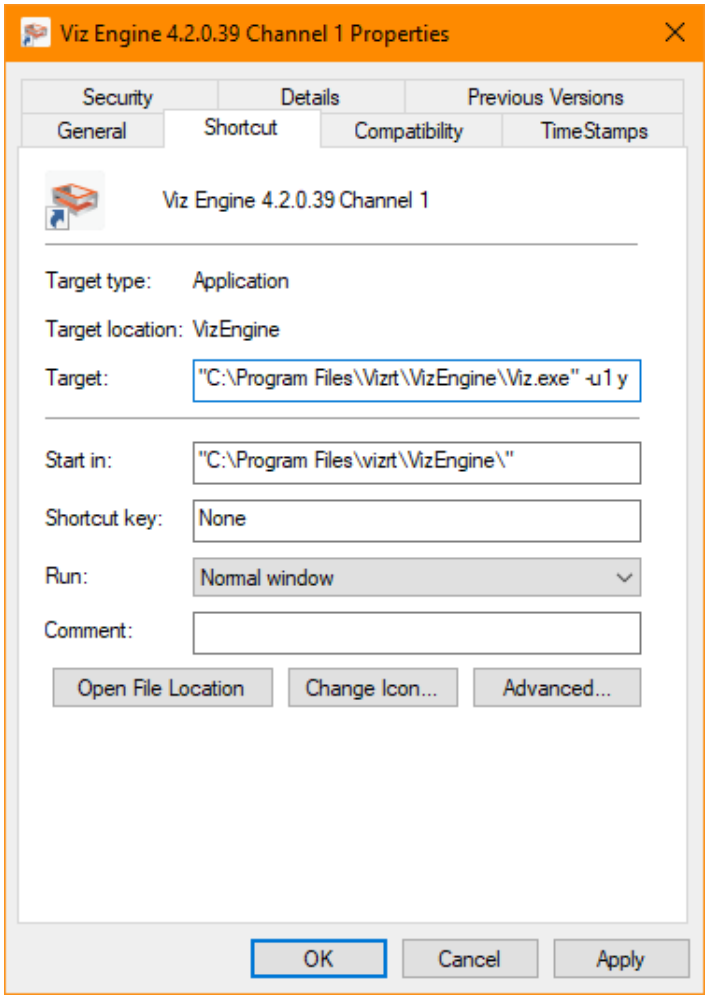
1. Install Viz Engine.
2. Create the necessary shortcuts according to the table.

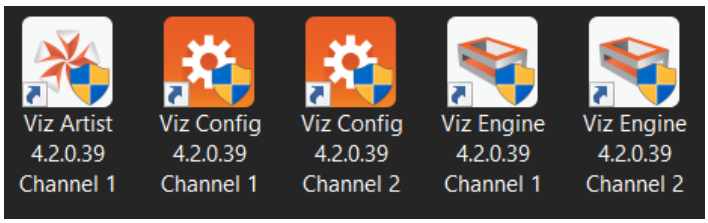
Shortcut Name	Shortcut Target Parameters
Viz Artist Channel 1	`-u1 -y`
Viz Config Channel 1	`-u1 -c`
Viz Config Channel 2	`-u2 -c`
Viz Engine Channel 1	`-u1 -n`
Viz Engine Channel 2	`-u2 -n`

Note: Viz Artist can only operate on the first Viz Engine instance.

Make sure that all shortcuts have the field **Start in:** set to your Engine installation directory. The picture shows an example for the **Viz Engine 4.2.0.39 Channel 1** shortcut.

Copyright © 2026 Vizrt





3. Open the **Viz Config Channel 1** application.
4. In **Database**, set **Auto log in** to **Yes** (active).
5. Click **Load...**



6. Select *<viz install folder>/Configuration Profiles/dualchannel-0.cfg* and click **OK**. For more information regarding the different ports and other required configuration settings see Installed Configuration Profiles.
7. Configure Viz Artist/Engine as required, by setting the **Output Format**, etc. Then click **Save** and exit Viz Config.
8. Open the **Viz Config Channel 2** application and repeat the above steps to configure the second channel, using the *dualchannel-1.cfg* configuration file.

Copyright © 2026 Vizrt

See Also

- Viz Trio User Guide
- Viz Pilot User Guide
- Installed Configuration Profiles

Copyright © 2026 Vizrt

5.13 Trio Box CG Mode

This section details how to configure the Trio Box CG (Character Generator) platform of Viz Artist/Engine.

Trio Box CG mode is a video version with, typically, **one program** and **one preview** output (fill and key on two channels). To support program and preview output, this option requires two graphics cards.

Once installed and configured, use Viz Trio to control the Viz Engines.

5.13.1 To Configure Trio Box CG

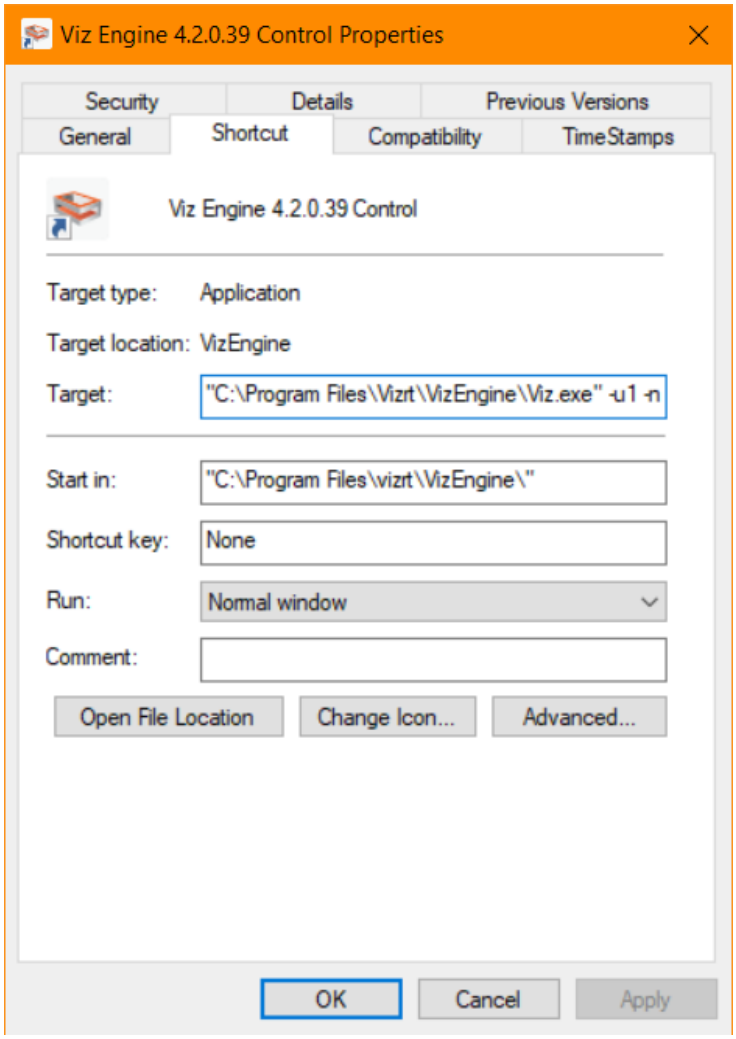
1. Install Viz Engine.
2. Create the necessary shortcuts similar to the examples in the table.

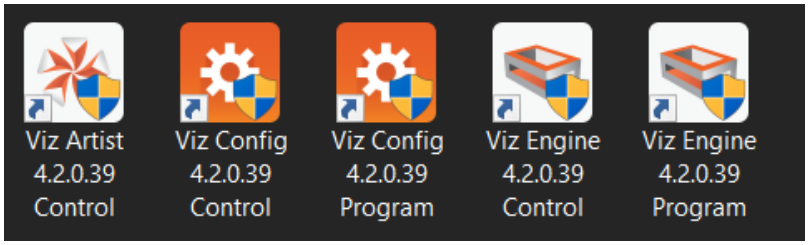
Shortcut Name	Shortcut Target Parameters
Viz Artist Control	`-u1 -y -M`
Viz Config Control	`-u1 -c -M`
Viz Config Program	`-u2 -c`
Viz Engine Control	`-u1 -n -M`
Viz Engine Program	`-u2 -n`

Note: Viz Artist can only operate on the first Viz Engine instance.

Make sure that all shortcuts have the field **Start in:** set to your Engine installation directory.
The picture shows an example for the **Viz Engine Control** shortcut.

Copyright © 2026 Vizrt





3. Open the **Viz Config Control** application.
4. In **Database**, set **Auto log in** to **Yes** (active).
5. Click **Load...**



6. Select *<viz install folder>/Configuration Profiles/trioonebox-0.cfg* and click **OK**.
7. Configure Viz Artist/Engine as required, by setting the **Output Format**, etc. Then click **Save** and exit Viz Config.
8. Open the **Viz Config Program** application and repeat the above steps to configure the second channel, using the *trioonebox-1.cfg* configuration file.

Copyright © 2026 Vizrt

Information: The preview channel needs to have a DVI out license to host the preview window.

Copyright © 2026 Vizrt

5.14 Dolby E Support

Viz Engine is certified by Dolby for decoding Dolby E streams from the inputs into the Viz Engine, and encoding the first eight audio channels back to Dolby E on the output.

This section contains the following topics:

- Dolby E Features
    - Dolby E License
- Dolby E Configuration
    - To Enable Dolby E Functionality
    - To Set Audio in Channels as Dolby E Inputs
    - To Enable Dolby E Encoding

5.14.1 Dolby E Features

The Dolby E feature set in Viz Artist/ Engine is defined as follows:

- Dolby E is supported on Matrox Video I/O hardware only
- Dolby E is supported both as AES or embedded audio on the input and output side
- Decoding of one Dolby E stream (minimum eight/maximum 16 Dolby E channels) on all inputs.

> ⚠ **Note:** The Dolby E stream (encoded in a stereo pair) must be on the first two AES or embedded audio channels

- Encoding of one Dolby E stream (minimum eight/maximum 16 Dolby E channels) on the fill output.

> ⚠ **Note:** The Dolby E signal is encoded on the first two AES or embedded audio channels on the output.

- All audio channels from the Viz timeline can be mixed, as usual, with the decoded Dolby E signal from the inputs, and is output together with the input audio as Dolby E.
- Dolby E encoding and decoding can be configured independently. It is possible to input PCM audio and output Dolby E or vice versa.
- You can also have PCM audio on one or more inputs and Dolby E audio on different input and mix them together.
- Dolby E decoding from clip channels is **not** supported.

Dolby E License

Each Dolby E stream processed in the system requires a license and a physical Dolby-E dongle.

For example, if there are two input signals with Dolby E and the output is to deliver Dolby E as well, two decoder and one encoder licenses are required. The Dolby E licenses are software licenses which reside on a software dongle and can hold multiple licenses for one system.

Copyright © 2026 Vizrt

5.14.2 Dolby E Configuration

This section details the setup required in the Viz Config file and GUI.

To Enable Dolby E Functionality

The Dolby E signal can be present in the embedded audio of the video signals, or on the AES inputs of the Matrox card.

1. Open the Viz Config file.
2. Set **DolbyEEnabled** to `1`.

To Set Audio in Channels as Dolby E Inputs

The Dolby E stream must be present in the sub-channel 0 and sub-channel 1 of the input.

1. Open the Viz Config file.
2. Configure these settings as shown:
   - ChannelDolbyEEnabled__0 = `1`
   - ChannelDolbyEEnabled__1 = `0`
   - ChannelDolbyEEnabled__2 = `0`
   - ChannelDolbyEEnabled__3 = `0`
   - ChannelDolbyEEnabled__4 = `0`
   - ChannelDolbyEEnabled__5 = `0`
   - ChannelDolbyEEnabled__6 = `0`
   - ChannelDolbyEEnabled__7 = `0`

To Enable Dolby E Encoding

This enables the encoding of the first eight internal audio channels to a Dolby E stream which are sent to the output.

- Set **DolbyEOutput** to `1`.

> ⚠ **Note:** Only live video inputs are supported.

Copyright © 2026 Vizrt

5.15 EVS Video Server Control

An EVS Control Plug-in lets the Viz Engine control an EVS video server (like a tape deck over RS422). This gives Viz Engine the ability to load and control playback of EVS video server clips.

This section contains information on the following topics:

- Setup Requirements
- RS422 and XtenDD35 Configuration
- RS422 Pin-out for the Connector Cable
- RS422 Controller Set Up Examples
- Bluestorm LP PCI Card Configuration
- ExSys EX-1303 USB to RS422 Connector Configuration

5.15.1 Setup Requirements

- The computer which runs the Viz Engine must be equipped with an RS422 controller that maps the controller ports to the Windows COM ports.
- The RS422 port must be connected to an RS422 remote controller port of the EVS video server.
- The EVS player, controlled by the RS422 port, must be set up to use the protocol *XtenDD35* (No other protocol is currently supported).

5.15.2 RS422 and XtenDD35 Configuration

1. You need a RS422 controller that installs the RS422 port as a new COM port in Windows.
2. Set up the *XtenDD35* protocol on the used remote port of the EVS video server.

IMPORTANT! This must be done before an attempt to connect.

3. Manually set the Windows COM port settings to:
   - `38400 baud`
   - `8 bytes`
   - `one stop bit`
   - `odd parity`

IMPORTANT! This must be set before Viz Artist is started.

4. Usually, a special RS422 cable is required to connect the controller to the EVS video server. It is recommended to use a connector cable to connect the RS422/DB9 connector to a standard RS422 cable. The RS422 cable should work with a normal EVS video server controller.
5. Every RS422 controller has a different pin-out setting and requires a different connector cable. The table below shows which signal of the RS422 controller must be connected to which pin on the EVS side:

Copyright © 2026 Vizrt

5.15.3 RS422 Pin-out for the Connector Cable

Signal type of RS422 controller	Cable pin on EVS side
RxD B+ (in)	7
TxD B+ (out)	3
TxD A- (out)	8
RxD A- (in)	2
Sig Ref / Gnd	1

5.15.4 RS422 Controller Set Up Examples

The connection of a RS422 controller to the EVS is always different for each controller. Here are two examples:

5.15.5 Bluestorm LP PCI Card Configuration

Viz Engine side (Bluestorm LP card)		EVS side	
Signal type	**Pin**	**Pin**	**Signal type**
TxD+	2	3	RxD+
TxD-	3	8	RxD-
RxD-	4	2	TxD-
RxD+	1	7	TxD+
Sig Ref	5	1	Sig Ref

5.15.6 ExSys EX-1303 USB to RS422 Connector Configuration

Viz Engine side (ExSys EX-1303)		EVS side	
Signal type	**Pin**	**Pin**	**Signal type**

Copyright © 2026 Vizrt

Viz Engine side (ExSys EX-1303)		EVS side	
TxD+	2	3	RxD+
TxD-	1	8	RxD-
RxD-	4	2	TxD-
RxD+	3	7	TxD+
Sig Ref	5	1	Sig Ref

Copyright © 2026 Vizrt

5.16 Integration with Viz One

Viz Engine can be integrated with Viz One so that video clips can be transferred, and monitored, to and from Viz One. The following sections describe how to install the Transfer and Monitor services to the Viz Engine and how to enable clip playout.

A Viz One system, which is configured to communicate with the Viz Engine, must be running during the installation and configuration of the two services. Administrator rights are required to complete the install (see Prerequisites).

This section contains information on the following topics:

- Configure Viz Engine
  - To Configure the Viz Engine
- Install Transfer and Monitor Services on Viz Engine
  - To Install the Transfer and Monitor Services
  - To Remove the Transfer and Monitor Services
- To Configure Viz One
- Configure Local Preview of Video Files

5.16.1 Configure Viz Engine

Viz Engine must be configured for the transfer and playing-out of video clips from Viz One.

To Configure the Viz Engine

1. Open **Viz Configuration**.
2. Click on **Output Format**.
3. Select the correct output format as used in Viz One.
4. Click on **Video Board**.
5. Click on **Video Input**.
6. Make sure that at least one Clip Channel is active.
7. Click **Save**.
8. Restart Viz Engine.

5.16.2 Install Transfer and Monitor Services on Viz Engine

The **Mediaftp** (file transfer) and **Fsmon** (file system monitor) Services are required for Viz One a Viz Engine to exchange data. Both services are part of the Viz Artist Bundle installation, but must be installed (activated) or removed (deactivated) through the Viz Configuration.

> ⚠ **Note:** If these Services are not required, make them unavailable when Viz Artist is installed. The Mediaftp Service prints some feedback to the Viz Engine Console. The Fsmon Service does not.

The files for Mediaftp and Fsmon are installed into the folders:

- *C:\Program Files\Vizrt\Fsmon*
- *C:\Program Files\Vizrt\Mediaftp*

Log-files are written to the folders:

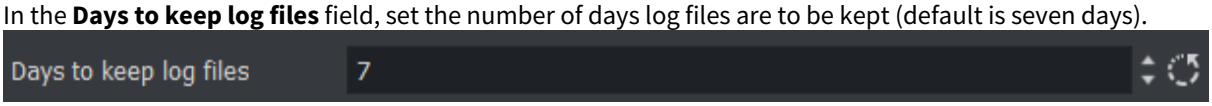
Copyright © 2026 Vizrt

- *<viz data folder>\Fsmon*
- *<viz data folder>\Mediaftp*

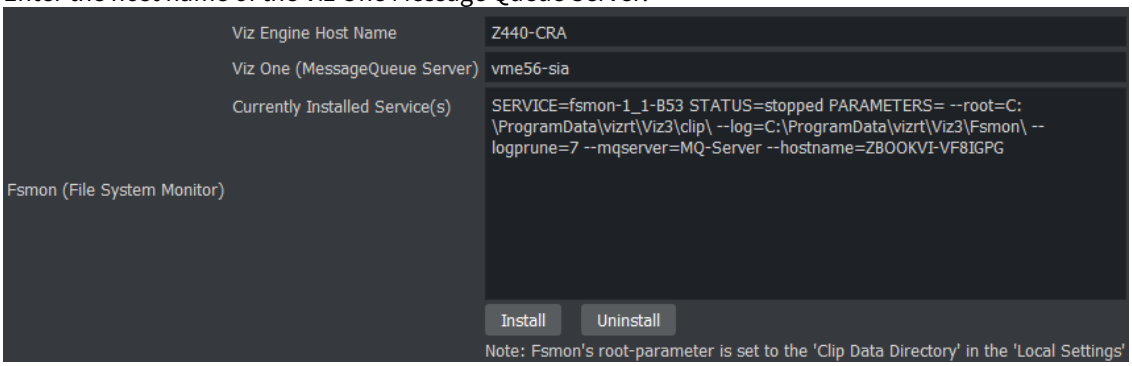
To Install the Transfer and Monitor Services

IMPORTANT! Make sure that the Clip Data Directory is set correctly before the services are installed (see Video Board).

1. Open **Viz Config**.
2. Click on Viz One.
3. In the **Days to keep log files** field, set the number of days log files are to be kept (default is seven days).

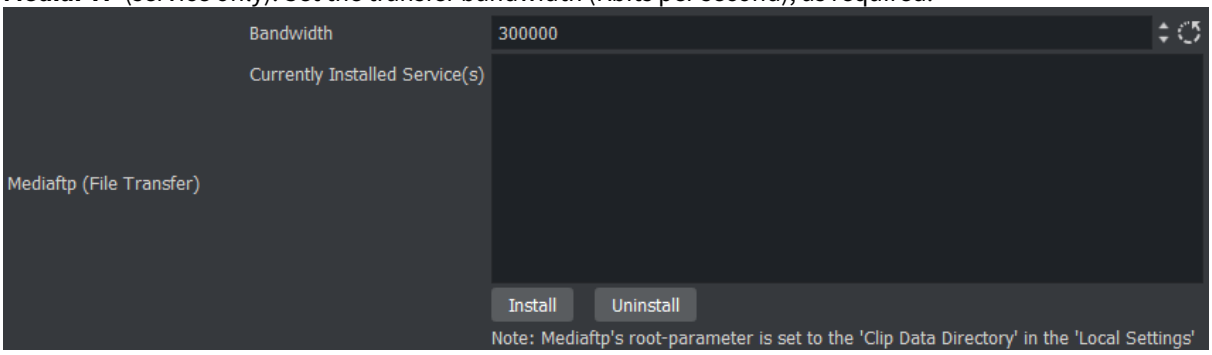


4. **Fsmon** (service only):
   a. Enter the host name of the localhost.
   b. Enter the host name of the Viz One Message Queue Server.



IMPORTANT! The localhost name must be exactly the same string that was entered in the Viz One as host name for the Viz Engine (it must be the exact same string as seen beside Host on the Servers page).

5. **MediaFTP** (service only): Set the transfer bandwidth (Kbits per second), as required.



Note: Mediaftp is installed with the `-a` flag. This assumes that the default Viz One user/password combination is: vtrsync/vtrsync. Mediaftp installs, but does not work, if the user/password combination is different (see **Adding a Viz Engine** in the Viz One Administrators Guide).

Copyright © 2026 Vizrt

6. Click on **Install** for each required Service.

> **Note:** Any currently installed Services must be removed before a new Service can be installed.

To Remove the Transfer and Monitor Services

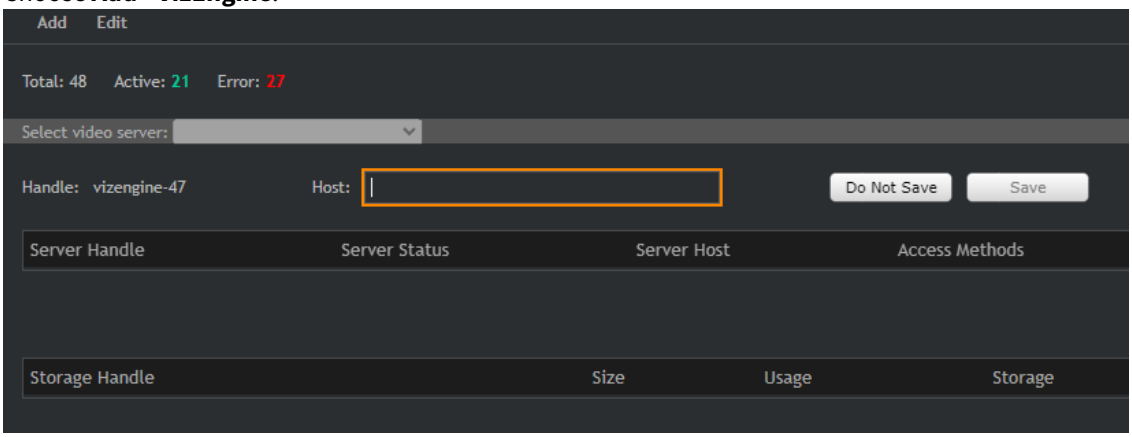
> **Note:** When a Service is removed, the Service is made **inactive**. The Service is still available, and if required again, click **Install** to make **active**.

1. Open **Viz Config**.
2. Click on **Viz One**.
3. Click on **Uninstall** for each required Service.

5.16.3 To Configure Viz One

Viz Engines are configured a video server with local storage on Viz One side. For detailed information, please refer to the Viz One documentation. Before adding any Viz Engine to Viz One, make sure that FSMon and MediaFTP are running on Viz Engine.

- Login to Viz One Studio.
- Switch to **Administration**.
- Click on **VideoServers**.
  - Choose **Add - VizEngine**.



  - Enter the Viz Engine host name. Be aware that this is case sensitive.
- Click on **Servers**
  - Verify the Credentials for the FTP/ARDFTP access are correct.

Once configured correctly, Viz Engine must show up as **Active** in the Servers Dashboard.

5.16.4 Configure Local Preview of Video Files

The VML player is recommended for previewing proxy clips from Viz One.

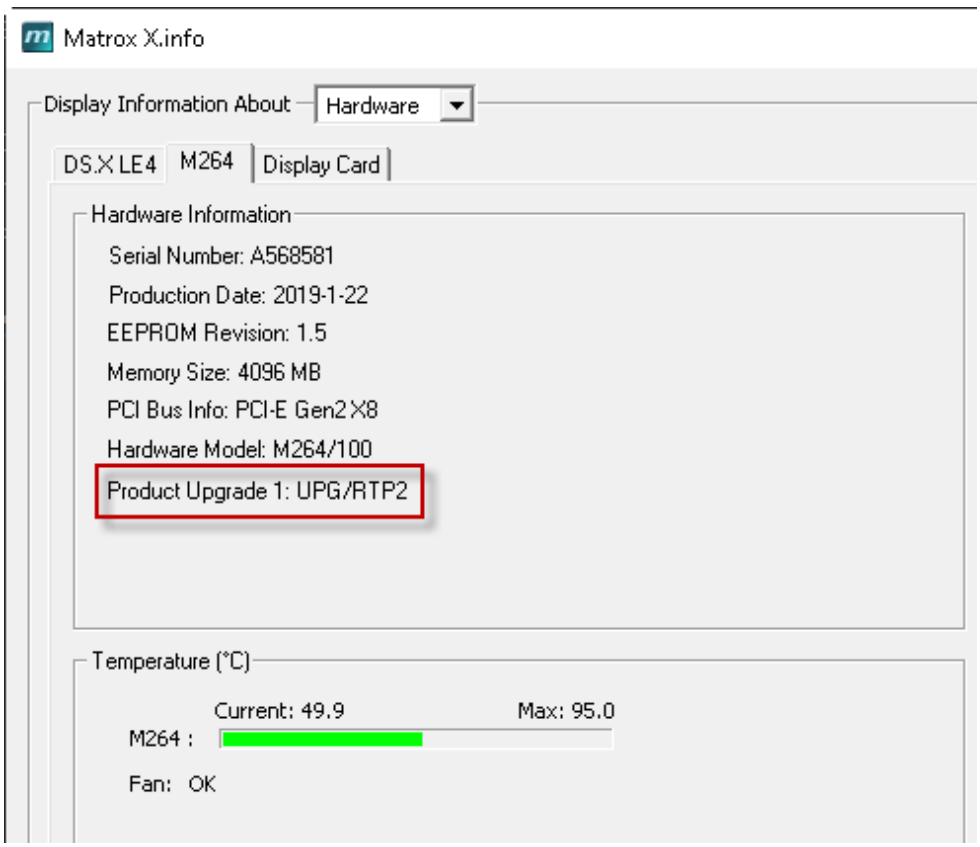
For more information about using the VML player, please see VML Clip Player.

Copyright © 2026 Vizrt

5.17 Matrox Stream

- Matrox License
- Example Matrox License Configuration
- Output Mode
- Configuring a Stream
- Example Sending MPEG-TS over RTP and Receiving with Viz Engine on localhost
- Example Receiving MPEG-TS over UDP on localhost
- Example Receiving SRT on localhost
- Example Receiving RTSP on localhost
- Example Receiving RTMP on localhost
- More Advanced Example Using ffmpeg
- Receiving Different Input Types
- Limitations
- Known Issues and Troubleshooting

Viz Engine can send MPEG-TS over RTP and receive MPEG-TS over RTP/UDP, SRT, RTSP, RTMP streams by using DSX.Core or any Matrox board that has an RTP / RTP2 / STMP upgrade.



Important: MPEG-TS over RTP/UDP, SRT, RTSP, RTMP streaming require a Mezzanine IP license. Progressive formats are supported for output only. However, both progressive and interlaced formats are supported for input.

Features	Input	Output
Progressive	✓	✓
Interlaced	✓	✗
MPEG-TS over RTP	✓	✓
MPEG-TS over UDP	✓	✗
RTSP	✓	✗
RTMP	✓	✗
SRT	✓	✗

Note: Only H.264 video and AAC audio are supported. Moreover, only audio with two channels is supported. If a different audio characteristic is detected in the stream, the audio is ignored and only video is displayed.

Note: Since only broadcast resolution is supported and there is no framerate conversion between input and output, input and output framerates must match.

5.17.1 Matrox License

An RTP / RTP2 / STMP Matrox license is required in order to be able to receive any kind of stream. Which license is required depends on your existing license.

Important: The main board, the first one appearing in *Matrox. Devices* in the Viz Engine configuration file, must have decoding capabilities. A board has decoding capabilities if it is /500 or /550, or if it has any of the following upgrades: U50, U55 or STMP.

In order to receive a stream, one of the following streaming licenses is also required for the board:

- **RTP:** Lets you receive any kind of stream except RTMP and MPEG-TS, since it lacks a demuxer.
- **RTP2:** Lets you receive any kind of stream except MPEG-TS, since it lacks a demuxer.
- **STMP:** Lets you receive any kind of stream.

Note: To be able to receive a stream, the license can be installed in a secondary board, it does not need to be installed in the main board.

Copyright © 2026 Vizrt

Boards with decoding capabilities also support muxing and demuxing. For example, if a board has decoding capabilities (if it's /500, has U50 upgrade, etc.) and has an RTP license, the board is also able to receive MPEG-TS.

Boards with STMP do not need to have any other upgrades to be able to receive and decode streams; this license is usually used with /100 boards and when clip playback is not needed.

To summarize:

- **RTP / RTP2:** Are only for receiving and do not provide decoding. Additional upgrades are therefore needed, usually in the form of U50 or U55.
- **STMP:** Offers everything and is therefore usually paired with a /100 board.

There are exception to the above rules:

- **DSX.Core:** By default supports any kind of stream **except RTMP**. However, to support decoding, the license should be at least /500.
- **M264:** By default supports **only video** decoding. However, to support streaming, either of the following receiving licenses must be added: RTP / RTP2.

5.17.2 Example Matrox License Configuration

For a machine that has the following boards:

DSX.LE4L/8/100F with UPG/100/U55

M264/100 with UPG/RTP2

- Choosing the DSX.LE4 as main board and the M264 as secondary board, the machine can:
  - Use the following input types: SDI, MPEG-TS over RTP/UDP, SRT, RTSP, RTMP, NDI, Clip.
  - Have SDI as output.
  - Use H.264 hardware decoding to decode the incoming streams.
- Choosing only DSX.LE4 as main board and no secondary board, the machine can:
  - Use the following input types: SDI, NDI, Clip.
  - Have SDI as output.
- Choosing only M264 as main board and no secondary board, the machine can:
  - Use the following input types: SRT, RTSP, RTMP, NDI, Clip (only uncompressed clips).
  - Audio decoding is not supported.

⚠ **Note:** In the case of a M264 main board, the machine does **NOT** have any output since RTP requires a muxer, which is only provided in a board with /500 /550 or upgrades U50, U55 or STMP. The machine does **NOT** support audio decoding or receiving MPEG-TS over RTP/UDP for the same reason.

5.17.3 Output Mode

There are two different output modes: SDI or RTP. To enable output to SDI, a Matrox board supporting SDI is needed and must be chosen in the configuration file: Matrox.Devices = **A520888**.

If the SDI board does not support an RTP upgrade, another board must be installed and must be included as a secondary board: Matrox.Devices = **A520888, A520898**.

Copyright © 2026 Vizrt

If the main board, the first one appearing in Matrox.Devices, has the corresponding Matrox license to support RTP, then RTP output can be enabled.

To do so, change the entry *RtpOut1.Enable* in the configuration file to `1` : **RtpOut1.Enable = 1**.

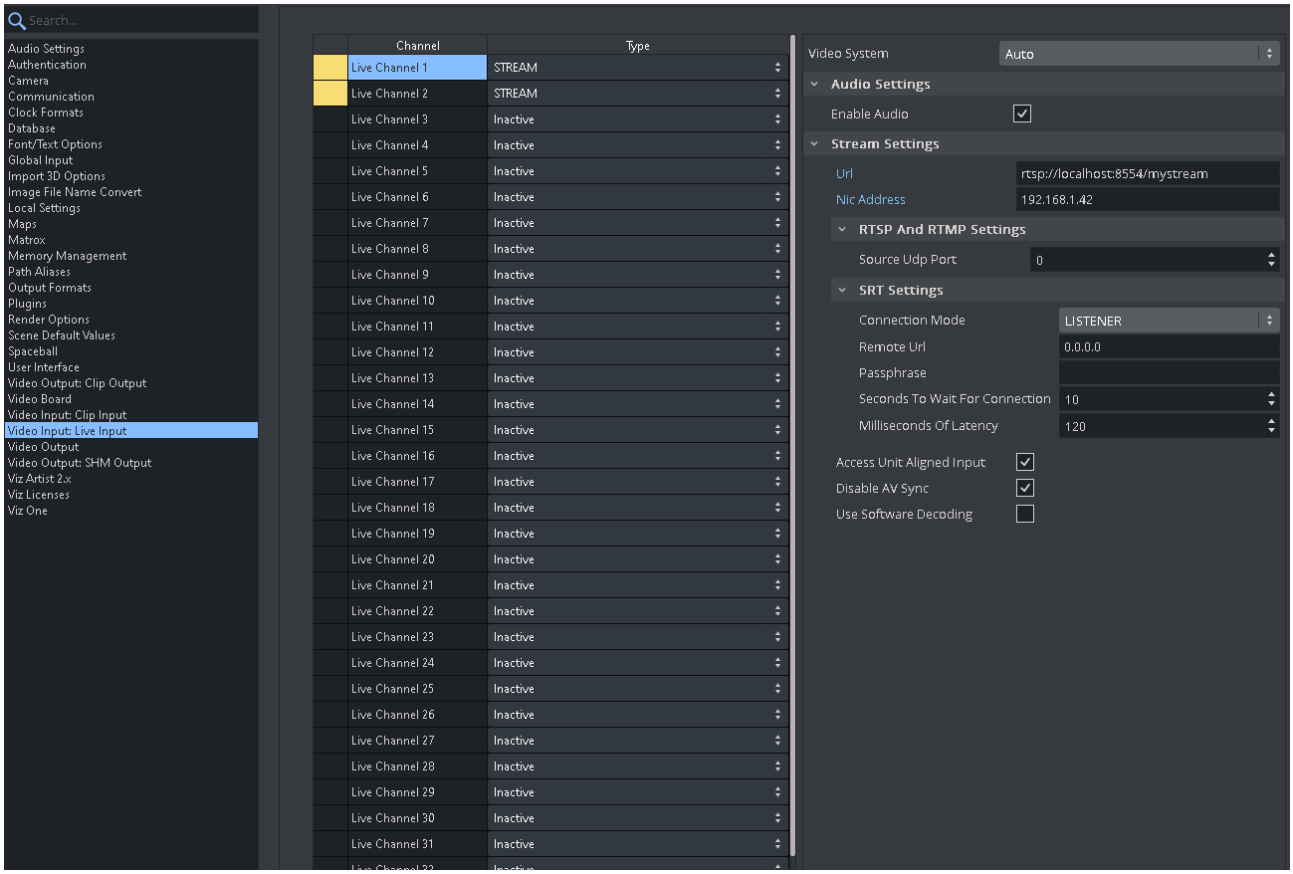
Specify a time server in the configuration file. It is also recommended to set an NTP time server: **ntp_server = 131.107.13.100**.

> **Info:** SDI Out, NDI Out, RTP output can be enabled at the same time.

For additional parameters to configure bitrates, enable verbose settings in the configuration file: **VerboseConfig = 1**.

5.17.4 Configuring a Stream

A stream can now be configured using the GUI interface under Live Input:



The only thing that differentiates different type of streams are their URLs:

```
rtp://localhost:22404
udp://localhost:22200?pkt_size=1316
srt://localhost:2345
```

Copyright © 2026 Vizrt

```
rtsp://localhost:8554/mystream
rtmp://localhost:30000/live/1
```

The best way to test the functionality of a stream is by using ffmpeg. ffmpeg supports the sending of simple pattern that allows us to do basic sanity test.

> **Note:** Even if a stream is being sent on localhost it is advisable to use one of the NIC as IP since otherwise it could cause problems. This issue is only known to happen to **RTP** stream.

5.17.5 Example Sending MPEG-TS over RTP and Receiving with Viz Engine on localhost

The following command can be used to simulate a source:

```
ffmpeg.exe -f lavfi -re -i smptehdbars=s=1280x720:r=60[out0] -c:v libx264 -profile:v
high422 -pix_fmt yuv422p -x264-params "nal-hrd=cbr" -b:v 2M -minrate 2M -maxrate 2M
-bufsize 2M -f rtp_mpegts -bsf:v h264_mp4toannexb "rtp://10.251.2.32:22404"
```

Viz Engine can then be configured with:

```
live_type1 = STREAM
live_system1 = 720P_6000_SMPTE296
LiveIn1.Url = rtp://10.251.2.32:22404
LiveIn1.NICAddress = 10.251.2.32
```

In this case, since we are sending to localhost, NIC address can be any of the installed network adapter IP. As mentioned previously, this configuration can be done via GUI.

5.17.6 Example Receiving MPEG-TS over UDP on localhost

The following command can be used to simulate a source:

```
ffmpeg.exe -f lavfi -re -i smptehdbars=s=1280x720:r=60[out0] -c:v libx264 -profile:v
high422 -pix_fmt yuv422p -x264-params "nal-hrd=cbr" -b:v 2M -minrate 2M -maxrate 2M
-bufsize 2M -f mpegts -bsf:v h264_mp4toannexb "udp://10.251.2.32:22404?pkt_size=1316"
```

Viz Engine can then be configured with:

```
live_type1 = STREAM
live_system1 = 720P_6000_SMPTE296
LiveIn1.Url = udp://10.251.2.32:22404
LiveIn1.NICAddress = 10.251.2.32
```

In this case, since we are sending to localhost, NIC address can be any of the installed network adapter IP. As mentioned previously, this configuration can be done via GUI.

Copyright © 2026 Vizrt

5.17.7 Example Receiving SRT on localhost

For this **srt-live-transmit** is needed to wrap a UDP stream. First, send the UDP stream as described in the example above and then run the following:

```
srt-live-transmit "udp://10.251.2.32:22200?pkt_size=1316" srt://10.251.2.32:2345
```

Viz Engine can then be configured with:

```
live_type1 = STREAM
live_system1 = 720P_6000_SMPTE296
LiveIn1.Url = srt://10.251.2.32:2345
LiveIn1.NICAddress = 10.251.2.32
```

In this case, since we are sending to localhost, NIC address can be any of the installed network adapter IP. As mentioned previously, this configuration can be done via GUI.

5.17.8 Example Receiving RTSP on localhost

For this an RTSP server is needed. **rtsp-simple-server** is probably easiest way to get an RTSP server. After running the server run the following:

```
ffmpeg.exe -f lavfi -re -stream_loop -1 -i smptehdbars=s=1280x720:r=60[out0] -c:v
libx264 -profile:v high422 -pix_fmt yuv422p -x264-params "nal-hrd=cbr" -b:v 2M
-maxrate 2M -minrate 2M -bufsize 2M -f rtsp rtsp://10.251.2.32:8554/mystream
-rtsp_transport tcp
```

The URL and port depend on how nginx has been configured.

Viz Engine can then be configured with:

```
live_type1 = STREAM
live_system1 = 720P_6000_SMPTE296
LiveIn1.Url = srt://10.251.2.32:2345
LiveIn1.NICAddress = 10.251.2.32
LiveIn1.SrcUdpPort = 22200
```

In this case, since we are sending to localhost, NIC address can be any of the installed network adapter IP. As mentioned previously, this configuration can be done via GUI.

For receiving multiple RTSP stream the configuration SrcUdpPort must be different for each of them. This port number should also be separated by 4. In this example, the next live would be, LiveIn2.SrcUdpPort = 22204.

Copyright © 2026 Vizrt

5.17.9 Example Receiving RTMP on localhost

For this an RTMP sever is needed. **nginx** is probably easiest way to get an RTMP server. After running the server run the following:

```
ffmpeg.exe -f lavfi -re -stream_loop -1 -i smptehdbars=s=1280x720:r=60[out0] -c:v
libx264 -profile:v high422 -pix_fmt yuv422p -x264-params "nal-hrd=cbr" -b:v 2M
-maxrate 2M -minrate 2M -bufsize 2M -f flv rtmp://10.251.2.32:30000/live/1
```

The URL and port depend on how nginx has been configured. Viz Engine can then be configured with:

```
live_type1 = STREAM
live_system1 = 720P_6000_SMPTE296
LiveIn1.Url = rtmp://10.251.2.32:30000/live/1
LiveIn1.NICAddress = 10.251.2.32
```

In this case, since we are sending to localhost, NIC address can be any of the installed network adapter IP. As mentioned previously, this configuration can be done via GUI.

For receiving multiple RTSP stream the configuration SrcUdpPort must be different for each of them. This port number should also be separated by 4. In this example, the next live would be, LiveIn2.SrcUdpPort = 22204.

5.17.10 More Advanced Example Using ffmpeg

It is possible to send clip content using ffmpeg and receive this on the Viz Engine. The clip must be first transcoded to a format that is supported by Viz Engine. This can be done in the following way:

```
ffmpeg.exe -i "clip_input.mp4" -filter:v
"fps=fps=60,scale=1280:720,setdar=16/9"-minrate 2M -maxrate 2M -bufsize 2M -vcodec
libx264 -preset slow -ac 2 -ab 320k -acodec aac -strict -2 -t 00:05:00
"clip_output.mp4"
```

The above command generates a clip that Viz Engine can support. It also makes sure that the audio is two channels and has a constant bit rate. The generated clip can then be send using the same commands as described in the previous sections.

It is also possible to use CUDA with ffmpeg to reduce the CPU usage in case testing multiple sources is needed:

```
ffmpeg -vsync 0 -c:v h264_cuvid -re -stream_loop -1 -i clip_output.mp4 -c:v
h264_nvenc -profile:v high422 -pix_fmt yuv422p -x264-params "nal-hrd=cbr" -b:a 320k
-acodec aac -strict -2 -b:v 2M -maxrate 2M -minrate 2M -bufsize 2M -f rtsp rtsp://
10.251.2.32:8554/mystream -rtsp_transport tcp
```

When sending it is advice to force the audio bit rate to constant even if the clip itself is already.

Copyright © 2026 Vizrt

5.17.11 Receiving Different Input Types

It is possible to combine different input types in the same scene. The possible input types are: SDI / IP 2110, MPEG-TS over RTP/UDP, SRT, RTSP, RTMP, NDI, Clip.

5.17.12 Limitations

There are a lot of factors that must be taken into account to know how many streams can be received into the engine. However the following hard limit is good to have in mind. The engine **might** be able to receive up to 16x1080p60 streams. To achieve this the Engine/system must have the following minimum requirements:

- P6000 or superior.
- M264S2 or superior.
- Network must be free from congestion.
- Stream must be 4:2:0 8-bit Long GOP.
- The scene must be simple.

Receiving 16x1080p60 is the **absolute upper limit** and use most of the system resources. There is **no guarantee** that this can be achieved since there are too many variables in place.

5.17.13 Known Issues and Troubleshooting

- Artifacts: This is usually due to the characteristics of the stream (decoding parameters such as GOP size, bandwidth size, buffer size, etc.). The best way to know if the stream or the network is the issue, is to use **ffplay** and try to receive the stream with it. **ffplay** usually throws a lot of errors and it also shows the stream with artifacts. If this is the case we know the issue is in the source.
- Can not receive stream: Try to receive the stream first with ffplay.

Copyright © 2026 Vizrt

5.18 VML Clip Player

VML Clip Player is a direct replacement for DirectShow Clip Player and an alternative to Matrox Clip Player. The main goal of VML Clip Player is to be able to play non-broadcast resolution while offering similar functionality and behavior to Matrox Clip Player.

The user can choose the **preferred fallback mode**, between Matrox with Fallback, VML Only, and Matrox Only, via the following configuration (*VerboseConfig* must be set to `1`):

```
# Default clip player to be used for this channel.  0=Matrox With Fallback 1=VML Only
2=Matrox Only
ClipIn1.PlayerFallbackMode: Default=0 Min=0 Max=2
ClipIn1.PlayerFallbackMode = 0
```

The default preferred fallback mode in the Matrox workflow is Matrox Clip Player with Fallback to VML Player. It is also possible to change the preferred fallback mode during runtime via the following command:

```
MAIN*CONFIGURATION*CHANNELS*CLIPIN_0*PLAYERFALLBACK SET MATROX_WITH_FALLBACK
MAIN*CONFIGURATION*CHANNELS*CLIPIN_0*PLAYERFALLBACK SET VML_ONLY
MAIN*CONFIGURATION*CHANNELS*CLIPIN_0*PLAYERFALLBACK SET MATROX_ONLY
```

5.18.1 Feature Comparison between Clip Players

Not all functionalities are available for VML. The following is a feature comparison between the three different supported clip players:

Functionality	VML	Matrox	DirectShow (Viz 4.x and earlier)
Basic Functions (Play, Pause, Stop, Load, Scrub, Continue, Flush, Autorun)	✓	✓	✓
In and out setting and getting	✓	✓	✓
Loop playback	✓	✓	✓
3-point loop / 4-point loop	✓	✓	✗
Upscale luma	✗	✓	✗
Repeat field	✓	✓	✗
Reverse field	✓	✓	✗

Copyright © 2026 Vizrt

Functionality	VML	Matrox	DirectShow (Viz 4.x and earlier)
Playback speed	✓	✓	✓
Native pending support	✓	✓	✓
Embedded alpha format	✓	✓	✓
Multiple files (alpha / fill / audio / VBI files)	✗	✓	✗
Timecode	✓	✓	✗
VBI Passthrough (to output)	✗	✓	✗
Timecode Passthrough (to output)	✗	✓	✗
CC Extraction	✗	✓	✗
Audio	✓	✓	✗
SHM key events	✓	✓	✓
Contiguous seek performance	✓	✓	✗
DVE	✗ (1)	✓	✗ (1)
Playlist management (insert, remove, replace, etc.)	✓	✓	✗
Mode on load error (pending)	✓	✓	✗
V210 (10-bit support)	✓	✓	✗
Broadcast clip status	✓	✓	✓
Replay mode (TDIR)	✓	✓	✗
Growing files	✓	✓	✗
Dealing with display aspect ratio	✓	✓	✗

Copyright © 2026 Vizrt

Functionality	VML	Matrox	DirectShow (Viz 4.x and earlier)
Delay audio and video	✗	✓	✗
Hardware-accelerated decoding	✓ (4)	✓ (2)	✗
Mixed resolution mode	✓ (3)	✓ (3)	✓ (3)

(1) Simulated DVE is supported.
(2) With an additional Matrox M264 board.
(3) With a slight difference in behavior documented below.
(4) With an NVIDIA GPU that supports NVDEC.

5.18.2 Mixed Resolution Mode in Matrox and VML Clip Player

Mixed resolution mode means playing clips with different resolution than the configured Viz Engine output resolution and/or the configured Viz Engine clip channel resolution. Matrox and VML clip player support the playback in mixed resolution. In Viz 4.x and earlier, Matrox clip player was already able to playback clips in mixed resolution, as long as the clips have the same *framerate family* as the output. However from Viz Engine 5.0 and later, Matrox clip player also support the playback of **selected** clips coming from smartphones with *framerate family* different than the configured Viz Engine output resolution. VML has always supported clips with *framerate family* different than the configured Viz Engine output resolution since its inception.

Even if both Matrox and VML clip players support mixed resolution, the following differences in behavior needs to be taken into account. This is referring only to mixed resolution in which the clip has the same framerate family as the configured Viz Engine output resolution.

Player	Output	Channel	Clip	STATUS GET AT EOF	SCRUBBING	CONCLUSIONS
VML	**p50**	**p50**	**p50**	duration 1000 current_position 999 play_mode scrubbing EOF file_name D:/clips_reference/1080p50/1080p50_Video_ProRes_Encoder_ProRes[ch=2 bits=24-32]_matrox.mov frame_rate 50.00000000 markin 0 markout 1000 output_frame_rate 50.00000000	Exactly the same between Matrox and VML. ✓	Exactly the same between Matrox and VML. ✓

Copyright © 2026 Vizrt

Player	Output	Channel	Clip	STATUS GET AT EOF	SCRUBBING	CONCLUSIONS
MATROX	**p50**	**p50**	**p50**	duration 1000 current_position 999 play_mode scrubbing EOF file_name D:/clips_reference/1080p50/1080p50_Video_ProRes_Encoder_ProRes[ch=2 bits=24-32]_matrox.mov frame_rate 50.00000000 markin 0 markout 1000 output_frame_rate 50.00000000	Exactly the same between Matrox and VML. ✅	Exactly the same between Matrox and VML. ✅
VML	p50	i25	p50	duration 1000 current_position 998 play_mode scrubbing EOF file_name D:/clips_reference/1080p50/1080p50_Video_ProRes_Encoder_ProRes[ch=2 bits=24-32]_matrox.mov frame_rate 50.00000000 markin 0 markout 1000 output_frame_rate 25.00000000	Matrox displayed is -1 compared to VML on scrubbing to 00:00:05:00. ❌	Matrox **displayed** last position is -1 compared to VML at scrubbing EOF. ❌
MATROX	p50	i25	p50	duration 1000 current_position 998 play_mode scrubbing EOF file_name D:/clips_reference/1080p50/1080p50_Video_ProRes_Encoder_ProRes[ch=2 bits=24-32]_matrox.mov frame_rate 50.00000000 markin 0 markout 1000 output_frame_rate 25.00000000	Matrox displayed is -1 compared to VML on scrubbing to 00:00:05:00. ❌	Matrox **displayed** last position is -1 compared to VML at scrubbing EOF. ❌

Copyright © 2026 Vizrt

Player	Output	Channel	Clip	STATUS GET AT EOF	SCRUBBING	CONCLUSIONS
VML	p50	p50	i25	duration 2000 **current_position 1999** play_mode scrubbing EOF file_name D:/clips_reference/1080i50/1080i50_Video_AVC-Intra_Encoder_AVCIntraClass100[TimeCode] [VBI] [ch=2 bits=24-32]_avcintra.mxf frame_rate 25.00000000 markin 0 markout 2000 output_frame_rate 50.00000000	Exactly the same between Matrox and VML. ✅	Matrox **displayed** last position and Current Position is -1 compared to VML at scrubbing EOF. ❌
MATROX	p50	p50	i25	duration 2000 **current_position 1998** play_mode scrubbing EOF file_name D:/clips_reference/1080i50/1080i50_Video_AVC-Intra_Encoder_AVCIntraClass100[TimeCode] [VBI] [ch=2 bits=24-32]_avcintra.mxf frame_rate 25.00000000 markin 0 markout 2000 output_frame_rate 50.00000000	Exactly the same between Matrox and VML. ✅	Matrox **displayed** last position and "Current Position" is -1 compared to VML at scrubbing EOF. ❌
VML	p50	i25	p50	duration 2000 current_position 1998 play_mode scrubbing EOF file_name D:/clips_reference/1080i50/1080i50_Video_AVC-Intra_Encoder_AVCIntraClass100[TimeCode] [VBI] [ch=2 bits=24-32]_avcintra.mxf frame_rate 25.00000000 markin 0 markout 2000 output_frame_rate 25.00000000	Exactly the same between Matrox and VML. ✅	Exactly the same between Matrox and VML. ✅

Copyright © 2026 Vizrt

Player	Output	Channel	Clip	STATUS GET AT EOF	SCRUBBING	CONCLUSIONS
MATROX	p50	i25	p50	duration 2000 current_position 1998 play_mode scrubbing EOF file_name D:/clips_reference/1080i50/1080i50_Video_AVC-Intra_Encoder_AVCIntraClass100[TimeCode] [VBI] [ch=2 bits=24-32]_avcintra.mxf frame_rate 25.00000000 markin 0 markout 2000 output_frame_rate 25.00000000	Exactly the same between Matrox and VML. ✅	Exactly the same between Matrox and VML. ✅
VML	**i25**	**i25**	**i25**	duration 1000 current_position 999 play_mode scrubbing EOF file_name D:/clips_reference/1080i50/1080i50_Video_AVC-Intra_Encoder_AVCIntraClass100[TimeCode] [VBI] [ch=2 bits=24-32]_avcintra.mxf frame_rate 25.00000000 markin 0 markout 1000 output_frame_rate 25.00000000	Exactly the same between Matrox and VML. ✅	Exactly the same between Matrox and VML. ✅
MATROX	**i25**	**i25**	**i25**	duration 1000 current_position 999 play_mode scrubbing EOF file_name D:/clips_reference/1080i50/1080i50_Video_AVC-Intra_Encoder_AVCIntraClass100[TimeCode] [VBI] [ch=2 bits=24-32]_avcintra.mxf frame_rate 25.00000000 markin 0 markout 1000 output_frame_rate 25.00000000	Exactly the same between Matrox and VML. ✅	Exactly the same between Matrox and VML. ✅

Copyright © 2026 Vizrt

Player	Output	Channel	Clip	STATUS GET AT EOF	SCRUBBING	CONCLUSIONS
VML	i25	p50	i25	duration 1000 current_position 999 play_mode scrubbing EOF file_name D:/clips_reference/1080i50/1080i50_Video_AVC-Intra_Encoder_AVCIntraClass100[TimeCode] [VBI] [ch=2 bits=24-32]_avcintra.mxf frame_rate 25.00000000 markin 0 markout 1000 output_frame_rate 50.00000000	Exactly the same between Matrox and VML.	Matrox **displayed** last position is -1 compared to VML at scrubbing EOF.
MATROX	i25	p50	i25	duration 1000 current_position 999 play_mode scrubbing EOF file_name D:/clips_reference/1080i50/1080i50_Video_AVC-Intra_Encoder_AVCIntraClass100[TimeCode] [VBI] [ch=2 bits=24-32]_avcintra.mxf frame_rate 25.00000000 markin 0 markout 1000 output_frame_rate 50.00000000	Exactly the same between Matrox and VML.	Matrox **displayed** last position is -1 compared to VML at scrubbing EOF.
VML	i25	i25	p50	duration 500 current_position 499 play_mode scrubbing EOF file_name D:/clips_reference/1080p50/1080p50_MPEG-2__MpegIFrame422ProfileHighLevel_422[bitrate=300].avi frame_rate 50.00000000 markin 0 markout 500 output_frame_rate 25.00000000	Matrox displayed is -1 compared to VML on scrubbing to 00:00:05:00.	Matrox **displayed** last position is -1 compared to VML at scrubbing EOF.

Copyright © 2026 Vizrt

Player	Output	Channel	Clip	STATUS GET AT EOF	SCRUBBING	CONCLUSIONS
MATROX	i25	i25	p50	duration 500 current_position 499 play_mode scrubbing EOF file_name D:/clips_reference/1080p50/1080p50_MPEG-2__MpegIFrame422ProfileHighLevel_422[bitrate=300].avi frame_rate 50.00000000 markin 0 markout 500 output_frame_rate 25.00000000	Matrox displayed is -1 compared to VML on scrubbing to 00:00:05:00.	Matrox **displayed** last position is -1 compared to VML at scrubbing EOF.
VML	i25	p50	p50	duration 500 **current_position 499** play_mode scrubbing EOF file_name D:/clips_reference/1080p50/1080p50_MPEG-2__MpegIFrame422ProfileHighLevel_422[bitrate=300].avi frame_rate 50.00000000 markin 0 markout 500 output_frame_rate 50.00000000	Exactly the same between Matrox and VML.	Matrox Current Position (but not displayed one) is the same as duration but VML is duration -1.
MATROX	i25	p50	p50	duration 500 **current_position 500** play_mode scrubbing EOF file_name D:/clips_reference/1080p50/1080p50_MPEG-2__MpegIFrame422ProfileHighLevel_422[bitrate=300].avi frame_rate 50.00000000 markin 0 markout 500 output_frame_rate 50.00000000	Exactly the same between Matrox and VML.	Matrox Current Position (but not displayed one) is the same as duration but VML is duration -1.

5.18.3 Non-broadcast Clip Support in Matrox and VML Clip Player

Broadcast clips are defined by their resolution (width and height), framerate, aspect ratio, scan mode and audio sample rate (48khz). The codec and format can also define a clip as broadcast clip. Examples of broadcast clips can be found in the **Media Assets > Supported Codecs > Supported Codecs for Matrox Clip Playback** section of the Viz Artist User Guide.

From Viz Engine version 5.0 and later, Matrox Clip Player also supports non-broadcast clips. These clips usually comes from smartphones. All of these clips are progressive resolution with non-broadcast resolution, aspect ratio,

Copyright © 2026 Vizrt

framerate and audio sample rate (41.1khz). The following needs to be taken into account regarding non-broadcast clip support in Matrox clip player:

- Loading a clip with different framerate than the configured Viz Engine Output framerate, converts the clip framerate to the Viz Engine Output framerate.
- DVE or Hybrid is not supported for non-broadcast clip.
- Reverse playback might not be in real time.
- Supported containers for non-broadcast clips are mostly *.mov* or *.mp4*.

5.18.4 VML Clip Player as a Fallback Clip Player

When using Matrox clip player as the preferred clip player and the input clip is not supported, Viz Engine automatically tries to load the clip with VML clip player. However, this is not always possible, such as in the following case:

- If the clip channel is configured as DVE or Hybrid.

Three possible fallback modes can be set starting from Viz Engine 5.4 using the following command:

```
MAIN*CONFIGURATION*CHANNELS*CLIPIN_0*PLAYERFALLBACKMODE GET/SET
```

The possible fallback modes are MATROX_WITH_FALLBACK, VML_ONLY, and MATROX_ONLY.

To know which clip player is being used at the moment, the following command can be called:

```
MAIN_SCENE*VIDEO*CLIPIN*1*PLAYER GET
```

Calling this command after loading a clip tells you if a fallback has been triggered and VML is being used instead of Matrox Clip Player, if the preferred clip player is configured as Matrox Clip Player. If the preferred clip player is configured as VML, the command always returns VML.

The preferred clip player is always used first in an attempt to load a new clip. If the load fails, then the fallback triggers. An example of a workflow if the fallback mode is configured as MATROX_WITH_FALLBACK:

- The user attempts to load a supported clip; Matrox Clip Player is used.
- The user then tries to load an unsupported clip: Matrox Clip Player is used first, Matrox Clip Player reports the clip is not supported, fallback triggers and VML Clip Player is instead used to load the clip.
- The user then tries to load a supported clip: Matrox Clip Player is used first (since it is the preferred clip player) in an attempt to load the clip.

The following warning messages appear in the console when a fallback has triggered, depending on whether the target clip player inside Matrox Clip Player is active or pending:

```
WARNING: Unable to open file with Matrox clip player (active). Attempting to open
file with VML clip player.
WARNING: Unable to open file with Matrox clip player (pending). Attempting to open
file with VML clip player.
```

The following message also appears after a VML Clip Player has been used to load an unsupported clip, and a new clip is about to be loaded in Matrox Clip Player:

Copyright © 2026 Vizrt

Warning: VML Clip Player was used to open an unsupported clip. Using Matrox Clip Player again as the preferred clip player.

The messages are there to warn the user that a clip player switch has been performed.

Warning: Do not use clip fallback in stage workflow since it is not fully supported. The console also displays a warning when the pending clip player is the one failing to load a clip. Basic stage workflow which do not include back to back playback or 3-4 point loop work fine. If an unsupported clip is needed, then the best course of action is to change the preferred clip player of the clip channel to *VML: ClipIn1.PlayerfallbackMode= 1*.

The following warning appears in the console when the pending clip player is used to attempt to load a clip but fails:

Warning: If this is part of a stage workflow (B2B, 3-4 point loop) clip fallback is not fully supported!

Fallback Mode in Viz Engine Version 5.3.2 and Earlier

Viz Engine version 5.3.2 and earlier only have two different fallback modes: MATROX and VML. Their behavior is the same as the fallback mode, MATROX_WITH_FALLBACK and VML_ONLY, respectively. The command to set and get them was:

```
MAIN*CONFIGURATION*CHANNELS*CLIPIN_0*PLAYER GET/SET
```

This command is still usable in Viz Engine 5.4 and later; however, it is not possible to set the new MATROX_ONLY mode, and **the get command returns the new values MATROX_WITH_FALLBACK, VML_ONLY, and MATROX_ONLY**.

5.18.5 Clip Player AutoFormat

From Viz Engine version 5.1, VML and Matrox Clip Player automatically decides which format to use based on the clip characteristics. The following configurations are ignored when *ClipInX.AutoFormat* is enabled:

- *clip_bpcX*
- *ClipInX.ContainsAlpha*
- *ClipInX.UseV210*

If for any reason the old behavior is needed, disable *ClipInX.AutoFormat* (for example, ClipInX.AutoFormat = 0).

Info: The stage b2b workflow does not work with Matrox Clip Player if two clips have different formats (for example, one clip has alpha and the next one does not). If this workflow is needed with Matrox Clip Player, disable the AutoFormat setting or use VML Clip Player as the default clip player.

5.18.6 VML Clip Player with NVDEC

It is possible to decode a clip using NVDEC. Depending on the NVIDIA GPU in use, there are different capabilities in terms of which codec can be decoded. This is an experimental feature and can be enabled in the configuration file:

Copyright © 2026 Vizrt

```
## Allow using NVDEC to decode clips (experimental)
#* clip_use_NVDEC: Default=0
# clip_use_NVDEC = 0
```

Copyright © 2026 Vizrt

6 Hardware Related Information

Note: As a general recommendation for all hardware types:

- Disable all power saving options in Windows or in the BIOS Settings.
- Disable Hibernate mode.
- Disable Secure boot for driver installations (optional).

The installation of video, audio or graphics cards is not required on a new system. However, for maintenance or upgrades, it is important to know how to install the different models.

All machines ordered from Vizrt are pre-installed and do not require any changes.

Information on how to replace a Matrox X.mio board and connect an audio-extension card can be found in the Matrox chapter.

IMPORTANT! Be careful when handling cards, see Handling and Installing Cards for some advice.

Copyright © 2026 Vizrt

6.1 Handling and Installing Cards

Static electricity from your body can damage your cards or your computer. Although you may not notice it, static electricity is generated every time you move. It's often too small to cause a spark, but it can still cause damage to sensitive electronic components or at least reduce their lifespan.

To avoid damage, please observe the following precautions:

- Do not remove cards from their anti-static bags until you're ready to install them. Before removing the cards, place the packages within easy reach of the area where you intend to perform the installation.
- You should avoid touching the chips and other components on the circuit boards. Try to handle the cards by their edges.
- Try to work in an area where the relative humidity is at least 50%.
- Do not wear wool or synthetic clothing. These fabrics tend to generate more static electricity than cotton, which is best for this kind of work.
- Turn off the power switches on your computer and its connected components.
- Once you've opened your computer, drain static electricity from your body by touching a bare metal surface on your computer chassis before you install or remove any parts of your system. If you have a grounding wrist strap, use it while handling and installing any components in your computer.

Copyright © 2026 Vizrt

6.2 Graphics Boards

This section contains information on the following topics:

- NVIDIA Driver Configuration
- Supported GPUs and Driver History
- Virtualized GPUs
- Working with Synchronous Output
- Working with Two or More GPUs

Copyright © 2026 Vizrt

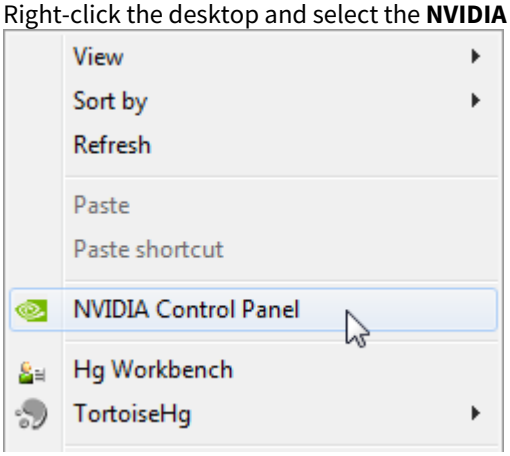
6.2.1 NVIDIA Driver Configuration

This section contains information on the following procedures:

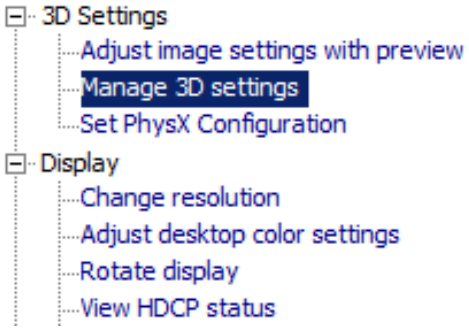
- To Configure NVIDIA Driver Settings for Viz Engine, Video Mode
- Multi-GPU Systems
- Vertical Sync
- To Configure Mosaic
- NVIDIA G-SYNC Card
    - To Configure G-SYNC on One Viz Engine
    - To Configure the G-SYNC Card on Viz Engine Slaves
    - To Check the G-SYNC Status
    - Genlock (House Sync)

To Configure NVIDIA Driver Settings for Viz Engine, Video Mode

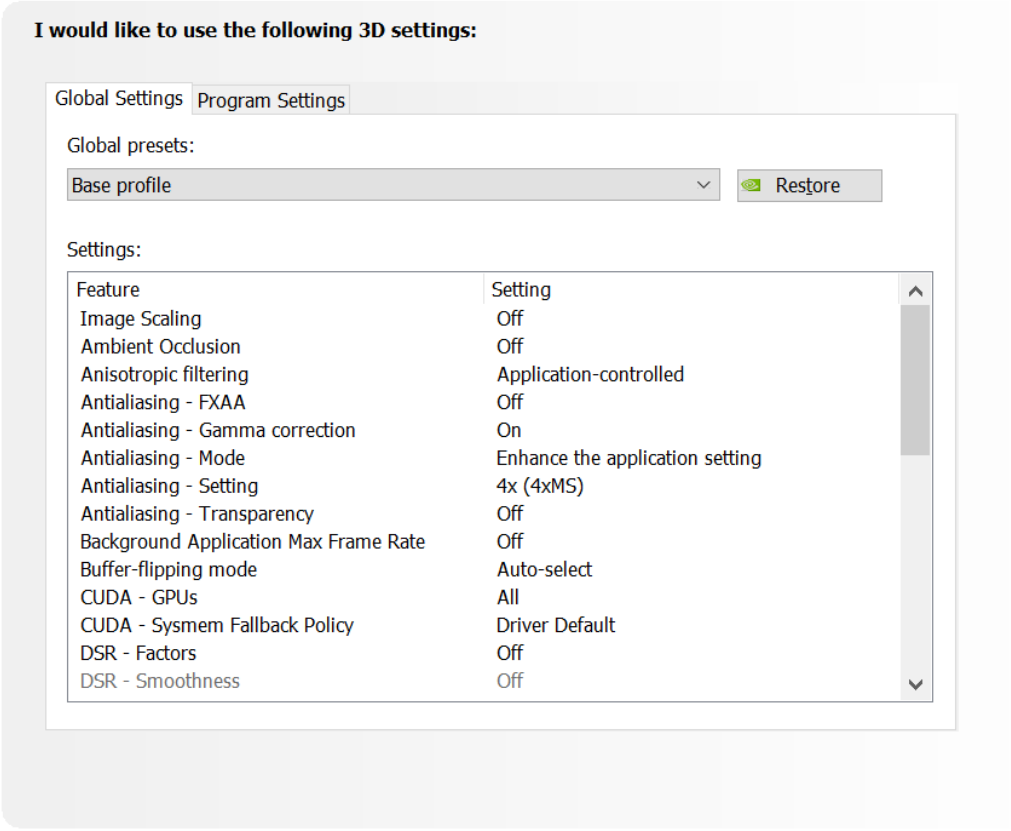
1. Right-click the desktop and select the **NVIDIA Control Panel**.



2. Click **Manage 3D settings**.



3. In **Global Settings** select the Global presets option **Base Profile**.



4. Set the following parameters:
    - **Ambient Occlusion:** `Off`.
    - **Anisotropic filtering:** `Application controlled`.
    - **Antialiasing-Gamma correction:** `Off`.
    - **Antialiasing-FXAA:** `Off`.
    - **Antialiasing-Mode:** `Enhance the application setting.` (In videowall setups, `Override any application setting` has proven to produce better results on geometry edges.)
    - **Antialiasing-Setting:** 4x (4xMS).
    - **Antialiasing-Transparency:** `Off`.
    - **CUDA-GPUs:** `All`
    - **Power management mode:** `Prefer maximum performance`.
    - **Vertical Sync:** `Off` (see **Vertical Sync** below).
5. In the **Global presets** drop-down box select **Workstation App - Dynamic Streaming**.
6. Click on **Apply**.

Copyright © 2026 Vizrt

Multi-GPU Systems

A GPU without a connected display stays in the idle power state (P8) and runs at strongly reduced clocks, even under full load. Scenes rendering on such a GPU show severe performance drops. The **Power management mode** setting does not apply to GPUs without a display.

Every GPU used by a Viz Engine or Unreal Engine instance must have a display: connect a monitor, or load an EDID file on one of the GPU's connectors (NVIDIA Control Panel → Workstation → View System Topology → Manage EDID → Load), then reboot.

Vertical Sync

Notes for Vertical Sync:

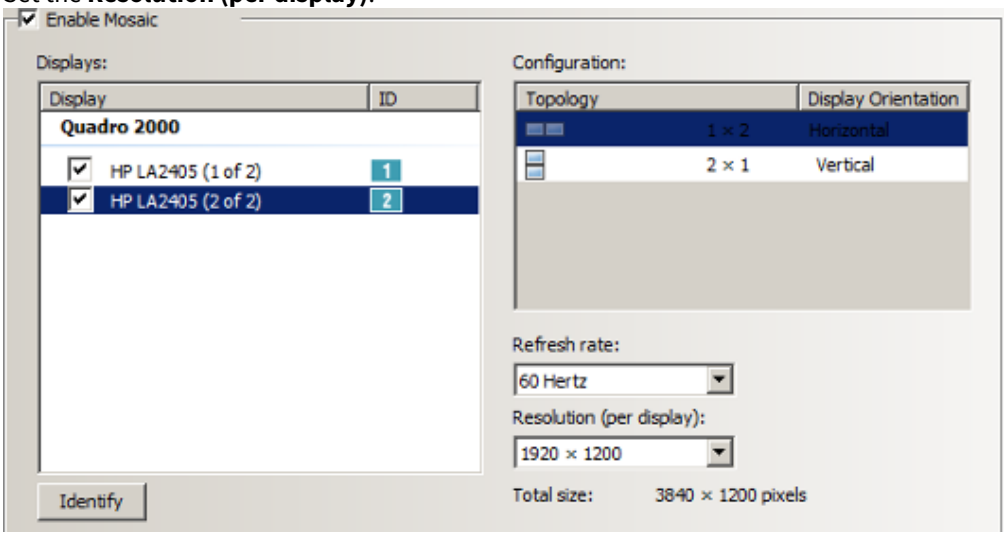
- **Use the 3D application setting:** Use for Viz Engine installations running in VGA mode (former Standard PC mode).
- **Off:** Use for Viz Engine installations where video is in use (unless used for video walls).

Videowall: For video wall installations, Vertical Sync must always be `On`.

To Configure Mosaic

If setting up a Video Wall, please refer to the Video Wall Configuration chapter.

1. Right-click the desktop and select the **NVIDIA Control Panel**.
2. Click **Set up Mosaic**.
3. Tick the **Enable Mosaic** box.
4. Tick all required displays.
5. Set the **Refresh rate**.
6. Set the **Resolution (per display)**.



7. Click **Apply**.
8. Open Viz Config.
9. Click on **User Interface**.

Copyright © 2026 Vizrt

10. Set **Screen Layout Mode** to **Mosaic Horizontal** or **Mosaic Vertical**.

NVIDIA G-SYNC Card

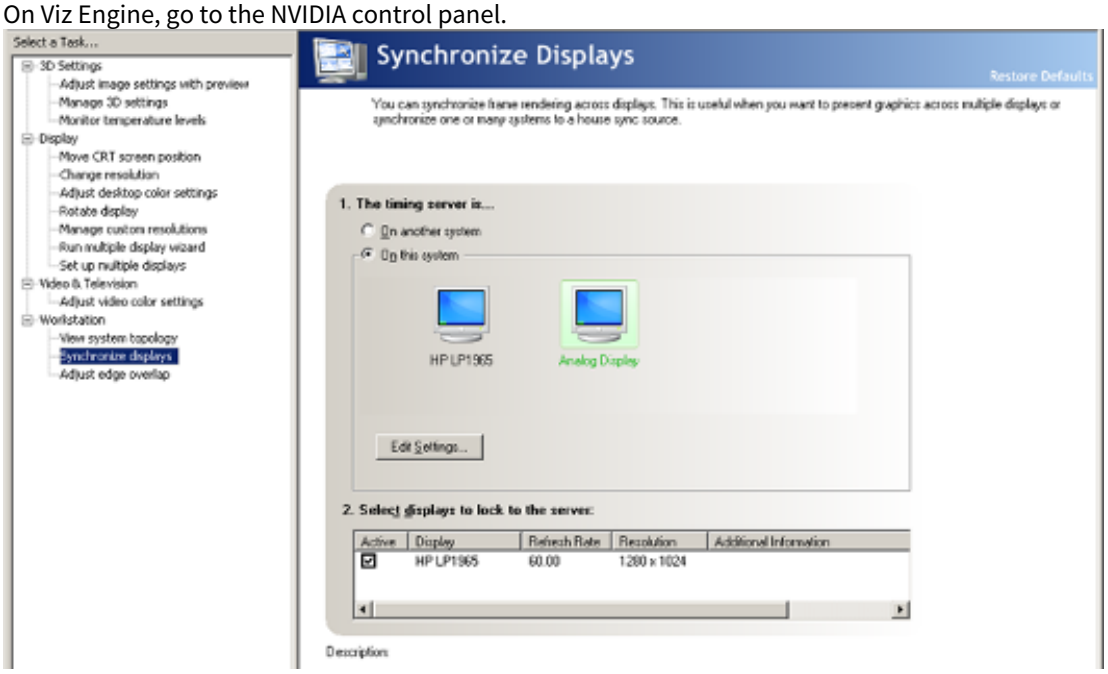
The G-SYNC card makes sure that all screens, which make up a Video Wall, are synchronized. When there are several Viz Engines in use, one is defined as the master and all the others as slaves. Pascal GPUs and higher require a G-SYNC Gen2 version.

Make sure your G-SYNC is running on the latest firmware:

Generation	Version
Quadro Sync 2	2.02
Quadro Sync	7.01

To Configure G-SYNC on One Viz Engine

1. On Viz Engine, go to the NVIDIA control panel.



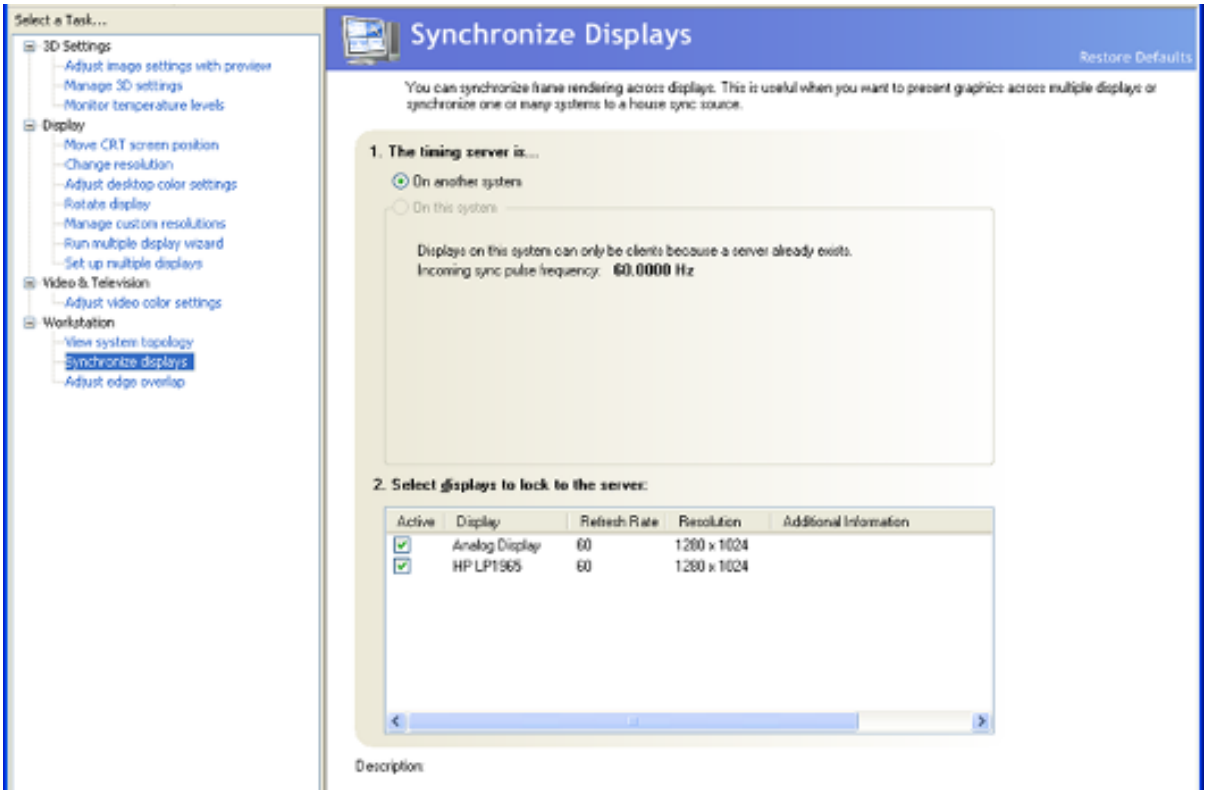
2. In **Workstation**, click **Synchronize displays**.
3. In **The timing server is...**, click **On this system**.
4. In **Select displays to lock to the server**, click each available display, in the field below, to make them active.

To Configure the G-SYNC Card on Viz Engine Slaves

Do this procedure on each Viz Engine slave.

1. On the Viz Engine slave, open the NVIDIA control panel.

Copyright © 2026 Vizrt



2. In **Workstation**, click **Synchronize displays**.
3. In **The timing server is...**, click **On another system**.
4. In **Select displays to lock to the server**, click each available display, in the field below, to make them active.

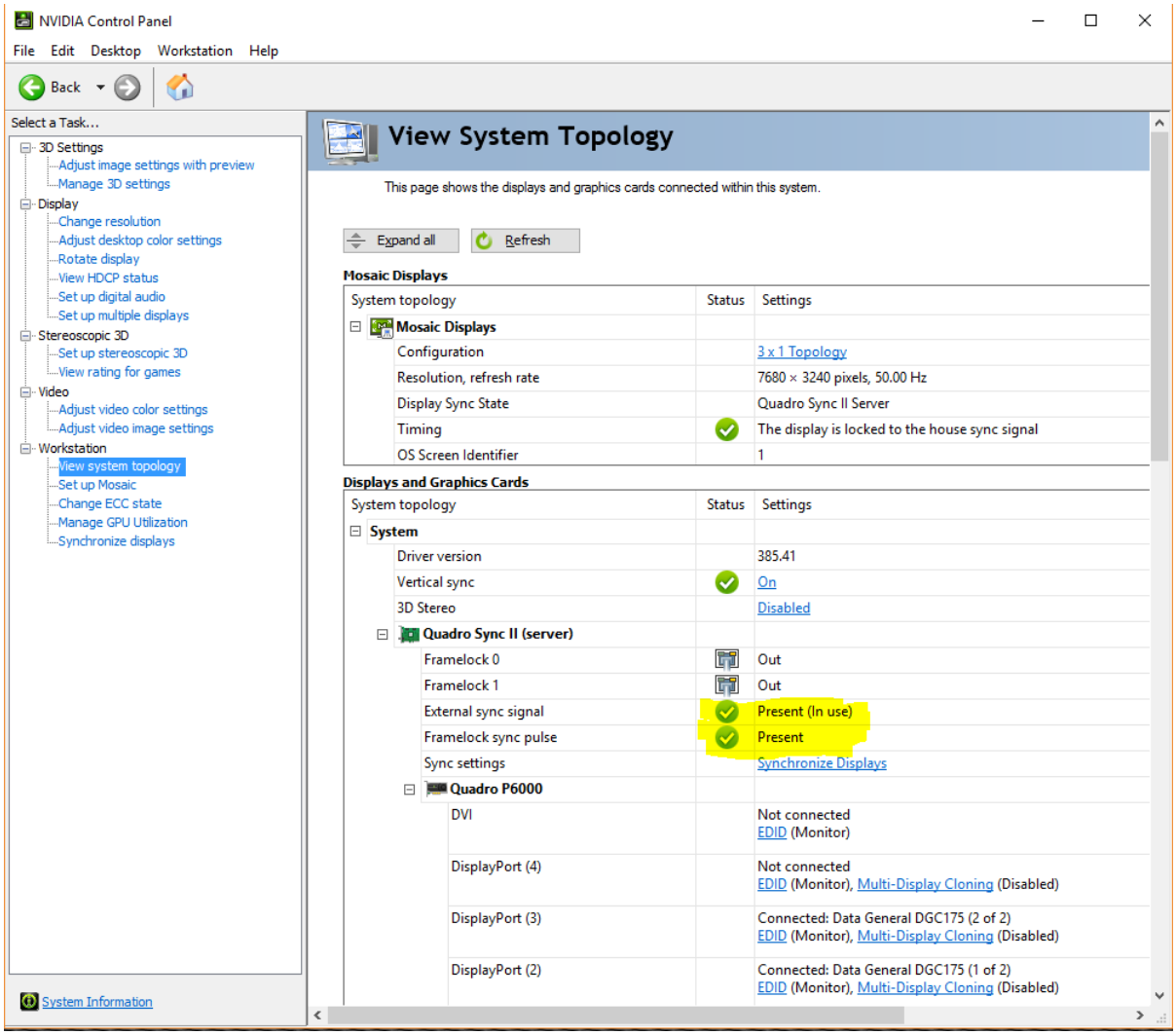
To Check the G-SYNC Status

1. On each machine, in turn, open the NVIDIA control panel.
2. In **Workstation**, click **View system topology**.
3. On the slave system, make sure that **Frame lock sync pulse is present** shows in the G-SYNC status portion on the lower part of the screen.

Genlock (House Sync)

There also the option to use a genlock (house sync) signal from the studio as an input to the G-SYNC card. In this case, once connected, check the signal in the **View system topology** screen, in **Workstation**. **House sync signal is not present** (1) means that no external genlock is connected and that the pulse is generated internally. If the external genlock is not used, this message can be ignored.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

6.2.2 Supported GPUs and Driver History

- Supported NVIDIA GPUs
- Virtualized GPUs
- Driver History

Supported NVIDIA GPUs

Blackwell GPUs	Ada Lovelace GPUs	Ampere GPUs	Turing GPUs
Blackwell 6000 Max Q Blackwell 5000	RTX 6000 Ada	RTX A6000	RTX 6000
Blackwell RTX 4000 Pro Blackwell RTX 2000 Pro	RTX 5000 Ada	RTX A5500	RTX 5000
Mobile GPUs Blackwell RTX 2000 Pro Blackwell RTX 3000 Pro Blackwell RTX 4000 Pro Blackwell RTX 5000 Pro	RTX 4500 Ada	RTX A5000	RTX 4000
	RTX 4000 Ada SFF	RTX A4500	RTX 3000
	RTX 2000 Ada	RTX A4000	T1000
		RTX A2000	
		RTX A3000 (mobile)	
		RTX A2000 (mobile)	
		RTX A1000 (mobile)	
Orange entries are recommended for rendering photo-realistic graphics on the Viz Engine Pipeline.			

Virtualized GPUs

The following GPUs have been tested in virtualized environments, the listed driver version is the one being used. The following GPUs are currently supported.

Copyright © 2026 Vizrt

Model	Driver	Platform
NVIDIA RTX PRO Server 6000 (Blackwell)	596.86	AWS (g7e instances) Proxmox 9.2.2 (12q instance)
NVIDIA RTX PRO Server 6000 (Blackwell)	596.86	AWS (g7 instances)
NVIDIA L4 (Ada)	596.86	AWS (g6 instances, gr6 instances)
NVIDIA L40S (Ada)		AWS (g6e instances)
NVIDIA A10G (Ampere)	596.86	AWS (g5 instances)
NVIDIA T4 (Turing)	596.86	AWS (g4dn instances)
NVIDIA A40	582.78	vSphere 8.0 (VGPU 19.6, 8q instance)
NVIDIA RTX6000	582.78	vSphere 8.0 (VGPU 19.6, 8q instance)

Driver History

Starting with Viz Engine 5.6 we recommending a minimum driver version only.

Viz Artist/Engine Version	Minimum recommended driver
5.6.0	596.86 for Blackwell, Ada Lovelace, Ampere and Turing GPUs
5.5.0	573.76 for Ada Lovelace GPUs 573.76 for Ampere, Turing, Pascal, Maxwell, Volta GPUs [1]
5.4.1	572.83 for Ada Lovelace GPUs 573.76 for Ampere, Turing, Pascal, Maxwell, Volta GPUs [1]
5.4.0	572.83 for Ada Lovelace GPUs 572.83 for Ampere, Turing, Pascal, Maxwell, Volta GPUs [1]
5.3.1, 5.3.2	553.24 for Ada Lovelace GPU 553.24 for Ampere, Turing, Pascal, Maxwell, Volta GPUs [1]

Copyright © 2026 Vizrt

Viz Artist/Engine Version	Minimum recommended driver
5.3.0	552.22 for Ada Lovelace GPUs 552.22 for Ampere, Turing, Pascal, Maxwell, Volta GPUs [1]
5.1.0, 5.1.1, 5.2.0, 5.2.1	528.89 for Ada Lovelace GPUs 516.59 for Ampere, Turing, Pascal, Maxwell, Volta GPUs [1]
5.0.0, 5.01	516.59 473.47 for Kepler boards
4.4.0, 4.4.1	472.12
4.3.0, 4.3.1	462.31
4.2.0	442.19
3.13.x, 3.14.x,4.0, 4.1	419.17
3.9.x, 3.10.0, 3.11.x, 3.12.x	385.41
3.8.3	368.86
3.8.2	361.91
3.8.1	353.82
3.8.0	347.25
3.6.4	341.21
[1] When used in combination with Unreal Engine 5.3 and higher, 572.83 is mandatory.	

If there is any security concerns about the CVE's you can upgrade to a newer version at your own risk.
Note: There is no driver recommendation for virtual environments as they rely on the host operating system being used.

See Also

- Video Wall Configuration

Copyright © 2026 Vizrt

6.2.3 Virtualized GPUs

Viz Engines requires a suitable GPU to operate correctly. If Viz Engine is used in a cloud environment, the correct display driver needs to be used. Cloud instances are mainly used for compute tasks, not every GPU can be used as rendering device. For example, NVIDIA Tesla GPUs are intended primarily for compute tasks, using the GPU for machine learning, AI etc. If you want to use these GPUs for other tasks, like rendering, you need to install a different driver.

There are several types of drivers available:

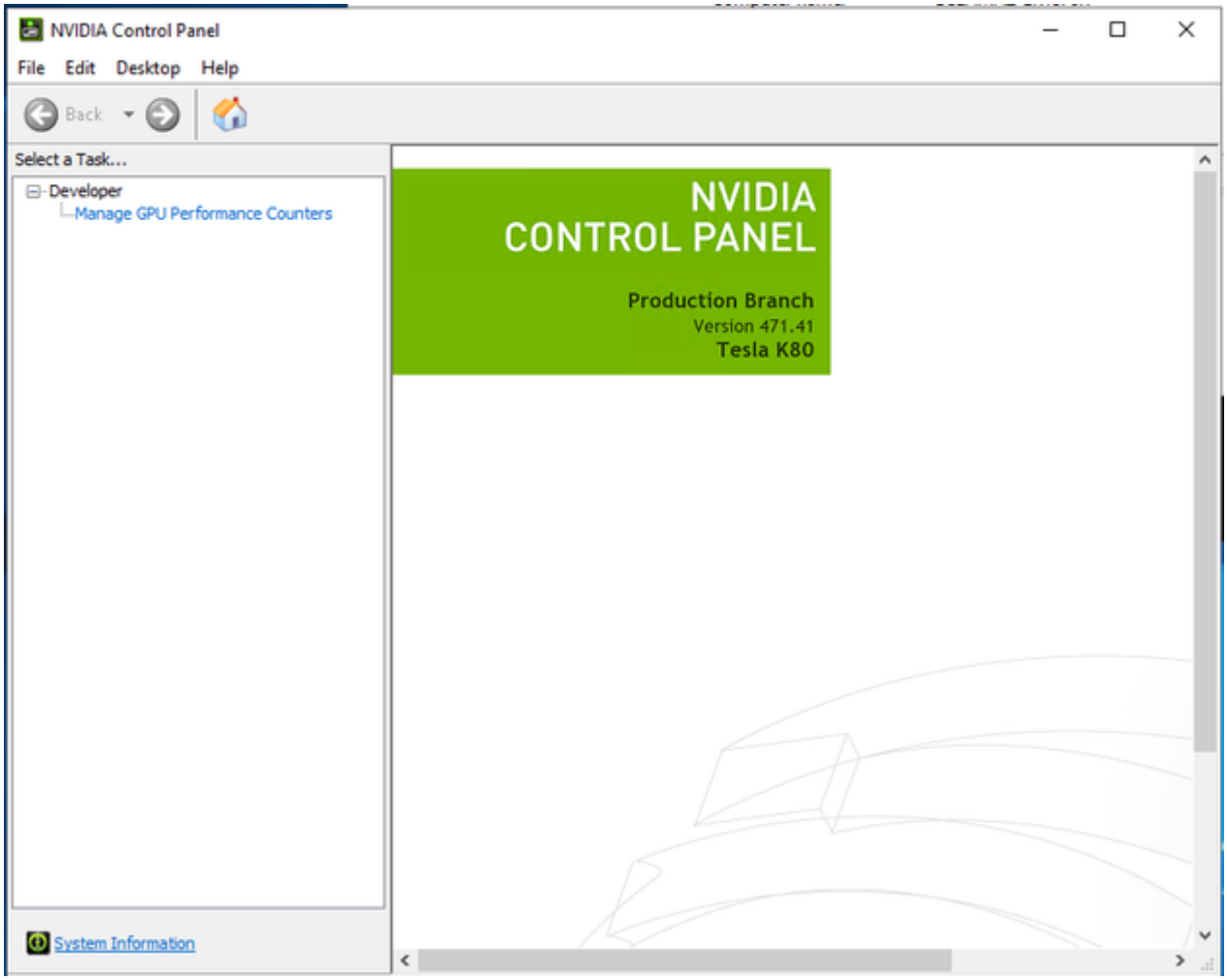
- **Tesla drivers / Datacenter drivers:** These drivers are optimized to be used for compute tasks like machine learning. They support technologies like OpenCL, CUDA and nvEnc.
- **Grid drivers** or **Gaming drivers:** Drivers for graphics tasks. Grid drivers are mainly used for professional virtual workstations whereas gaming driver are being updated more frequently and contain a lot of specify optimizations, mainly for popular game titles. Supported technologies are OpenGL, DirectX, Vulkan, CUDA and nvEnc.

A driver can run in two modes, TCC (Tesla Compute Cluster) or in WDDM mode (Windows Display Driver). **Viz Engine requires a WDDM driver**. To check in which mode your driver is running, use the *nvidia-smi* tool. This tool is located either in the NVIDIA driver directory or in the Windows System32 directory.

Call `nvidia-smi.exe -q`.

This gives you details about the driver. Check for the line **Driver Model**. If it says TCC, your driver is running in Tesla Compute Cluster mode and can not be used by Viz Engine. As an alternative, open the NVIDIA Control panel. If there is no section called **Manage 3D Settings**, your board is most likely running in TCC mode:

Copyright © 2026 Vizrt



Some drivers can be changed to operate in WDDM mode. To do so:

- Call `C:\Program Files\NVIDIA Corporation\NVSMI>nvidia-smi.exe -g 0 -dm 0`.
- Set driver model to WDDM for GPU 00000000:00:1E.0.
- Reboot your system.

This attempts to change the driver mode. Not every GPU and every driver can be forced to operate in WDDM mode.

GPU	Tesla driver ([1])	GRID driver	Gaming driver
L4	✅	✅	✅ (G6 only, not Gr6)
L40s	✅	✅	✅ (G6e)
L40	✅	✅	❌
Tesla T4	✅	✅	✅

Copyright © 2026 Vizrt

GPU	Tesla driver ([1])	GRID driver	Gaming driver
Ampere A10g	✅	✅	✅

[1] Works only in WDDM mode.

To Install the Correct Driver

Drivers can be downloaded either from NVIDIA's Driver section or you can install the drivers provided by the cloud provider. AWS offers customized drivers for GRID and gaming purposes, therefore it is recommended to use these drivers instead of the NVIDIA ones.

Prerequisites

Windows PowerShell Tools for AWS are required. Please follow these steps:

1. To install, run `Install-Module -Name AWS.Tools.Installer`.
2. Create an Access key and obtain a Secret access key.
3. Save them as profile using the `Set-AWSCredential`.
4. Use this profile as the default: `Initialize-AWSDefaultConfiguration`.

Installing a GRID Driver (AWS)

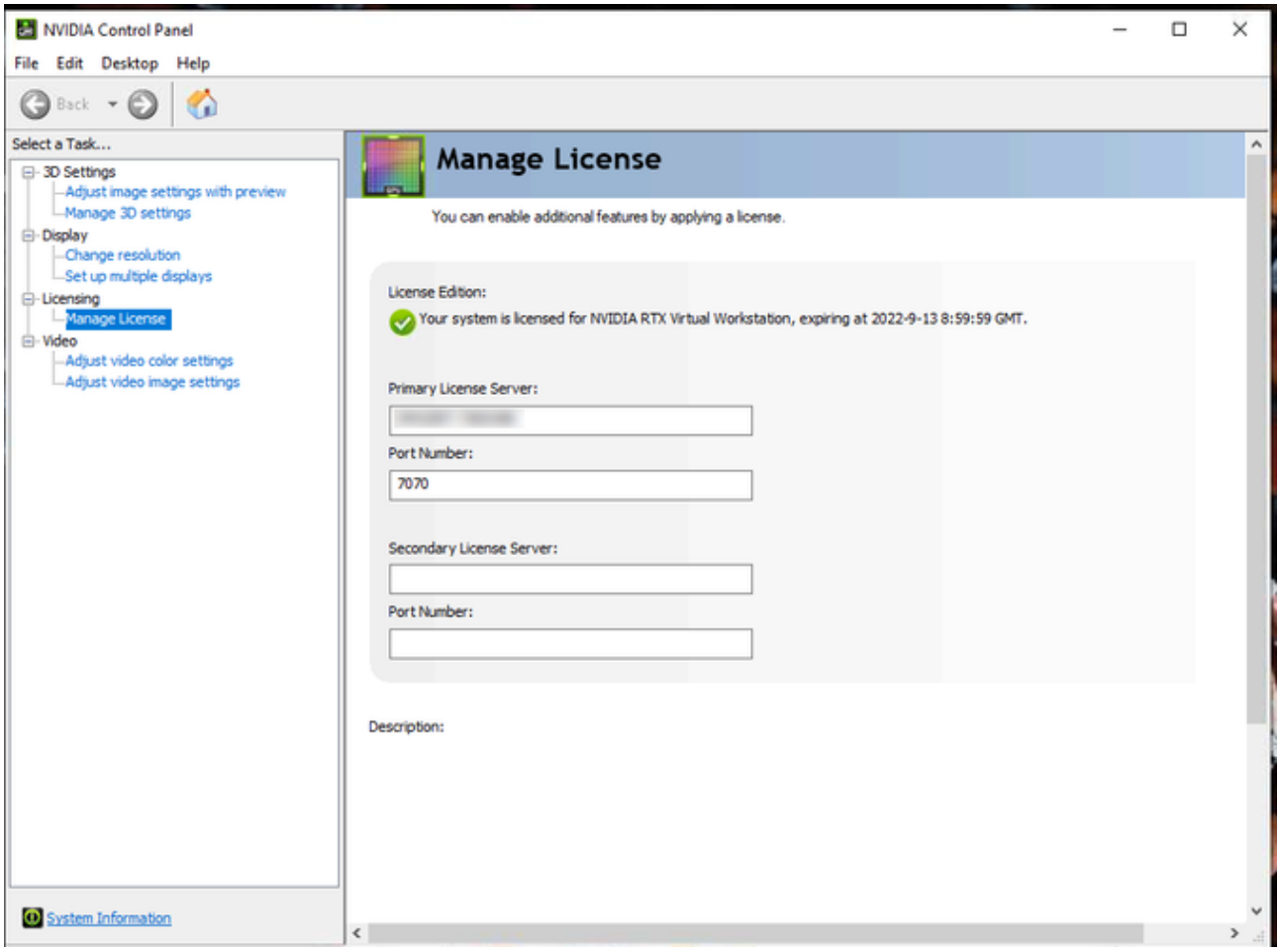
Make sure your credentials are valid and have been configured using the PowerShell tools.

Run the following command:

```
$Bucket = "ec2-windows-nvidia-drivers"
$KeyPrefix = "latest"
$LocalPath = "$home\Desktop\NVIDIA"
$c = Get-AWSCredential -ProfileName MyNewProfile
$Objects = Get-S3Object -BucketName $Bucket -KeyPrefix $KeyPrefix -Region us-east-1
 -Credential $c
foreach ($Object in $Objects) {
    $LocalFileName = $Object.Key
    if ($LocalFileName -ne '' -and $Object.Size -ne 0) {
        $LocalFilePath = Join-Path $LocalPath $LocalFileName
        Copy-S3Object -BucketName $Bucket -Key $Object.Key -LocalFile $LocalFilePath
-Region us-east-1 -Credential $c
     }
}
```

The region parameters may need to be changed to match your location. Run the setup procedure as usual. Tesla M60 and other GRID GPUs might require an NVIDIA license server. Make sure the license server is valid and reachable via DNS.

Copyright © 2026 Vizrt



To prevent users from changing the license setting, the above entry can be hidden by changing *HKLM:\SOFTWARE\NVIDIA Corporation\Global\GridLicensing\NvCplDisableManageLicensePage DWord=1* in the Windows registry.

Installing a Gaming Driver (AWS)

If your GPU has gaming drivers available, (T4, A10, V100 GPUs), they can be downloaded as described in the previous steps, however you need to change the Bucket to be *nvidia-gaming*.

To install these drivers, you need to run the following PowerShell script within your AWS installation.

```
$Bucket = "nvidia-gaming"
$KeyPrefix = "windows/latest"
$LocalPath = "$home\Desktop\NVIDIA"
$Objects = Get-S3Object -BucketName $Bucket -KeyPrefix $KeyPrefix -Region us-east-1
foreach ($Object in $Objects) {
    $LocalFileName = $Object.Key
    if ($LocalFileName -ne '' -and $Object.Size -ne 0) {
        $LocalFilePath = Join-Path $LocalPath $LocalFileName
        Copy-S3Object -BucketName $Bucket -Key $Object.Key -LocalFile $LocalFilePath
-Region us-east-1
```

Copyright © 2026 Vizrt

```
    }
}
```

Note: p3 instances require `$KeyPrefix = "grid-9.1"`.

Run the *nvidia-smi* tool to verify if your license is okay. Check for **License Status**. If the license status does not report as *licensed*, you need to run:

```
New-ItemProperty -Path "HKLM:\SOFTWARE\NVIDIA Corporation\Global" -Name
"vGamingMarketplace" -PropertyType "DWord" -Value "2"
Invoke-WebRequest -Uri "https://nvidia-gaming.s3.amazonaws.com/GridSwCert-Archive/
GridSwCertWindows_2021_10_2.cert" -OutFile "$Env:PUBLIC\Documents\GridSwCert.txt"
```

This downloads a certificate file. A restart is required.

See Also

- Windows Accelerated Computing Instances (external)

Copyright © 2026 Vizrt

6.2.4 Working with Synchronous Output

Information: This is supported on the Classic Render Pipeline only!

With a dual channel setup it is easy to create two fill/key pairs for the left and right eye/camera during a stereo production. A special version of the Video Wall Distributor synchronously distributes one command to the two instances of the Viz Engine. It includes a built-in locking mechanism that makes sure both Viz Engines stay in sync, even if one of them drops a frame. This feature can be set On or Off by a simple command sent through the distributor.

This section contains information on the following topics:

- Hardware Requirements
- Software Requirements
- Synchronous Output Configuration
- To Configure a Machine with Two Graphics Cards

Hardware Requirements

- Certified Vizrt workstation.
- Two NVIDIA Quadro (GPUs must be identical).
- Matrox video in- and output card.

Software Requirements

- Viz Artist/Viz Engine version 3.3 or later.

Synchronous Output Configuration

The Viz Engine installer includes an option for dual channel support. Choosing this option generates desktop icons to start the configuration and the Viz Engine for channel 1 as well as for channel 2.

Note: Viz Artist is only available for channel 1 as it always runs on the first GPU.

The configurations use one video input per channel and embedded audio on the channel. The table below shows the main differences in the configurations:

	Viz Engine 1	Viz Engine 2
Stereo Mode	LEFT_EYE	RIGHT_EYE
Video In A	Video1	Unused
Video In B	Unused	Video 1
Video In C+D	Unused	Unused

Copyright © 2026 Vizrt

	Viz Engine 1	Viz Engine 2
Clip A	Clip 1	Clip 1
Clip B	Clip 2	Clip 2
Video Out A	Fill	Unused
Video Out B	Unused	Fill
Video Out C	Key	Unused
Video Out D	Unused	Key
Communication Port	6700	6800

Since this configuration puts an extra load on the Matrox card, it is important to use the available resources with care:

- In the video section of the scene, switch off all unused layers.
- Let the GPU do the color conversion.

The Matrox card itself is synced through the first instance of Viz Engine.

It is important that only **one** display is active in the NVIDIA control panel. If you span the desktop across multiple GPUs, the affinity mask has no effect and both Viz Engines render on GPU.

To Configure a Machine with Two Graphics Cards

1. Install Viz Engine and create shortcuts for two channels using the `-u1` and `-u2` parameter.
2. Create two copies of the selected Viz Config file and place them in your Viz directory and rename one to *VIZENGINE-0.cfg* and the other *VIZENGINE-1.cfg.*
3. Load the correct output format using the configuration profiles and apply *dualchannel-0.cfg* for instance 1 and *dualchannel-1.cfg* for instance 2.
4. Start Viz Config for both instances (channel 1 and channel 2) and enter settings for the Viz Graphics Hub (see Database) and enable Auto Log-in.
    - Auto log-in is needed as dual channel Engines start without a UI.
5. Start Viz Engine for channel 1 and then Viz Engine for channel 2.
    - Channel 1 signals are available on Matrox OutA (fill) and OutC (key).
    - Channel 2 signals are available on Matrox OutB (fill) and Out D (key).

See Also

- Viz Command Line Options
- The Camera section of the Viz Artist User Guide
- The Configuration Interface section of the Viz Trio User Guide

Copyright © 2026 Vizrt

6.2.5 Working with Two or More GPUs

Multiple instances of Viz Engine can be spawned along multiple graphic cards. The assignment of the GPU to be used is done automatically if the `-u1` `-u2` parameters are used. If in addition the `-G1`, `-G2` parameters are provided, the GPU can be bound to the Viz Engine instance.

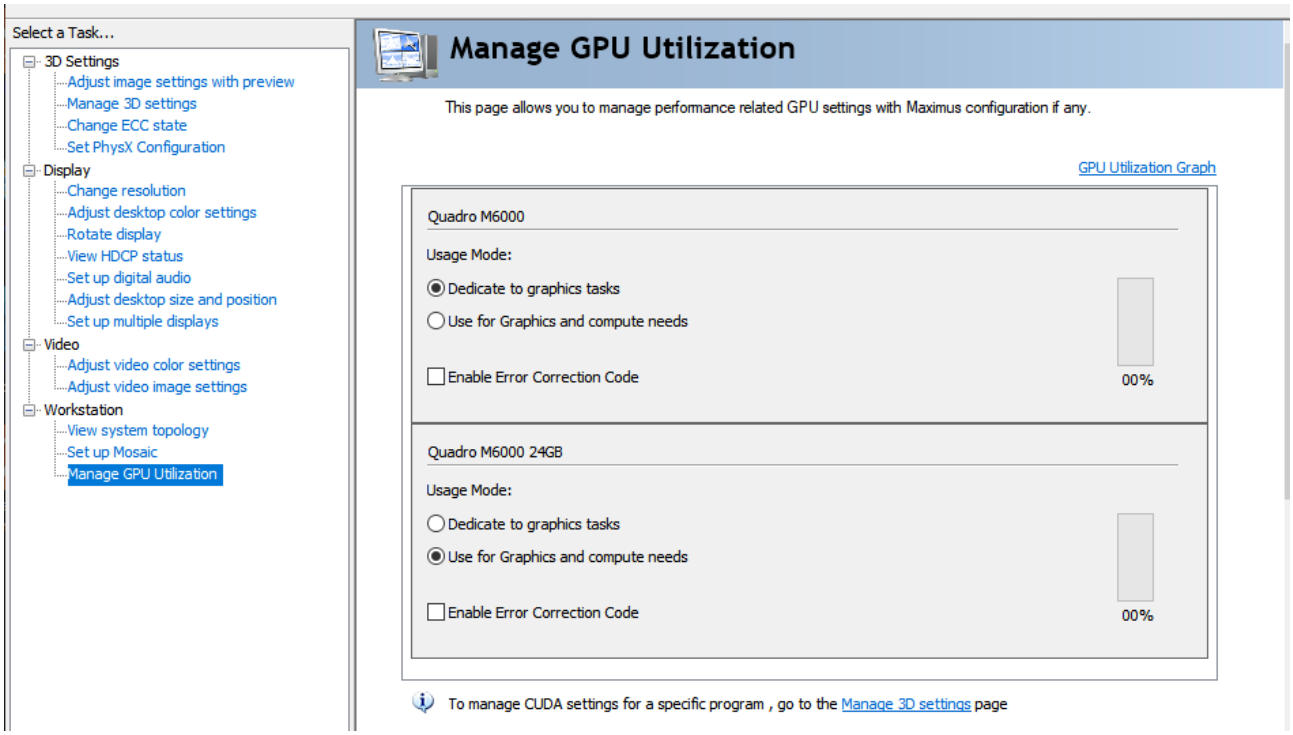
Some example use cases:

- **Two Program Outputs:** Using two instances that can run two program channels on a Dual Channel system.
- **Viz Trio Onebox Installation:** Using one instance to render a full SDI/IP Program channel on a strong GPU and a less powerful GPU to render a Viz Engine hosted by the Viz Trio client.
- **One Program Output and One Preview Output:** Running a control application with one instance for preview and one instance for program output with video. Ideal for outside broadcast environments.
- Multiple frame servers on one machine.

> **Note:** The limitations to such systems (in general and not limited to Vizrt systems alone) are memory usage, access and a lack of power redundancy. Also keep in mind that the memory usage of each machine needs to be covered by the system (for example, two P6000 with 24 GB of texture memory require a machine with at least 48 GB).

A Dual GPU setup can also be used with Viz Engine version 5.3 or later to distribute the load between third party render engines (Unreal) and Viz Engine. For more details, please see **Setting up a Multi-GPU Workflow** in the Viz Artist User Guide.

When using a Dual GPU Setup and hardware accelerated Clip Playback, you can force the NVIDIA driver to do the Clip decoding on a certain GPU by setting the GPU Utilization in the NVIDIA Control Panel.



See Also

- Working with Synchronous Output

6.3 Supported Systems

Note: As a general recommendation for all hardware types:

- Disable all power saving options in Windows or in the BIOS Settings.
- Disable Hibernate mode.
- Disable Secure boot for driver installations (optional).

The installation of video, audio or graphics cards, is not required on a new system. However, for maintenance or upgrades, it is important to know how to install the different models.

This section describes the all supported systems and their configuration supported by Viz Engine.

IMPORTANT! Any other setup than those described is not guaranteed to be supported by Vizrt and may cause problems during operation.

Note: Vizrt may make changes to specifications and product descriptions at any time, without notice.

This section contains information on the following topics:

- HP Z4 G5
- HP Z4 Rack G5
- HP Z8 G5 Fury
- Lenovo P3 Ultra / P3 Ultra Gen 2
- Lenovo P620
- Lenovo SR655 V3
- HP Z4 G6i
- HP Z8 Fury G6i

All machines ordered from Vizrt are pre-installed and do not require any changes. the following table list the current installation base:

Machine	OS	GPU
HP Z4 G5	Windows 10 LTSC IoT 21H2, Windows 11 LTSC IoT 24H2	Ampere, Ada, Blackwell 5000+6000 Single GPU 4000 Dual GPU
HP Z4 G5 Rack	Windows 10 LTSC IoT 21H2, Windows 11 LTSC IoT 24H2	Ampere, Ada, Blackwell Single 4000 only
HP Z8 G5 Fury	Windows 10 LTSC IoT 21H2, Windows 11 LTSC IoT 24H2	Ampere, Ada, Blackwell Dual GPU
Lenovo P620 End of Sale 04/2026	Windows 10 LTSC IoT 21H2, Windows 11 LTSC IoT 24H2	Ampere, Ada, Dual GPU

Copyright © 2026 Vizrt

Machine	OS	GPU
Lenovo P3 Ultra	Windows 10 LTSC IoT 21H2, Windows 11 LTSC IoT 24H2	Ada 2000 or 4000SFF
Lenovo P3 Ultra Gen2	Windows 11 LTSC IoT 24H2	Ada 2000 or 4000SFF Blackwell 2000
Lenovo SR655 V3	Windows 10 LTSC IoT 21H2, Windows 11 LTSC IoT 24H2, Windows Server 2022, Windows Server 2025	Ampere, Ada, Blackwell Dual GPU
Z4 G6i	Windows 11 LTSC IoT 24H2	Blackwell Dual GPU
Z8 Fury G6i	Windows 11 LTSC IoT 24H2	Blackwell Dual GPU

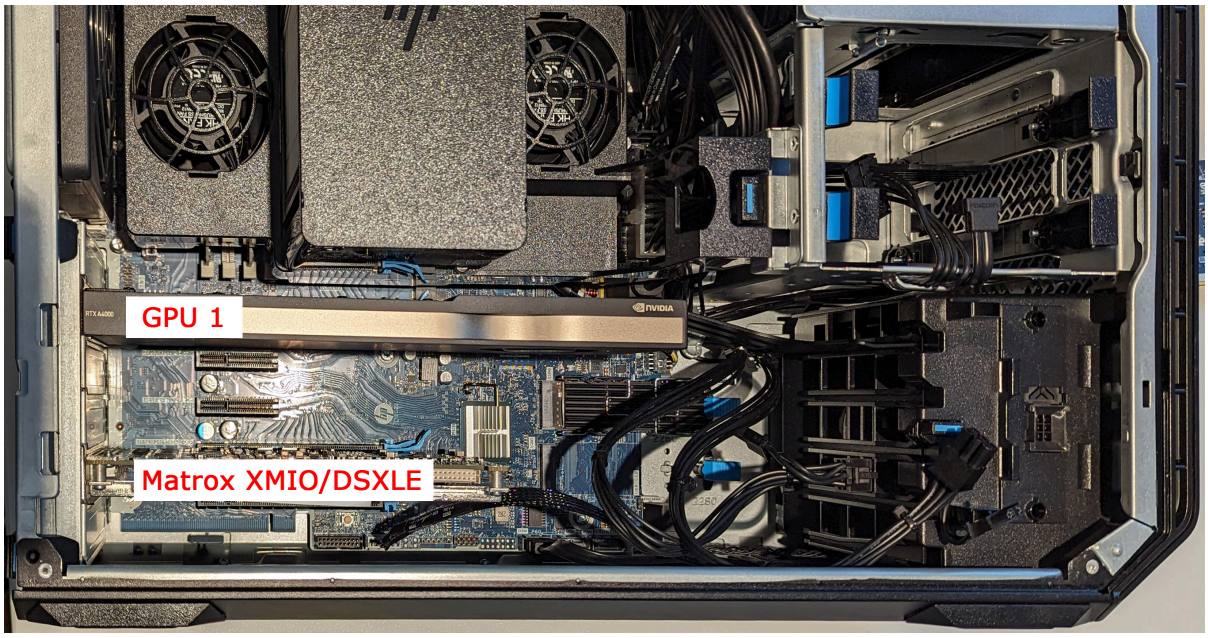
Copyright © 2026 Vizrt

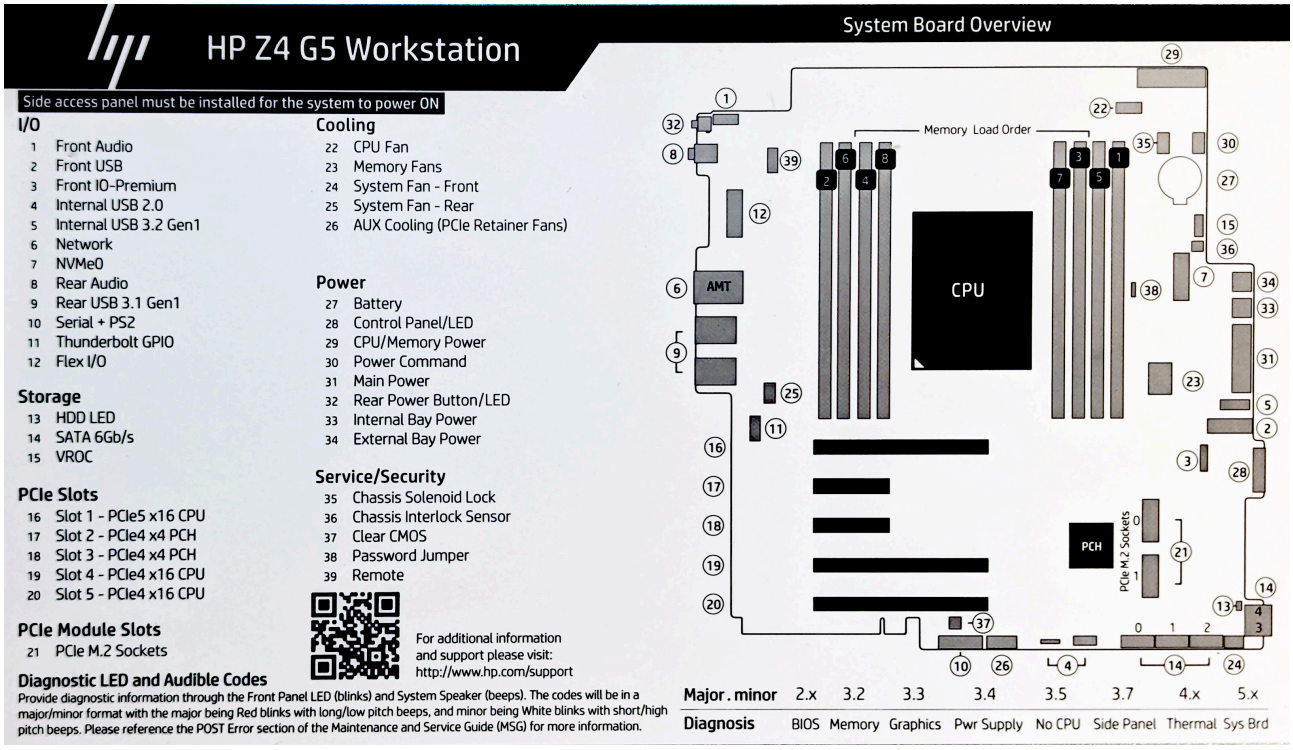
6.3.1 HP Z4 G5





Slot/Card Installation





Copyright © 2026 Vizrt

Slot Configuration

Slot	Usage
Slot 1 Gen5 x16	Graphics Card
Slot 2 Gen4 x8 (PCH)	Free or used by Dual Slot GPU
Slot 3 Gen4 x8 (PCH)	Other Expansion Card (Serial, GPI)
Slot 4 Gen4 x16	Matrox Video Card (X.mio5, DSX LE5)
Slot 5 Gen4 x16	Other Expansion Card (Serial, GPI)

Default Hardware

- Single CPU 10-Cores
- Power Supply: 1000W Platinum
- PCIe Gen4+Gen5 Slots
- 5 PCIe Slots
- 64GB DDR5 1x512GB M.2 + 1x 1TB M.2
- 1x Gbit RJ45 onboard
- 1x 10Gbit RJ45 Flex IO
- OS Support:
    - Windows 10 LTSC 2021
    - Windows 11 LTSC 2024
- GPU Support:
    - Nvidia Ampere: RTX A2000, RTX A4000, RTX A5000, RTX A6000
    - Nvidia Ada: RTX 2000 Ada, RTX 4000 Ada, RTX 5000 Ada, RTX 6000 Ada
    - Nvidia Blackwell: RTX PRO 2000 Blackwell, RTX PRO 4000 Blackwell, RTX PRO 5000 Blackwell, RTX PRO 6000 Blackwell Max-Q (300W)
- Matrox Support: XMIO5, DSXLE5

Bios Settings

- Security/Secure Boot Configuration Secure Boot: Enable
- Advanced/Built-In Device Options:
    - Increase PCIe Idle Fan Speed(%): 30
- Advanced/Power Management Options:
    - Hardware P-States: Disable
    - S4/S5 Maximum Power Savings: Disable
    - PCI Express Power Management: Disable
- Advanced/Performance Options:

Copyright © 2026 Vizrt

- Workload Configuration: I/O-Focused

Copyright © 2026 Vizrt

6.3.2 HP Z4 Rack G5





Slot/Card Configuration

Slot	Usage
Slot 1 - Gen5 x16	Graphics Card (single Slot only)
Slot 2 - Gen5 x16 or x8	Matrox Video Card (X.mio5, DSX LE5) only runs x8 speed if Slot 3 is used
Slot 3 - Gen5 x8	Other Expansion Card (Serial, GPI)

Information: Matrox AES Audio can not be installed. Larger Graphic cards will not run in Power redundant mode and are not tested by Vizrt.

Default Hardware

- Single CPU 10-Cores
- Dual redundant Power
    - Supply 675W Platinum redundant, hot-swap
- PCIe Gen5 Slots
- Three PCIe Slots
- 64GB DDR5
- 1x 512GB M.2 (Front) + 1x 1TB M.2 (internal)
- 1x Gbit RJ45 onboard
- 1x 10Gbit RJ45 Flex IO Slot
- OS Support:

Copyright © 2026 Vizrt

    - Windows 10 LTSC 2021
    - Windows 11 LTSC 2024
- OS Support:
    - Windows 10 LTSC 2021
    - Windows 11 LTSC 2024
- GPU Support:
    - Nvidia Ampere: RTX A2000, RTX A4000
    - Nvidia Ada: RTX 2000 Ada, RTX 4000 Ada
    - Nvidia Blackwell: RTX PRO 2000 Blackwell, RTX PRO 4000 Blackwell
- Matrox Support: XMIO5, DSXLE5

Bios Settings

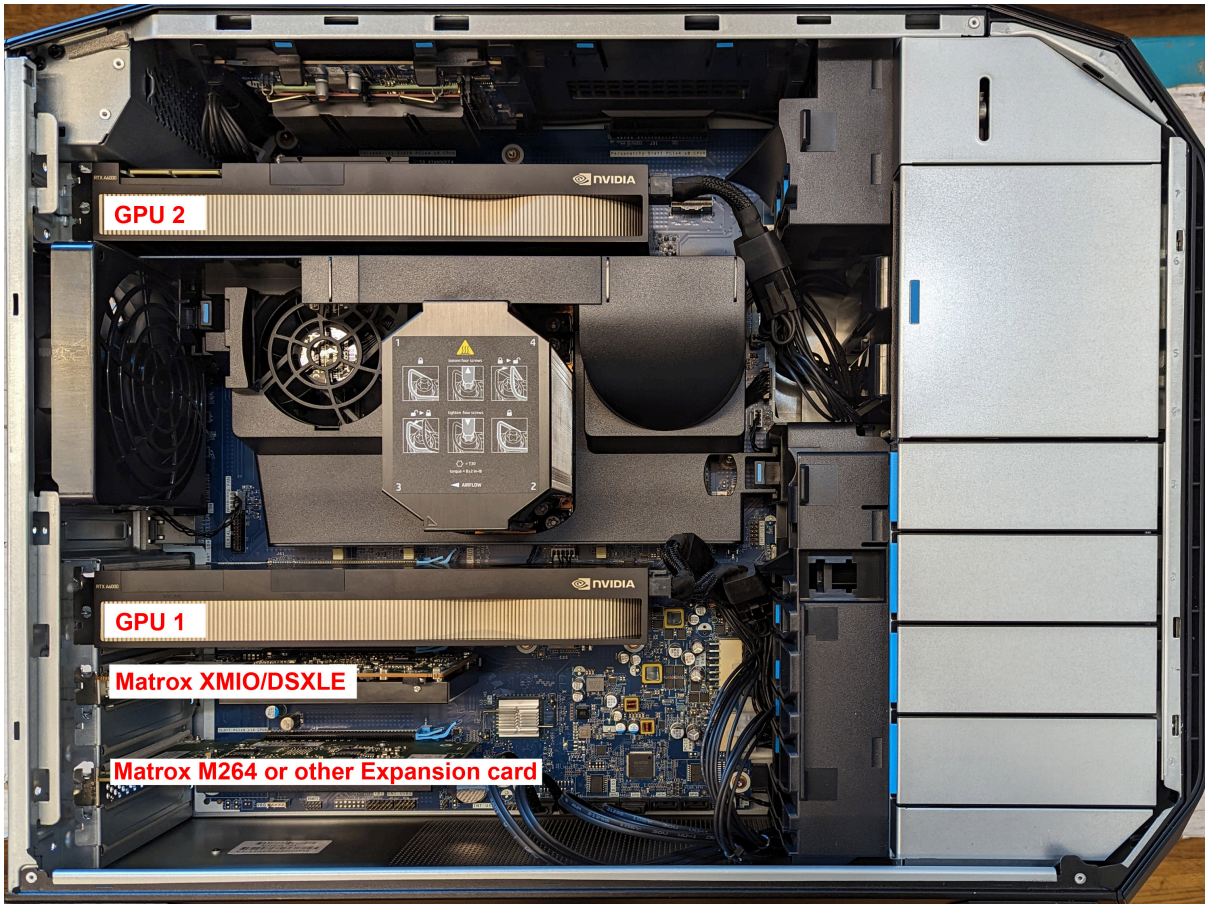
- Security/Secure Boot Configuration Secure Boot: Enable
- Advanced/Built-In Device Options:
    - Increase PCIe Idle Fan Speed(%): 30
- Advanced/Power Management Options:
    - Hardware P-States: Disable
    - S4/S5 Maximum Power Savings: Disable
    - PCI Express Power Management: Disable
- Advanced/Performance Options:
    - Workload Configuration: I/O-Focused
- Advanced/Power Supply Options:
    - Power Supply Mode: Redundant

Copyright © 2026 Vizrt

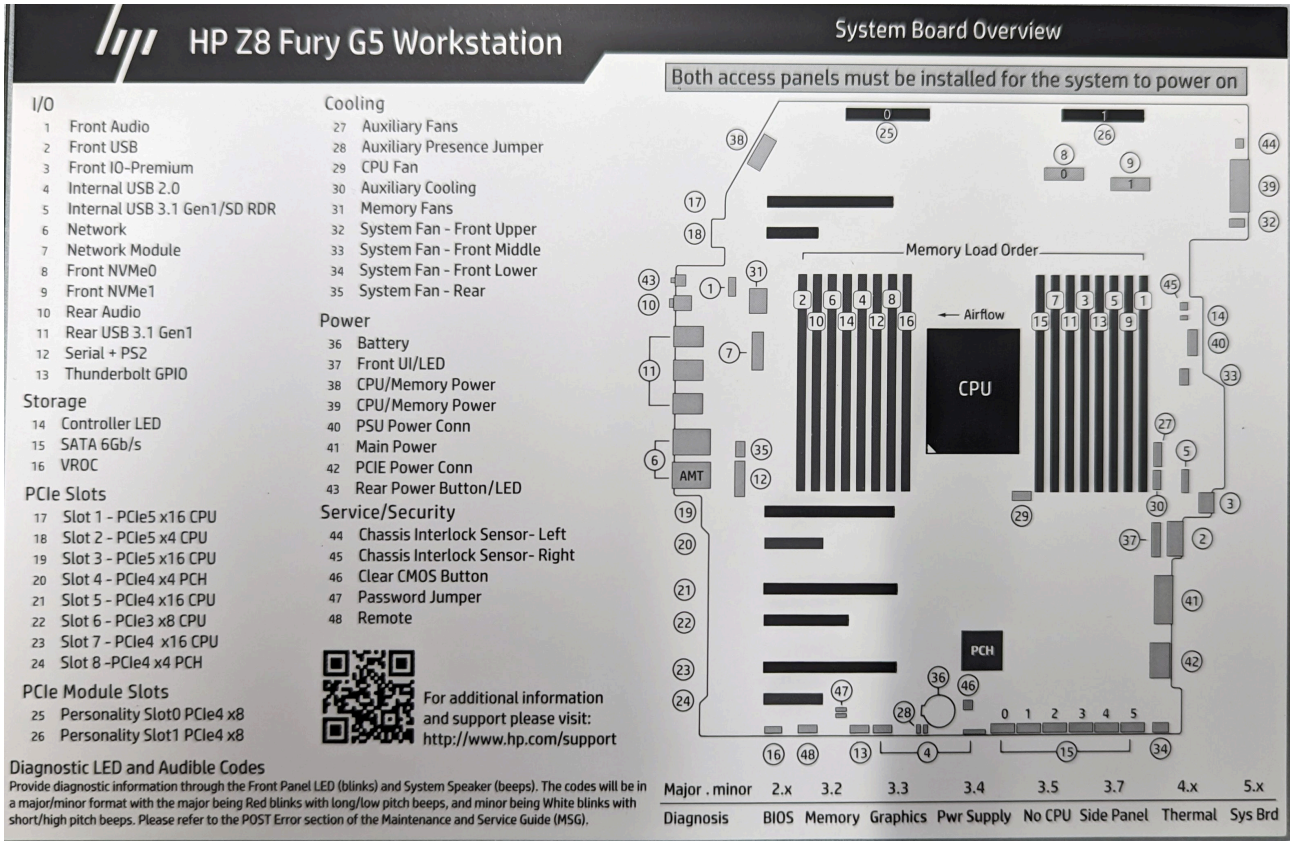
6.3.3 HP Z8 G5 Fury



Slot/Card Installation



Copyright © 2026 Vizrt



Default Hardware

- Single CPU 20-Cores
- Dual redundant Power Supply
    - 1125/1450W Platinum redundant, hot-swap
- PCIe Gen4 + Gen5 Slots
- 8 PCIe Slots
- 64GB DDR5
- 1x 512GB M.2 + 1x 1TB M.2
- 2x Gbit RJ45 onboard
- OS Support:
    - Windows 10 LTSC 2021
    - Windows 11 LTSC 2024
- GPU Support:
    - Nvidia Ampere: RTX A2000, RTX A4000, RTX A5000, RTX A6000
    - Nvidia Ada: RTX 2000 Ada, RTX 4000 Ada, RTX 5000 Ada, RTX 6000 Ada
    - Nvidia Blackwell: RTX PRO 2000 Blackwell, RTX PRO 4000 Blackwell, RTX PRO 5000 Blackwell, RTX PRO 6000 Blackwell Max-Q (300W)
- Matrox Support: XMIO5, DSXLE5, M264, DSXLE6D100

Copyright © 2026 Vizrt

PCI Configuration

Slot	Usage
Slot 1 Gen5 x16	Second Graphics Card (no Monitor)
Slot 2 Gen5 x4	Free or used by Dual Slot GPU
Slot 3 Gen5 x16	First Graphics Card (with Monitor)
Slot 4 Gen4 x4 (PCH)	Free or used by Dual Slot GPU
Slot 5 Gen4 x16	Matrox Video Card (X.mio3, X.mio5, DSX LE4)
Slot 6 Gen3 x8	Empty
Slot 7 Gen4 x16	Matrox M264 or other Expansion Card (Serial, GPI)
Slot 8 Gen4 x4 (PCH)	Other Expansion Card (Serial, GPI)

Bios Settings

- Security/Secure Boot Configuration Secure Boot: Enable
- Advanced/Built-In Device Options:
    - Increase PCIe Idle Fan Speed(%): 50
    - PCIe Idle Fan Tracking mode: Enable
- Advanced/Power Management Options:
    - Hardware P-States: Disable
    - S4/S5 Maximum Power Savings: Disable
    - PCI Express Power Management: Disable
- Advanced/Performance Options:
    - Workload Configuration: I/O-Focused
- Advanced/Power Supply Options:
    - Power Supply Mode: Redundant

Copyright © 2026 Vizrt

6.3.4 Lenovo P3 Ultra / P3 Ultra Gen 2

The Lenovo P3 Ultra is a performance workstation that is utilized in Vizrt's Go product lines. More details about the Lenovo P3 Ultra can be found on Lenovo's website: Lenovo P3 Ultra and P3 Ultra Gen2.

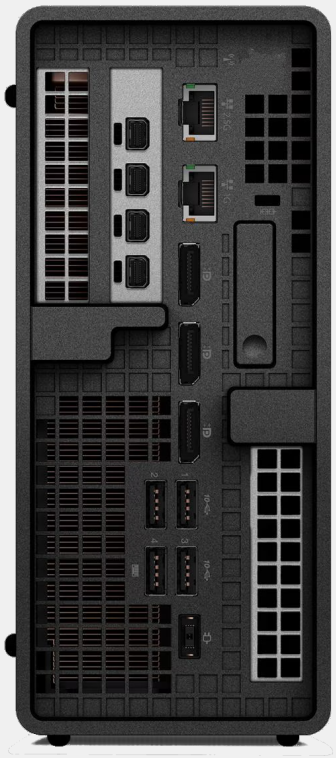


Default Hardware

- CPU: Intel i5-13600 / Core Ultra 5 235
- RAM: 32 GB (2x 16GB)
- Disk: 1TB NVMe SSD
- OS Support:
    - Windows 10 LTSC 2021 (Gen1)
    - Windows 11 LTSC 2024 (Gen1 and Gen2)
- GPU Support:
    - Gen1: Nvidia Ampere: RTX A2000
    - Gen1 and Gen2: Nvidia Ada: RTX 2000 Ada, RTX 4000 Ada SFF (75W)
    - Gen2: Nvidia Blackwell: RTX PRO 2000 Blackwell, RTX PRO 4000 Blackwell SFF (75W)

Copyright © 2026 Vizrt

Slot Configuration





Slot	Usage
Slot 1 (Top left) PCIe Gen4 x16 Low Profile Double Wide	NVIDIA RTX 4000 Ada SFF 20GB or NVIDIA RTX 2000 Ada
Slot 2 (Bottom right) PCIe Gen3 x4 Low Profile	Other Expansion Card (Serial, GPI)

Bios Settings

- Advanced - CPU Setup
  - Efficient-Cores: Disabled

Copyright © 2026 Vizrt

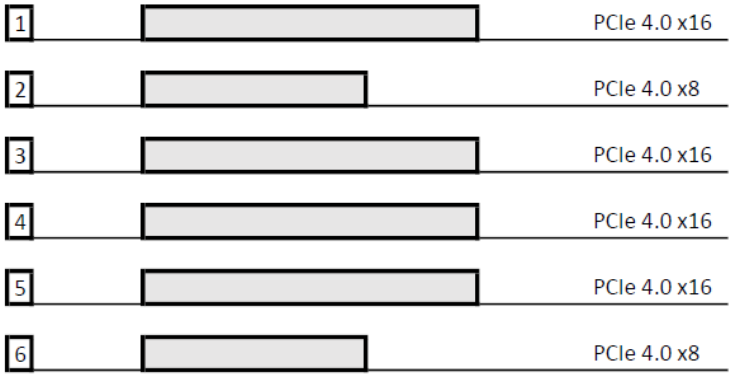
6.3.5 Lenovo P620

This section describes how to setup a Lenovo P620 machine with the different cards provided by Vizrt. The Lenovo P620 is a high performance workstation that can be configured with two CPUs, has eight memory slots, multiple storage and PCIe configuration options.

More details about the Lenovo P620 can be found on Lenovo's website: Lenovo P620.



Info: Only newer Matrox Boards like X.mio3 and DSX LE4 are shipped with P620 workstations. Older videoboards are not tested by Vizrt.



Copyright © 2026 Vizrt

PCI Configuration for Single GPU

Slot	Usage
Slot 1 – PCIe4 x16	Matrox AES Audio G-Sync III Blue Storm Sea Level Single Port Network Card
Slot 2 – PCIe4 x8	Empty
Slot 3 – PCIe4 x16	Graphics Card (Single / Dual Slot)
Slot 4 – PCIe4 x16	Used if Dual Slot GPU is installed
Slot 5 – PCIe4 x16	If one Graphic Card is installed, this slot is used for Video Cards (Matrox Video Boards)
Slot 6 – PCIe4 x8	Empty or can be used by Matrox AES Audio if Slot 1 is used

PCI Configuration for Single GPU with M264

Slot	Usage
Slot 1 – PCIe4 x16	M264
Slot 2 – PCIe4 x8	Matrox AES Audio Blue Storm Sea Level Single Port Network Card
Slot 3 – PCIe4 x16	Primary Graphics Card (Single / Dual Slot)
Slot 4 – PCIe4 x16	Primary Graphics Card (Dual Slot)
Slot 5 – PCIe4 x16	Video Card (Matrox)
Slot 6 – PCIe4 x8	Quadro Sync

Copyright © 2026 Vizrt

PCI Configuration for Dual GPU

Slot	Usage
Slot 1 – PCIe4 x16	Primary Graphics Card (Single / Dual Slot)
Slot 2 – PCIe4 x8	Primary Graphics Card (Dual Slot)
Slot 3 – PCIe4 x16	Video Card (Matrox)
Slot 4 – PCIe4 x16	Matrox AES Audio G-Sync III Blue Storm Sea Level Single Port Network Card
Slot 5 – PCIe4 x16	Secondary Graphics Card (Single / Dual Slot)
Slot 6 – PCIe4 x8	Secondary Graphics Card (Dual Slot)

Default Hardware

- Tower 4RU
- CPU: AMD Threadripper Pro 3955WX (16-Core, 3.90GHz, 8MB) or 5955WX (16-Core, 4.00GHz, 8MB)
- RAM: 64 GB DDR 4 (4x 16GB)
- SSD1: 512 GB M.2 NVMe (Operating System)
- SSD2: 512 GB M.2 NVMe (Data and Clips)
- Power Supply: 1x 1000W Power Supply
- OS Support:
  - Windows 10 LTSC 2021
  - Windows 11 LTSC 2024
- GPU Support:
  - Nvidia Ampere: RTX A2000, RTX A4000, RTX A5000, RTX A6000
  - Nvidia Ada: RTX 2000 Ada, RTX 4000 Ada, RTX 5000 Ada, RTX 6000 Ada
- Matrox Support: XMIO3, XMIO5, DSXLE4, DSXLE5, M264

Info: Certain third party vendor drivers require secure boot to be disabled to install properly.

Copyright © 2026 Vizrt

6.3.6 Lenovo SR655 V3

Slot/Card Installation

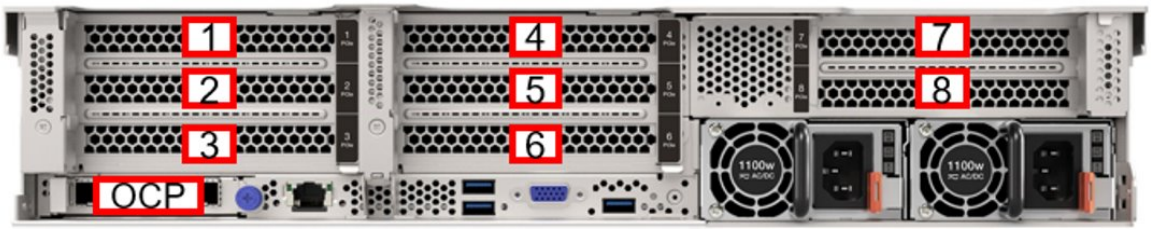


Default Hardware

- Single CPU 16-Cores AMD EPYC 9124 3GHz
- Dual Redundant Power Supplies, hot swap (220V ONLY! 1800W C13 connector)
- PCIe Gen4 + Gen5 Slots
- 6 PCIe Slots + 2 Slots for Double Wide GPU
- 64GB DDR5 (4x 16GB) up to 12x 16GB
- 2x 920GB U.2 Front Mount Hot Swap
- 4x Gbit RJ45 onboard in OCP Slot
- OS Support:
    - Windows 10 LTSC 2021
    - Windows 11 LTSC 2024
    - Windows Server 2022
    - Windows Server 2025
    - VMWare ESXi
- GPU Support:
    - Nvidia Ampere: RTX A2000, RTX A4000, RTX A5000, RTX A6000
    - Nvidia Ada: RTX 2000 Ada, RTX 4000 Ada, RTX 5000 Ada, RTX 6000 Ada
    - Nvidia Blackwell: RTX PRO 2000 Blackwell, RTX PRO 4000 Blackwell, RTX PRO 5000 Blackwell, RTX PRO 6000 Blackwell Max-Q (300W)
    - Virtualized with ESXi:
        - Nvidia A40, RTX A6000, RTX 6000 Ada
- Matrox Support: XMIO5, DSXLE5, M264

Copyright © 2026 Vizrt

Slot Configuration



Slot	Usage
Slot 1	Unused or used by Dual Slot GPU
Slot 2 Gen5 x16	First Graphics Card (with Monitor)
Slot 3 Gen4 x16	Raid Controller
Slot 4	Unused or used by Dual Slot GPU
Slot 5 Gen5 x16	Second Graphics Card (no Monitor)
Slot 6 Gen4 x16	Other Expansion Card (Serial, GPI)
Slot 7 Gen5 x16	Matrox Video Card (X.mio5), Matrox M264 or other Expansion Card (Serial, GPI)
Slot 8 Gen4 x16	Matrox Video Card (X.mio3, DSX LE4), Matrox M264 or other Expansion Card (Serial, GPI)

Bios Settings

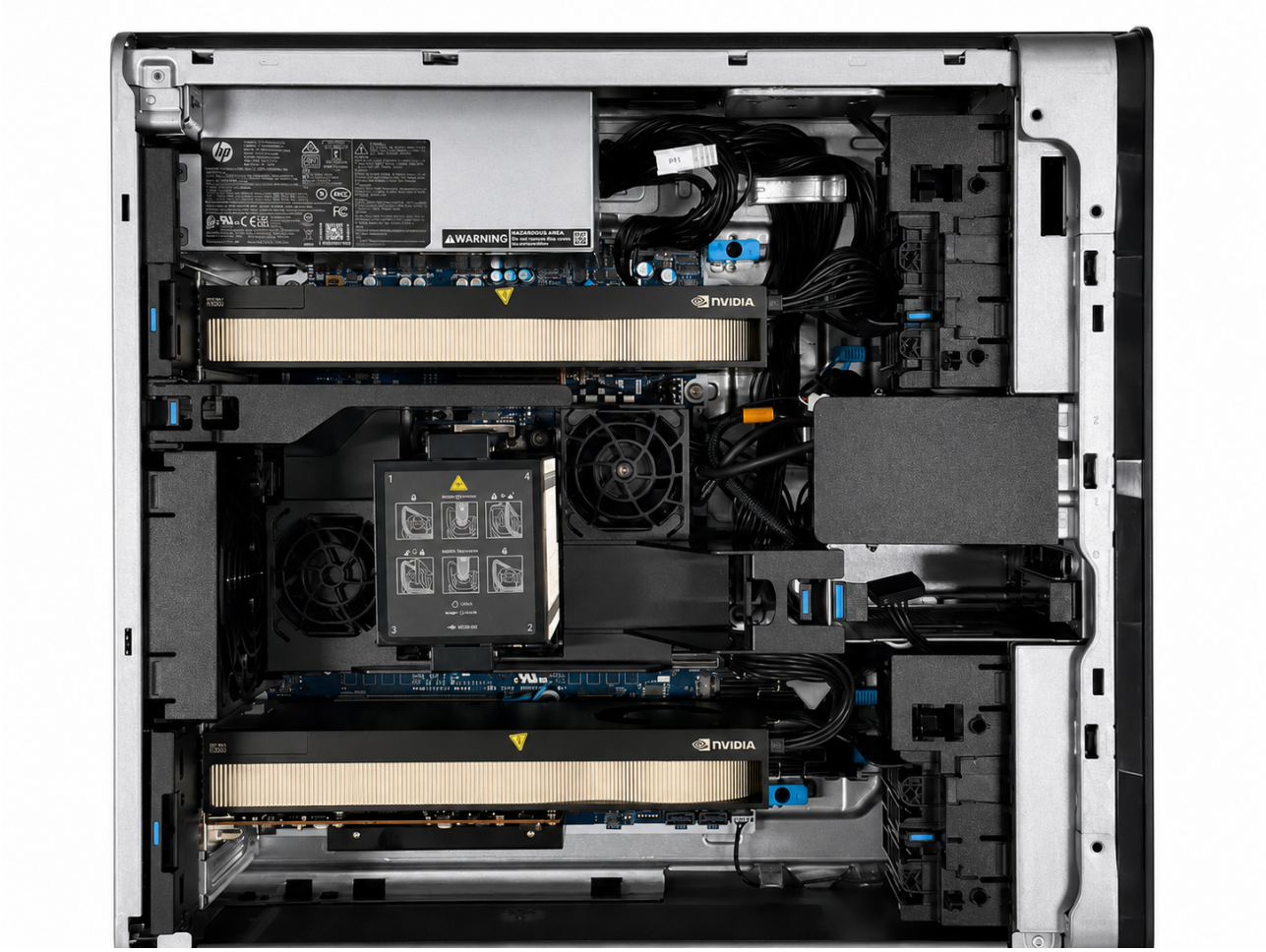
- System Settings / Devices and I/O Ports
  - Active Video: Add-in Device
- System Settings / Enable/Disable Onboard Devices
  - Onboard Video: Disabled
- System Settings / Operating Modes
  - Choose Operating Mode: Maximum Performance
- System Settings / Power
  - PCIe Power Brake: Disabled

Copyright © 2026 Vizrt

6.3.7 HP Z4 G6i



Slot/Card Installation



Slot Configuration

Slot	Usage
Slot 1 Gen5 x16	First Graphics Card
Slot 2 Gen5 x4	Free or used by Dual Slot GPU
CPU and Memory	
Slot 3 Gen5 x16	Second Graphics Card or Matrox M264

Copyright © 2026 Vizrt

Slot	Usage
Slot 4 Gen4 x4	Other Expansion Card (Serial, GPI)
Slot 5 Gen5 x16	Matrox Video Card (X.mio3, X.mio5, DSX LE4)
Slot 6 empty	

Default Hardware

- Single CPU 12-Cores
- Power Supply: 1350W @110V or 1700W @200V, Platinum Rated (92% Efficient)
- PCIe Gen4+Gen5 Slots
- 5 PCIe Slots
- 64GB DDR5 2x 1TB M.2
- 1x Gbit RJ45 onboard
- 2x 10Gbit RJ45 Flex IO
- OS Support:
    - Windows 11 LTSC 2024
- GPU Support:
    - Nvidia Blackwell: RTX PRO 2000 Blackwell, RTX PRO 4000 Blackwell, RTX PRO 5000 Blackwell, RTX PRO 6000 Blackwell Max-Q (300W)
- Matrox Support: XMIO5, DSXLE5, M264

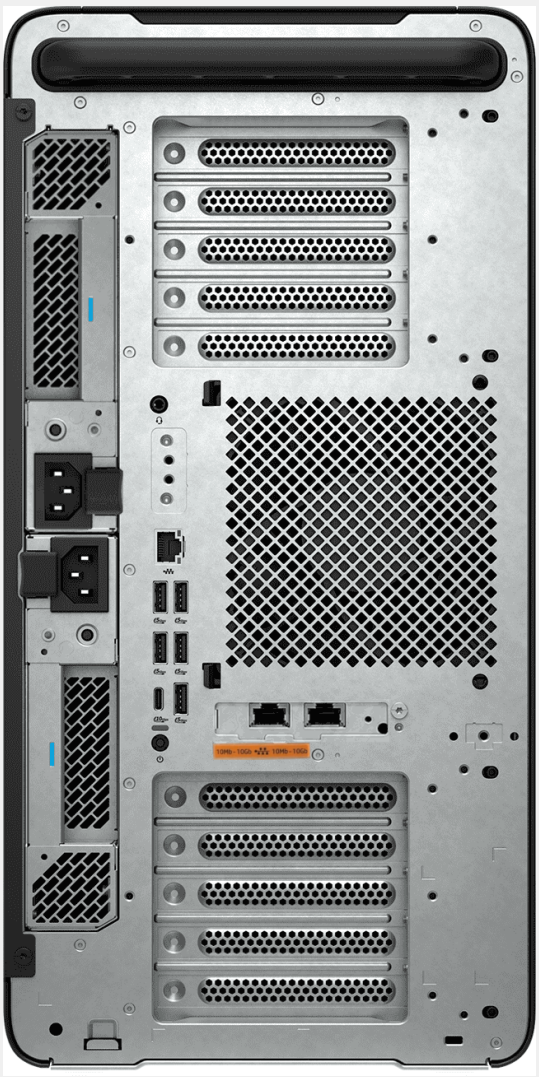
Bios Settings

- Security/Secure Boot Configuration Secure Boot: Enable
- Advanced/Built-In Device Options:
    - Increase PCIe Idle Fan Speed(%): 30
- Advanced/Power Management Options:
    - Hardware P-States: Disable
    - S4/S5 Maximum Power Savings: Disable
    - PCI Express Power Management: Disable
- Advanced/Performance Options:
    - Workload Configuration: I/O-Focused

Copyright © 2026 Vizrt

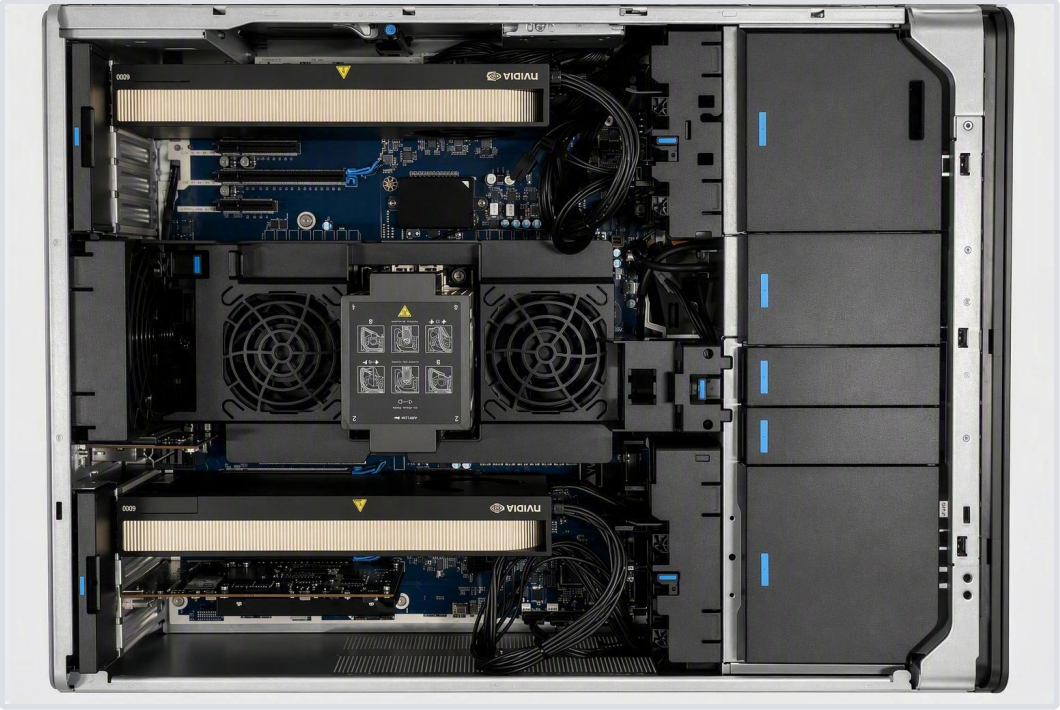
6.3.8 HP Z8 Fury G6i





Copyright © 2026 Vizrt

Slot/Card Installation



Slot Configuration

Slot	Usage
Slot 1 Gen5 x16	Second Graphics Card
Slot 2 empty	
Slot 3 Gen5 x8	Nvidia Sync Board
Slot 4 Gen5 x16	Matrox M264
Slot 5 Gen5 x4	
CPU and Memory	
Slot 6 Gen5 x16	First Graphics Card
Slot 7 Gen5 x8	

Copyright © 2026 Vizrt

Slot	Usage
Slot 8 Gen5 x8	
Slot 9 Gen5 x16	Matrox Video Card (X.mio5, DSX LE5)
Slot 10 Gen4 x4	

Default Hardware

- Single CPU 20-Cores
- Dual Redundant Power Supply: 1350W @110V or 1700W @200V, Platinum Rated (92% Efficient)
- PCIe Gen4+Gen5 Slots
- 9 PCIe Slots
- 128GB DDR5 2x 1TB M.2
- 1x Gbit RJ45 onboard
- 2x 10Gbit RJ45 Flex IO
- OS Support:
    - Windows 11 LTSC 2024
- GPU Support:
    - Nvidia Blackwell: RTX PRO 2000 Blackwell, RTX PRO 4000 Blackwell, RTX PRO 5000 Blackwell, RTX PRO 6000 Blackwell Max-Q (300W)
- Matrox Support: XMIO5, DSXLE5, M264, DSXLE6D100

Bios Settings

- Security/Secure Boot Configuration Secure Boot: Enable
- Advanced/System Options
    - USB Type-C Connector System Software Interface (UCSI): Disable
- Advanced/Built-In Device Options:
    - Increase PCIe Idle Fan Speed(%): 30
    - PCIe Idle Fan Tracking mode: Enable
- Advanced/Power Management Options:
    - Hardware P-States: Disable
    - Runtime Power Management: Disable
    - BIOS Energy/Performance Bias: Performance
    - S4/S5 Maximum Power Savings: Disable
    - PCI Express Power Management: Disable
- Advanced/Performance Options:
    - Performance Control: Ultimate Performance Mode
    - Workload Configuration: I/O-Focused
- Advanced/Power Supply Options:
    - Power Supply Mode: Redundant

Copyright © 2026 Vizrt

6.4 Video Boards

This section describes the all supported Video IO boards, their configuration and drivers supported by Viz Engine.

IMPORTANT! Any other setup than those described is not guaranteed to be supported by Vizrt and may cause problems during operation.

Note: Vizrt may make changes to specifications and product descriptions at any time, without notice.

This section contains information on the following topics:

- SMPTE ST 2110 Configuration
- Matrox Hardware
- BlueFish444
- Aja Hardware

Copyright © 2026 Vizrt

6.4.1 SMPTE ST 2110 Configuration

The configuration of the IP settings was moved to an XML file containing IP relevant parameters only. The file *ipconfig.xml* can be found in *%ProgramData%\Vizrt\VizEngine*.

Warning: When editing the XML file, do not delete any tags and only edit values between *<tag>* and *</tag>*.

Important: Never change any of the name parameters in the tags. For example, *<SFP name="SFP A">* or *<Video name="IP video OUT 1">*.

This section contains information on the following topics:

- How to Setup X.mio5 Separate Flows
- NMOS Configuration

General Settings

Setting	Description
AutoRecovery	When set to `true`, every change for parameters (either set by direct command or NMOS requests) is saved to the config file. When Viz Engine is restarted, those settings are used to configure the IP connectors.
Is07Receivers	Currently not in use.

SFP Settings

Setting	Description
IPv4Address	X.mio3: the static unicast IP v4 address used as source address and for the PTP connection. X.mio5: the static IP v4 address used for the PTP connection, if DHCP is disabled. Can be 0.0.0.0 otherwise.
IPV4Gateway	X.mio3: the static IP v4 gateway address used as source address and for the PTP connection. X.mio5: the static IP v4 gateway address used for the PTP connection, if DHCP is disabled. Can be 0.0.0.0 otherwise.
IPv4Netmask	X.mio3: the network subnet mask used for the source address and the PTP connection. X.mio5: the network subnet mask used for the PTP connection, if DHCP is disabled. Can be 0.0.0.0 otherwise.

Copyright © 2026 Vizrt

Setting	Description
Smpte2059SfpSettings::DhcpEnable	X.mio5 only: enables DHCP support for the SFP unicast PTP settings. X.mio3 and DSX LE 4 don't support DHCP and need static configuration using the fields described above.
Smpte2059SfpSettings::TypeOfServiceDSCP	Indicates that the Type of Service (ToS) is Differentiated Service Code Point (DSCP). Range is [0..63] (6 bits).
Smpte2059SfpSettings::DelayMechanism	Indicates the type of delay mechanism to use for the time server connection. 0 = Invalid 1 = EndToEnd 2 = PeerToPeer
Smpte2059SfpSettings::IpMode	Indicates the type of internet protocol mode to use for the time server connection. 0 = Invalid 1 = Multicast 2 = Unicast 3 = Hybrid
Smpte2059SfpSettings::MasterClockDomainNumber	Indicates the time server clock domain number to use.
Smpte2059SfpSettings::AnnounceReceiptTimeout	Number of 'Announce intervals' to wait: Range is [2..10] with a default of 6.
Smpte2059SfpSettings::JoinType	Indicates the type of membership request made. 0 = Invalid 1 = None 2 = IGMPv2 3 = IGMPv3
Smpte2059SfpSettings::IgmpV3Settings::FilterType	Only needed when the join type is set to 3 = IGMPv3. Indicates the filter type applied. 0 = Invalid 1 = Inclusion 2 = Exclusion
Smpte2059SfpSettings::IgmpV3Settings::FilterListCount	Indicates the number of IPv4 source addresses in the filter list. Maximum is four.
Smpte2059SfpSettings::IgmpV3Settings::FilterList	Indicates the list of IPv4 source addresses to filter separated by commas.

Copyright © 2026 Vizrt

Important: The IPv4 settings for the source port on X.mio5 must be set in the Windows 10 network settings. If DHCP is not available on the essence network, static addresses must be used. **Every SFP of an X.mio5 needs two unicast IPv4 addresses.** One is for the PTP settings set in *ipconfig.xml*, and the other is set in the Windows 10 network settings, used as source address (for example, the address used in an IGMPv3 environment for source specific multi-casting).

General Output Settings

Setting	Description
Redundancy	If set to `true`, enables the redundant stream for seamless reconstruction conforming to SMPTE ST 2022-7.

Video Output Settings

Setting	Description
EnableFlow	Set to `true` to enable the flow at startup. This can be controlled using a command during run-time.
RtpPayloadId	Indicates the RTP (Real-time Transfer Protocol) payload ID.
TimeToLive	Indicates the time in which packets can be used in seconds. That is, it indicates the packets' Time to live (TTL).
TypeOfServiceDSCP	Indicates that the Type of Service (ToS) is Differentiated Service Code Point (DSCP). Range is [0..63] (6 bits).
TypeOfServiceECN	Indicates that the Type of Service (ToS) is Explicit Congestion Notification (ECN). Range is [0..3] (2 bits).
GroupHint	Unique name used as identifier for a group of flows belonging together.
Primary::SrcUdpPort	Indicates the User Datagram Protocol (UDP) port of the sender (that is, transmitter).
Primary::DstAddress	Indicates the IPv4 address of the destination (that is, receiver). Only used in multicast.
Primary::DstUdpPort	Indicates the UDP port of the destination (that is, receiver).

Copyright © 2026 Vizrt

Setting	Description
Secondary::SrcUdpPort	Indicates the redundant stream UDP port of the sender (that is, transmitter).
Secondary::DstAddress	Indicates the redundant stream IPv4 address of the destination (that is, receiver). Only used in multicast.
Secondary::DstUdpPort	Indicates the redundant stream UDP port of the destination (that is, receiver).

Audio Output Settings

Setting	Description
EnableFlow	Set to `true`, to enable the flow at startup. This can be controlled using a command during run-time.
RtpPayloadId	Indicates the RTP (Real-time Transfer Protocol) payload ID.
TimeToLive	Indicates the time in which packets can be used in seconds. That is, it indicates the packets' Time to live (TTL).
TypeOfServiceDSCP	Indicates that the Type of Service (ToS) is Differentiated Service Code Point (DSCP). Range is [0..63] (6 bits).
TypeOfServiceECN	Indicates that the Type of Service (ToS) is Explicit Congestion Notification (ECN). Range is [0..3] (2 bits).
GroupHint	Unique name used as identifier for a group of flows belonging together.
AudioPacketDuration	Indicates the outgoing audio packet duration. 1 = 125 µs 4 = 1 ms All other values are not supported.
TrackCount	Indicates the number of tracks in that flow.
Primary::SrcUdpPort	Indicates the User Datagram Protocol (UDP) port of the sender (that is, transmitter).

Copyright © 2026 Vizrt

Setting	Description
Primary::DstAddress	Indicates the IPv4 address of the destination (that is, receiver). Only used in multicast.
Primary::DstUdpPort	Indicates the UDP port of the destination (that is, receiver).
Secondary::SrcUdpPort	Indicates the redundant stream UDP port of the sender (that is, transmitter).
Secondary::DstAddress	Indicates the redundant stream IPv4 address of the destination (that is, receiver). Only used in multicast.
Secondary::DstUdpPort	Indicates the redundant stream UDP port of the destination (that is, receiver).

Ancillary Data Output Settings

Setting	Description
EnableFlow	Set to `true`, to enable the flow at startup. This can be controlled using a command during run-time.
RtpPayloadId	Indicates the RTP (Real-time Transfer Protocol) payload ID.
TimeToLive	Indicates the time in which packets can be used in seconds. That is, it indicates the packets' Time to live (TTL).
TypeOfServiceDSCP	Indicates that the Type of Service (ToS) is Differentiated Service Code Point (DSCP). Range is [0..63] (6 bits).
TypeOfServiceECN	Indicates that the Type of Service (ToS) is Explicit Congestion Notification (ECN). Range is [0..3] (2 bits).
GroupHint	Unique name used as identifier for a group of flows belonging together.
SMPTE352Payload	If set to 1, enables SMPTE 352 packets.
Primary::SrcUdpPort	Indicates the User Datagram Protocol (UDP) port of the sender (that is, transmitter).

Copyright © 2026 Vizrt

Setting	Description
Primary::DstAddress	Indicates the IPv4 address of the destination (that is, receiver). Only used in multicast.
Primary::DstUdpPort	Indicates the UDP port of the destination (that is, receiver).
Secondary::SrcUdpPort	Indicates the redundant stream UDP port of the sender (that is, transmitter).
Secondary::DstAddress	Indicates the redundant stream IPv4 address of the destination (that is, receiver). Only used in multicast.
Secondary::DstUdpPort	Indicates the redundant stream UDP port of the destination (that is, receiver).

General Input Settings

Setting	Description
Redundancy	If set to `true`, enables the redundant stream for seamless reconstruction conforming to SMPTE ST 2022-7.
JoinType	Indicates the type of membership request made. 0 = Invalid 1 = None 2 = IGMPv2 3 = IGMPv3

Video Input Settings

Setting	Description
EnableFlow	Set to `true`, to enable the flow at startup. This can be controlled using a command during run-time.
RtpPayloadId	Indicates the RTP (Real-time Transfer Protocol) payload ID.
GroupHint	Unique name used as identifier for a group of flows belonging together.
Primary::DstAddress	Indicates the reception multicast IPv4 address.

Copyright © 2026 Vizrt

Setting	Description
Primary::DstUdpPort	Indicates the reception User Datagram Protocol (UDP) port.
Primary::PacketIntervalThreshold	Indicates the threshold for generating the time interval between the IP packets alarm on the video flow. The range is from 6.4 ns to 419424.0 ns in intervals of 6.4 ns.
Primary::IgmpV3Settings::FilterType	Only needed when the join type is set to 3 = IGMPv3. Indicates the filter type applied. 0 = Invalid 1 = Inclusion 2 = Exclusion
Primary::IgmpV3Settings::FilterListCount	Indicates the number of IPv4 source addresses in the filter list. Maximum is four.
Primary::IgmpV3Settings::FilterList	Indicates the list of IPv4 source addresses to filter separated by commas.
Secondary::DstAddress	Indicates the redundant stream reception multicast IPv4 address.
Secondary::DstUdpPort	Indicates the redundant stream reception UDP port.
Secondary::PacketIntervalThreshold	Indicates the threshold for generating the time interval between the IP packets alarm on the video flow. The range is from 6.4 ns to 419424.0 ns in intervals of 6.4 ns.
Secondary::IgmpV3Settings::FilterType	Only needed when the join type is set to 3 = IGMPv3. Indicates the filter type applied. 0 = Invalid 1 = Inclusion 2 = Exclusion
Secondary::IgmpV3Settings::FilterListCount	Indicates the number of IPv4 source addresses in the filter list. Maximum is four.
Secondary::IgmpV3Settings::FilterList	Indicates the list of IPv4 source addresses to filter separated by commas.

Copyright © 2026 Vizrt

Audio Input Settings

Setting	Description
EnableFlow	Set to `true`, to enable the flow at startup. This can be controlled using a command during run-time.
RtpPayloadId	Indicates the RTP (Real-time Transfer Protocol) payload ID.
GroupHint	Unique name used as identifier for a group of flows belonging together.
AudioPacketDuration	Indicates the outgoing audio packet duration. 1 = 125 µs 4 = 1 ms All other values are not supported.
TrackCount	Indicates the number of tracks allowed in that flow.
Primary::DstAddress	Indicates the reception multicast IPv4 address.
Primary::DstUdpPort	Indicates the reception User Datagram Protocol (UDP) port.
Primary::PacketIntervalThreshold	Indicates the threshold for generating the time interval between the IP packets alarm on the audio flow. The range is from 102.4 ns to 6710681.6 ns in intervals of 102.4 ns.
Primary::IgmpV3Settings::FilterType	Only needed when the join type is set to 3 = IGMPv3. Indicates the filter type applied. 0 = Invalid 1 = Inclusion 2 = Exclusion
Primary::IgmpV3Settings::FilterListCount	Indicates the number of IPv4 source addresses in the filter list. Maximum is four.
Primary::IgmpV3Settings::FilterList	Indicates the list of IPv4 source addresses to filter separated by commas.
Secondary::DstAddress	Indicates the redundant stream reception multicast IPv4 address.
Secondary::DstUdpPort	Indicates the redundant stream reception UDP port.

Copyright © 2026 Vizrt

Setting	Description
Secondary::PacketIntervalThreshold	Indicates the threshold for generating the time interval between the IP packets alarm on the audio flow. The range is from 102.4 ns to 6710681.6 ns in intervals of 102.4 ns.
Secondary::IgmpV3Settings::FilterType	Only needed when the join type is set to 3 = IGMPv3. Indicates the filter type applied. 0 = Invalid 1 = Inclusion 2 = Exclusion

Ancillary Data Input Settings

Setting	Description
EnableFlow	Set to `true`, to enable the flow at startup. This can be controlled using a command during run-time.
RtpPayloadId	For SMPTE ST 2110 only: Indicates the RTP (Real-time Transfer Protocol) payload ID.
GroupHint	Unique name used as identifier for a group of flows belonging together.
Primary::DstAddress	Indicates the reception multicast IPv4 address.
Primary::DstUdpPort	Indicates the reception User Datagram Protocol (UDP) port.
Primary::PacketIntervalThreshold	Indicates the threshold for generating the time interval between the IP packets alarm on the ancillary data flow. The range is from 819.2 ns to 53685452.8 ns in intervals of 819.2 ns.
Primary::IgmpV3Settings::FilterType	Only needed when the join type is set to 3 = IGMPv3. Indicates the filter type applied. 0 = Invalid 1 = Inclusion 2 = Exclusion
Primary::IgmpV3Settings::FilterListCount	Indicates the number of IPv4 source addresses in the filter list. Maximum is four.

Copyright © 2026 Vizrt

Setting	Description
Primary::IgmpV3Settings ::FilterList	Indicates the list of IPv4 source addresses to filter separated by commas.
Secondary::DstAddress	Indicates the redundant stream reception multicast IPv4 address.
Secondary::DstUdpPort	Indicates the redundant stream reception UDP port.
Secondary::PacketIntervalThreshold	Indicates the threshold for generating the time interval between the IP packets alarm on the ancillary data flow. The range is from 819.2 ns to 53685452.8 ns in intervals of 819.2 ns.
Secondary::IgmpV3Settings::FilterType	Only needed when the join type is set to 3 = IGMPv3. Indicates the filter type applied. 0 = Invalid 1 = Inclusion 2 = Exclusion

Change IP Properties

Although the IP properties may be changed by editing the configuration file directly, the IP address and port numbers can also be set via the command line interface. Because channel numbering is zero-based here, **Output Channel 1** is referred to as `VIDEOOUT_0`. To check the currently configured values call the command using GET (instead of SET) without parameters.

Copyright © 2026 Vizrt

To Change/Check the Destination Address and Port of the Output Channels

```
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*ENABLEDSTATE SET ON/OFF

MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*DSTADDRESS SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*DSTUDPPORT SET xxxxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*REDUNDANCYDSTADDRESS SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*REDUNDANCYDSTUDPPORT SET xxxxx

MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*DSTADDRESSKEY SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*DSTUDPPORTKEY SET xxxxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*REDUNDANCYDSTADDRESSKEY SET
xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*REDUNDANCYDSTUDPPORTKEY SET xxxxx

MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*DSTADDRESSAUDIO SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*DSTUDPPORTAUDIO SET xxxxx

MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*REDUNDANCYDSTADDRESSAUDIO SET
xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*REDUNDANCYDSTUDPPORTAUDIO SET xxxxx

MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*DSTADDRESSANC SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*DSTUDPPORTANC SET xxxxx

MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*REDUNDANCYDSTADDRESSANC SET
xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*MATROX*VIDEOOUT_x*REDUNDANCYDSTUDPPORTANC SET xxxxx
```

Copyright © 2026 Vizrt

To Change the Source Address of the Input Channels

```
MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*ENABLEDSTATE SET ON/OFF

MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*DSTADDRESS SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*DSTUDPPORT SET xxxxx

MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*REDUNDANCYDSTADDRESS SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*REDUNDANCYDSTUDPPORT SET xxxxx

MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*DSTADDRESSAUDIO SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*DSTUDPPORTAUDIO SET xxxxx

MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*REDUNDANCYDSTADDRESSAUDIO SET
xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*REDUNDANCYDSTUDPPORTAUDIO SET xxxxx

MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*DSTADDRESSANC SET xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*DSTUDPPORTANC SET xxxxx

MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*REDUNDANCYDSTADDRESSANC SET
xxx.xxx.xxx.xxx
MAIN*CONFIGURATION*CHANNELS*LIVEIN_x*REDUNDANCYDSTUDPPORTANC SET xxxxx
```

Viz Engine currently supports NMOS IS-04 and NMOS IS-05 protocols for IP enabled IO boards.

Copyright © 2026 Vizrt

How to Setup X.mio5 Separate Flows

General Information

Several use cases require the increased flexibility to combine multiple flows and treat them as one entity or channel.

To achieve this we need to specify the definition of a channel (in IP) first. A channel consists of:

- One or two video flows (two flows AKA fill/key are not yet implemented for IP inputs).
- 0..n audio flows containing up to 64 tracks of audio (follow the correct naming here).
- 0..n ancillary data flows (currently only one ancillary data flow is supported).

Attention: There are two system wide limitations: the maximum amount of connectors/flows on an X.mio5 Q25 is 200, and the maximum amount of audio tracks across all flows is 2048. An X.mio5 D25 can host 100 flows and 1024 tracks of audio.

Group Hints, which are part of the IP configuration, will be used to create a channel.

Attention: Matrox X.mio3 does not support separate flows.

Configuration

All configuration must be done in the *ipconfig.xml* file, which can be found in *%PROGRAMDATA%\vizrt\VizEngine*.

- *<GroupHint>* entries are used to determine which connectors belong to which channel. The definition always start with one video flow. It is not possible to create an audio only channel.
- Group hints must follow the rules of BCP-002-01, specifically the rules for the Group Hint Tags detailed in the NMOS parameter registers.
- Key connectors are used automatically when configured. The group hint is ignored in that case.
- Make sure, that the amount of audio flows for inputs and outputs is identical.
- It is necessary to set the *<TrackCount>* per audio flows for both inputs and outputs to be the same.
- Audio track configuration for X.mio5 is done in the *ipconfig.xml* only. Track settings in the Viz Engine configuration file are ignored.

Known Limitations

- The audio input and output configuration must be identical.
- It is necessary to feed exactly the amount of tracks configured.
- It is highly recommended to use power of two track configurations only (2, 4, 8, 16, 32, 64).

Copyright © 2026 Vizrt

NMOS Configuration

General Information

NMOS control is enabled on a driver level. No changes are needed in Viz Engine. NMOS connection requests must be sent out-of-band for X.mio3. For X.mio5, the requests can be either out-of-band or in-band.

- **Out-of-band:** RDS and receiver NIC on the attached machines must be located in a network separated from the essence network, which hosts video, audio and ancillary data flows.
- **In-band:** The RDS can be hosted in the essence network, NMOS requests are sent alongside the essence flows.

Important: PTP signals are always in-band.

Configuration

- Configure Viz Engine as usual.
- Open *%ProgramFiles%\Matrox DSX-TopologyUtils\System64\Axxxxxx.json* with any text editor, where *Axxxxxx* is the serial number of your installed Matrox IP card.
- Edit the mandatory entries:
  - *enabled* must be set to `true` to enable NMOS functionality.
  - *host name* and *port* in the section *fallback registration server* must point to the RDS (Registration & Discovery Server) for manual configuration.
  - *use service discovery* must be set to `true` for automatic discovery of the RDS.

  It is recommended to set the Control IP and port to the values shown in the following example:

Example:

```
"enabled": true,
"local host name": "0.0.0.0",
"local port": 8080,
"fallback registration server": {
    "host name": "10.1.1.240",
    "port": 5432,
    "api version": "auto"
},
"use service discovery": true,
"use dns-sd unicast only": false,
"heartbeat interval": 5,
"use secure communication": false,
```

- Edit all the labels, so that they can easily be identified in control applications or NMOS explorers (optional).

Copyright © 2026 Vizrt

- **Save** the file and restart either the machine or *mvNetworkService* from Windows services.

Special Firewall Rules

In case the a firewall is active on the system, two rules must be added for the NMOS service to run properly:

- If the default Windows firewall is **not** used, UDP port `5353` must be opened for the process *mDNSResponder.exe* for service discovery to work. If the Windows firewall is used, this is done by the driver installer.
- The Matrox NMOS service (mvNetworkService) must be accessible. Use the following in an administrator console to add the rule to the Windows firewall:
  `netsh advfirewall firewall add rule name="Matrox NMOS API" protocol=TCP dir=in localport={local port} action=allow program="C:\Program Files\Matrox DSX-TopologyUtils\System64\mvNetworkService.exe"`
  where *{local port}* is the listening port for the incoming requests as configured in JSON file mentioned above.

Copyright © 2026 Vizrt

6.4.2 Matrox Hardware

Viz Engine supports a wide range of Matrox products. This chapter provides an overview of the different Matrox video cards supported by Viz Engine, as well as providing their common installation procedures.

The Matrox X.mio series boards are used as Vizrt's HD, SD and UHD multi-channel video- and audio I/O solution.

- **DSX LE6:** The next generation IP card, SMPTE ST 2110 over 100GbE.
- **X.mio5/DSX LE 5 series:** Integrates Viz Engine into an IP based workflow (2110) (optional SDI).
- **DSX Core:** Allows to playback Matrox encoded clips without the need of a physical hardware.
- **X.mio3/DSX LE 4 series:** Introduced in 2015 and is supported by Viz Engine 3.8 and higher.
- **M264:** Hardware H.264 encoding and decoding card.

This section contains information on the following topics:

- Configuration History for Matrox X.mio and DSX Series
- Audio Cable Assignment
- Matrox DSX.Core
- Matrox M264
- Matrox X.mio3 12G
- Matrox X.mio3 - DSX LE 4
- Matrox X.mio5 IP (D25 / Q25)
- Matrox X.mio5 - DSX LE5 SDI
- Matrox Driver Installation
- Troubleshooting Matrox Video Hardware
- Matrox DSX LE6
- Matrox DSX LE5 IP (D25 / Q25)

See Also:

- Configuration History for Matrox X.mio and DSX Series

Copyright © 2026 Vizrt

Configuration History for Matrox X.mio and DSX Series

Viz Artist/ Engine Version	Driver	Supported Hardware
5.6.0	Topology Utils 11.2.102.2632	X.mio5 IP (D25/Q25), X.mio5 SDI, DSX LE5 IP (D25/Q25), DSX LE5 LP, DSX LE6, X.mio3, X.mio3 12G, DSX LE4, M264
5.5.0 - 5.5.3	Topology Utils 11.2.101.2608	X.mio5 IP, X.mio5 SDI, DSX LE5 IP (D25/Q25), DSX LE5 LP, DSX LE6, X.mio3, X.mio3 12G, DSX LE4, M264
5.4.0, 5.4.1	Topology Utils 11.1.101.2163	X.mio5 IP, X.mio5 SDI, DSX LE6, X.mio3, X.mio3 12G, DSX LE4, DSX LE5 LP, M264
5.3.2	Topology Utils 10.4.102.1388 SP2	X.mio5 IP, X.mio5 SDI, X.mio3, X.mio3 12G, DSX LE4, DSX LE5 LP, M264
5.3.1	Topology Utils 10.4.102.1346	X.mio5 IP, X.mio5 SDI, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, DSX LE5 LP, M264
5.3.0	Topology Utils 10.4.102.1342	X.mio5 IP, X.mio5 SDI, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, DSX LE5 LP, M264
5.2	Topology Utils 10.4.101.1285	X.mio5 IP, X.mio5 SDI, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264
5.1	Topology Utils 10.3.102.783	X.mio5 IP, X.mio5 SDI, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264
5.0	Topology Utils 10.3.100.679	X.mio5 IP, X.mio5 SDI, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264
4.4.0, 4.4.1	Topology Utils 10.2.102.26084	X.mio5 IP, X.mio5 SDI, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264
4.3.0, 4.3.1	Topology Utils 10.2.100.26040	X.mio5, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264
4.2.0	Topology Utils 10.1.101.24973	X.mio5, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264

Copyright © 2026 Vizrt

Viz Artist/ Engine Version	Driver	Supported Hardware
4.1.0, 3.14.4	Topology Utils 10.1.100.24874	X.mio5, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264
4.0.0	Topology Utils 10.1.020.24098	X.mio5, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264
3.14.5	Topology Utils 10.1.102.24988	X.mio5, X.mio3 IP, X.mio3, X.mio3 12G, DSX LE4, M264
3.14.3	DSX.utils/Topology Utils 10.1.060.24670	X.mio5, X.mio3 IP, X.mio3, X.mio3 12G, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, M264
3.14.1, 3.14.2	DSX.utils 10.1.040.24351	X.mio5, X.mio3 IP, X.mio3, X.mio3 12G, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, M264
3.14.0	DSX.utils 10.1.020.24097	X.mio5, X.mio3 IP, X.mio3, X.mio3 12G, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, M264
3.13.0	DSX.utils 10.0.100.23773	X.mio3 IP, X.mio3, X.mio3 12G, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, M264
3.12.1	DSX.utils 9.9.1. 23136	X.mio3 IP, X.mio3, X.mio3 12G, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, M264
3.12.0	DSX.utils 9.9.1. 23136	X.mio3 IP, X.mio3, X.mio3 12G, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, M264
3.11.1	DSX.utils 9.9.0. 23060	X.mio3 IP, X.mio3, X.mio3 12G, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, M264
3.11.0	DSX.utils 9.9.0.23060	X.mio3 IP, X.mio3, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, M264
3.10.0	DSX.utils 9.8.1.22400	X.mio3 IP, X.mio3, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, DSX LE2/CG
3.9.1	DSX.utils 9.8.1.22400	X.mio3 IP, X.mio3, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, DSX LE2/CG
3.9.0	DSX.utils 9.8.0.22358	X.mio3 IP, X.mio3, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, DSX LE2/CG

Copyright © 2026 Vizrt

Viz Artist/ Engine Version	Driver	Supported Hardware
3.8.3	DSX.utils 9.7.0.21682	X.mio3 IP, X.mio3, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, DSX LE2/CG
3.8.2	DSX.utils 9.6.0.18841	X.mio3, X.mio2 Plus, X.mio2, DSX LE4, DSX LE3, DSX LE2/CG
3.8.1	DSX.utils 9.6.0.18836	X.mio3, X.mio2 Plus, X.mio2, DSX LE3, DSX LE2/CG
3.8.0	DSX.utils 9.5.0.17766	X.mio3, X.mio2 Plus, X.mio2, DSX LE3, DSX LE2/CG
3.7.0	DSX.utils 9.4.0.9040 (release)	X.mio2 Plus, X.mio2, X.mio, DSX LE3, DSX LE2/CG
3.6.4	DSX.utils 9.2.2.2343	X.mio2 Plus, X.mio2, X.mio, DSX LE3, DSX LE2/CG
3.6.3	DSX.utils 9.2.2.2335 (SP2)	X.mio2 Plus, X.mio2, X.mio, DSX LE3, DSX LE2/CG
3.6.2	DSX.utils 9.2.2.2331 (SP2)	X.mio2 Plus, X.mio2, X.mio, DSX LE3, DSX LE2/CG
3.6.0	DSX.utils 9.2.2.2317 (SP2)	X.mio2 Plus, X.mio2, X.mio, DSX LE3, DSX LE2/CG
3.5.3	DSX.utils 7.5.2.1448	X.mio2, X.mio, DSX LE2/CG
3.5.0 - 3.5.2	DSX.utils 7.5.2.447 (SP2)	X.mio2, X.mio, DSX LE2/CG
3.3.0	DSX.utils 7.5.2.443	X.mio2, X.mio
3.2.2	DSX.utils 5.0.3.171	X.mio2, X.mio
3.1.0 - 3.2.1	DSX.utils 5.0.3.166	X.mio2, X.mio

Copyright © 2026 Vizrt

Audio Cable Assignment

This section contains information on the following topics:

- AES Input Cable Assignment
- AES Output Cable Assignment

AES Input Cable Assignment

Card	Viz Audio Config	AES Cable	Matrox Video Channel
X.mio3	2 tracks	A IN 1/2 A IN 3/4 A IN 5/6 A IN 7/8 B IN 1/2 B IN 3/4 B IN 5/6 B IN 5/6 A IN 1/2+3/4	Video IN A Video IN B Video IN C Video IN D Video IN E Video IN F Video IN G Video IN H Video IN A
	4 tracks	A IN 1/2+3/4 A IN 5/6+7/8 B IN 1/2+3/4 B IN 5/6+7/8	Video IN A Video IN B Video IN C Video IN D
	8 tracks	A IN 1/2+3/4+5/6+7/8 B IN 1/2+3/4+5/6+7/8	Video IN A Video IN B
	16 tracks	A IN 1/2+3/4+5/6+7/8 B IN 1/2+3/4+5/6+7/8	Video IN A

AES Output Cable Assignment

Card	Viz Audio Config	AES Cable	Matrox Video Channel
X.mio3	2 tracks	A Out 1/2 A Out 3/4 A Out 5/6 A Out 7/8 B Out 1/2 B Out 3/4 B Out 5/6 B Out 7/8	Video Out A Video Out B Video Out C Video Out D Video Out E Video Out F Video Out G Video Out H

Copyright © 2026 Vizrt

Card	Viz Audio Config	AES Cable	Matrox Video Channel
	4 tracks	A Out 1/2+3/4 A Out 5/6+7/8 B Out 1/2+3/4 B Out 5/6+7/8	Video Out A Video Out B Video Out C Video Out D
	8 tracks	A Out 1/2+3/4+5/6+7/8 B Out 1/2+3/4+5/6+7/8	Video Out A Video Out B
	16 tracks	A Out 1/2+3/4+5/6+7/8 B Out 1/2+3/4+5/6+7/8	Video Out A

Copyright © 2026 Vizrt

Matrox DSX.Core

Overview

A DSX.Core is a software-only module allowing you to decode and encode clips or Matrox streams. DSX.Core is mainly used in virtual environments. It supports the same codecs as a real hardware board. Since Engine 4.4.1, licensing is done directly via Viz Engine and does not require any additional configuration on Matrox side.

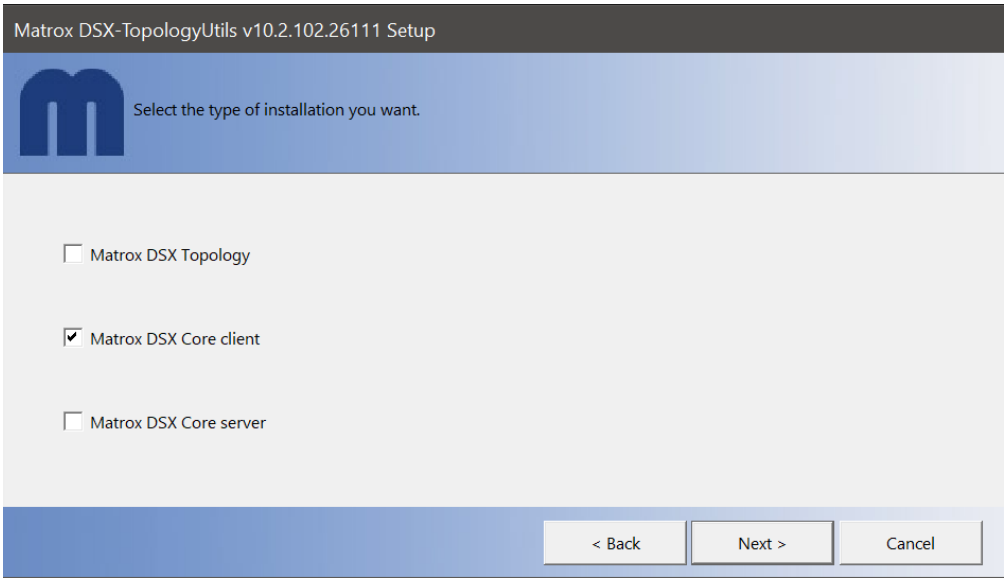
DSX Core requires one of the two licenses:

- DSX Core HD license
- DSX Core UHD license

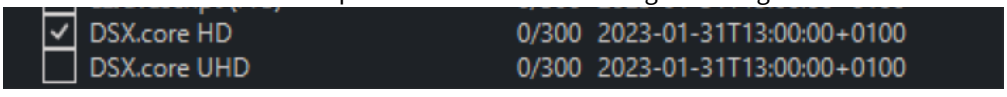
To Install DSX.Core

Note: Make sure you start the installation with Administrator rights.

- Launch *DSX-TopologyUtils.exe*.
- Be sure to select Matrox DSX Core client:



- Be sure to select one of the required licenses within Viz Engine Configuration:



Depending on your setup, you may now use it to play back clips using Matrox codecs, use Matrox Streams or the ClipOut Channel. Further information about DSX.Core is available at matrox.com.

Note: Matrox Section and Clip Output are disabled in Configuration mode due to technical reasons.

Known Issue: After the installation of the DSX.Core client version of the driver, perform the following steps:

Copyright © 2026 Vizrt

1. Unregister *mvfDsxCore.dll*.
    a. Click **Start > Run** (or use the Windows command line: **Search > CMD >** (Right click) **Run as Administrator**).
    b. Type `REGSVR32 /U "C:\Program Files\Matrox DSX-TopologyUtils\System64\mvfDsxCore.dll"` and press **ENTER**.
2. Shut down *X.info* in the task manager (*mveXinfo.exe*).
3. Delete *mvfDsxCore.dll* from the folder *C:\Program Files\Matrox DSX-TopologyUtils\System64*.
4. Start *X.info (mveXinfo.exe).*

Copyright © 2026 Vizrt

Matrox M264

Matrox M264 cards feature hardware-based H.264 encoding and decoding capabilities. Depending on the card used, up to four UHD XAVC streams can be encoded/decoded simultaneously. Viz Engine and Channel Recorder are both capable of using the M264 cards for playback and capture of XAVC material, without affecting CPU performance.



Key Features

- PCI Express Cards
- H.264 video encoding and decoding
- UHD resolution support
- 4:2:2 10-bit
- Sony XAVC compliant encoding including 4K XAVC Long and 4K Intra Class 480
- Panasonic AVC-Ultra compliant

Performance for 4:2:2 10-bit Streams for M264 (S1)

Resolutions	Structure	Format	Number of streams
2160p	I-frame	XAVC Class 300 & Class 480 / AVC-Intra 4K	1
1080p	I-frame	XAVC Class 200 / AVC-Intra 200	3
1080i	I-frame	XAVC Class 100 / AVC-Intra 100	10

Copyright © 2026 Vizrt

2160p	Long GOP	Generic / XAVC 4K Long	1
1080p	Long GOP	Generic / XAVC Long50 / AVC-LongG50	5
1080i	Long GOP	Generic / XAVC Long25 / AVC-LongG25	10
720p	Long GOP	Generic / XAVC Long25 / AVC-LongG25	10

Note: Since the S2, S3 or S4 models host more encoder/decoder chips respectively, the number of possible streams is doubled, tripled or quadrupled.

M264 Configuration

Viz Engine

How to Choose Which Cores to Use

Viz Engine automatically uses all M264 cores available from the installed M264 boards in the system. There is a configuration to be able to choose the cores to be used by a specific Viz Engine instance:

```
## Specify which M264 cores the engine is allowed to use (e.g. 0,1,2). Empty string
means the engine can use all of them. Use any non-numerical string, such as None, to
not use any M264 cores.
#* Matrox.Clip.M264CoresToUse: Default=
# Matrox.Clip.M264CoresToUse =
```

For example, when using dual channel with an M264S4, it might be desirable for the first instance to use a different set of cores than the second instance, so the configuration for the first instance would look like this: `Matrox.Clip.M264CoresToUse = 0,1`

The second instance configuration would look like this: `Matrox.Clip.M264CoresToUse = 2,3`

These configurations mean that the first instance can only use the first and second core, while the second instance can only use the third and the fourth core of the M264S4.

How to Seek Frame Accurate XAVC Long GOP Clips

By default, Viz Engine is not able to seek XAVC Long GOP clips accurately. However, if an M264 is used, it is possible to seek frame accurately using the following configuration:

```
## Enable/Disable frame accurate seek for long GOP AVC clips when using M264
#* ClipIn1.FrameAccurateSeek: Default=0
# ClipIn1.FrameAccurateSeek = 0
```

Copyright © 2026 Vizrt

Channel Recorder

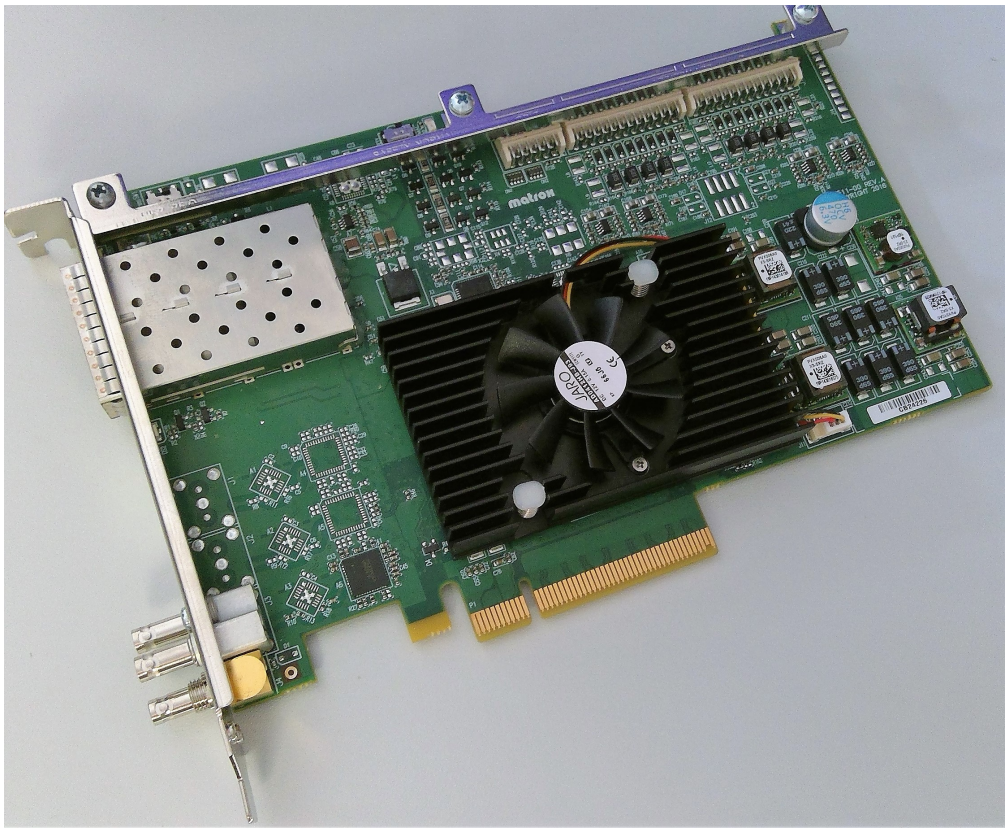
When the XAVC UHD codec is used, Channel Recorder automatically detects any installed M264 card and uses it for encoding. No further configuration is necessary.

Copyright © 2026 Vizrt

Matrox X.mio3 12G

The Matrox X.mio3 12G video board is targeted for the growing UHD market. It is a half-length, full height PCI Express card that offers two 12G SDI inputs and two 12G SDI outputs, using one input and one output module manufactured by Embrionix. The card is based on the original Matrox X.mio3 design and is capable of handling one input and fill/key output, or two inputs and one fill output.

Important: Due to the increased bandwidth of the signal, it is important to use high quality cables and as few connection points as possible. It is not recommended to use converter cables from Mini-BNC to standard BNC.



There is one Mini-BNC connector used to connect the card to the house genlock signal. The other two connectors are not implemented.

Key Features

- Half-length PCI Express card.
- Up to two 12G SDI inputs.
- Up to two 12G SDI outputs.
- Frame synchronizers on each input.
- VANC and HANC support for each input and output.
- Analog blackburst reference input (tri-level or bi-level).

Copyright © 2026 Vizrt

- Onboard 4K scaler.
- Onboard 4K compositor.
- Up to 16 channels of AES/EBU inputs and outputs.

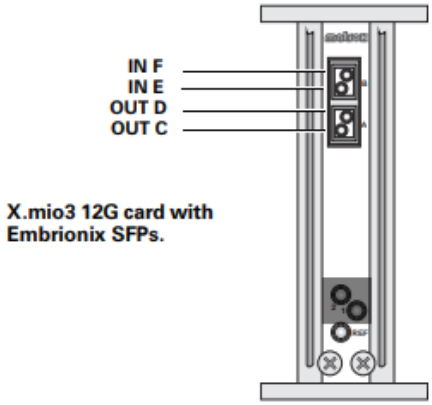
Matrox X.mio3 12G Configuration

The main video properties match those of a Matrox X.mio3, meaning you can have either one UHD input and one Fill/Key UHD output, or two UHD inputs and one Fill UHD output. Please see Matrox Configuration Section for further details.

X.mio3 12G connections

	Type of connector	Supported models	Label	Channel
X.mio3 12G	SFP (RX) (12G SDI)	EB12HD2R-MNR-2 EB12LC2R-MN-P	B	IN F/ Key E
				IN E
	SFP (TX) (12G SDI)	EB12HD2T-MNR-2 EB12LC2T-SN-13D	A	OUT D
				OUT C / Key D
	HD-BNC (3G SDI)	not supported	2	OUT B/ Key A
	HD-BNC (3G SDI)		1	OUT A

For more information on the supported Embrionix SFP models, see "X.mio3 12G validated SFPs" in the *Matrox X.mio3 Installation Manual*



X.mio3 12G card with Embrionix SFPs.

Attention: The HD-BNC connectors labeled 1 and 2 are not supported by Viz Engine.

Embrionix SFP I/O Modules

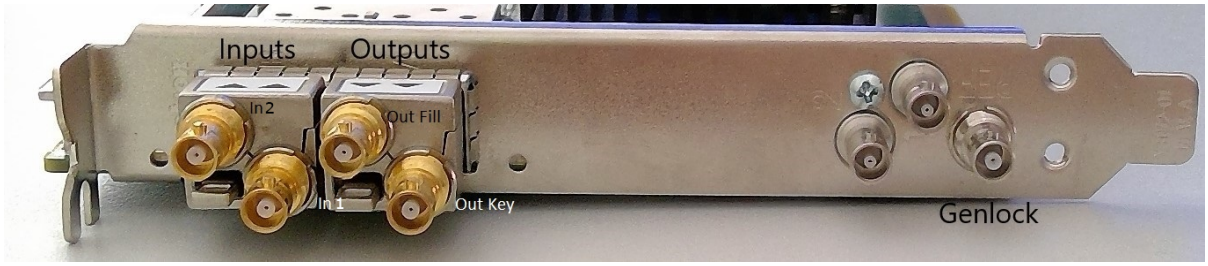
The needed 12G SFPs are available in two flavors, one for input and one for output. To connect to the SFPs standard Mini-BNC cables can be used.

Copyright © 2026 Vizrt

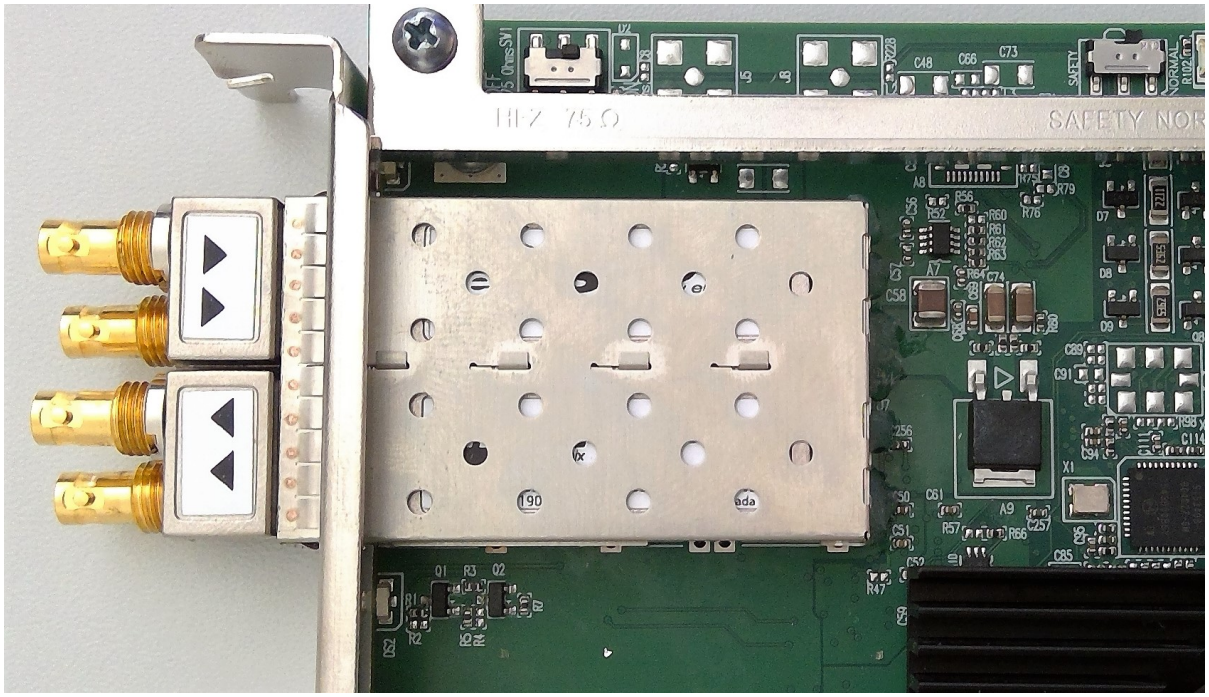


Matrox X.mio3 12G SFP Configuration

It is important that the seating of SFPs is correct, otherwise inputs and outputs are not available. The output SFP goes into the lower bay and the input SFP into the upper one. See pictures for reference.



Copyright © 2026 Vizrt

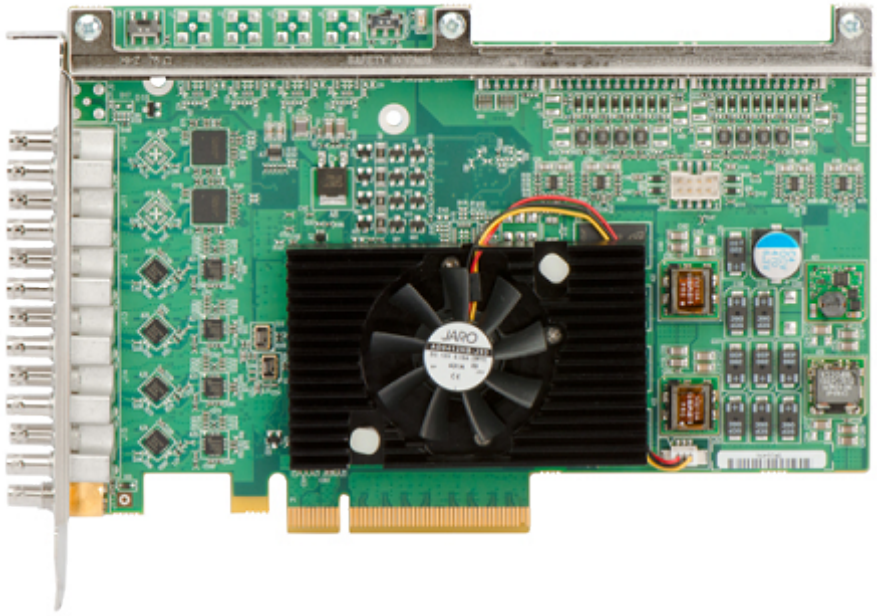


Copyright © 2026 Vizrt

Matrox X.mio3 - DSX LE 4

This section first gives an overview of the Matrox X.mio3 video cards, then describes the details specific to the full height X.mio3 FH and low profile X.mio3 LP versions of the card. The Matrox X.mio3 IP - DSX LE 4 IP board is discussed in a separate section.

DSX LE 4 cards have the same capabilities as their X.mio3 counterparts, but lack an onboard video mixer, so DVE is not usable on those boards. Whenever X.mio3 is mentioned in this document, the same is true for DSX.LE 4.



IMPORTANT! X.mio3 and DSX LE 4 boards must be installed in a PCIe Gen 4 or Gen 3 slot. The boards will not work in PCIe Gen 5 or higher slots.

In this section:

- Matrox X.mio3 Overview
- X.mio3 FH
    - Key Features
- X.mio3 LP
    - Key Features
- X.mio3 Connector Mapping Reference
    - FH X.mio3 X2 Connector Mapping for Quad-Link Inputs
- X.mio3 I/O Port Configuration
- Viz Engine Matrox Video Mapping Configuration
- To Upgrade X.mio Class and Firmware
- X.mio3 AES Audio Kit
- Upgrading Matrox Firmware for Two Sample Interleave

Copyright © 2026 Vizrt

Matrox X.mio3 Overview

The X.mio3 can be ordered in two configurations: FH (full height, 12 physical HD-BNC connectors, plus one connector for sync) and LP (low profile, eight physical HD-BNC Connectors, plus one connector for sync). Both are half length PCI express Gen2 x8 cards.

- X.mio3FH (Full Height, half length) with up to 12 I/O ports, SD to 4K capable. This card can be ordered with six, eight or X2 (12) I/O. Eight ports are hardwired through relays: Four inputs to four outputs. Model /6 means six I/Os are available, model /8 eight I/Os. Model X2 (twelve ports) has four freely configurable ports. There can be maximum eight I/Os of one type (either in or out) enabled on a card.
- X.mio3LP (Low Profile, half length) with up to eight I/O ports, SD to 4K capable.
- The I/O ports are configurable. This means that for instance the X.mio3 model /8 can be configured to have from 8/0 (for example, 8 channels in, zero channels out) to 0/8 (zero channels in, eight channels out). Only even combinations are allowed.
- One connector is reserved for a reference signal (sync) marked as REF.
  The default card is an X.mio3/6/100, meaning X.mio3 FH with six connectors class 100. The class 100 can not play any video clips and upgrade to class 500 is required to enable both SD and HD clip playback. For Apple ProRes support, the class 550 is needed.

The I/O connectors for the X.mio3 are of the Mini-BNC (also called HD-BNC) type. The use of cable converters from Mini-BNC to BNC may be required.

The X.mio3 cards can be combined with the AES Audio Kit for 16 AES audio channels In/Out, see X.mio3 AES Audio Kit.



X.mio3 FH

X.mio3 FH provides multichannel SDI IO with hardware based video processing in a full-height, half-length PCI express card with 13 physical HD-BNC connectors. This card provides up to 12 reconfigurable I/Os, from SD to 4K, one connector is reserved for sync. Support for AES/EBU, LTC and GPIO provide for versatile connectivity. The multi-channel hardware processing accelerates compute-intensive operations including motion-adaptive de-interlacing, up/down/cross scaling and mixing/compositing for all resolutions, including 4K.

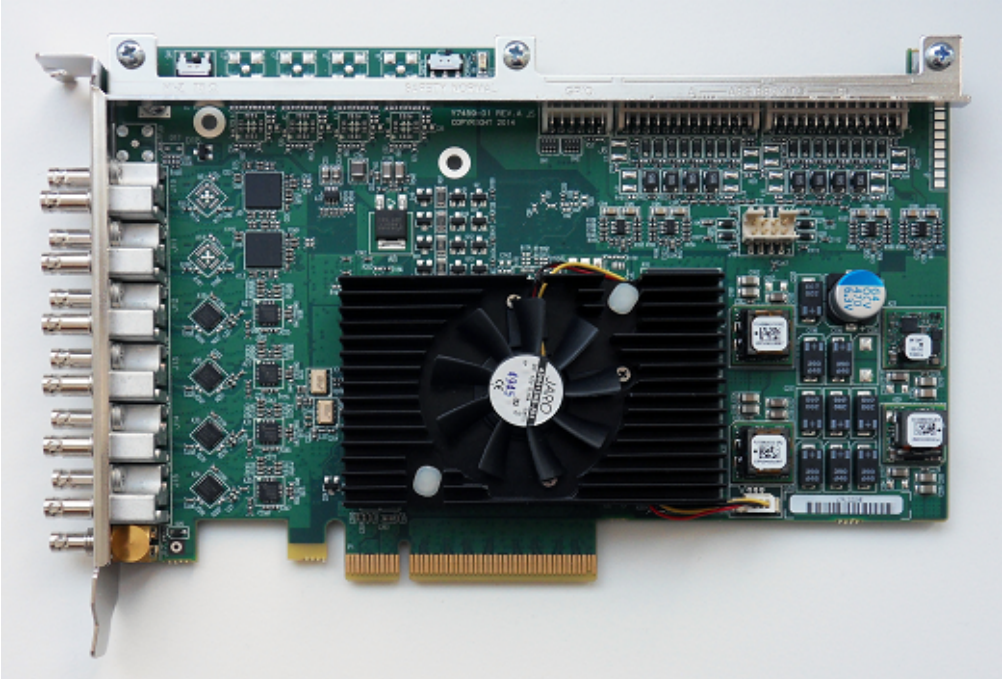
Key Features

- Half-length PCI express card

Copyright © 2026 Vizrt

- Re-configurable IO that can support up to 12 SDI IO
- Frame synchronizers
- VANC and HANC support for each input and output
- Analog black burst reference input (tri-level or bi-level)
- On-board multi-channel MADI (Motion Adaptive De-Interlacer)
- On-board multi-channel Up/Down/Cross scaler
- On-board multilayer compositor
- Automatic video relay bypass
- Live zero-frame delay video and audio mixers
- Up to 16 channels of AES/EBU inputs and outputs
- Up to eight LTC inputs and outputs
- RS422 control
- Single slot all inclusive option

IMPORTANT! Not all X.mio3 hardware features are supported by Viz Engine. Features available depends on drivers, SDK and Viz Engine versions in use.



X.mio3 LP

Matrox X.mio3 LP is a low-profile, half-length PCIe card with up to eight reconfigurable SDI I/Os from SD to 4K.

Key Features

- Low profile, half-length PCI-e card.
- Re-configurable I/O that can support up to eight SDI inputs or outputs.

Copyright © 2026 Vizrt

- Frame synchronizers.
- VANC and HANC support for each input and output.
- Analog black burst reference input (tri-level or bi-level).
- On-board multi-channel MADI (Motion Adaptive De-Interlacer).
- On-board multi-channel Up/Down/Cross scaler.
- On-board multilayer compositor.
- Automatic video relay bypass (optional).
- Live zero-frame delay video and audio mixer.

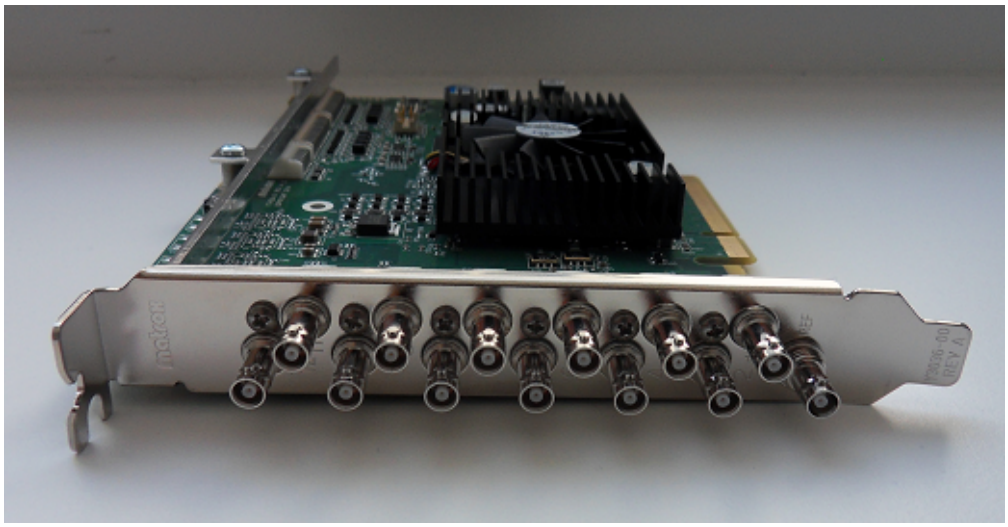
IMPORTANT! Not all X.mio3 hardware features are supported by Viz Engine. Features available depends on drivers, SDK and Viz Engine versions in use.

The X.mio3 LP models are: /4, /6, /8 - corresponding to the number of I/O port the configuration supports.

X.mio3 Connector Mapping Reference

This section is a reference for X.mio3 connector mappings.

Upgrades and firmware setup of I/O mapping should be done by Vizrt or a qualified technician.



The X.mio3 connectors are labeled 1 through 12 (plus one connector for sync), eight connectors for the LP (low profile) card. As explained, the connectors can be configured for in or out and with various capabilities for clip playback depending on class - with possibilities for firmware class upgrades.

IMPORTANT! There can be maximum eight ports of the same type (either in or out), this restriction also applies to the model x2 that has 12 connectors.

Connectors 1,3,5,7 are always reserved for **input**, connectors 2,4,6,8 for **output**. Hence, if the Watchdog is active port 1 (in) goes to port 2 (out) with relay, port 3 (in) to port 4(out) and so on.

The x2 model can as mentioned only have eight I/Os of the same type; 8in - 4out or 4in - 8out.

To configure the X.mio3 card for usage:

Copyright © 2026 Vizrt

- Make sure that the card has the required mix of inputs and outputs required for the intended usage, for instance to satisfy fill, key, full-screen graphics requirements. This is done, if required, with the *mvConnectorConfig.exe* utility described in X.mio3 I/O Port Configuration.
- Map the I/Os from step 1 above to the Viz Video channels using the Viz Engine Configuration utility. This is mostly the same procedure for X.mio3 as for previous Matrox cards, see Viz Engine Matrox Video Mapping Configuration.

FH X.mio3 X2 Connector Mapping for Quad-Link Inputs

The following table lists the connectors used for quad-link inputs, depending on the I/O port mapping of the card. This has been tested on X.mio3 X2 and approved by QA. It may differ from card to card. Viz Engine does not support fill+key for quad-link inputs and the mapping between Engine input channels and ports is hard-coded.

FPGA configuration	Mapping	Input	Connectors used (Matrox Terminology)	Connectors used (Viz Terminology)
2SI Default	4in/8out	1	A-C-E-G	1-3-5-7
	8in/4out	1	A-C-I-J	1-3-9-10
		2	E-G-K-L	5-7-11-12
Square Division (SD) & 2SI Alternate	4in/8out	1	A-C-E-G	1-3-5-7
	8in/4out	1	A-C-E-G	1-3-5-7
		2	I-J-K-L	9-10-11-12

IMPORTANT! Default mapping for 2SI does not support quad-link inputs. Please flash the card to **Alternate mapping** to enable this feature.

X.mio3 I/O Port Configuration

Normally the I/O ports are pre-configured when delivered from Vizrt. If the I/O configuration needs to be changed, the Matrox command-line utility *mvConnectorConfig.exe* normally found in: *C:\Program Files\Matrox DSX-TopologyUtils\drivers* must be used. *mvConnectorConfig.exe* can be called from the command prompt without options to get a helpful usage message. To configure all X.mio3 cards in a system to use four inputs and two outputs the syntax is:

```
C:\Program Files\Matrox DSX-TopologyUtils\drivers\mvConnectorConfig.exe -4in2out
```

To configure six outputs and no inputs the syntax is:

```
C:\Program Files\Matrox DSX-TopologyUtils\drivers\mvConnectorConfig.exe -0in6out
```

Copyright © 2026 Vizrt

and so on for the various I/O possibilities.

The computer should be stable with low load and no Viz programs running when using the configuration utility. Never interrupt a firmware upgrade. After making changes to the Matrox I/O configuration, the computer must be re-started with a complete power off.

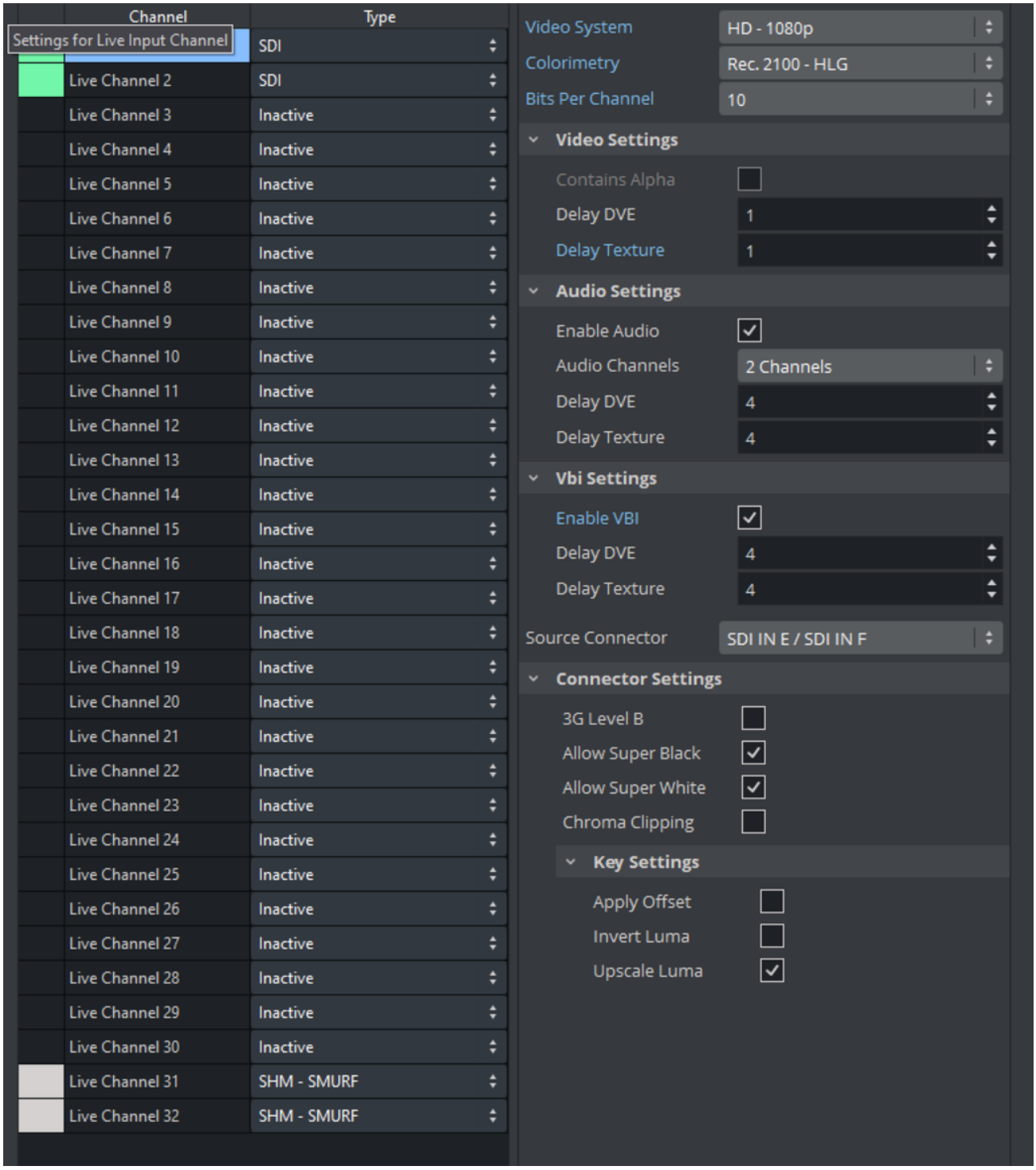
> **Note:** For X.mio3/X2 cards, only 4/8 and 8/4 are supported. Please check the connector mapping document for more information.

Viz Engine Matrox Video Mapping Configuration

Open **Viz Configuration > Matrox** to configure the Matrox video channel mapping and mapping type.

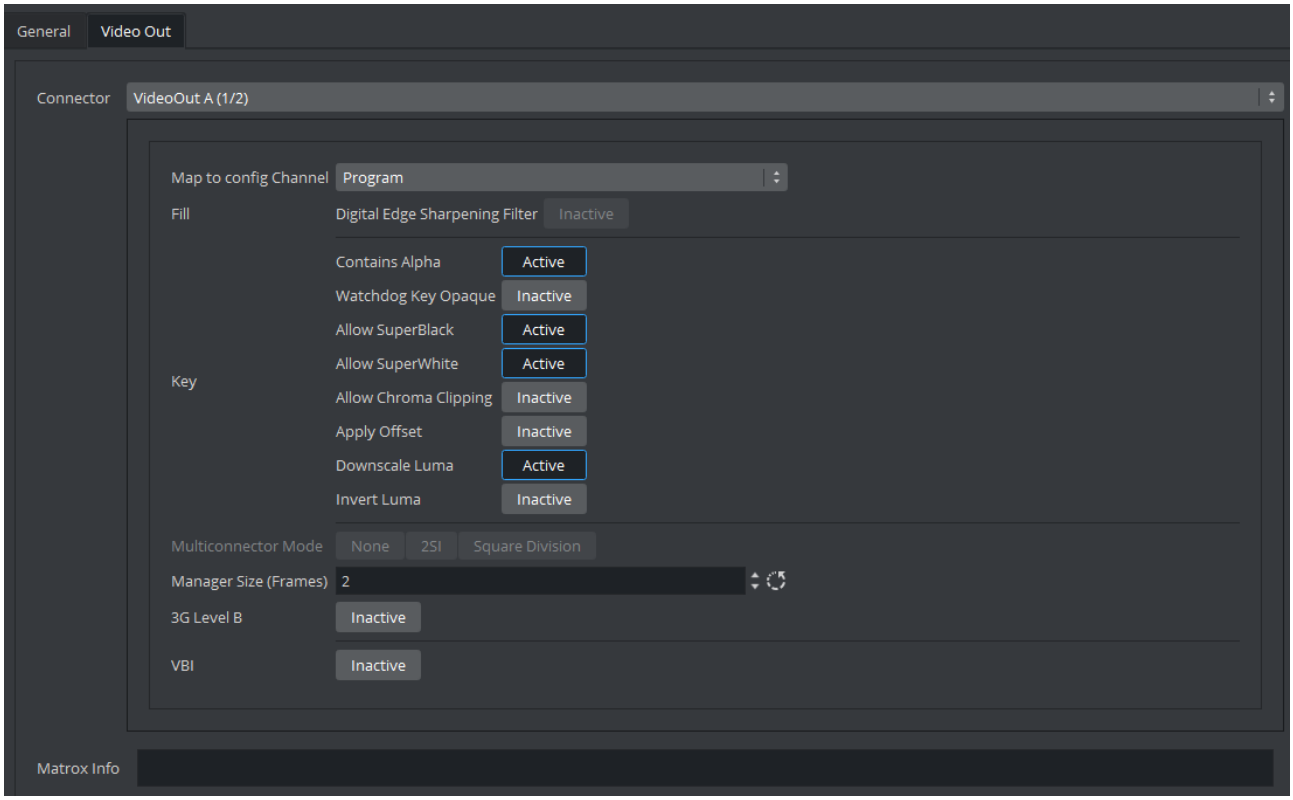
The Viz video inputs are named alphabetically and limited to the number of available I/O channels the system has. The number in parenthesis indicates which channel is being used for fill and key. If you do not need a key signal in the output, de-select the **Key > Contains Alpha** option to make the I/O channel available for other usages. Example configuration:

Copyright © 2026 Vizrt



Navigate to **Config > Matrox > Video Out** for an overview showing which I/O channels are used for fill and key:

Copyright © 2026 Vizrt



To Upgrade X.mio Class and Firmware

To upgrade your X.mio3 card, for example to enable more IO-ports or a codec upgrade for HD-Clipback, you must use the *mvDongleUpdater.exe* update utility with your Vizrt supplied license upgrade file *<filename>.OPT*. This utility is installed with the Matrox driver package and is normally installed to *C:\Program Files\Programs\Matrox DSK.utils\drivers\mvDongleUpdater.exe.*

To upgrade the card use this syntax:

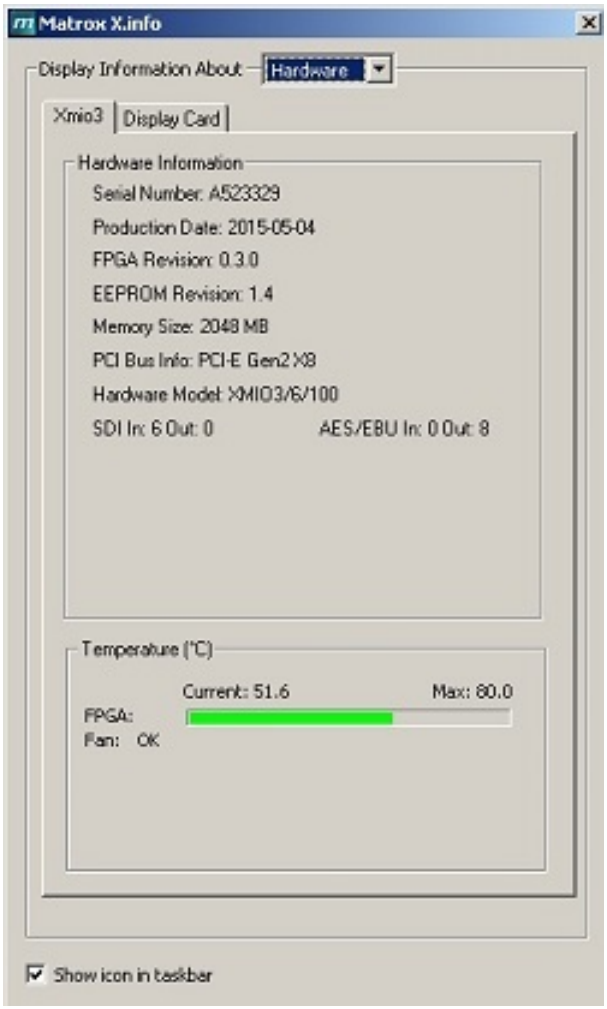
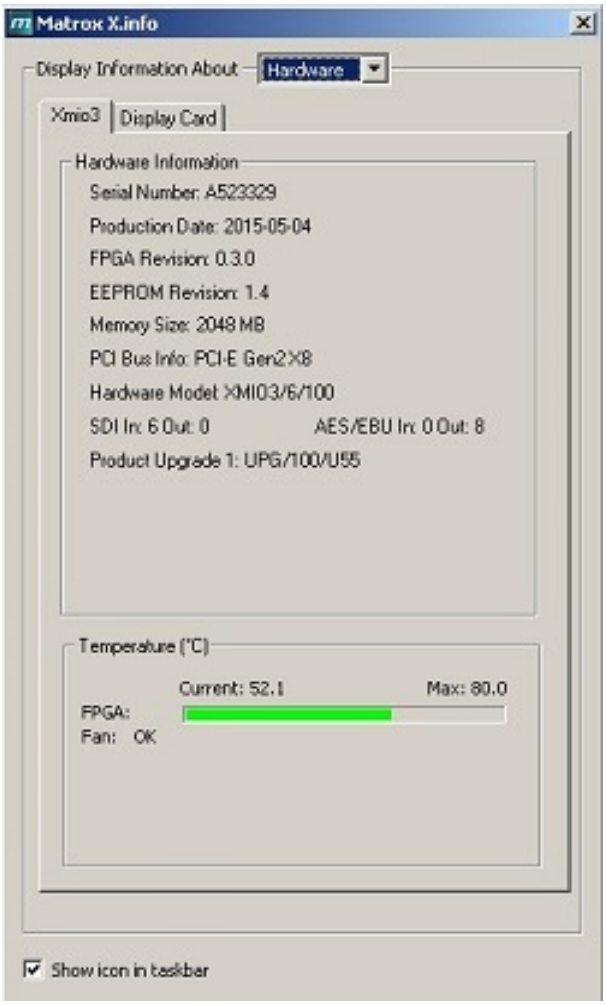
```
C:\Program Files\Matrox DSX.utils\drivers\mvDongleUpdater.exe upgrade
-sn="SerialNumberofthecard" -f="Path to the upgrade file"
```

The command above must be run from a Windows command line window (**WINDOWS BUTTON > cmd > ENTER**).

The Matrox X.info utility is used to display information about the card class, how the Input and Outputs are configured, serial number and more. This utility can normally be started from the Windows taskbar. A typical information window would be:

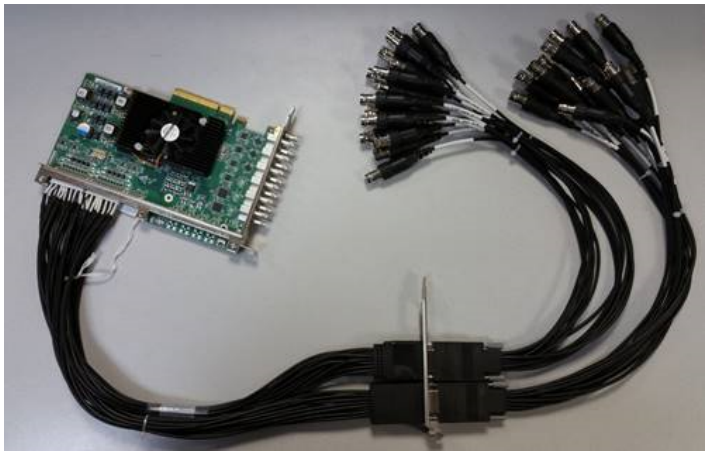
Copyright © 2026 Vizrt

X.mio3 AES Audio Kit

X.mio3 Class 100	X.mio3 Class 550
	

The X.mio3 AES Audio Kit connects to the X.mio3 card internally and takes up one slot as indicated in the picture below:

Copyright © 2026 Vizrt



The kit contains:

- Two internal cables with bracket.
- Two external cables, each cable has:
  - Eight AES/EBU in and eight AES/EBU out (total 16 AES/EBU in and out).
  - Four LTC in and four LTC out (total eight LTC in and out).

Upgrading Matrox Firmware for Two Sample Interleave

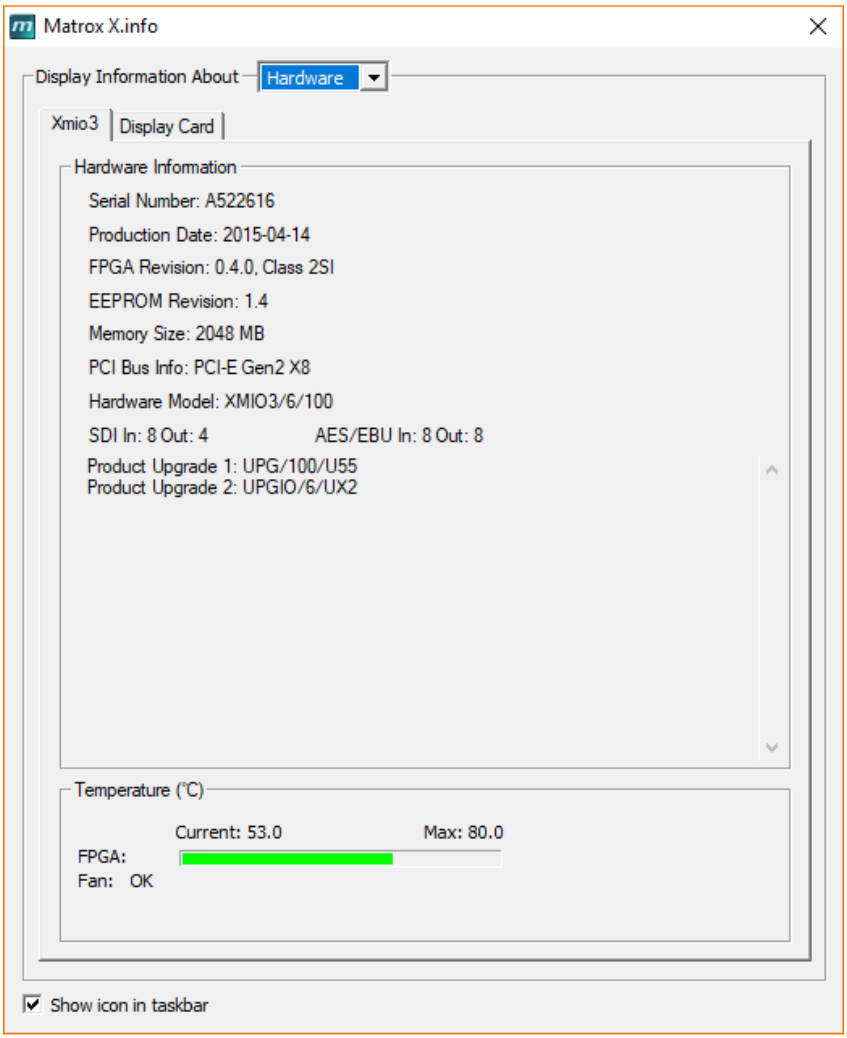
To make your card capable of two SI UHD:

- Check if you have an X.mio3 X2 series SDI card.
- Flash the firmware. Open a command line and run the following command:

```
cd C:\Program Files\Matrox DSX.utils\drivers
mvConnectorConfig.exe -2SI=on
```

- Restart the machine.
- Check *X.Info*, which should state that the card is now in 2SI mode.

Copyright © 2026 Vizrt



Refer to the connector mapping in the section above for 2SI, 2SI alternate connector mapping.

Copyright © 2026 Vizrt

Matrox X.mio5 IP (D25 / Q25)

The Matrox X.mio5 video board is the next generation IP card supporting SMPTE ST 2110-20/30/40 protocols. It is equipped with 25GbE SFP28 cages and is available in two models:

- X.mio5 Q25 - four (two dual) SFP28 cages, delivering an effective, fully redundant bandwidth of 50GbE, capable of providing four UHD inputs and outputs.
- X.mio5 D25 - two (one dual) SFP28 cages, delivering an effective, fully redundant bandwidth of 25GbE, capable of providing two UHD inputs and outputs.

Both models support 10GbE and 25GbE SFP28 transceivers. (The SFP+ form factor is identical to SFP28: SFP+ designates 10GbE, SFP28 designates 25GbE; both are compatible with the card.)



Copyright © 2026 Vizrt



Key Features (both models)

- Native SMPTE ST 2110 support with no CPU usage.
- ST 2110-21 packet pacing in hardware for narrow senders (Type N).
- Wide asynchronous receivers (Type A).
- Audio flows from one to 64 tracks.
- Multi-channel HDR conversion.
- On-board de-interlacing, scaling and compositing.
- Integrated hardware PTP for ST 2059-2.
- Built-in NMOS IS-04 and IS-05 support.

i Large Canvas Output: The Matrox X.mio5 D25 supports Large Canvas Output for SMPTE ST 2110-driven video walls, allowing custom non-broadcast canvas resolutions of up to 8192 × 4320 pixels, composed from up to eight progressive, broadcast-resolution cutout flows. The D25 supports a maximum of two UHD outputs. Appropriate output license(s) (4K, 8K, …) are required. Large Canvas Output is not supported on the X.mio5 Q25. See Large Canvas Output (Large Canvas Output for configuration details.

Per-model capacity

	Q25	D25
UHDp60/50 in & out	Up to four	Up to two
HD in & out (over 25GbE)	32	16

Copyright © 2026 Vizrt

	Q25	D25
Audio flows	256	128

General Notes

- An X.mio5 uses a **16x PCIe** slot (x8 electrically). It can't use the same slot as an X.mio3.
- To keep PTP stable, all Power Management options in Windows and BIOS must be **disabled** (Runtime power management disabled in BIOS settings).
- Each SFP of an X.mio5 board requires two IP addresses. This is mandatory. (One for the Matrox network adapter and one for genlock over IP.)
- NTSC and PAL resolutions are not supported by the SMPTE ST 2110 standard.
- UHD is only supported with 25GbE SFP cables. UHD is not possible with 10GbE SFP cables.
- On the Q25, the second SFP pair (SFP C/D) provides connectors starting at video 17 / audio 129 / ANC 33. See Explicit Usage of the Second SFP Pair on Matrox X.mio5 Q25. The D25 has only the first SFP pair (SFP A/B) and therefore no second pair.
- Ethernet packet capture (mvNetworkCapture) is not supported on the D25.

Matrox X.mio5 IP Configuration

Please see section SMPTE ST 2110 Configuration for further details.

Copyright © 2026 Vizrt

Matrox X.mio5 - DSX LE5 SDI

In this section:

- Supported Cards
  - X.mio5 X2
    - Key Features
    - Connector Mapping
  - X.mio5/8
    - Key Features
    - Connector Mapping
  - DSX LE5/8 Low Profile
    - Key Features
    - Connector Mapping
  - DSX LE5/4 Low Profile
    - Key Features
    - Connector Mapping
- Changing Connector Mapping
- Multi-link Channels
  - Dual Link - Fill + Key
  - Quad Link UHD
  - Quad Link UHD2

Matrox X.mio5 and DSX LE5 are two closely related series of PCI Express video boards with configurable SDI connectors. All connectors support HD and 3G SDI signals, and some support 12G as well. The main difference between the two series is that X.mio5 cards have an onboard video mixer to perform DVE effects, which DSX LE5 cards lack. The cards use either 8 or 16 PCI Express Gen 3 lanes.

Compared to their predecessors X.mio3 and DSX LE4, the series 5 cards lack several features:

- Watchdog functionality is not supported.
- SD formats are not supported.
- AES/EBU audio is not supported.

Important: Due to the increased bandwidth, it is important to use high quality cables and as few connection points as possible for 12G signals. It is not recommended to use converter cables from Mini-BNC to standard BNC.

Supported Cards

Viz Engine only supports a few cards from this series: X.mio5 X2, X.mio5/8, DSX LE5/4 LP and DSX LE5/8 LP. The IP models of these series are covered in Matrox X.mio5 IP (D25 / Q25) and Matrox DSX LE5 IP (D25 / Q25).

Copyright © 2026 Vizrt

X.mio5 X2

The Matrox X.mio5 X2 is the strongest card of the series with 12 SDI connectors, one Genlock and a powerful onboard compositor.

This card requires Viz Engine 4.4 or newer.



Key Features

- Up to four 12G inputs (texture) and four 12G outputs with four 3G reconfigurable I/O.
- 12 completely reconfigurable 3G I/O connectors.
- Up to three DVE inputs in UHD.

Connector Mapping

The pin files needed to change the configuration of SDI connectors are located in the sub folder *Xmio5* (see Section Changing Connector Mapping below). The table below shows all possible combinations. Connectors capable of **12G** signals are marked with bold text, while connectors limited to *3G* are in italic.



X.mio5 X2	Group 1				Group 2								
Connector	**0 in 12 out**	**1 in 11 out**	**2 in 10 out**	**3 in 9 out**	**4 in 8 out**	**5 in 7 out**	**6 in 6 out**	**7 in 5 out**	**8 in 4 out**	**9 in 3 out**	**10 in 2 out**	**11 in 1 out**	**12 in 0 out**

12	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	*In L*
11	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	*In K*	*In K*
10	**Out J**	**Out J**	**Out J**	**Out J**	**Out J**	**Out J**	**Out J**	**Out J**	**Out J**	**Out J**	*In J*	*In J*	*In J*
9	**Out I**	**Out I**	**Out I**	**Out I**	**Out I**	**Out I**	**Out I**	**Out I**	**Out I**	*In I*	*In I*	*In I*	*In I*
8	*Out H*	*Out H*	*Out H*	*Out H*	*Out H*	*Out H*	*Out H*	*Out H*	*In H*	*In H*	*In H*	*In H*	*In H*
7	*Out G*	*Out G*	*Out G*	*Out G*	*Out G*	*Out G*	*Out G*	*In G*	*In G*	*In G*	*In G*	*In G*	*In G*
6	*Out F*	*Out F*	*Out F*	*Out F*	*Out F*	*Out F*	*In F*	*In F*	*In F*	*In F*	*In F*	*In F*	*In F*
5	*Out E*	*Out E*	*Out E*	*Out E*	*Out E*	*In E*	*In E*	*In E*	*In E*	*In E*	*In E*	*In E*	*In E*
4	*Out D*	*Out D*	*Out D*	*Out D*	**In D**	**In D**	**In D**	**In D**	**In D**	**In D**	**In D**	**In D**	**In D**
3	*Out C*	*Out C*	*Out C*	**In C**	**In C**	**In C**	**In C**	**In C**	**In C**	**In C**	**In C**	**In C**	**In C**
2	*Out B*	*Out B*	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**
1	*Out A*	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**

Copyright © 2026 Vizrt

X.mio5/8

The Matrox X.mio5/8 has eight SDI connectors, one Genlock and an onboard compositor.

This card requires Viz Engine 5.0 or newer.



Key Features

- Up to four 12G inputs (texture) and four 12G outputs.
- Eight completely reconfigurable 3G I/O connectors.
- Up to two DVE inputs in UHD.

Connector Mapping

The pin files needed to change the configuration of SDI connectors are located in the sub folder *Xmio5lt* (see Section Changing Connector Mapping below). The table below shows all possible combinations. Connectors capable of **12G** signals are marked with bold text, while connectors limited to *3G* are in italic.



X.mio5/8	Group 1								
Connector	0 In 8 Out	1 In 7 Out	2 In 6 Out	3 In 5 Out	4 In 4 Out	5 In 3 Out	6 In 2 Out	7 In 1 Out	8 In 0 Out

Copyright © 2026 Vizrt

8	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	**Out L**	*In L*
7	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	**Out K**	*In K*	*In K*
6	**Out J**	**Out J**	**Out J**	**Out J**	**Out J**	**Out J**	*In J*	*In J*	*In J*
5	**Out I**	**Out I**	**Out I**	**Out I**	**Out I**	*In I*	*In I*	*In I*	*In I*
4	*Out D*	*Out D*	*Out D*	*Out D*	**In D**	**In D**	**In D**	**In D**	**In D**
3	*Out C*	*Out C*	*Out C*	**In C**	**In C**	**In C**	**In C**	**In C**	**In C**
2	*Out B*	*Out B*	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**
1	*Out A*	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**	**In A**

DSX LE5/8 Low Profile

The Matrox DSX LE5 LP/8 is a low profile card with eight I/O SDI connectors (four 12G and four 3G connectors) and one Genlock connector.

This card requires Viz Engine version 5.3.1 or later.



Key Features

- Eight freely configurable mixed 12G/3G SDI connectors.
- Low profile card.

Connector Mapping

The pin files needed to change the configuration of SDI connectors are located in the sub folder *Dsxle5lp* (see Section Changing Connector Mapping below). The table below shows all possible combinations. All connectors capable of handling **12G** signals are marked with bold text, while connectors limited to *3G* are in italic.

Important: If used for a fill/key signal the used connectors are paired based on their possible bandwidth. If only one connector of the pair is available fill/key I/O is not possible.

DSX LE5/8 LP	Group 1	Group 2	Group 3

Copyright © 2026 Vizrt

Connector	0 In 8 Out	1 In 7 Out	2 In 6 Out	3 In 5 Out	4 In 4 Out	5 In 3 Out	6 In 2 Out	7 In 1 Out	8 In 0 Out
8	**Out H**	**Out H**	**Out H**	**Out H**	**Out H**	**Out H**	**Out H**	**Out H**	**In H**
7	*Out G*	*Out G*	*Out G*	*Out G*	*Out G*	*Out G*	*In G*	*In G*	*In G*
6	**Out F**	**Out F**	**Out F**	**Out F**	**Out F**	**Out F**	**Out F**	**In F**	**In F**
5	*Out E*	*Out E*	*Out E*	*Out E*	*Out E*	*In E*	*In E*	*In E*	*In E*
4	**Out D**	**Out D**	**In D**	**In D**	**In D**	**In D**	**In D**	**In D**	**In D**
3	*Out C*	*Out C*	*Out C*	*Out C*	*In C*	*In C*	*In C*	*In C*	*In C*
2	**Out B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**	**In B**
1	*Out A*	*Out A*	*Out A*	*In A*	*In A*	*In A*	*In A*	*In A*	*In A*

DSX LE5/4 Low Profile

The Matrox DSX LE5 LP/4 is a low profile card with four I/O SDI connectors and one Genlock.

This card requires Viz Engine version 5.3.0 or later.



Key Features

- Four freely configurable 12G SDI connectors.
- Low profile card.

Connector Mapping

The pin files needed to change the configuration of SDI connectors are located in the sub folder *Dsxle5lp* (see Section Changing Connector Mapping below). The table below shows all possible combinations. All connectors are capable of handling **12G** signals and are marked with bold text.

DSX LE5/4 LP	Group 1

Copyright © 2026 Vizrt

Connector	0 In 4 Out	1 In 3 Out	2 In 2 Out	3 In 1 Out	4 In 0 Out
4	Out F	Out F	Out F	Out F	In F
3	Out E	Out E	Out E	In E	In E
2	Out B	Out B	In B	In B	In B
1	Out A	In A	In A	In A	In A

Changing Connector Mapping

The I/O configuration can be changed by using a read-only configuration file located in the folder ***Xmio5Le5ConnectorMapping***, where the Matrox drivers are installed (normally under *C:\Program Files\Matrox DSX-TopologyUtils\drivers*).

1. Make sure that neither Viz Engine nor any other processes are using the card.
2. From the drivers folder that contains *mvConnectorConfig.exe*, open the command prompt (administrator privileges are requested).
3. Call `mvConnectorConfig.exe load -f="C:\path\to\file.pin" -sn=Axxxxxx`. Replace `Axxxxxx` with the serial number of the card you want to configure.

Pin files for different card models are located in different sub folders of `Xmio5Le5ConnectorMapping`. The name of each file indicates the number of inputs and outputs. Provide the full path, within quotation marks, to the desired file. Using relative paths does not work and can flash the card to an invalid status. If this happens, flash the card again with the absolute path.

Example: To configure an X.mio5 X2 with serial number A123456 to use the first **four** connectors as inputs and the other **eight** as outputs, use the command: `mvConnectorConfig.exe Load -f="C:\Program Files\Matrox DSX-TopologyUtils\drivers\Xmio5Le5ConnectorMapping\Xmio5\xmio5_x2_04i08o.pin" -sn=A123456`

Information: On some cards, the connector mappings are split into different groups, as marked in the respective tables. When changing between configurations inside the same configuration group, the command should take one to two seconds to process the new configuration and a restart of the computer is not needed.
When the change of configuration involves going from one configuration family to the other, this will take some time, since a firmware upgrade takes place (empirically five to ten minutes) and it requires a system restart (command line waits for user input on whether the restart should happen now or not).

Multi-link Channels

Input or output channels of Viz Engine require multiple SDI connectors if they include a key signal, use 3G quad link, or both.

Copyright © 2026 Vizrt

Dual Link - Fill + Key

For channels that use a single fill connector (including 12G UHD), the easy rule to determine the associated key connector is "Fill A Key B", "Fill C Key D", and so on. This applies to both input and output channels. The main connector of a channel is always the fill connector. A channel that is mapped to connector B, D, and so on, can not have a key signal.

Exception: on DSX LE5/8 LP the pair is the connector two positions later, so fill A/B/E/F take key C/D/G/H respectively; connectors C, D, G and H cannot carry a fill with key.

Quad Link UHD

Viz Engine only allows quad 3G configurations on connectors A, E and I (both for inputs and outputs). If one of those connectors is used as the base connector of a quad link channel, the next three connectors are automatically assigned to that channel and no longer available. For example, if connector A is chosen, it is the first one in the set and B, C and D are automatically used. Same for E (F, G, H) and I (J, K, L).

Exceptions:

- **DSX LE5/4 LP** only has connectors A, B, E, and F. Only connector A is available as the base connector of a quad-link channel and the remaining connectors are used in order.
- **DSX LE5/8 LP** has no connector I, so only A and E are available as base connectors. In addition, as soon as at least one input connector is configured, E is the only base connector available for a quad-link **output**.

Each quad-link connector can be configured to use either square division or 2SI. See Multiconnector Mode under Video Output for outputs, and Video Input: Live Input for inputs.

It is possible to combine quad link with a key signal. In that case, there are only two possible connector mappings on X.mio5 X2, assuming that the connectors are available: If fill uses [A,B,C,D] then key automatically uses [E,F,G,H]. If fill uses [E,F,G,H], then key automatically picks [I,J,K,L]. On X.mio5/8, there is only one option: Fill uses [A,B,C,D] and key uses [I,J,K,L]. On DSX LE5/8 LP there is also only one option, but a different one: fill uses [A,B,C,D] and key uses [E,F,G,H]. On DSX LE5/4 LP, quad link cannot be combined with a key signal, because the card does not have enough connectors.

Information: On X.mio5/8 and DSX LE5/8 LP the key always uses the second group of four connectors. The mappings look different only because the two cards label those connectors differently (I-L versus E-H).

Note: For outputs, multiconnector mode None (one single 12G connector) can only be selected on connector I, or on connector A of a DSX LE5/4 LP configured with four outputs. On any other base connector, only Square Division and 2SI are offered.

Quad Link UHD2

UHD2 requires an X.mio5/X2, and the card is limited to one input and one output (fill only) quad link (4x12G) 2SI. The input uses connectors A-D, and the output uses connectors I-L. The only supported frame rate is 50 FPS.

Hardware DVE does not work due to performance limits and needs to be disabled in the configuration.

Both a latest generation workstation and a high end GPU are required (NVIDIA 6000 Ada is recommended).

Copyright © 2026 Vizrt

Matrox Driver Installation

The first time a machine is started with a Matrox board or dongle installed, the operating system prompts the user to install the necessary Matrox drivers. Before installing a new driver, or upgrading the existing drivers, any currently installed drivers must be removed, see To Remove the Matrox Driver below.

Please pay attention to the following considerations:

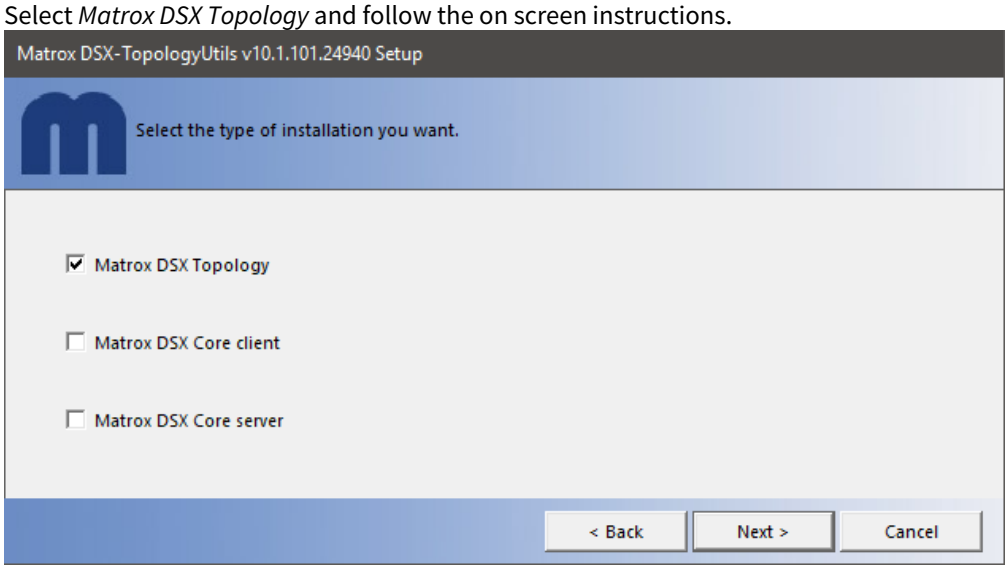
- A new driver should not be installed before the new hardware is installed.
- The use of a different driver version than that which was shipped with the video board is not recommended.
- In most cases, using a driver version that is not recommended causes the system to be unresponsive.
- Use the supplied driver installer application and do not rely on drivers automatically installed by the operating system.
- Please refer to the Drivers and Configuration History section for information on which driver version is recommended for the Matrox video card and Viz Engine version in use. As there are many driver versions available, it is important to compare the versions reported by the Matrox X.info utility to check that the driver and firmware versions match. A warning is displayed if the versions do not match.

This section contains information on the following topics:

- To Install the Matrox Driver
- To Remove the Matrox Driver
- To Upgrade the Matrox Driver
    - Important:
- To Check the Installation
- To Check the Installation with Windows Device Manager

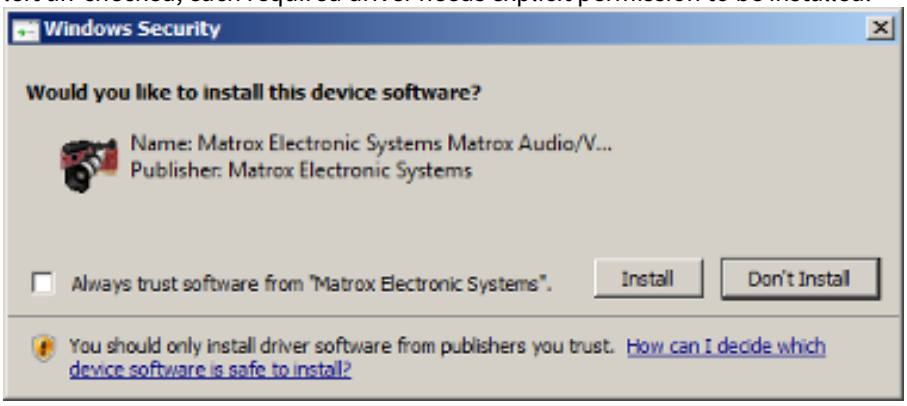
To Install the Matrox Driver

1. Locate the latest Matrox driver (DSX-Topology Utils): *ftp://ftp.vizrt.com/products/Vizrt%20Drivers/Matrox/Xmio/<driver>/*.
2. Download and save the installation file to the local hard drive.
3. Open the downloaded *DSX-TopologyUtils.exe* file. This file automatically extracts the installation files and launches the installation wizard.
4. Select *Matrox DSX Topology* and follow the on screen instructions.

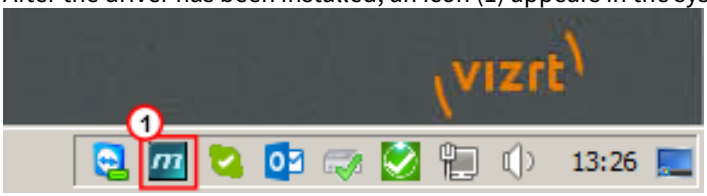


Copyright © 2026 Vizrt

5. In case the Windows Security window pops up, tick the **Always trust software from Matrox Electronic Systems** check-box. This allows for a faster installation procedure in any subsequent driver installations. If left un-checked, each required driver needs explicit permission to be installed.



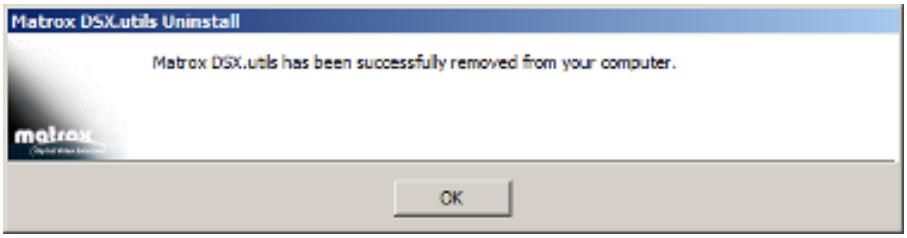
6. Click **Install** to start the installation. The installation can take several minutes.
7. After the driver has been installed, an icon (1) appears in the system tray.



Note: The firmware upgrade now starts. This can take a few minutes to complete.

To Remove the Matrox Driver

1. Go to **Start > Control Panel > Programs and Features**.
2. Locate the *Matrox DSX.utils <version>*.
3. Click the **Uninstall** button, or right click the entry and select **Uninstall** from the context menu.
4. Follow the on screen instructions.



5. Reboot the system.

Tip: In some cases, the Matrox driver removal process automatically reboots the system.

Copyright © 2026 Vizrt

To Upgrade the Matrox Driver

Important:

When upgrading from 10.4 to 11.x Matrox driver, follow these mandatory steps:

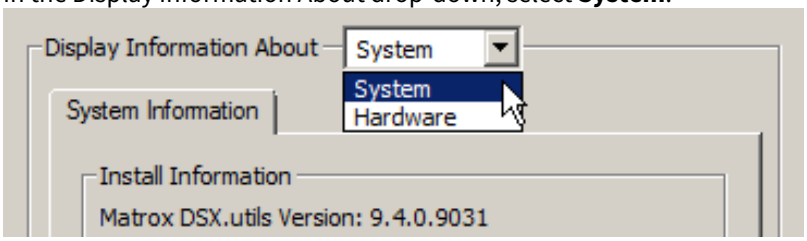
- Uninstall old Matrox driver
- Cold Boot (Turn off computer for at least 20 seconds)
- Switch Computer on
- Continue installation of new driver

To Check the Installation

1. Click the Matrox system tray icon.
2. Select **Open X.info**.

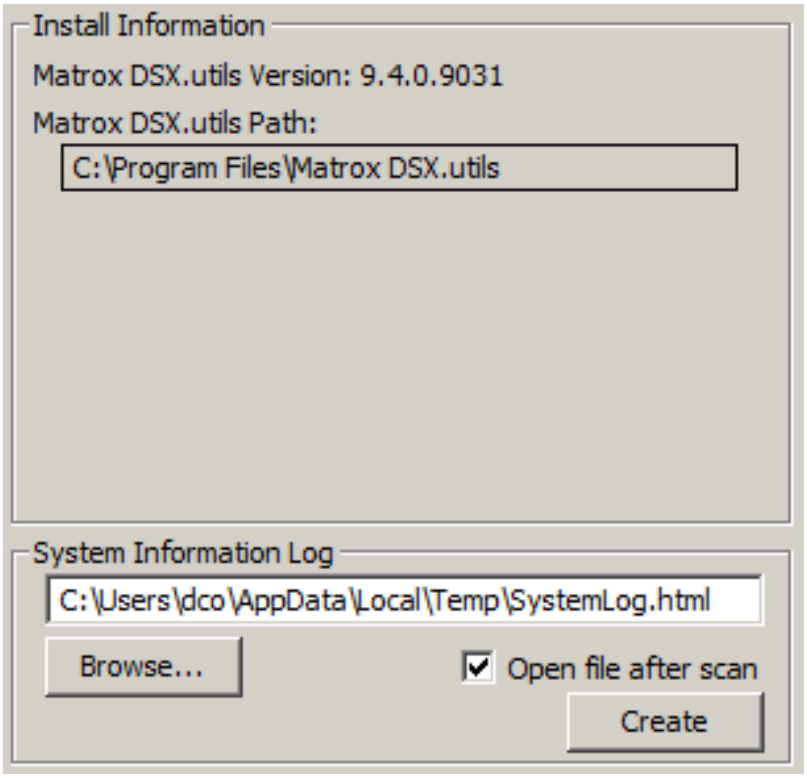


3. In the Display Information About drop-down, select **System**.



4. Check that the correct driver version was installed.

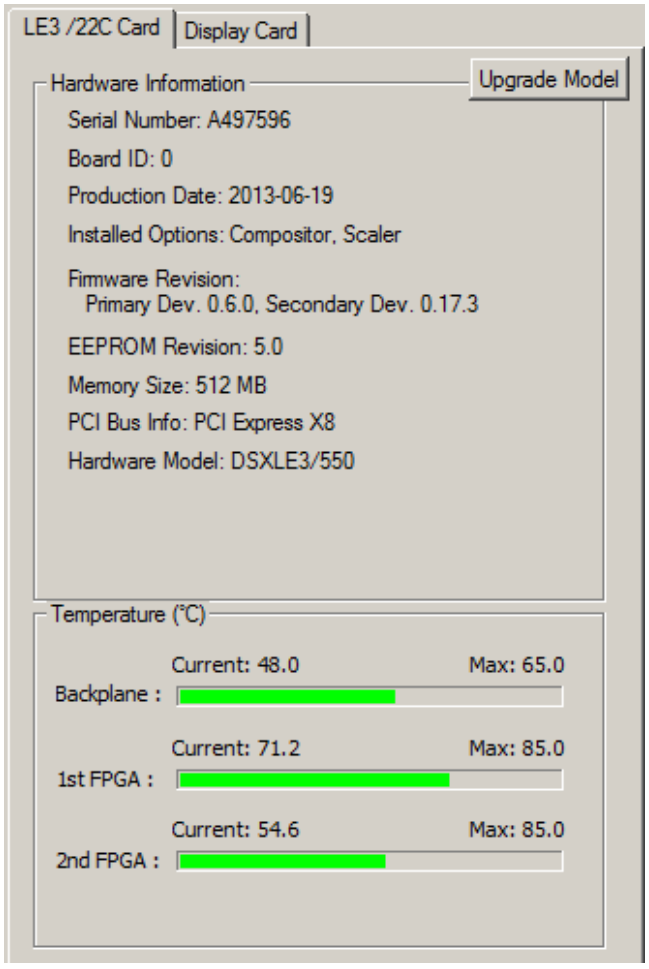
Copyright © 2026 Vizrt



5. In the **Display Information About** drop-down, select **Hardware** and click on the installed card tab, for example **LE3/22C**.

Note: The information shown in the panel can look different depending on the installed Matrox card.

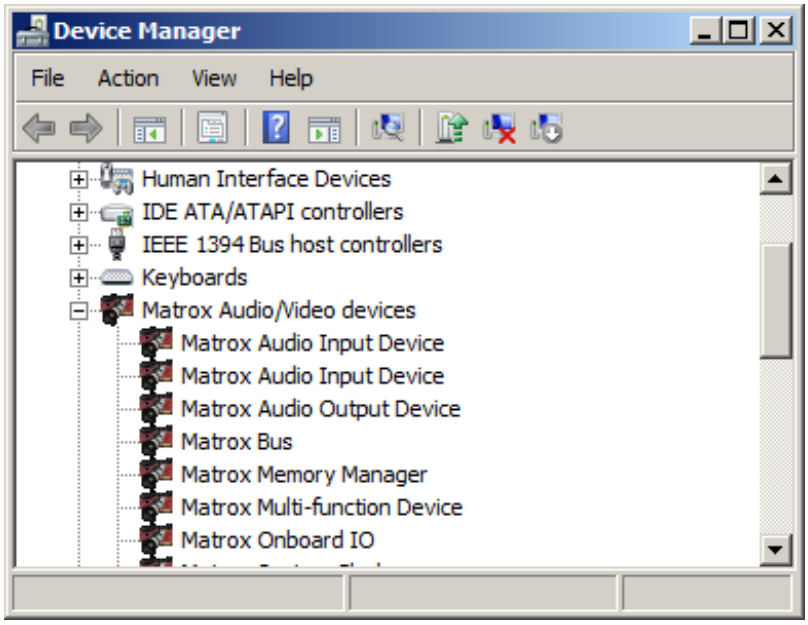
Copyright © 2026 Vizrt



6. Check the these details:
    a. Under Hardware Information, check the Installed Options for Compositor, to confirm that the board is taking advantage of the on-board compositor. Some boards do not have an on-board compositor. The Matrox X.mio2 Plus does have an on-board compositor, even though this is not listed.
    b. See PCI Bus Info and check that the board was put into a PCIe slot operating at the right frequency.
    c. See Hardware Model, for example, X.mio2 Plus 8500, to check for Mixed Mode Video Support.

Copyright © 2026 Vizrt

To Check the Installation with Windows Device Manager



1. Start the **Device Manager**, by either:
    a. Click **Start**, then **Run**, and type `devmgmt.msc` , or
    b. Right-click **My Computer**, select **Manage** and then **Device Manager**, or
    c. Right-click **My Computer**, select **Properties** and click the **Hardware** tab, then click **Device Manager**, or
    d. Run the command star `devmgmt.msc` from the **Command Prompt**
2. Expand the **Matrox Audio/Video devices** node to see the devices that are installed.

Copyright © 2026 Vizrt

Troubleshooting Matrox Video Hardware

It might be necessary to reset the Matrox board. This needs to be done for example when Viz Engine reports:

- `"0xefac9019: Generic driver error: The requested operation was unsuccessful" at startup.`
- `"0xeddd8008: Topology driver error: Watchdog specified already in use by another application." at startup.`
- `"VideoClipInOut::SequenceVideoOut SequenceVideoOut no output node." during operation.`

For this a reset mechanism has been implemented for the Viz Engine. To activate this mechanism, do this procedure:

1. Stop all Viz Engines (in dual channel setups both Viz Engines must be stopped).
2. Open the Viz Configuration file.
3. Set `Matrox0.ResetTopology = 1`.
4. Start up the respective Viz Engines.

> **Note:** The `Matrox0.ResetTopology` setting is automatically set back to `0` after the reset has been performed.

The reset feature also clears the on-board memory of the Matrox board. This operation removes all Matrox topologies from the on-board memory. After the reset, everything is restored automatically according to the configuration settings. This also applies to both Viz Engines in a Dual Channel setup.

Disable Warmboot

Matrox boards do not support Warmboot or Fastboot options. Please make sure the registry entries for `HiberbootEnabled: Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Power` and `HibernateEnabled: Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Power` are both set to be deactivated.

Replacing a Video Board

This procedure describes how to safely replace a video board on a system with an existing video board, or a system that previously has had a video board installed.

1. Remove the video board drivers.
2. Shut down the machine.
3. If present, remove the currently installed video board.
4. Install the replacement video board.
5. Install the video board drivers.

Copyright © 2026 Vizrt

IMPORTANT! When changing a Matrox video board the *Matrox.Devices* setting, set in the Viz Config file's `SECTION MATROX_CONFIG`, is not updated. The Matrox support is not correct and the new board does not work. In this case, the setting must be removed and Viz Engine restarted such that Viz Engine can insert the new serial number.

Copyright © 2026 Vizrt

Matrox DSX LE6

The Matrox DSX LE6 video board is the next generation IP card supporting SMPTE ST 2110-20/30/40 protocols. It is equipped with two (one dual) QSFP28 cages in which to connect QSFP28 transceivers. Only 100GbE transceivers are supported.



Key Features

- Native SMPTE ST 2110 supports over 100 GbE with no CPU usage.
- Up to eight UHDp60/50 inputs and outputs over 100GbE.
- ST 2110-21 packet pacing in hardware for narrow senders (Type N).
- Wide asynchronous receivers (Type A).
- 16x HD inputs and outputs over 100GbE.
- 128 audio flows from one to 64 tracks.
- Integrated hardware PTP for ST 2059-2.
- Built in NMOS IS-04 and IS-05 support.
- UHD-2 support will come with future Matrox driver versions.

General Notes

- Please be aware that a DSX LE6 uses a **16x PCIe** slot. It cannot use the same slot as an X.mio3.
- To make sure the PTP is stable, all Power Management options in Windows and BIOS need to be **disabled** (*Runtime power management* disabled in BIOS settings).
- Each SFP of a DSX LE6 board requires two IP addresses. This is mandatory.
- NTSC and PAL resolutions are not supported by the SMPTE ST 2110 standard.

Copyright © 2026 Vizrt

Matrox DSX LE5 IP (D25 / Q25)

The Matrox DSX LE5 video board is an IP card supporting SMPTE ST 2110-20/30/40 protocols. It is equipped with 25GbE SFP28 cages and is available in two models:

- DSX LE5 Q25 — four (two dual) SFP28 cages, delivering an effective, fully redundant bandwidth of 50GbE, capable of providing four UHD inputs and outputs.
- DSX LE5 D25 — two (one dual) SFP28 cages, delivering an effective, fully redundant bandwidth of 25GbE, capable of providing two UHD inputs and outputs.

Both models support 10GbE and 25GbE SFP28 transceivers. (The SFP+ form factor is identical to SFP28: SFP+ designates 10GbE, SFP28 designates 25GbE; both are compatible with the card.)

Key Features

- Native SMPTE ST 2110 support with no CPU usage.
- ST 2110-21 packet pacing in hardware for narrow senders (Type N).
- Wide asynchronous receivers (Type A).
- Integrated hardware PTP for ST 2059-2.
- Built-in NMOS IS-04 and IS-05 support.
- Genlock over IP (one genlock-over-IP setting per SFP, conforming to SMPTE ST 2059-2).

Per-model capacity

	DSX LE5 Q25	DSX LE5 D25
SFP28 cages	Four (two dual) — A/B/C/D	Two (one dual) — A/B
Effective redundant bandwidth	50 GbE	25 GbE
UHDp60/50 in & out	Up to four	Up to two
HD in & out (over 25GbE)	32	16
Audio flows	256	128
Ancillary-data connectors	1-64	1-32
Status LEDs	4	2
Power (with active transceivers)	38 W	34 W
SMPTE compliance	-10/-20/-21/-22/-30/-31/-40, ST 2059-2, ST 2022-7	identical

Copyright © 2026 Vizrt

General Notes

- The DSX LE5 is a full-height, ½-length PCIe Gen3 x16 card (x8 electrically) and uses a **16x PCIe** slot. It can't use the same slot as an X.mio3.
- To keep PTP stable, all Power Management options in Windows and BIOS must be disabled (Runtime power management disabled in BIOS settings).
- Each SFP of a DSX LE5 board requires two IP addresses (one for the Matrox network adapter and one for genlock over IP). This is mandatory.
- NTSC and PAL resolutions are not supported by the SMPTE ST 2110 standard.
- UHD is only supported with 25GbE SFP cables; it is not possible with 10GbE SFP cables.
- On the Q25, the second SFP pair (SFP C/D) provides connectors starting at video 17 / audio 129 / ancillary 33. See Explicit Usage of the Second SFP Pair. The D25 has only the first SFP pair (SFP A/B).
- Changing a transceiver's speed (10GbE ⟷ 25GbE), or fitting a 25GbE transceiver where none was present, requires a system restart (default link speed is 10GbE when no transceiver is connected).

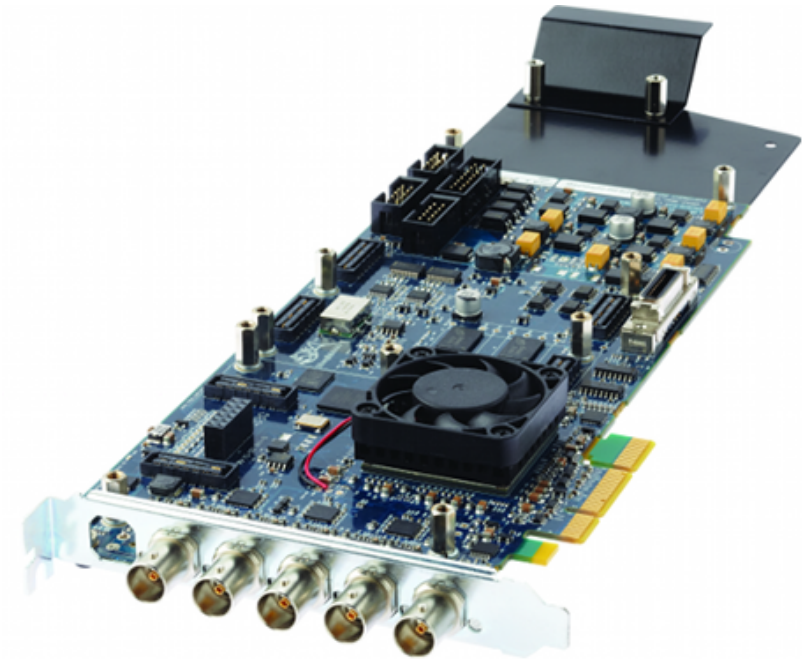
i Large Canvas Output: The Matrox DSX LE5 (both D25 and Q25) supports Large Canvas Output for SMPTE ST 2110-driven video walls, with custom non-broadcast canvas resolutions up to 8192 × 4320 pixels composed from up to eight progressive, broadcast-resolution cutout flows. Appropriate output license(s) (4K, 8K, …) are required. See Large Canvas Output (Large Canvas Output for configuration details.

Matrox DSX LE5 IP Configuration

Please see section SMPTE ST 2110 Configuration for further details.

Copyright © 2026 Vizrt

6.4.3 BlueFish444



Viz Engine supports the following BlueFish444 video boards:

- Epoch | 4K Supernova
- Epoch | 4K Supernova S+
- Epoch | Neutron (introduced in Viz version 3.7.0)
- Kronos | K8 (introduced in Viz version 5.0.0)

Some of the main supported features are:

- Embedded audio input and output.

> **Note:** All 16 channels are used; however, there are currently no configuration options available for these channels (for example, routing of channels, enable/disable channels etc.).

- ANC data (both HANC and VANC), such as VITC and RP188 time codes, is supported. Data from the input is laid over the output. Ingestion of time codes from time code reader boards is supported as well.
- Two video inputs and outputs and one Genlock.
- Automatic bypass (aka Special Configuration Options for Bluefish444) of video if the Viz Engine crashes.
- Constant delay of four frames from input to output.

> **Note:** Other BlueFish444 and Digital Voodoo boards are no longer supported.

Copyright © 2026 Vizrt

Configuration History for BlueFish444

Use the driver version available on the Vizrt FTP server to make sure of correct functionality and ringbuffer support.

The following driver and firmware versions are supported:

Viz Release	Driver Version	Supernova Firmware	Neutron Firmware	Kronos K8 Firmware
5.5.x	6.6.1.4	2i2o 162 0i4o 162	1i2o V35 3i V32 3o V13 ASI V31	V1.6.12
5.0 - 5.4.x	6.5.1.22	2i2o 162 0i4o 162 (1)	1i2o 35	4i4o 1.4 build 41
4.0, 4.1, 4.2, 4.3, 4.4	5.11.0.45	2i2o 145	1i2o 027	N/A
3.9.0 - 3.14.x	5.11.0.45	2i2o 145	1i2o 027	N/A
3.8.1 - 3.8.2	5.11.0.39	2i2o 134	1i2o 025	N/A
3.8.0	5.11.0.25	2i2o 127	1i1o 067	N/A
3.7.1	5.11.0.14	2i2o 108	1i1o 054	N/A
3.7.0	5.11.0.7	2i2o 096 Rev. 2	1i1o 016 Rev. 2	N/A
3.6.4	5.11.0.3	2i2o 096	N/A	N/A
3.6.3	5.10.2.18	2i2o 094	N/A	N/A
3.6.1 - 3.6.2	5.10.2.4	2i2o 053	N/A	N/A
3.5.1 - 3.5.4	5.10.1.11	2i2o 031	N/A	N/A
3.5.0	5.9.0.78	442	N/A	N/A

(1) 5.3 and later

Copyright © 2026 Vizrt

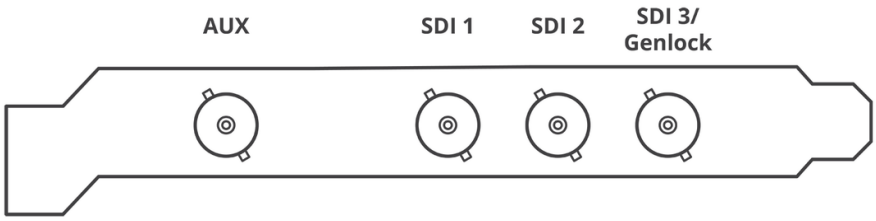
Epoch Neutron



The Epoch | Neutron is a full-length PCIe card with three bi-directional multi-format I/O BNC connectors that support dual link configurations. For use with Viz Engine, the card needs to be configured with one input and one output. Of the three BNC connectors, the first two do SD, HD1.5G, 3G, ASI, AES and LTC as either input or output. The third is used for Genlock.

Tip: The Epoch | Neutron is capable of downstream keying. To achieve this, add a Media Asset as DVE when creating a scene in Viz Artist.

Epoch | Neutron Connectors



On the Epoch | Neutron, the default connector configuration for the four connectors are:

- AUX connector (Not Used)
- (empty space)
- Output one (fill)
- Input one

Copyright © 2026 Vizrt

- Reference IN

If the `dual_link_output` flag is set to `1` in the Viz configuration file, the key output is enabled on the bottom BNC connector. The Genlock signal gets moved to the AUX connector at the top.

- Reference In
- (empty space)
- Output one (fill)
- Input one
- Output two (key)

Please refer to the Special Configuration Options section for more information.

See Also

- Configuration History for BlueFish444
- Special Configuration Options
- Video Board
- Media Assets as DVE in the Viz Artist Guide

Copyright © 2026 Vizrt

Epoch Supernova

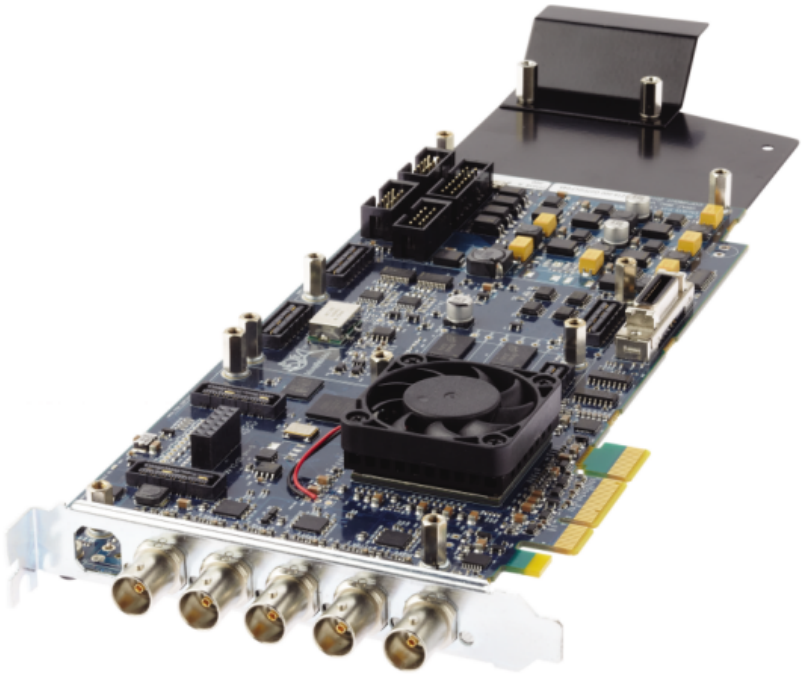
The BlueFish444 Epoch | 4K Supernova, Epoch | 4K Supernova S+ and Epoch | Supernova CG boards are intended for character generators such as Viz Trio, but can also be used as an alternative where only two inputs and outputs are required.

> **Note:** When the 0in4out firmware is used on the Epoch Supernova cards, two instances of Viz Engine can run simultaneously (Dual Channel setup). The assignment of fill/key connectors is automatic and cannot be changed in this case.

This section contains information on the following:

- Epoch | 4K Supernova
  - Epoch | 4K Supernova Connectors
- Epoch | 4K Supernova S+
  - Epoch | 4K Supernova S+ Connectors
- Epoch | Supernova CG
  - Epoch | Supernova CG Connectors

Epoch | 4K Supernova

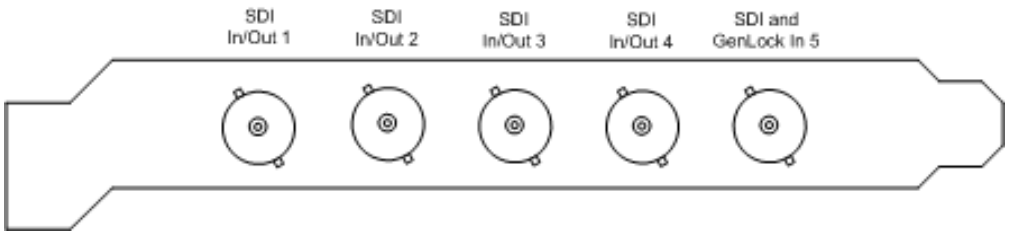


The Epoch | Supernova is a full-length PCIe card with five bi-directional multi-format I/O BNC connectors that support a number of configurations with up to two dual link channels or four single link channels. For use with Viz Engine, the card needs to be configured with two inputs and two outputs.

Copyright © 2026 Vizrt

Epoch | 4K Supernova Connectors

With five BNC connectors, each of the first four do SD, HD, 3G, ASI, AES and LTC as either input or output. The fifth is the designated GenLock or SD, HD, 3G, ASI or AES (no LTC).

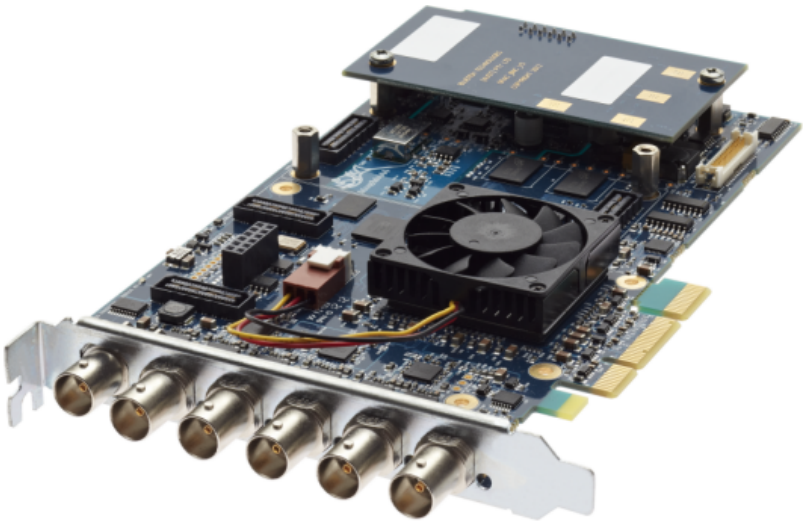


On the Epoch | 4K Supernova, the default connector configuration for the five connectors are:

- Output one
- Input one
- Output two
- Input two
- Reference In (Genlock)

If the `dual_link_input` flag is set to `1` in the Viz Configuration file, the two inputs are treated as one channel with fill on the first and key on the second. Please refer to the Special Configuration Options for Bluefish444 section for more information.

Epoch | 4K Supernova S+

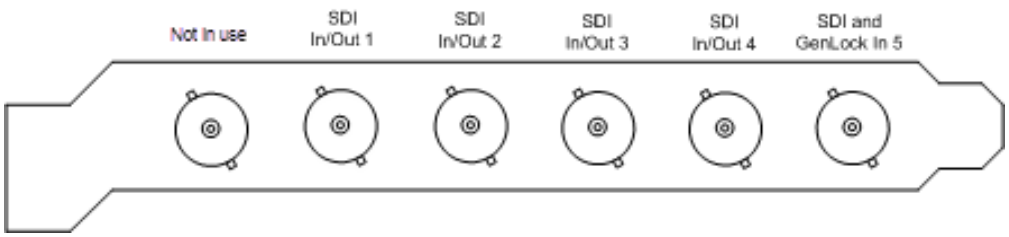


The Epoch | Supernova S+ is a 2/3-length PCIe card with bi-directional multi-format I/O BNC connectors that support a number of configurations, up to two dual link channels or four single link channels. For use with Viz Engine, the card needs to be configured with two inputs and two outputs. With six BNC connectors, each of the first

Copyright © 2026 Vizrt

four do SD, HD, 3G, ASI, AES and LTC as either input or output. The fifth is the designated GenLock or SD, HD, 3G, ASI or AES (no LTC).

Epoch | 4K Supernova S+ Connectors



On the Epoch | 4K Supernova S+, the default connector configuration for the six connectors are:

- *Not in use*
- Output one
- Input one
- Output two
- Input two
- Reference In (Genlock)

Note: Viz Engine does not utilize the top BNC connector on the Epoch | 4K Supernova S+ video card, recognized as not being flush with the other five.

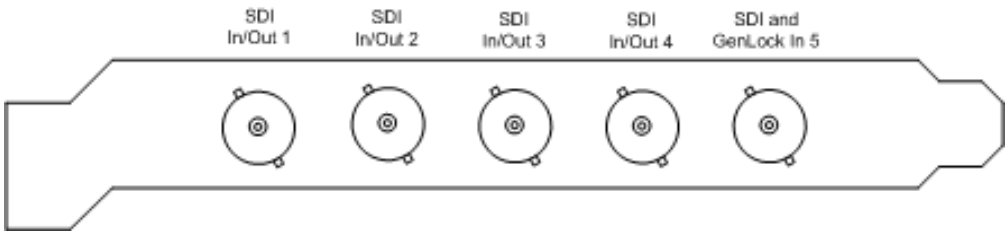
If the `dual_link_input` flag is set to `1` in the Viz Configuration file, the two inputs are treated as one channel with fill on the first and key on the second. Please refer to the Special Configuration Options for Bluefish444 section for more information.

Epoch | Supernova CG



The Epoch | Supernova CG is a 2/3-length PCIe card with bi-directional multi-format I/O BNC connectors that support a number of configurations, up to two dual link channels or four single link channels. For use with Viz Engine, the card needs to be configured with two inputs and two outputs. With five BNC connectors, each of the first four do SD, HD, 1.5G, ASI, AES and LTC as either input or output. The fifth is the designated GenLock or SD, HD, 1.5G, ASI or AES (no LTC).

Epoch | Supernova CG Connectors



On the Epoch | Supernova CG, the default connector configuration for the five connectors are:

- Output one
- Input one
- Output two
- Input two
- Reference In (Genlock)

If the `dual_link_input` flag is set to `1` in the Viz Configuration file, the two inputs are treated as one channel with fill on the first and key on the second. Please refer to the Special Configuration Options for Bluefish444 section for more information.

See Also

- Configuration History for BlueFish444
- Special Configuration Options for Bluefish444
- Video Board

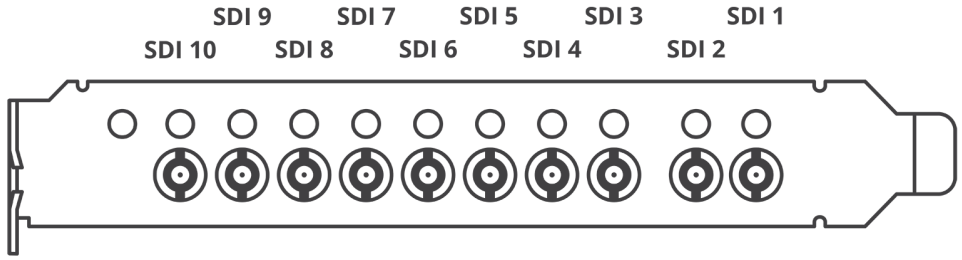
Copyright © 2026 Vizrt

Kronos



The Kronos K8 is a full-length PCIe card with ten HDBNC connectors. Eight of those are used for SDI IO, the other two are reference connectors. For use with Viz Engine, the card needs to be configured with four inputs and four outputs. The connectors support SD, HD 1.5G, 3G, ASI, AES and LTC as either input or output. In addition the card has a reference input and a reference output connector.

Kronos K8 Connectors



SDI Connector	Function
1	Reference In
2	Reference Out
3	Output Fill

Copyright © 2026 Vizrt

SDI Connector	Function
4	Input 1
5	Output Key
6	Input 2
7	-
8	Input 3
9	-
10	Input 4

Copyright © 2026 Vizrt

Special Configuration Options for Bluefish444

This section describes the procedures to configure the following options when running a Bluefish444 setup:

- To Enable Dual Link Input for Epoch | 4K Supernova and Epoch | 4K Supernova S+ Boards
- To Enable Dual Link Output for Epoch | Neutron
- To Enable Automatic Bypass for BlueFish444 Epoch Boards

To Enable Dual Link Input for Epoch | 4K Supernova and Epoch | 4K Supernova S+ Boards

1. Stop Viz Engine.
2. Open the **Viz Configuration file** (for example: *VIZENGINE-0.cfg*).

> **Note:** The default location for this file is *%ProgramData%\Vizrt\VizEngine*.

3. Locate `SECTION VIDEO`.
4. Enable the **Dual Link Input** setting by changing the default value `0` to `1`: `dual_link_input = 1`.
5. Save the file.
6. Start Viz Engine.

To Enable Dual Link Output for Epoch | Neutron

1. Stop Viz Engine.
2. Open the **Viz Configuration file** (for example: *VIZENGINE-0.cfg*).

> **Note:** The default location for this file is *%ProgramData%\Vizrt\VizEngine*.

3. Locate `SECTION VIDEO`.
4. Enable the **Dual Link Output** setting by changing the default value `0` to `1`: `dual_link_output = 1`.
5. **Save** the file.
6. Start Viz Engine.

To Enable Automatic Bypass for BlueFish444 Epoch Boards

Viz Engine supports automatic mechanical bypass (copper-to-copper) of video for BlueFish444 Epoch boards. To use this feature enable the feature.

1. Stop Viz Engine.
2. Open the **Viz Configuration file**
3. Locate `SECTION VIDEO`.
4. Enable the Watchdog setting by changing the default value `0` to `1`: `video_use_watchdog = 1`.
5. **Save** the file.
6. Start Viz Engine.

See Also

- Configuration History for BlueFish444

Copyright © 2026 Vizrt

- Video Board

Copyright © 2026 Vizrt

6.4.4 Aja Hardware

Viz Engine with AJA Hardware supports up to two 3G-SDI inputs and fill/key output.

This section contains information on the following topics:

- Overview and Connectors
- Configuration
- Audio Configuration
- Timecode
- Video Configuration
- AJA KONA IP (J2K)
- Configuration Utilities
- Firmware Upgrade

Overview and Connectors

AJA boards support current SD and HD SDI workflows with frame rates up to 50p/59.94p. Audio is supported as embedded signal in the SDI stream.

Configuration

IMPORTANT! For 1080p, only 3G Level A is supported.

Audio Configuration

Embedded audio is supported, always with 16 channels in and out. However, mixer settings are active, and respected. Example: if the user requests only two-channel output, the remaining 14 channels are silent.

Timecode

Viz Engine reads RP188 timecode embedded in the SDI input signal. Two conditions are evaluated from the board's RP188 DBB register: whether timecode is present in the input signal, and whether the board is configured to take the output timecode directly from the input. If the latter bypass is active, Viz Engine disables it so that the output carries Viz Engine's own timecode.

Video Configuration

The default configuration is:

Live Input 1	SDI 1
Live Input 2	SDI 2
Fill Output	SDI 3

Copyright © 2026 Vizrt

Key Output	SDI 4
Genlock	[REF IN]

If Live Input 2 is inactive in the configuration, SDI 2 is enabled for preview output.

Info: For 1080p, only one input is supported.

AJA KONA IP (J2K)

Viz Engine also detects AJA KONA IP boards with JPEG 2000 firmware and configures them automatically. Two variants are recognized:

- 1 input / 1 output (KONAIP_1RX_1TX_1SFP_J2K): SDI 1 is the input, SDI 2, 3 and 4 are outputs.
- 0 inputs / 2 outputs (KONAIP_2TX_1SFP_J2K): all four SDI connectors are outputs.

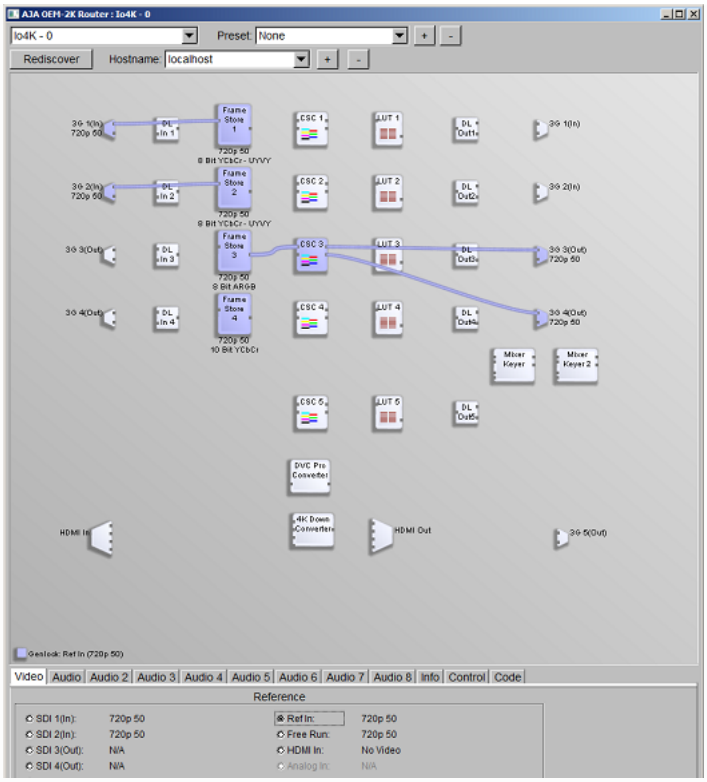
The topology is chosen from the board's device ID, so no extra configuration is required.

Configuration Utilities

Vizrt includes two configuration utilities from AJA:

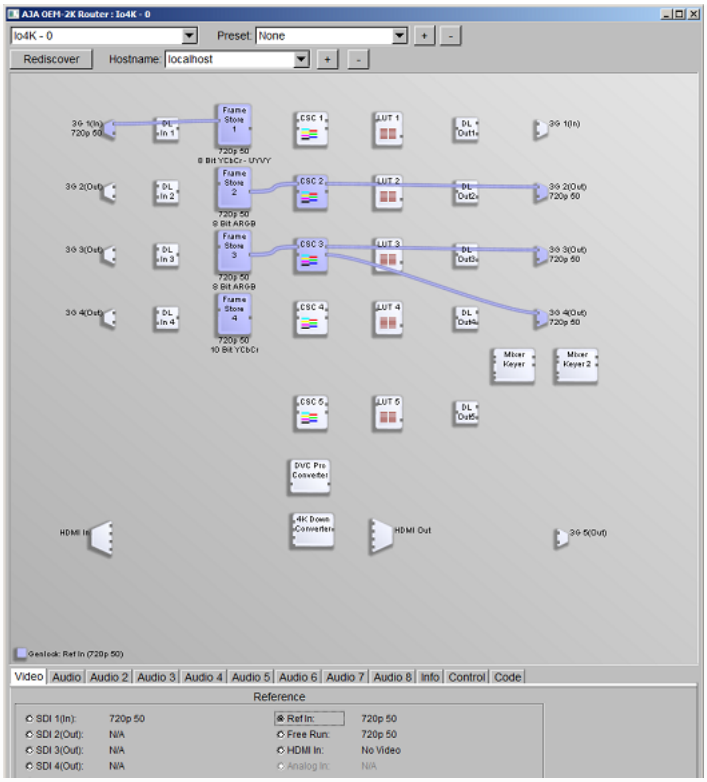
- **cables.exe:** This utility shows the current topology running: Inputs, outputs, color space conversion, fill and key paths. Firmware and driver version information are also accessible. Screen-shots of *cables.exe* use showing two typical topologies (two video inputs and one video and one preview) are shown below:

Example: Topology using two video inputs:



Example: One video input, one preview:

Copyright © 2026 Vizrt



- **watcher.exe:** This utility shows register settings. You may also write out a status log file. The auto-circulate engine is also observable. This utility is mostly of value for support specialists.

Firmware Upgrade

Only do an upgrade of Io4K firmware if instructed to do so by Vizrt or AJA support. All files for firmware upgrade are provided in a separate directory, typically supplied as a zip-archive. To install the firmware:

1. Open a command window.
2. *cd* to the directory where the firmware upgrade files are located. Make sure no other programs are running that could interfere with the firmware upgrade.
3. Install the firmware, in this example named IO_XT_4K_13.bit, using the *ntv2firmwareinstaller* utility (for example, *ntv2firmwareinstaller -f IO_XT_4K_13.bit).*
4. The window then pauses for a few minutes. When the firmware upgrade is complete, the command line program asks for a return key press to exit. Please be patient, the firmware upgrade can take three to six minutes (or more, depending on hardware configuration).
5. Power-cycle the Io4K and reboot the computer to make sure the upgraded firmware configuration is active.

See Also

- Manufacturer's documentation (external link at www.aja.com): AJA Io 4K

Copyright © 2026 Vizrt

AJA Io 4K Plus

Overview and Connectors

The AJA Io 4K Plus is the next generation of this series. The only differences related to the implementation with Viz Engine and the hardware are the usage of Thunderbolt 3 as connection interface, and the fact that the Ref In/LTC In and LTC Out connectors changed places.

As with its predecessor, only the connectors SDI 1 - 4 and the Ref In connector are implemented for use in Viz Engine. For information on the connector mapping, please see AJA Hardware.

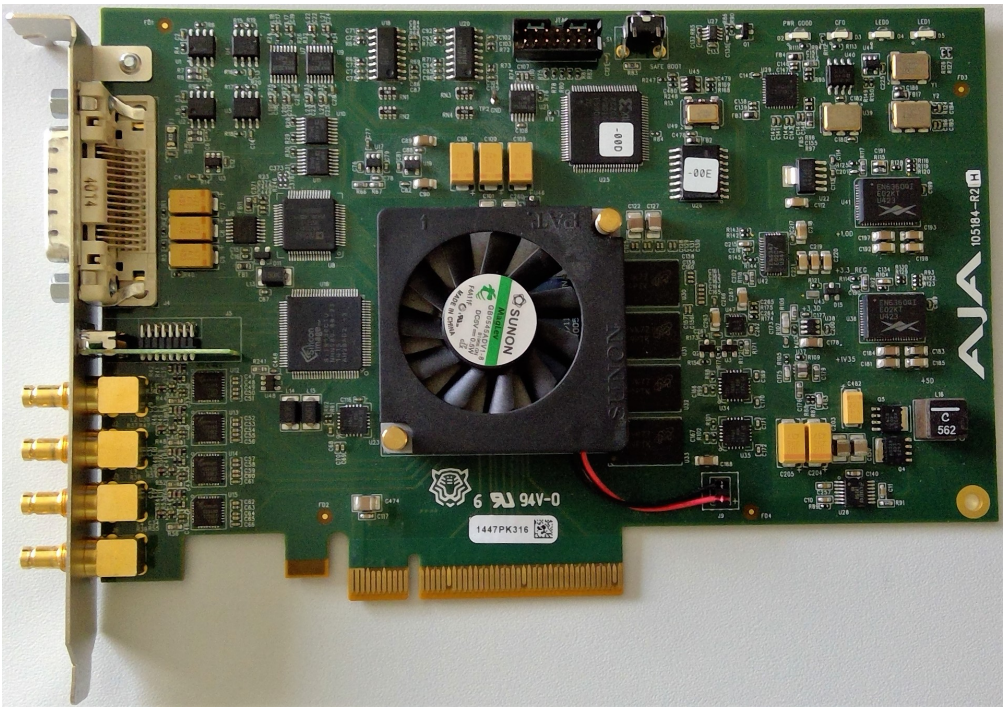


Information: Newer devices might require an update of the firmware. 16.1.0.3 driver has been tested with 2021/06/23 firmware.

AJA Kona 4

Overview and Connectors

Only the connectors SDI 1 - 4 and the Ref In connector are implemented for use in Viz Engine. For information on the connector mapping please see Aja Hardware.



7 Starting Viz Engine

To start and run Viz Engine, you need to connect to an available Graphic Hub. Viz Engine can also be started with various options included (see Viz Command Line Options). To start Viz Engine with an option or options, the command for that option must be added before Viz Engine start-up.

7.1 To Start Viz Engine

1. Start Viz Engine.
2. In the Graphic Hub login window, provide the required details:
    a. : Select a Host from the drop-down list. You can also type the host name, if the host does not appear in the list. This may happen if the server is on a different sub-net.
    b. : Select a Graphic Hub from the drop-down list.
    c. : Select a User from the drop-down list.
    d. : Type the correct password for the selected User.
    e. **Log me in automatically:** If auto log in is available, click to enable automatic log on. This makes Viz Engine automatically connect and log on to a pre-defined Graphic Hub on start-up.
3. Click **Log in**.

7.2 To Add a Viz Engine Startup Option

1. Right click the Viz Engine icon and select **Properties**.
2. In the **Shortcut** pane, apply a command in the **Target** field (see Viz Command Line Options).

> **Example:** `"%ProgramFiles%\vizrt\VizEngine\viz.exe" -n`

3. Click **OK**.
4. Start Viz Engine.

7.2.1 Viz Configuration

Viz Artist/Engine is mainly configured by the Configuring Viz application. All configuration settings are stored in the Viz Config file, found in the *<viz data folder>*. The Viz Config file uses the machine host name to uniquely identify which machine Viz Artist/Engine is installed on, as for example, *VizEninge-0.cfg*.

Any changes to the host name affect the Viz Artist/Engine. If a host name is changed, a new Viz Config file is created with a default setup. The old Viz Config file is not deleted, but left unused. If required, use the command `-g <configuration file>` (see Viz Command Line Options) to reassign the old Viz Config file.

> **IMPORTANT!** Viz Engine ignores text following a Hash tag. Do not use the Hash tag (#) in any folder, file or path name.

Copyright © 2026 Vizrt

7.2.2 To Migrate a Previous Viz Configuration

Viz Engine automatically migrates existing Viz 3 configurations if:

- There is no valid Viz config file.
- A sentinel file can not be found in the Viz folder.

7.2.3 Deleting Shader Cache

Sometimes it is required to remove the shader cache and rebuild it from scratch if Viz Engine hangs during startup, the GPU driver has changed or the GPU itself has been replaced.

To do so, remove the content of the the following folders:

- `%userprofile%\AppData\Local\NVIDIA\GLCache`
- `%programdata%\vizrt\VizEngine\ShaderCache`

> **Note:** Depending on your CPU, compiling Shaders might require up to 15 minutes.

Copyright © 2026 Vizrt

7.3 Viz Console

The Viz Console window is mainly used for debugging purposes, to display information about the running Engine or connected Graphic Hub, and for manually sending commands directly to the Engine. The Viz Console shows the commands that are used in communication between the Viz Engine Renderer and other components, such as Viz Artist, External Control Applications, plug-ins, and scripts.



For an overview of the available commands, open the console window and type `?` or `help`, then press enter.
This outputs a list of the available console commands. The complete documentation for the command interface is included with the Viz installation, and can be found under *<Viz Install Folder>\Documentation\viz-cmd-doc-webhelp*.

IMPORTANT! The Viz Console is case sensitive.

7.3.1 Autocompletion Feature

Introduced in Viz Engine version 5, the console offers and autocompletion feature when sending commands directly the console. To use this, hit the **TAB** key on any location.

Example

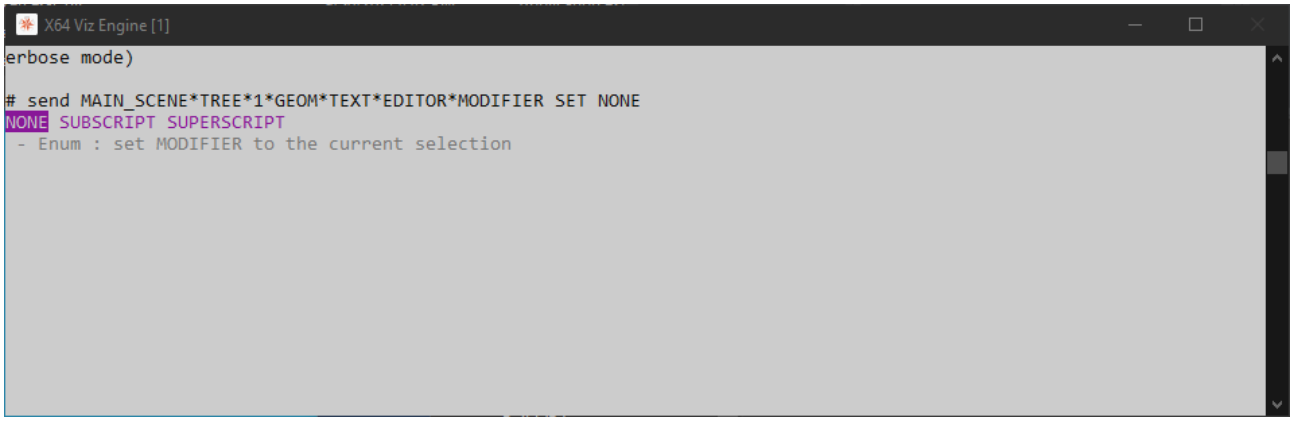
Type `send RENDERER` and hit the **CTRL+TAB** Key:

```
# send RENDERER*#%1
#%1                     PERFORMANCE             CLOSE                   SET_CAMERA_IMMEDIATE
ANIMATION               PICK_LAYER              COMMAND_COMPLETE        SET_CHANGED
BACK_LAYER              PICK_QUAD               COMMAND_EXIST           SET_CHANNEL
COMMANDS_XML            PICK_QUAD_DATA          COMMAND_INFO            SET_KEY
CONTENTS_VISIBLE        POST                    CREATE_FSWINDOW         SET_OBJECT
COORDINATE_SYSTEM       POST_MODE               DROP                    SET_PERFORMANCE
CSC                     SAA                     DROPINZ                 SET_SAFE_AREA
CURRENT_CAMERA          SHORTCUTS               DROPZ                   SET_TITLE_AREA
EYE_SEPARATION          SHOW_SAFE_AREA          GET_OBJECT              SET_VSYNC
FORMAT                  SHOW_TITLE_AREA         GET_OBJECT_LOCATION_PATH SHOW_BOUNDING_BOX
FRONT_LAYER             SIMULATE_INPUT          GET_WINDOW_INFO         SLEEP
GRID                    STEREO_PARAMETERS       INITIALIZE              SNAPSHOT
HUD                     TEXT_EDITOR             JOIN_SWAPGROUP          SYNC_DISPLAYS
KEY_INTERNAL            UPDATE                  PREVIEW                 SYNC_VIDEOINS
MAIN_LAYER              VGA_PREVIEW             PREVIEW_SIZE            UNPROJECT
MIDDLE_LAYER            VIEW                    PROPERTY_STATUS         WINDOW_OPEN
OBJECT_ID               ZERO_PARALLAX_DIST      REDRAW                  WINDOW_OPEN_ON_PARENT
OBJECT_TYPE             ANTIALIASE              RENDER_TIME             WINDOW_REPARENT
OVERWRITE_STEREO        CAPTURE_ENVIRONMENT_MAP SET_CAMERA              WINDOW_RESIZE
```

Copyright © 2026 Vizrt

All possible properties on this location are shown in blue, whereas all possible commands are shown in red.

Enums are shown in purple:



Shortcuts

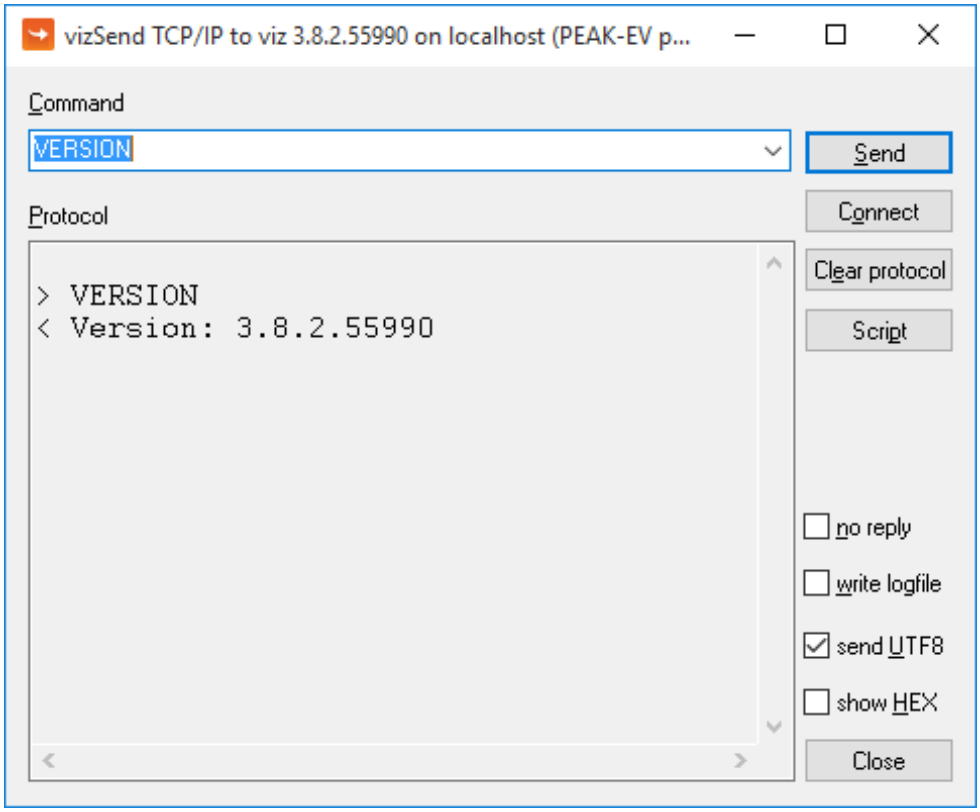
- **TAB:** Steps through the list of available commands or properties.
- **SHIFT + TAB:** Steps back.
- **PAGE UP** and **PAGE DOWN:** Selects the first or last entry in the active column.
- **ARROW KEYS:** Navigates through the list.
- **ENTER:** Selects the highlight possible word if shown otherwise send the command.
- **ESCAPE:** Aborts selection if shown, otherwise erase the command line.
- **SHIFT + ESCAPE:** Redraws current editing line.
- **ARROW UP/DOWN:** Shows sent commands from history.

Commands are auto converted to match correct upper or lower case. For example a *main_scene* is turned into *MAIN_SCENE*.

7.3.2 Issuing External Commands to Viz Engine via Console

External Control Applications, such as Viz Trio and Viz Pilot, communicate with Viz Engine through External Control commands. Viz Engine supports a wide range of such commands. Any installation of Viz Engine and Artist also come with the application **Viz Send**, which mimics the behavior of External Control Applications.

Copyright © 2026 Vizrt



External Commands can also be sent manually to Viz Engine through the console window, using the `send` command. External commands range from loading a scene and taking the scene On Air, to shutting down Viz Engine. To issue a command to the Engine, open the console window and type `send [COMMAND]`, then press enter. For example, the command VERSION can be used to display the current Viz version and build number: `send VERSION` `CONSOLE: answer <Version: 3.8.2.55990>`

Most commands also have a subset of commands. All commands in the Engine follow a specific syntax, and can be passed either properties or further commands. The general syntax structure is: `[Leading flag] [Location] [Command] [Value]`

Locations and commands may also have sub-locations or subsets of commands, respectively, in which case an asterisk serves as the divider between each part. Please see the examples below for a better understanding of how to address these.

- **Leading Flag:** Define if Viz Engine should send a reply to the command being issued. A leading `-1` indicates that no reply is required. Positive integers, including `0`, serve as a command ID. This allows an external control application to match the reply to the corresponding command request, if several commands are sent simultaneously.
- **Location:** The location can be the main object, configuration, an object ID, one of the object pools, a container or geometry, a key-frame, a shared memory (SHM) location, a layer, and so on. In addition, the location can be any command or property of such. In essence, anything in Viz could serve as the location for a command. As an example, a scene can be loaded into any one of the three layers in Viz; **Back** layer, **Main** layer and **Front** layer. The layers can be addressed by name:

Copyright © 2026 Vizrt

- `RENDERER*MAIN_LAYER`
- `RENDERER*FRONT_LAYER`
- `RENDERER*BACK_LAYER`

  In addition, the scene can be accessed directly by referring to the scene via its database path, UUID, REST URL or temporary object ID:
- `SCENE*SceneLocation`

- **Command:** The Command part of the syntax can consist of either a property or another command.
- **Value:** The value for the command, which can be for example a numeric value, a location or a state.

> **Tip:** `COMMAND_INFO` is a very useful command, used to display help on any other command or subset. For example, passing the command `send RENDERER COMMAND_INFO` displays a list of all available properties and commands for the **RENDERER** command, while `RENDERER*BACKGROUND` displays properties and commands for the **BACKGROUND** sub-command of the `RENDERER` command.

Viz Engine can address locations in three ways, by either **UUID**, **Path** or **REST:**

- **UUID:** Universally Unique Identifiers are always provided angle brackets, for example `<1A85AC97-00AE-E74B-A236EDA262D69908>`.
- **Path:** The full path to the item, as seen in Viz Artist.
- **REST:** In addition to the built-in Viz Engine REST Interface, Viz Engine also interprets Graphic Hub REST URLs as part of commands. Even though the commands themselves are case sensitive, the host names and UUIDs in REST URLs are not. The URLs may also contain special characters, like a forward slash.

These are all valid as locations when passing commands to Viz Engine, although they change the appearance of the command itself. Here is an example command setting an image as background image for the loaded scene, for each location method:

- **UUID:** `RENDERER*BACKGROUND*IMAGE*IMAGE SET IMAGE*<1A85AC97-00AE-E74B-A236EDA262D69908>`
- **Path:** `RENDERER*BACKGROUND*IMAGE*IMAGE SET IMAGE*01_DOC_EXAMPLES/Images/Bear`
- **REST:** `RENDERER*BACKGROUND*IMAGE*IMAGE SET IMAGE*http://GH\-SERVER:19398/image/1A85AC97\-00AE\-E74B\-A236EDA262D69908/Bear`

> **Tip: Object IDs** are temporary shortcuts created on the fly when an item is loaded into memory, and removed when unloading the item or restarting Viz. To get an objects Object ID, pass the **GET** command to the **OBJECT_ID** location of the object. For example: `MAIN_SCENE*OBJECT_ID GET`.

7.3.3 Internal Commands

For ease of use, internal commands can be set to be hidden or visible. Under certain circumstances, some plug-ins may need to send commands on every frame. This can leave the Viz Console window flooded with text, leaving it unreadable. By default, internal commands are hidden, meaning these messages do not output to the console window. To see all internal commands by default, the Viz configuration file must be manually edited, setting `show_internal_commands` to 1. During normal operation, however, it would usually suffice to temporarily

Copyright © 2026 Vizrt

activate the output of internal commands to the console, by clicking the  (show commands) icon in the Viz Artist or Viz Engine GUI. This also makes the console window stay on top, meaning it is not hidden behind other full-screen applications, such as Viz Artist or an Engine in On Air Mode.

Copyright © 2026 Vizrt

7.4 Viz Command Line Options

You can start Viz Artist and Viz Engine with various startup options. These command line parameters must be passed to the Viz executable file, *viz.exe.* The batch file *viz.cmd* is only there for compatibility reasons. The table below details the most common Viz Command Line Options. You can get a list of all available startup options, by running `viz.exe -h` or `-?` from the command line.

Important Change From Viz Engine 4: Starting with Viz Engine version 5.0, the parsing is done according to the POSIX standard with GNU extensions. The most notable change is the space between the option character and the option argument if the option requires an argument and the required long option form for Graphic Hub parameters: *--db* instead of *-db*.

Command	Description
-B <path>	Specify the path where Viz Engine stores its temporary data (see Viz Engine Folders).
-c	Start in Viz Configuration mode (see also -u 1, -u 2, -u 3).
-C	Start without a console.
--db user:pw@server/name-server:port	Specify which Graphic Hub to connect to on Viz Engine startup. `password` can be omitted from the argument, but `user`, `server`, `name-server` and `port` must always be provided. **Example:** `viz.exe --db Guest:@VizDbServer/localhost:19396` In the example above, Viz Engine connects to a Graphic Hub server called `VizDbServer` (⚠ case sensitive) on the name-server `localhost` at port `19396`, as the user `Guest` with no password.
-g <config file>	Start with a predefined Viz Config file. This allows a User to have more than one startup option.
-G 1, -G 2, -G *n*	Sets the GPU affinity for the selected instance. For systems with two or more graphic cards (Viz Trio One Box / Dual Channel). See Systems with Two or More GPUs below. **Example:** *viz.exe -u 1 -y -n -G 1* starts Viz Engine with On Air GUI as first instance, bound to the first GPU. *viz.exe -u 1 -y -n -G 2* starts Viz Engine with On Air GUI as first instance, bound to the second GPU.

Copyright © 2026 Vizrt

Command	Description
-h, -?	Display the available commands.
-i	Enable pre-initialization of textures. Textures are generated on the graphics card immediately after loading an image.
-l	Specify a console title to distinguish Viz Engines in a Dual Channel setup (example: `-l <title>`).
-M	Used by Viz Trio to allow sharing of content between Viz Artist and Viz Trio.
-n	Start in Viz Engine mode (see also -u 1, -u 2, -u 3).
-o "<scene>" -o "<layer> <scene>"	There is also an optional argument may give a scene that is loaded upon startup with the following syntax: **SCENE*scene_id** ... load scene_id into the main layer **1 SCENE*scene_id** ... load scene_id into the main layer **0 SCENE*scene_id** ... load scene_id into the back layer **2 SCENE*scene_id** ... load scene_id into the front layer
-P	Disable automatic mouse capture.
-t	Enables non-interactive mode for all occurrences except dongle issues and sound driver setup. The non interactive mode was introduced for use cases where there is no user and when there is no GUI one can interact with.
-T	Keep the Viz Engine Console always on top.
-u 1, -u 2, -u 3 (up to a maximum of 24)	Sets the instance number.
-v [argument]	Enable verbose mode. This enables all possible information to be shown in the Viz Engine Console. The optional argument is a number composed by addition of: • **1:** produce verbose output to console • **2:** add a timestamp • **4:** log OpenGL • **16:** log 2d-texture messages • **32:** log Graphic Hub releated debug messages • **64:** log medium and high GL warnings

Copyright © 2026 Vizrt

Command	Description
-w	Start in Engine mode and show render window (videowall mode).
-W	Disable restart on crash.
-X	Write extended dump file (full memory dump) in case of a program crash.
-y	Start in Artist mode. Viz Artist is stated by Viz Engine after the Engine has started.
-Y <path>	Specify the path where Viz Engine stores its program data (see Viz Engine Folders).

7.4.1 Systems with Two or More GPUs

It is possible to specify which GPU Viz Engine should run on, by providing the flags -G *n,* whereas n means the index of the GPU (starting with 1). In combination with the -u flag, it is possible to start Viz on a dedicated GPU, no matter, where the primary monitor is connected to.

Providing the -G parameter is mainly used to bind certain instances to dedicated GPUs or to make sure, that a GPU is not occupied by any Viz Engine instance.

Example:
viz.exe -u 1 -y -n -G 1 starts Viz Engine with On Air GUI as first instance, bound to the first GPU.
viz.exe -u 2 -n -G 2 starts Viz Engine as second instance with console only, rendering on the second GPU.
viz.exe -u 2 -n -G 1 starts Viz Engine as second instance with console only, but rendering on the first GPU.

If only the -u parameters are provided, Viz uses the old behavior by automatically assigning the GPU to the instance started.

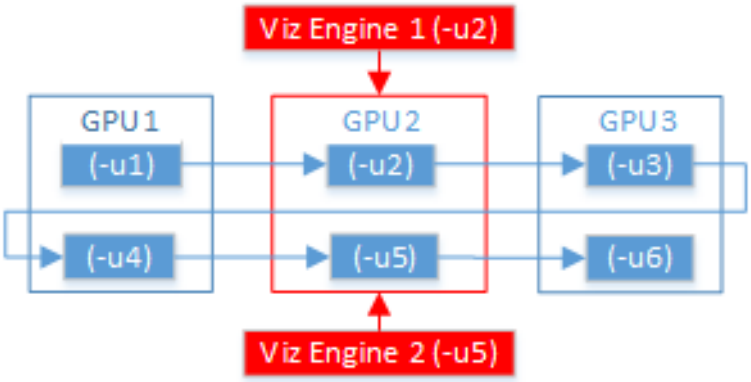
Formula: m%n, where:

- m = -u 1/-u 2/etc.
- n = number of GPUs

For example, with two Viz Engines:

- System with two GPUs: Engines run on GPU2: Engine 1 (-u 2) / Engine 2 (-u 4)
- System with three GPUs: Engines run on GPU2: Engine 1 (-u 2) / Engine 2 (-u 5)

Copyright © 2026 Vizrt



When either Viz Trio One Box or Dual Channel Viz Artist versions are installed, GPUs are selected by default.

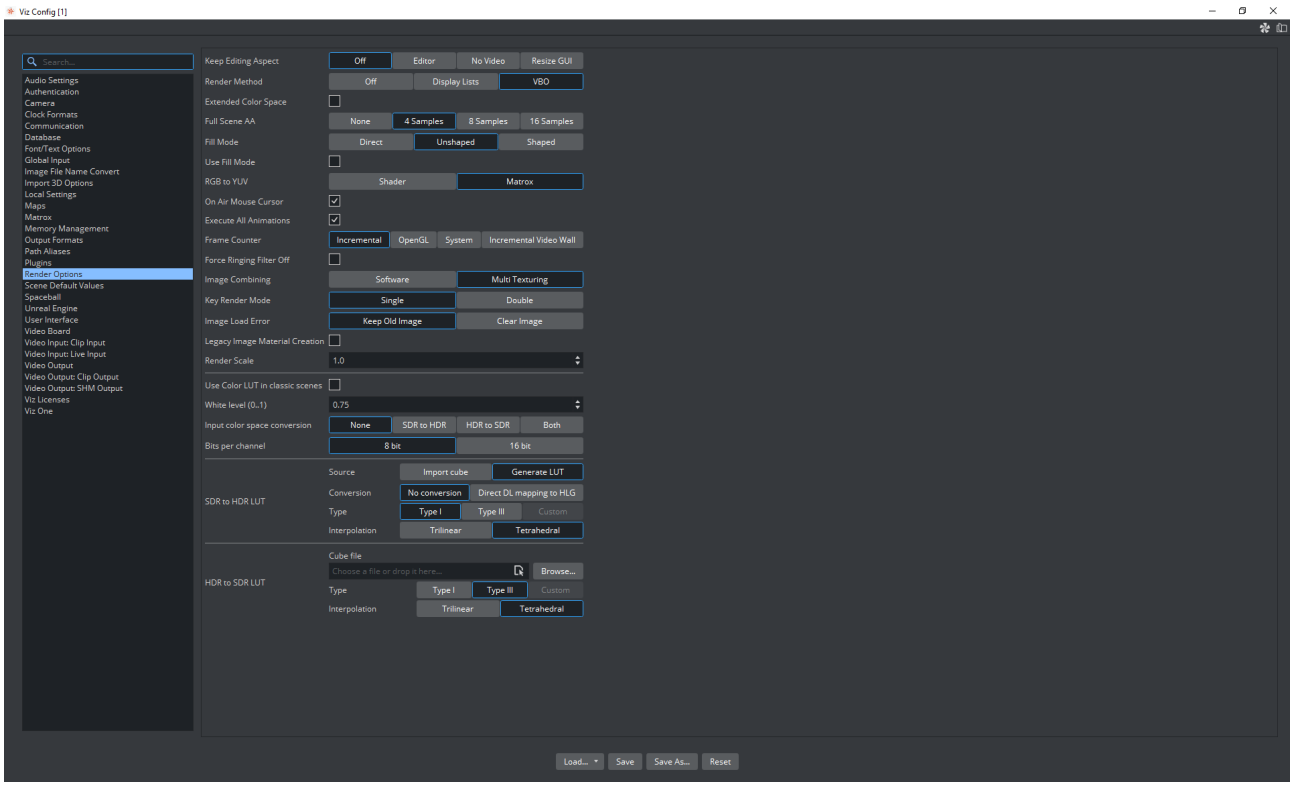
Copyright © 2026 Vizrt

8 Configuring Viz

This page contains information on the following topics:

- Working with Viz Configuration
  - To Start Viz Configuration
- Modify Viz Configuration
  - To Save the Current Configuration
  - To Reset the Viz Config File
  - To Restart Viz Configuration
- Installed Configuration Profiles
  - To Load a Pre-Installed Configuration Profile
  - To Save a Custom Profile
  - To Load a Custom Configuration Profile

Viz Configuration is the configuration interface for Viz Engine and other applications that integrate with Viz Engine.



The Viz Configuration user interface is divided in two parts:

- **Sections:** Shows a list of all the sections available for configuration.
- **Configuration Parameters:** Shows all available configuration parameters for each section.

Various parameters regarding the program functionality can be set in the configuration. Viz Artist/Engine is mainly configured with the Viz Configuration. All configuration settings are stored in a Viz Config file (located in the *<viz*

Copyright © 2026 Vizrt

data folder>).
Any changes to the host name affects the configuration.

If required, an optional Viz Config file can be reassigned with the command `-g <configuration file>`.

Caution: Make sure that any changes are saved before Viz Configuration is closed. Changes are not saved and do not take effect until Viz Artist has been closed and started again.

8.1 Working with Viz Configuration

To configure Viz Artist and Engine, click the **Config** button in Viz Artist or start the Viz Configuration application from the Start menu. Viz Artist/Engine also has a selection of Installed Configuration Profiles. When the configuration profiles are saved, only the settings that differ from the default configuration of a setup (VGA mode, Video mode, etc.) is saved.

Viz Configuration adjusts its size to the current screen resolution when it opens. When running Viz Configuration without Viz Artist, it checks the resolution and limits the size to `1430` x `750` pixels. This is useful for high-resolution configurations, especially during Video Wall Configuration.

8.1.1 To Start Viz Configuration

- Run Viz Configuration from the desktop shortcut or the **Start** menu.
- If Viz Artist is running, click **Config** or press **F11**.

8.2 Modify Viz Configuration

This section details how to Save, Save as..., Reset and Load Viz Configurations.

8.2.1 To Save the Current Configuration

1. Start Viz Configuration.
2. Change the configuration as required.
3. Click **Save**.

IMPORTANT! Changes made to the resolution can affect an in use IP streaming service. Make sure to check the IP streaming service configuration.

4. Click the **Restart** button to apply the saved changes. The Viz Config file is updated.

8.2.2 To Reset the Viz Config File

The **Reset** button sets the configuration to default settings.

1. Start Viz Configuration.
2. Click **Reset**.
3. Click **Restart** to apply the changes.

Copyright © 2026 Vizrt

8.2.3 To Restart Viz Configuration

Click the **Restart** button to save changes.

1. Click **Restart**.
2. Select from:
    - Current
    - Viz Engine w/GUI
    - Viz Engine w/o GUI
    - Viz Artist
    - Viz Config

8.3 Installed Configuration Profiles

Viz Artist/Engine is installed with a selection of Configuration Profiles. These profiles are a set of predetermined basic settings to run Viz Artist/Engine for specific purposes, for example, Dual Channel Mode, Trio Box CG Mode or a Video Wall Configuration.

Once loaded into the Viz Configuration, these files can be modified to refine the profile to specific needs, which can then be saved. Pre-installed Configuration Profiles are located in: *<viz install folder>\Configuration Profiles*.

8.3.1 To Load a Pre-Installed Configuration Profile

1. Start Viz Configuration.
2. Click **Load** and select **Installed Profile**. The correct folder opens automatically.
3. Open the required Configuration Profile.
4. Click **Restart** to load the Configuration Profile.

8.3.2 To Save a Custom Profile

A Configuration Profile cannot be saved to the *<viz install folder>*. A UAC requirement is that an application must not write to the installation folder (see User Account Control). The default location for custom Configuration Profiles is *%Programdata%\Vizrt\VizEngine\Configuration Profiles*.

1. Start Viz Configuration.
2. Change the configuration as required.
3. Click **Save As**.
4. Select a location to save the new Configuration Profile. The default location is *%Programdata%\vizrt\VizEngine*.
5. Type a name for the new Configuration Profile.
6. Click **OK**.

8.3.3 To Load a Custom Configuration Profile

1. Start Viz Configuration.
2. Click **Load** and select **Custom Profile**. The correct folder opens automatically.
3. Locate a saved Configuration Profile (**.cfg*).
4. Open the required Configuration Profile.

Copyright © 2026 Vizrt

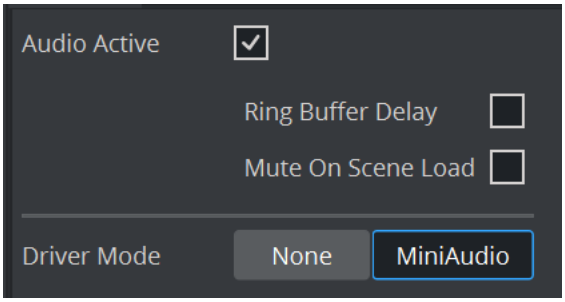
5. Click **Restart** to load the saved Configuration Profile.

8.4 Audio Settings

This section provides information about the following:

- Various Tab
- Channels Tab
  - To Add Multi-language Audio Channels
  - To Add Multiple Audio Channel Configurations
  - Setup Tab
- Manual Audio Configuration
  - To Map Audio Output for a Dual Channel Setup
  - To Manually Activate an Audio Device

8.4.1 Various Tab

- 

- **Audio Active:** Toggles audio in Viz Engine on/off.

On videoboard enabled system the following settings are also available:

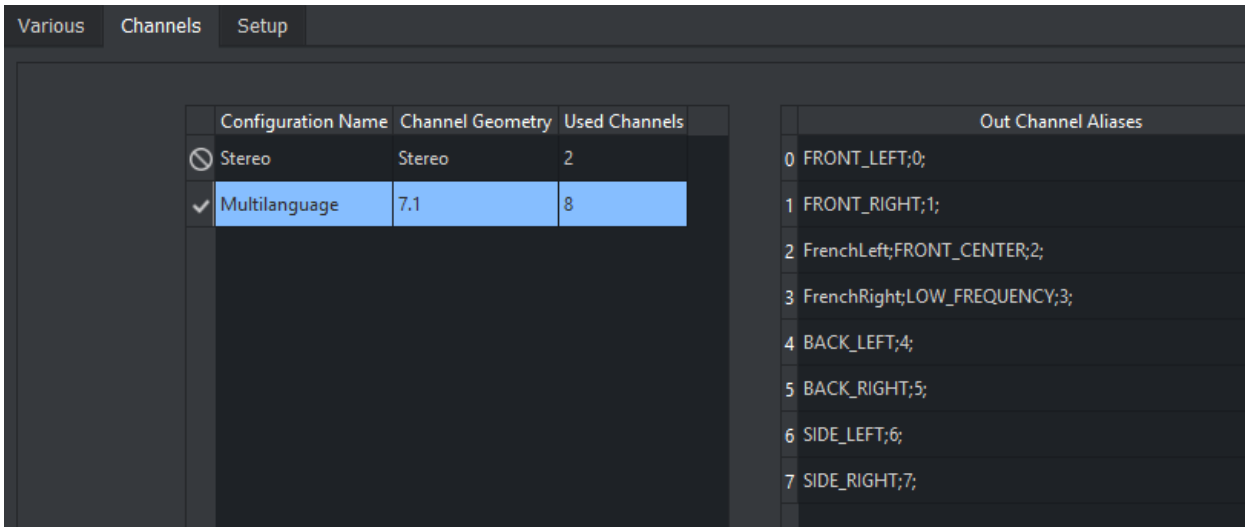
- **Ring Buffer Delay:** Makes the audio system compensate for the ringbuffer delay during clip playout. Ringbuffer is ignored when disabled.
- **Mute On Scene Load:** Mutes audio before any scene load commands are executed when set to `On`. This is necessary for video cards which do not mute the audio automatically, or when no video refresh happens.

> ⚠ **Note:** Additional commands to mute audio can be added in the Viz Config file.

- **Driver Mode:** Affects playback of sound on the computer running Viz Engine, for audio monitoring purposes.
  - **None:** Prohibits Viz Engine from playing out sound on the local system.
  - **MiniAudio:** Default OS Audio mode.

Copyright © 2026 Vizrt

8.4.2 Channels Tab



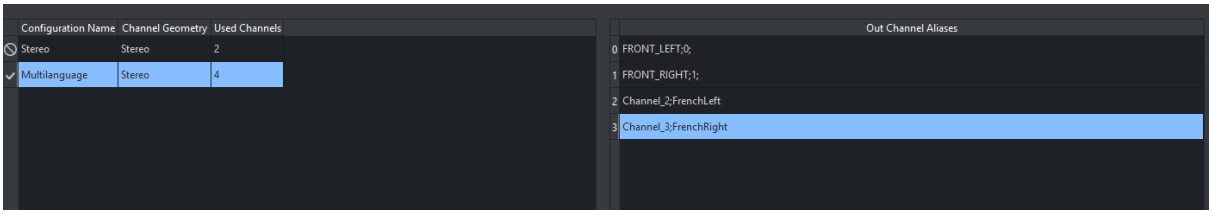
- **Configurations:** Shows a list of channel configurations.
- **Add:** Adds a new channel configuration to the Configurations list.
- **Delete:** Deletes the selected channel configuration from the Configurations list.
- **Name:** Sets the name of the selected channel configuration.
- **Channel Geometry:** Sets the channel geometry. Options are:
  - Mono
  - Stereo
  - 5.1
  - 7.1
  - Quad
- **Used Channels:** Sets the number of configurable channel alias fields that can be mixed by the internal channels in Viz Engine (software). On a Matrox system this number must be equal to the number of configured input channels (hardware). This setting is independent of the Channel Geometry setting.
- **Reset Aliases:** Sets the channel aliases to the default option (the custom entry is not removed).
- **Out Channel 0-15:** The Out Channels represent the internal Viz audio channels, which are mixed to the output device one by one. Alias names are useful to create multilingual systems, and also to define the channel geometry for the Default and FX audio clip mix modes. Aliases are separated by a semicolon. The **Audio** plug-in only uses the channel aliases to find the correct speakers for Pan and 3D sound effects in FX mode. With this functionality any of the 16 internal audio channels can be used to play any audio geometry.

To Add Multi-language Audio Channels

1. From the Channels tab, click **Insert**.
2. Enter the new name in the **Name** field.
3. Set the Channel Geometry to **Stereo.**
4. Set the number of **Used Channels** to, for example, `4` .

Copyright © 2026 Vizrt

5. In the **Channel 2** field, add the alias **FrenchLeft**.
6. In the **Channel 3** field, add the alias **FrenchRight**. Add the same configuration for English (EnglishLeft, EnglishRight) and German (GermanLeft, GermanRight) on the other machines.
7. Click **Save**.

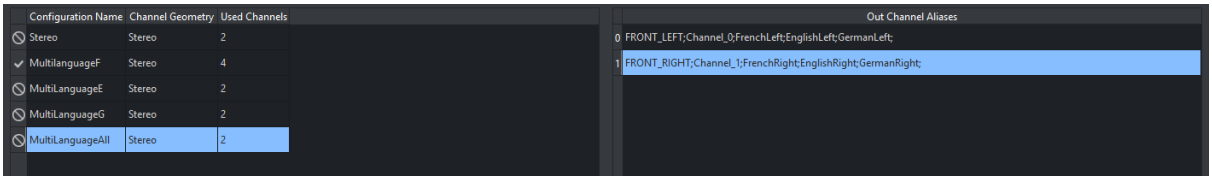


To Add Multiple Audio Channel Configurations

1. Repeat To Add Multi-language Audio Channels to add three more Multi-language Audio Channels. Name the configurations the following way:
   - **Multi-languageF**
   - **Multi-languageE**
   - **Multi-languageG**
2. Add a **Multi-languageAll** configuration.
3. In the **Channel 1** field, add the aliases **FrenchLeft**, **EnglishLeft** and **GermanLeft.**

```
FRONT_LEFT;
Channel_0;FrenchLeft;EnglishLeft;GermanLeft;
FRONT_RIGHT;
Channel_0;FrenchRight;EnglishRight;GermanRight;
```

4. In the **Channel 2** field, add the aliases **FrenchRight**, **EnglishRight** and **GermanRight.**
5. Click **Save**.



Setup Tab

The Setup tab lists all audio devices available to the Viz Engine audio mixer (see Audio in Viz). Sound card devices are the playback devices that Windows reports, such as sound cards, USB audio interfaces and HDMI audio (see Device Recognition and Selection). Channels without a configured device play on the Windows default playback device, which is set under **Sound** in the Windows settings.

When a Matrox card is installed, Matrox audio is available. With Matrox audio, there are different ways to capture audio for use with the Viz Engine audio mixer, before it is output:

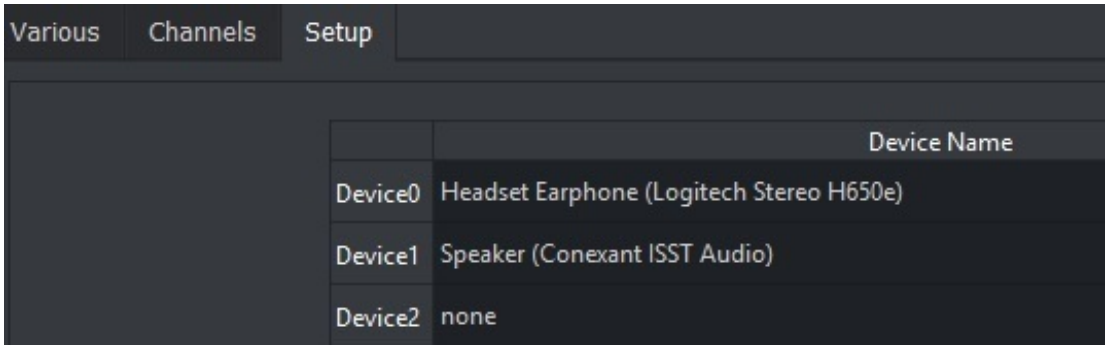
1. Capture audio embedded in the live video input signal from the live input connectors.
2. Capture audio from the AES connectors.

The final output is the same, either embedded as part of the live video output signal, or separated out to the AES output connectors. In addition, simply loop audio through the system. This makes the embedded or AES audio unavailable to the Viz Engine audio mixer.

Copyright © 2026 Vizrt

The **Default** mode captures audio and directs it to the default onboard audio device with no output on the Matrox card output connectors.

> **Note:** The Matrox audio-extension board is not configurable through this user interface.



- **Device *n* :** Shows the name of the audio card.
- **Mode:** Refers to the audio mode of the Matrox board. Options are:
  - **Embedded:** Captures audio from the Live video input connectors and is made available to the Viz Engine audio mixer, to mix it with other audio sources. Then output as embedded audio on the live video output connectors.
  - **AES:** Captures audio from the AES input connectors and is made available to the Viz Engine audio mixer, to mix it with other audio sources. Then output on the AES output connectors.
  - **Embedded -> AES:** Captures embedded audio from the live video input connectors and is made available to the Viz Engine audio mixer, to mix it with other audio sources. Then output on the AES output connectors.
  - **AES -> Embedded:** Captures audio from the AES input connectors and is made available to the Viz Engine audio mixer, to mix it with other audio sources. Then output as embedded audio on the live video output connectors.
  - **Loop:** Loops audio through. No audio is mixed.
  - **Default:** Captures audio, but no output on the Matrox card.

8.4.3 Manual Audio Configuration

Certain use-cases require manual configuration of audio. Make sure to create a backup of the Viz configuration file before performing the following procedures:

To Map Audio Output for a Dual Channel Setup

1. In a Dual Channel environment there are two Config files, one for each Viz Engine.
2. For the first Viz Engine (1) the configuration of the audio output channel mappings should look like this:

```
Matrox0.AudioOut1.MapToVizChannel = 0
Matrox0.AudioOut2.MapToVizChannel = -1
```

3. Note that the audio output for the first Viz Engine (1) should be according to the video output channel:

Copyright © 2026 Vizrt

```
Matrox0.VideoOut1.MapToVizChannel = 0
Matrox0.VideoOut2.MapToVizChannel = -1
```

4. For the second Viz Engine (2) the configuration should look like this:

```
Matrox0.AudioOut1.MapToVizChannel = -1
Matrox0.AudioOut2.MapToVizChannel = 0
```

Note: In a stereo setup, audio must be set to `Off` in the second Viz Engine (2).

5. Mind the difference to the video output settings:

```
Matrox0.VideoOut1.MapToVizChannel = -1
Matrox0.VideoOut2.MapToVizChannel = 0
```

6. All other audio output channels should be set to `Off`:

```
Matrox0.AudioOut3.MapToVizChannel = -1
Matrox0.AudioOut4.MapToVizChannel = -1
```

To Manually Activate an Audio Device

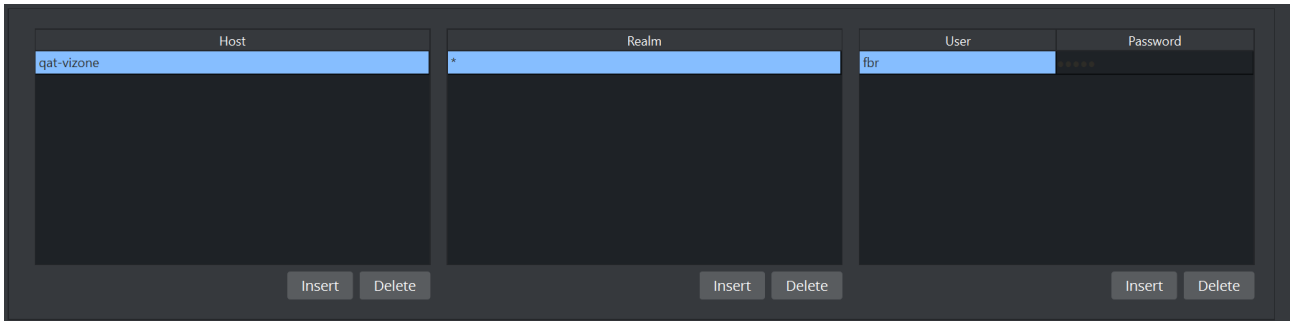
1. Open the Viz Config file.
2. Go to **SECTION AUDIO_CONFIG** and find the device name in the **Available** list (`Available0` , `Available1` and so on).
3. Enter that name, exactly as listed, in the **VIZChannelDevice** setting of each channel that should play on the device (`VIZChannelDevice0` to `VIZChannelDevice7`).
4. Save the Viz Config file and restart Viz Engine.

Example: `AudioDevice0 = Realtek HD Audio output`

Copyright © 2026 Vizrt

8.5 Authentication

The Authentication panel is for the authentication of one or more Viz One systems on Viz Artist. Enter the details of each Viz One system to connect to. To save a new user, all three panels of the Authentication panel must be completed. If a Realm is not required, enter `<empty>` in the Realm panel. If only one or two panels are completed, the information is not saved.



- **Host Info:** The host name of the Viz One server.
- **Realm:** Determines if an entry should be used in an authentication process for a given URI.
- **User Info:** Name and password for each User, for the selected Host and Realm.

8.5.1 Authentication Properties

Host Info Properties

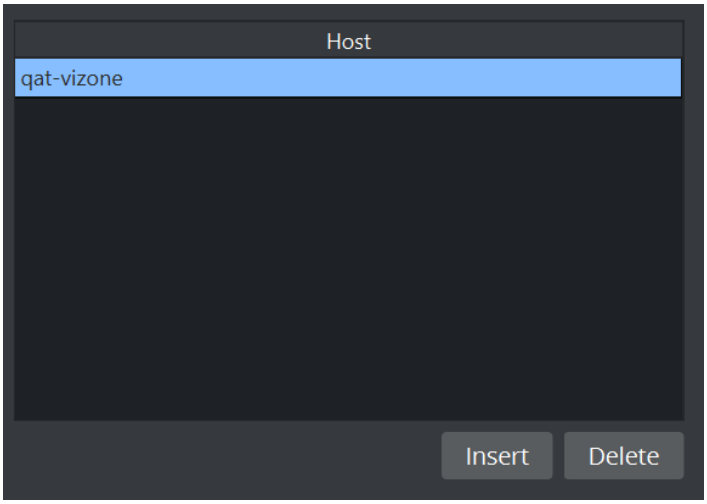
The host name of the Viz One server (see also Fsmon (File System Monitor)). The host info is in the form '`host:port`', where the colon and port number are optional. An asterisk value (`*`) shows that it is used against all hosts. The host info can be the Viz One host name or IP address.

Examples:

- `Host Info = myhost.mycompany.com:8080`
- `Host Info = 10.0.1.2:8080`
- `Host Info = *`

- A Host cannot be added without at least one Realm (or an empty Realm with *<empty>* entered) and one User defined.
- A Host can have more than one Realm.

Copyright © 2026 Vizrt



- **Insert:** Adds a host name. To change, double click on the entry.
- **Delete:** Removes a selected host name.

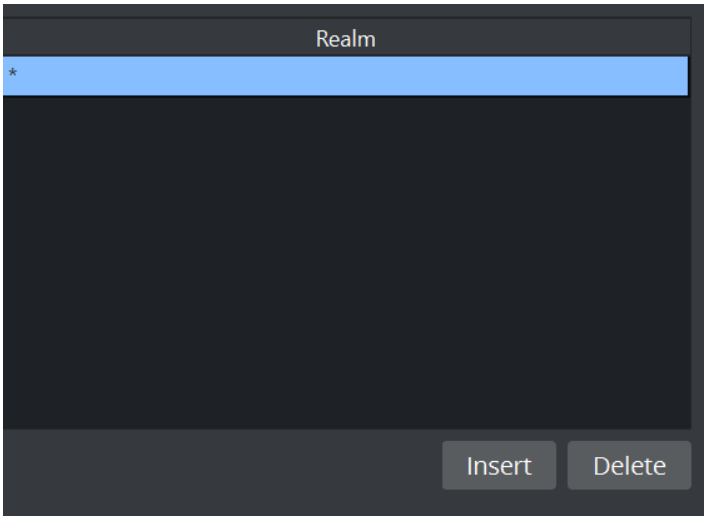
Note: Do not enter a URL as the Host Info name. The Host Info name format must be `host[:port]`, for example: *vizone.mycompany.com:8080*.

Realm Properties

Determines if an entry should be used in an authentication process for a given URI.

Example: A single value of shows that it should be used against all realms (Realm = [a-zA-Z_0-9] or ' * ').

- A Realm cannot be added to a Host without at least one User defined
- A Realm can have more than one User.



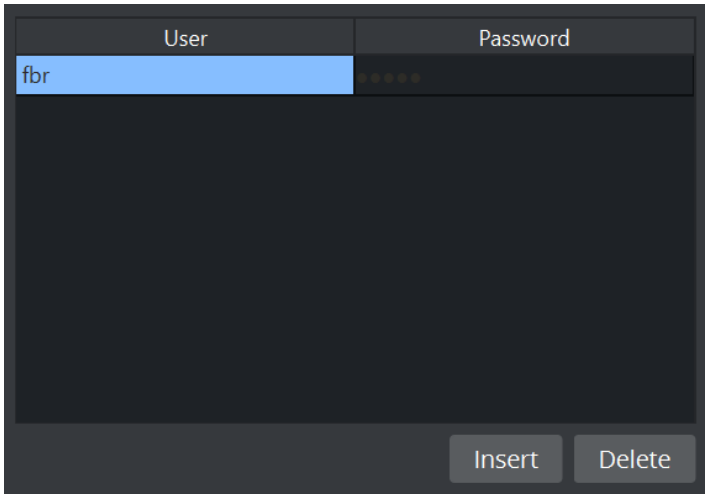
- **Insert:** Adds a Realm. To change, double click on the entry.
- **Delete:** Removes a selected Realm.

Copyright © 2026 Vizrt

- **Realm:** Displays the name of the required Realm.

User Info Properties

Name and password for each user, for the selected Host and Realm. Passwords are not mandatory.



- **Insert:** Adds a user. To change, double click on the entry.
- **Delete:** Removes a selected user.
- **User:** Enter the user name.
- **Password:** Enter a password for the user, if required.
- **Confirm PW:** Confirm the user password.

See Also

- Viz One Configuration Panel

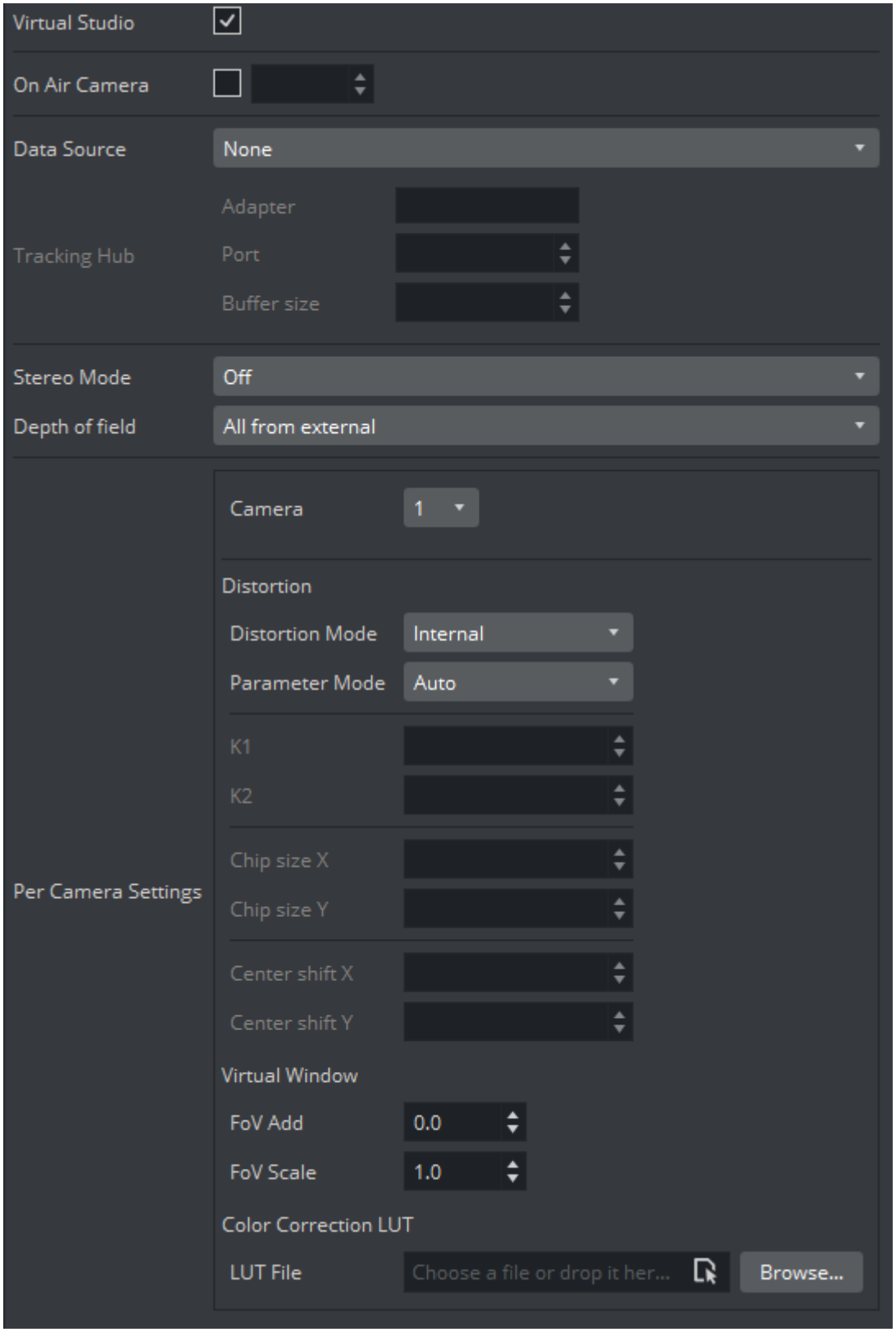
8.6 Camera

In the Camera section, special camera behavior settings which are used for virtual studio setups, can be set. Viz Tracking Hub is used as the studio configuration and calibration tool for enabling connectivity and control between all required studio devices such as cameras, routers, VTRs, video servers, audio mixers and other studio equipment.

Note: Tracking Hub and Studio Manager are the successors of VizIO. VizIO is not supported since Viz Engine version 4.0. To enable Tracking Hub, set the flag *use_trackinghub* to `1`, section CAMERA.

Copyright © 2026 Vizrt

8.6.1 Camera Properties



Copyright © 2026 Vizrt

Virtual Studio: Starts the tracking process when Viz Engine is started when set to `Active`.

- **On Air Camera:** Activates the selected camera that should be used when setting the scene in **On Air** mode. The camera is controlled by an external tracking device.
- **Data Source:** Defines the input for the camera:
  - **None**
  - **NDI/SMURF**
  - **Tracking Hub**
- **Tracking Hub:** Sets the parameter for the Tracking Hub:
  - **Adapter:** Specifies the IP address.
  - **Port:** Enter the port number.
  - **Buffer size:** Enter the buffer size for Tracking Hub.
- **Stereo Mode (Classic Render Pipeline only):** Stereo Mode settings only take effect when the design is taken On Air (license required):
  - **Off:** Makes Stereo mode unavailable and invisible in the GUI **Camera Editor**.
  - **Quad Buffered:** Alternates the image for the left/right eye and darkens the other eye when using nVision glasses together with a monitor with a frequency >100Hz. This configuration provides the full resolution.
  - **Over Under Left Top/Over Under Right Top:** Draws both images beneath each. Either side can be drawn first. The image height is halved, providing half resolution.
  - **Left Eye/Right Eye:** Renders either the left or right eye image for dual-channel setups in combination with a Video Wall / Stereo Distributor. Full resolution.
  - **Side By Side Left/Side By Side Right:** Renders both images side by side, either left or right first. The image width is halved, providing half resolution.
- **Depth of field:** Defines from where parameters are taken:
  - **Use editor:** Gets parameters for depth of field from the editor.
  - **Focal plane from external:** Gets only the focal plane from the external camera.
  - **All from external:** Gets all parameters from the external camera.
- **Per Camera Settings**
  - **Distortion:** Here you can set the lens distortion parameters for each camera individually.
    - **Distortion mode:** Sets the preferred distortion depending on the tracking system which is used. Available distortion modes are:
      - **Auto**
      - **Internal**
      - **Libero**
      - **Xpecto**
      - **Radial**
      - **Stype**
      - **Trackmen**
      - **ImageBased**
    - **Parameter mode:** Set the parameter for the lens distortion, depending on the tracking system which is used:
      - **Auto**
      - **Libero**
      - **Manual**
      - **Tracking Hub**

Copyright © 2026 Vizrt

- **Virtual Window:** Additional settings for Virtual Window.
  - **FoV Add:** Sets an offset of degrees to the tracking data.
  - **FoV Scale:** Scales the incoming FoV tracking data.
- **Color Correction LUT**
  - **LUT File:** Set the LUT file (.cube) that is applied to the camera

See Also

- **Advanced Lens Distortion** in the Viz Artist User Guide.

Copyright © 2026 Vizrt

8.7 Clock Formats



In this section, twenty various digital date and time formats can be set. Viz Engine distinguishes between uppercase and lowercase clock formats, allowing for even more customization options. Setting the clock format using lowercase letters forces Viz Engine to display the digit even if the value is `0`. By defining the clock format using uppercase letters, Viz Engine omits the value if it is zero. This allows the designer to omit unused digits while still planning for their use when designing the scene.

- **Format 1-20:** Sets and enables the clock formats that may be selected in Viz Artist during scene design.

Tip: Add a font GEOM in Viz Artist to see how the clock formats can be used.

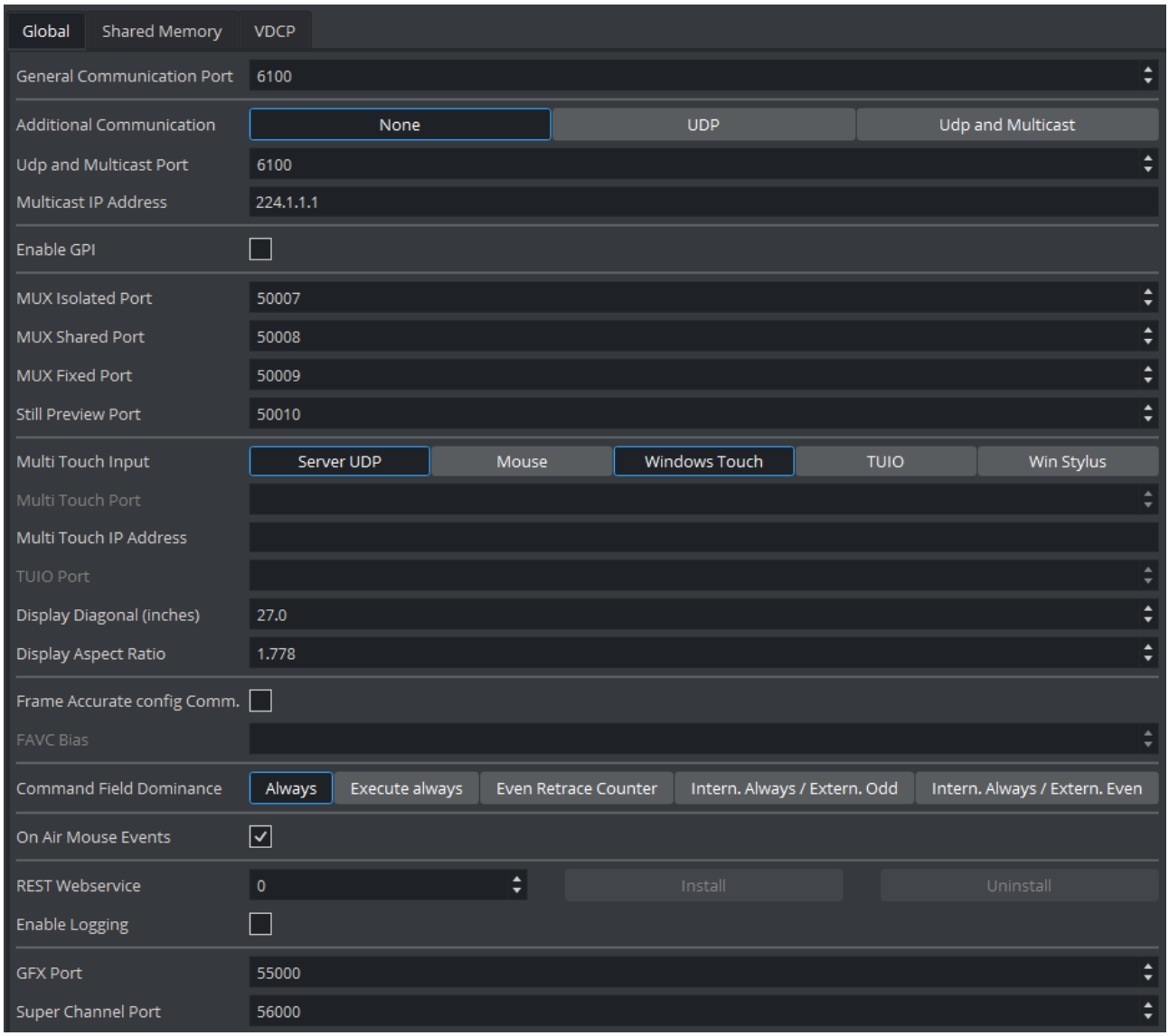
Copyright © 2026 Vizrt

8.8 Communication

In this section, network connections can be set. External control software, for example, Viz Trio, uses TCP/IP network connections to send commands to the Viz Engine renderer engine (some external communications are also through UDP). Viz Artist expects the commands at the ports which are defined here. The Communication panel has four tabs, Global, Shared Memory, VDCP (Video Disk Control Protocol) and Unreal Communication.

- Global Properties
- Shared Memory Properties
    - To Limit the Number of TCP Connections
- VDCP Properties

8.8.1 Global Properties



Copyright © 2026 Vizrt

- **General Comm. Port:** Sets the general communication port for receiving external commands when in On Air Mode. Make sure to update settings on the client side if you change the default port. The default is port `6100` (TCP).
- **Additional Communication:** Enables sending of commands to Viz Engine on **UDP** and **Multicast,** or a combination of the two.
- **Udp&Multicast Port Number:** Sets the port number for the computers that share the same virtual IP address.
- **Multicast IP Address:** Sets the shared virtual IP address.

> ⚠ **Note:** The default maximum number of TCP connections is limited to 255. Within this number of connections, a user defined limit of maximum connections can be set.

- **Enable GPI:** Enables initialization of supported Sealevel GPI/O devices when Viz Engine starts.
- **MUX Isolated Port:** Sets the port number for isolated sessions - no shared data (NLE).
- **MUX Shared Port:** Sets the port number for shared sessions - shared data (NLE).
- **MUX Fixed Port:** Sets the port number for fixed sessions - shared data, no reference counting (NLE).
- **Still Preview Port:** Sets the port number for still preview.
- **Multi Touch Input**
  - **Server UDP:** Multi Touch events are retrieved from the Viz Multi Touch Interface (see protocol documentation, Viz MultiTouchServer). An external server application connects to the touch device, translates the hardware messages into the Viz Multi Touch Protocol, and sends it via UDP to Viz Artist or Viz Engine. This triggers the internal Multi Touch events in scripts utilizing the plug-in API.
  - **Mouse:** Enable this to use a standard mouse to test Multi Touch trigger callbacks.
  - **Windows Touch:** Viz Artist and Viz Engine use Windows Touch messages to generate Multi Touch events.
  - **TUIO:** (Tangible User Interface Object) Select this if the touch device and/or application works with TUIO/OSC (Open Sound Control).
  - **Win Stylus:** Activates stylus pen input.
- **Multi Touch Port:** Sets the port number where Viz Engine listens for the multi touch server. Required for Server UPD communication.
- **Multi Touch IP Address:** Sets the IP address where Viz Engine listens for the multi touch server. Viz Artist and Viz Engine send keep-alive messages to the Multi Touch Server. You need to enter the IP address of this server. Required for Server UPD communication.
- **TUIO Port:** Sets the TUIO (Tangible User Interface Object) port number to communicate with a TUIO enabled multi-touch device. TUIO is a protocol for Table-Top Tangible User Interfaces. The default port for most TUIO applications is `3333`.
- **Display Diagonal (inches):** Viz Artist and Viz Engine can calculate transformations with momentum. For this, it needs to know the real, physical, screen-dimension to calculate the correct animation speeds. The value is in inches. Used for gesture recognition.
- **Display Aspect Ratio:** The real aspect ratio of the screen is required, as some screens do not have pixel aspect ratios of `1.0`. Used for gesture recognition.
- **Frame Accurate config. Comm.:** Frame accurate commands through TCP (commands delayed by ringbuffer and specified FAVC bias). Enable only for external control that supports special frame accurate command execution. Used with Frame Accurate Output.

Copyright © 2026 Vizrt

- **FAVC Bias:** Defines delay in fields, in addition to ringbuffer size, for frame accurate commands via TCP or GPI. This is the bias in frames for the commands, if **Frame Accurate Viz Communication.** is set to `On`. Allows negative values.
- **Command Field Dominance:** States when to handle the commands sent to Viz Engine. For example, when set to `Odd Retrace Counter`, Viz Engine handles all commands on odd fields. Not valid in progressive video output modes. Options are:
  - **Always.**
  - **Odd Retrace Counter:** Executes commands at an odd retrace counter.
  - **Even Retrace Counter:** Executes commands at an even retrace counter.
  - **Intern. Always/Extern. Odd:** Executes commands internally always and externally at an odd retrace counter.
  - **Intern. Always/Extern. Even:** Executes commands internally always and externally at an even retrace counter.
- **On Air Mouse Events:** Sets mouse events in On Air Mode to `On` or `Off`.
- **REST Webservice:** Sets the port number for communication with the Viz Engine REST Interface. The default is `0`, which deactivates the Webservice.
  - **Install:** Sets the port number and click **Install** to install the Webservice.
  - **Uninstall:** Removes the Webservice.

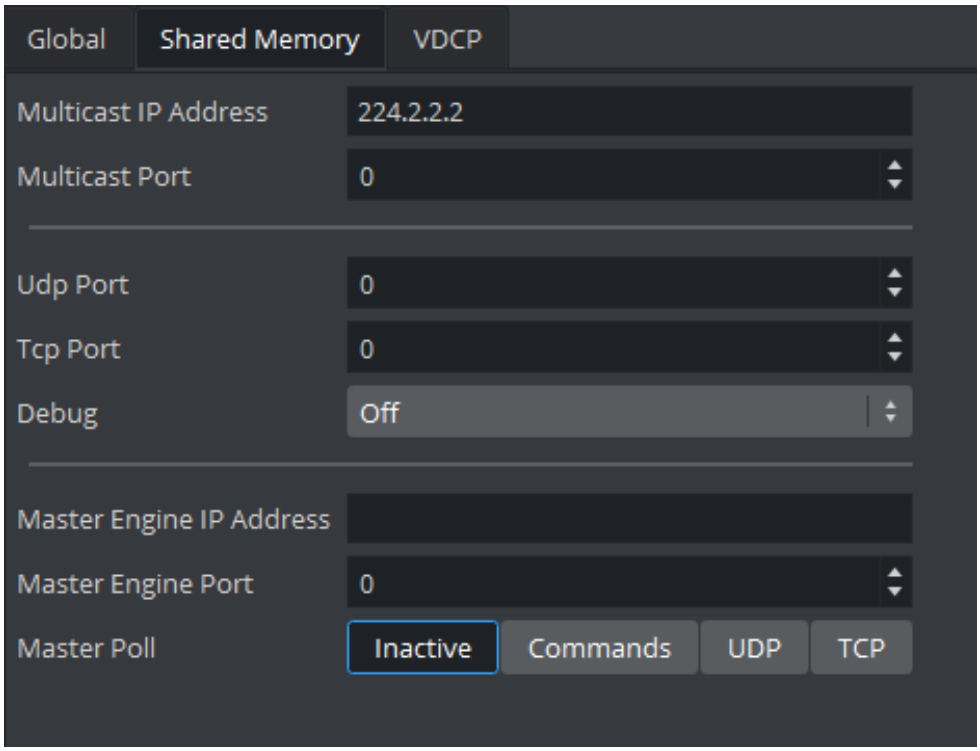
> ⚠ **Note:** To view the current documentation for the REST interface, go to *http://localhost:<port number>/#/documentation*.

> ⛔ **IMPORTANT!** The Webservice does not automatically install when User Account Control is active. Click **Install** to activate the Webservice.

- **Enable Logging:** Enables Webservice logging to the Viz Console.
- **GFX Port:** Sets the GFX Channel starting port number (GFX2: GFX Port+1, GFX3: GFX Port+2. For example: `55000`, `55001`, `55002`, etc.).
- **Super Channel Port:** Sets the Super Channel starting port number (SUPERCHANNEL2: Super Channel Port+1, SUPERCHANNEL3: Super Channel Port+2. For example: `56000`, `56001`, `56002`, etc.).

Copyright © 2026 Vizrt

8.8.2 Shared Memory Properties



- **Multicast IP Address:** Sets the address for synchronizing distributed shared memory map without a Graphic Hub.
- **Multicast Port:** Synchronizes shared memory between all Viz Engines listening to the multicast.
- **UDP Port:** Sets the UDP listening port for the shared memory input.
- **TCP Port:** Sets the TCP listening port for the shared memory input.
- **Debug:** Enables Shared Memory logging for UDP and TCP communication. Use Verbose to enable smm logging level with additional connection information.
- **Master Engine IP Address:** Sets the IP address of the master Viz Engine which holds the complete shared memory map (loaded during startup of Viz Engine).
- **Master Engine Port:** Sets the initializing port for the shared memory on startup (the command port of the master Viz Engine).
- **Master Poll:** Makes a Viz Engine load the shared memory map from the master Viz Engine, through the selected communication protocol. Available options are:
    - **Inactive**
    - **Commands**
    - **UDP**
    - **TCP**

Copyright © 2026 Vizrt

To Limit the Number of TCP Connections

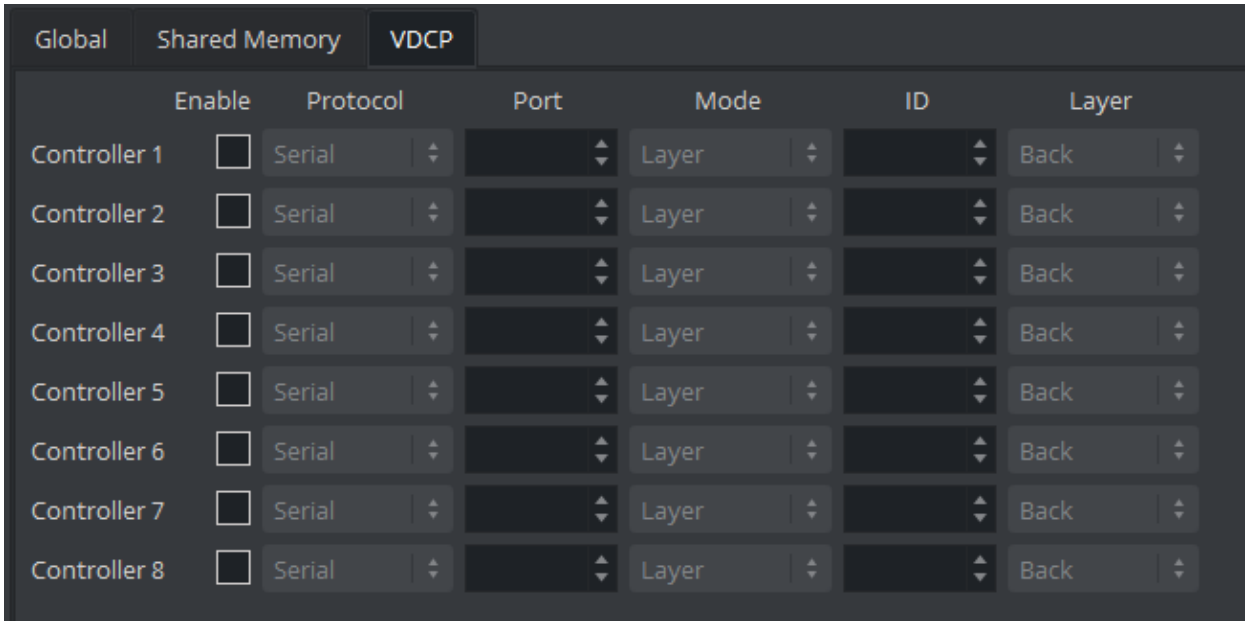
Since Viz Engine 3.3, the number of TCP connections to Viz Artist can be set in the Viz configuration. The maximum number of TCP connections is limited to 255. If set to `1`, the first control application connecting to Viz Artist/Viz Engine gets exclusive control over Viz Artist/Viz Engine.

1. Open the Viz Config file.
2. Under **SECTION COMMUNICATION** set '`max_tcp_connections`' to the number of TCP connections required.

This setting applies to the default port (`6100`) and the Multiplexing Ports.

8.8.3 VDCP Properties

The VDCP (Video Disk Control Protocol) tab enables the configuration of up to eight external controllers which can then have basic control over clip-channels and render-layers (Front, Main, Back).



Basic control is the ability to set a clip and to start, stop, pause or continue playback of a clip, or the scene animation in the specific layer.

- **Controller <1 to 8>:** Configure up to eight external controllers. Each Controller has these parameters:
  - **Enable:** Makes the Controller connection Active or Inactive.
  - **Protocol:** Selects a protocol (TCP/IP or Serial connections) for the external control device.
  - **Port:** Sets a port number where an external VDCP client can connect to the specified VDCP controller of the Viz Engine.
  - **Mode:** Select a mode:
    - **Clip Channel:** Enables the control of a selected video clip channel.
    - **Layer:** Enables the control of animation in a layer.

Copyright © 2026 Vizrt

- **ID:** Selects a Clip channel (an ID between `1` and `16`) to control. Available if **Mode** is set to **Clip Channel**.
- **Layer:** Selects the layer. Available if **Mode** is set to **Layer**. Available options are `Back`, `Main` or `Front`.

See Also

- Viz Engine Shared Memory
- **VizCommunication.Map** (see the Viz Artist User Guide)

Copyright © 2026 Vizrt

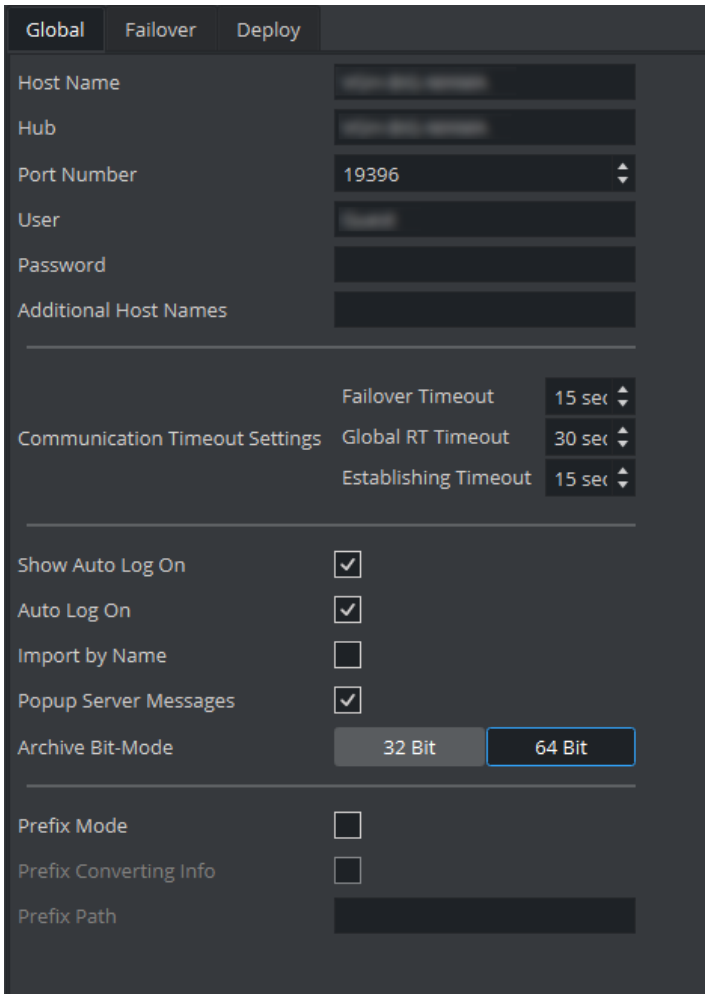
8.9 Database

The Database section has three tabs for setting connections (a Graphic Hub, Failover servers and Deploy servers).

- Global Properties
- Failover Properties
    - To Add Redundant Servers to the Failover List
- Deploy
    - To Add Deploy Servers

8.9.1 Global Properties

Use this section to configure the Graphic Hub database connection settings.



Copyright © 2026 Vizrt

- **Host Name:** Displays the name of the Graphic Hub naming service. The naming service is always a one to one map to the Host Name of the machine running a Graphic Hub.
- **Hub:** Displays the Graphic Hub server name.
- **Port Number:** Displays the listener port number for a Graphic Hub. The default port number is `19396`, and should normally not be changed.
- **User:** Sets the default user.
- **Password:** The password for the default user.
- **Additional Host Names:** Displays the host name or IP address of a Graphic Hub database located on a different subnet, to make it selectable in the log onscreen database drop-down list. Multiple entries can be added, and must be separated by a semicolon. Hosts located on different subnets are highlighted in turquoise in the drop-down list.
- **Communication Timeout Settings:**
  - **Failover Timeout (sec):** Sets the maximum time to wait before a fail over is initiated from the main to the replication Graphic Hub.
  - **Global RT Timeout (sec):** Sets the maximum response time for any request to a Graphic Hub.
  - **Establishing Timeout (sec):** Sets the maximum waiting time to establish a connection to a Graphic Hub.
- **Show Auto Log On:**
  - **Yes:** Shows the **Auto Log On** check box in the Graphic Hub log on window.
  - **No:** Hides the **Auto Log On** check box in the Graphic Hub log on window.
- **Auto Log On:** Enables or disables automatic log on to a Graphic Hub. This disables the log on screen for Viz Artist/Engine.
- **Import by Name:** Checks for objects by name rather than by UUID when set to `Yes`. Checks by UUID when set to `No`.
- **Popup Server Messages:** Enables or disables popup server messages. Disabled only works on local host.
- **Archive Bit-Mode:** Sets the bit-mode in which the archive is saved.

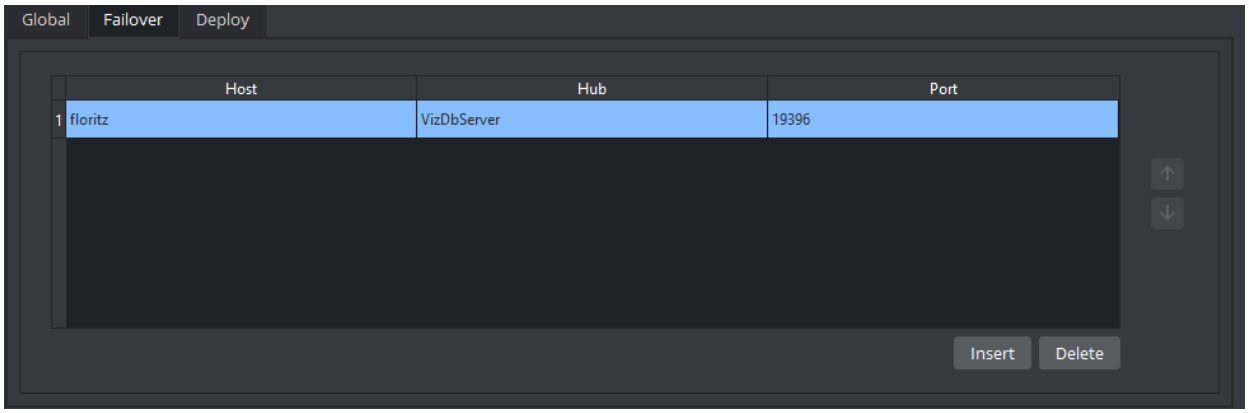
Note: For compatibility, 32-bit should be enabled if scenes are imported to Viz Artist versions prior to build 2310.

- **Prefix:** Needed if an external control application is used that sends commands containing certain path locations, but where the path of the files is a different one on the Graphic Hub (because they were deployed to a specific location).
  - **Prefix Mode:** Activates the prefix mode.
  - **Prefix Converting Info:** Shows the prefixed (final) paths in the console.
  - **Prefix Path:** Contains the prefix path string that is used for incoming commands containing path parameters.

IMPORTANT! Every time the Viz Engine connects to a Graphic Hub, it creates a mapping of all GH REST connections. This mapping is used to speed up images loading by directly downloading images from the Graphic Hub instead of using the GH REST. If the connected GH REST is reconfigured to a different Graphic Hub, then the mapping is not updated until the Viz Engine disconnects and reconnects to Graphic Hub.

Copyright © 2026 Vizrt

8.9.2 Failover Properties

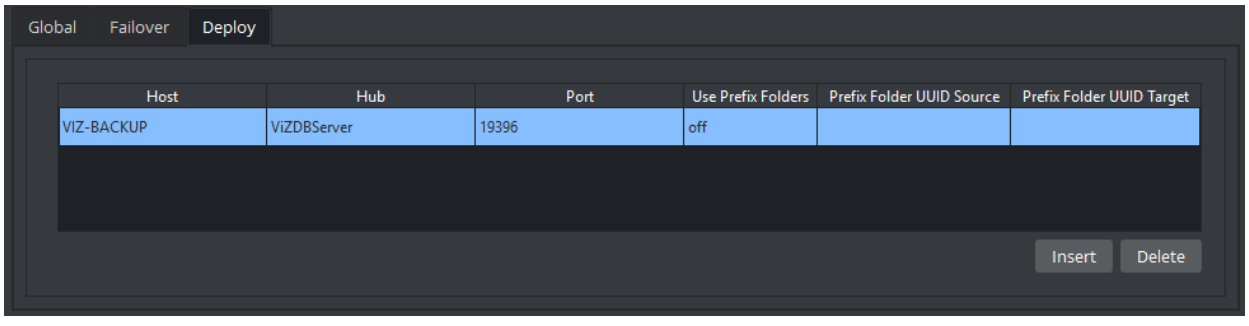


To Add Redundant Servers to the Failover List

1. Click **Insert**.
2. Enter the **Host Name** of a Failover server.
3. Enter the Graphic **Hub** instance for Failover.
4. Enter the **Port** number of the Graphic Hub for Failover.
5. Use the **Up** and **Down** buttons in the database Failover list to raise and/or lower a database's priority in the event of Failover.
6. Click **Save**.

8.9.3 Deploy

To Add Deploy Servers



Note: To configure deploy servers, the user you are currently logged in with on the source server must exist on the destination server with the same credentials.

1. Click **Insert**.

Copyright © 2026 Vizrt

2. Enter the **Host Name** of a deploy server.
3. Enter the name of the deploy Graphic Hub.
4. Enter the **Port** number of the deploy Graphic Hub. The default port is `19396`.
5. If specific source and destination folders are required, set **Use Prefix Folders** to `Yes`.
   Use Prefix Folders: Provides a simple server copy functionality that is able to rebuild the underlying structure from a source folder on the source server to a destination folder on the destination server. All references outside the source folder are rebuilt according to the original structure, under */data/*. Set to `Yes` to select a source and destination folder for the deployment. Set to `No` to keep the structure the same as on the source server. See To Add Deploy Servers below for an example setup.
   - **Prefix Folder UUID Source:** Displays the source folder UUID on the Graphic Hub to deploy from.
   - **Prefix Folder UUID Target:** Displays the target folder UUID on the Graphic Hub to deploy to **Use Prefix Folders.** Provides a simple server copy functionality that is able to rebuild the underlying structure from a source folder on the source server to a destination folder on the destination server. All references outside the source folder are rebuilt according to the original structure, under */data/*. Set to `Yes` to select a source and destination folder for the deployment. Set to `No` to keep the structure the same as on the source server. See To Add Deploy Servers.

   a. Enter the UUID of the **Prefix Folder UUID Source**.
   b. Enter the UUID of the **Prefix Folder UUID Target**.
6. Click Save.

See Also

- Graphic Hub User Guide

Copyright © 2026 Vizrt

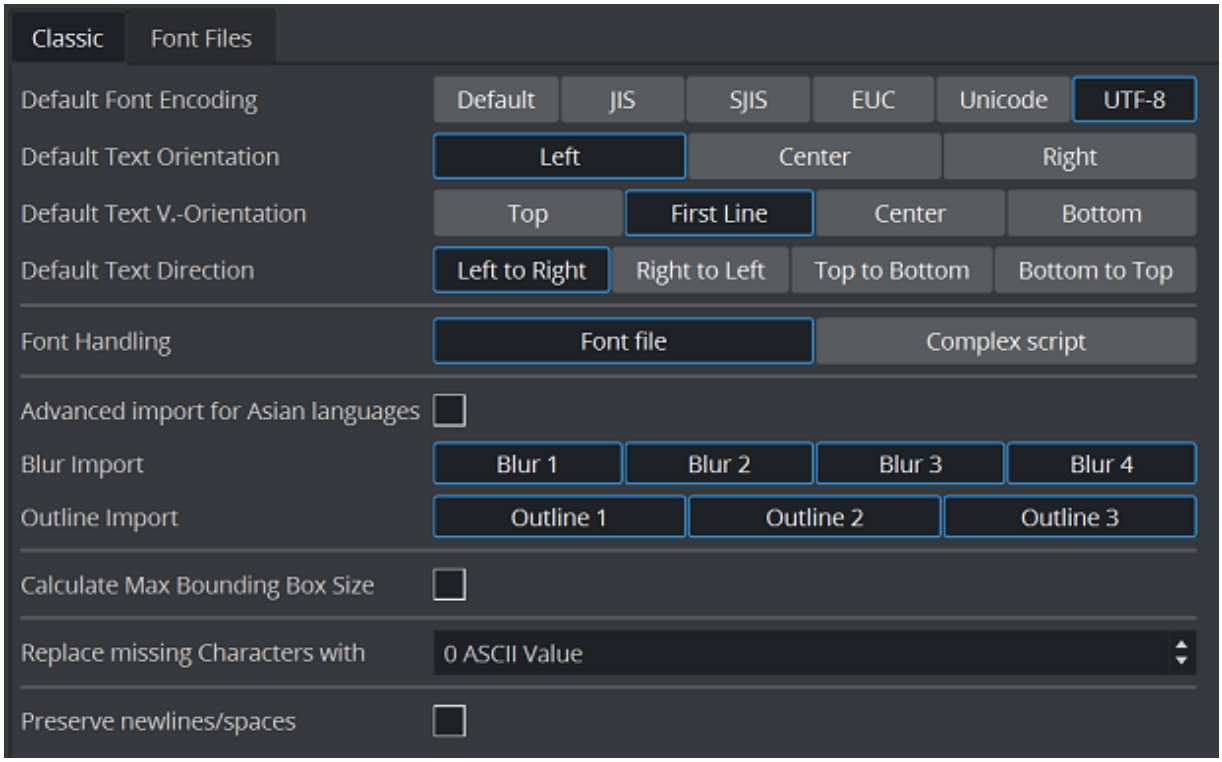
8.10 Font Text Options

In this section, there are two tabs.

- Classic
- Viz Engine

8.10.1 Classic

In this tab, the font and text options for the Classic Render Pipeline can be configured.



- **Font encoding**. Available options are:
  - **Default:** Sets the font encoding to single character interpretation (limited to 255).
  - **Japanese Industry Standard Code (JIS):** Sets Japanese industry standard code character encoding.
  - **Shifted Japanese Industry Standard Code (SJIS):** Sets the newer Shift JIS character encoding standard which sets aside certain character codes to signal the start of a two-character sequence.
  - **Extended Unix Code (EUC):** Sets Extended Unix Code (EUC) character encoding that is a multi byte character encoding system used primarily for Japanese, Korean, and simplified Chinese.
  - **Unicode:** Sets the Unicode character encoding where every two characters are inter-operated as one (not widely used).
  - **UTF-8:** Sets UTF-8 (8-bit UCS/Unicode Transformation Format) character encoding that is a variable-length character encoding for Unicode (default).
- **Default Text Orientation:** Sets the default horizontal text orientation.

Copyright © 2026 Vizrt

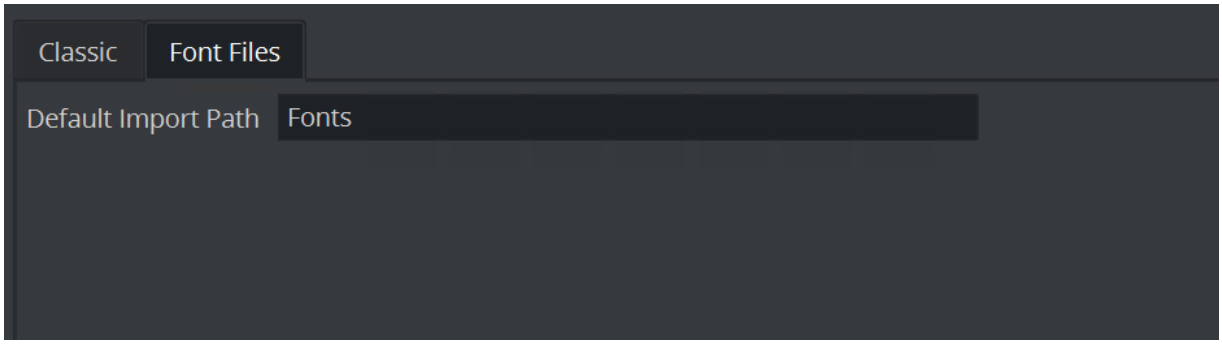
- **Default Text V. Orientation:** Sets the default vertical text orientation.
- **Default Text Direction:** Sets the default text direction.
- **Font Handling:** Defines how fonts are handled.
  - **Font file:** Sets the font kerning to Font file that is mainly used for languages written from left to right.
  - **Complex script:** Sets the font kerning to Complex script. Complex script is mainly used for languages written from right to left, or when one character is composed of one or several glyphs.

IMPORTANT! With font handling in Complex Script mode, all fonts used must be installed in Windows to avoid unpredictable text behavior.

- **Advanced Import for Asian Languages:** Imports fonts to be used with special features (including vertical text and EUDC) when enabled. This feature is useful for some Asian languages. The fonts must be installed in the operating system.
- **Blur Import:** Enables blur levels for imported fonts. These options are used for Classic fonts only.
- **Outline Import:** Enables outline levels for imported fonts. These options are used for Classic fonts only.
- **Calculate Max Bounding Box Size:** Enables Viz Artist/Engine 3.x to calculate the bounding boxes as they were calculated in older versions. In Viz Artist/Engine 3.x a text object's bounding box height increases if a capital character is entered (for example, an umlaut (double dots)). In Viz Artist/Engine 2.x, the bounding box height was always the same and independent of the characters in the text object. Available options are Active and Inactive. Default is Inactive (false).
- **Replace missing Characters with:** Replaces a missing font character in a font file with a default font character. The Unicode value refers to the decimal value of the replacement character in the Unicode table (valid values are `0-65553`). Normal usage would select a (42) or (95).
- **Preserve newlines/spaces:** Does not remove spaces and newlines at the end of the text when saving and reloading a scene when set to `Active`. This influences bounding boxes. The default state is *Inactive*.

8.10.2 Viz Engine

In this tab, Font and Text options for Viz Engine Render Pipeline can be configured.



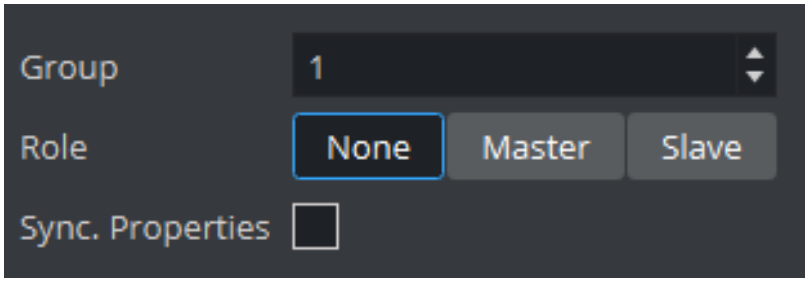
- **Default Import Path:** A default location in Graphic Hub to which the font importer imports new FONT_FILE(FL) or FONT_FACE(FF).

Copyright © 2026 Vizrt

8.11 Global Input

The Global Input settings influence the generation and handling of Six Degrees of Freedom (6DoF) messages that can be distributed to several Viz Engines.

6DoF is used when working in 3D space in combination with special input devices such as a mouse. A mouse uses two coordinates (xy) which Viz Artist/Engine is able to translate into three coordinates (xyz) based on a grid.



- **Group:** Defines which multicast group the generated or received messages belong to. If more than one group is to be defined, a unique Group number must be set for each group
- **Role:** Defines how messages are generated and processed. Alternatives are; None, Master and Slave.
  - **None:** Generates and processes messages on the local Viz Engine only.
  - **Master:** Creates messages for itself and the defined group.
  - **Slave:** Reads and processes 6DOF messages, but is not allowed to create them.
- **Synch. Properties:** Makes the synchronization of Viz Engine Scene properties `Active` or `Inactive`.

8.11.1 To Synchronize Multiple Viz Engines

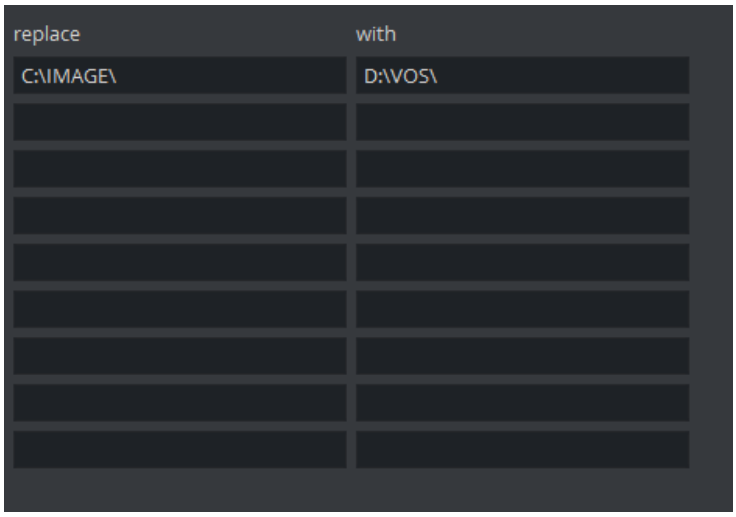
1. Start Viz Config on all involved render machines.
2. Set the same Group ID for all Viz Engines.
3. Set **Synch. Properties** to `Active`.
4. Save and close Viz Configuration on all machines.
5. Open the Control Panel on all render machines.
6. Deactivate all unused network connections.
   - Viz Engine always uses the first network connection setup by the Windows operating system.
   - Synchronized engines work within the same network segment only because it is using multicast, hence, it is important to use the right connection.
   - The first connection can be determined by setting a manual metric in Windows: see http://support.microsoft.com/kb/299540.
7. Start all Viz Engines again.
8. Create a simple test scene with a geometry and the Synchronized Properties plug-in (**Built Ins > Container > Global**) on the same container.
9. Save the Scene.
10. Open the Scene on all involved Viz Engines.
11. Move the geometry on one Viz Engine. All the other Viz Engines show the same object movement.

Copyright © 2026 Vizrt

8.12 Image File Name Convert

In this section, nine different replacement configurations can be set for image file names.

This can be useful if you need to remap certain locations, and is used by Viz Engine if an image cannot be found at the original location. The given replacements are used from first to last until the image can be loaded.



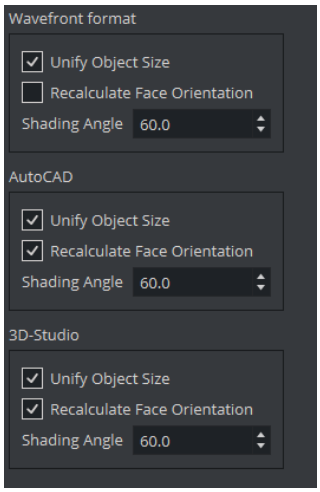
- **Replace:** Defines the string to be replaced.
- **With:** Defines the replacement string.

Example: If an image file name starts with the string `Replace` this part of the file name can be replaced by the string `With`.

8.13 Import 3D Options

In this section, parameters which influence the import of 3D objects can be configured. There are three different formats:

- Wavefront
- AutoCAD
- 3D-Studio

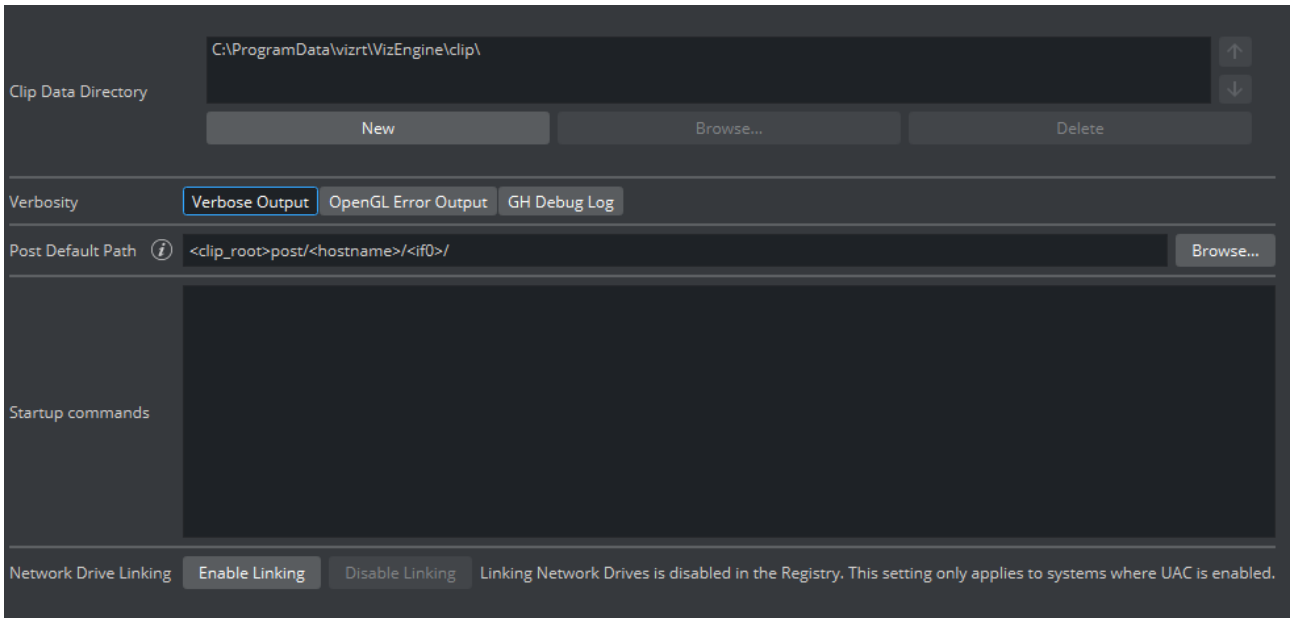


- **Unify Object Size:** Recalculates all vertices during import in a way that the object center is moved to the origin (0,0,0), and the size of the object is 100 cm in its largest extent when enabled. If disabled, all vertices retain their values as defined in the original file. An object could appear invisible in a Viz Artist scene because the object is translated a lot from the origin or is scaled up or down a lot. It may be necessary to deactivate the unification to be able to recombine several separately imported objects that must keep their size and relative position.
- **Face Orientation:** Rearranges the orientation of object faces during import when enabled. Polygonal 3D models often do not have a consistent face orientation, but for performance reasons, Viz Engine expects that all faces of an object point to the same direction.
- **Shading Angle:** Recalculates the normals from the geometry to make lighting possible, if the 3D object has no normal vector information. This recalculation is influenced by the shading angle, which acts as a threshold between sharp and soft edges.

Note: A shading angle value of 60 means that an edge between two faces is considered to be a soft edge for angles below 60 and a sharp edge above this level. `60` is the default shading angle.

Copyright © 2026 Vizrt

8.14 Local Settings



- **Clip Data Directory:** Sets the clip directory (default directory is *D:* drive). Multiple directories can also be selected (see Select Multiple Directories).

> **Note:** This directory is also used for Viz One installations, as the root parameter for the Fsmon and Mediaftp services (see Viz One).

- **Verbosity:** Selects content of log files, if no selection is made only the default content of the log file is created:
  - **Verbose Output:** Enables the most information in the Viz Engine Console.
  - **OpenGL Error Output (AMD):** Enables default content of the log file with the addition of OpenGL Error Output (AMD).
  - **GH Debug Log:** Enables default content of the log file including Write GH Connection Log.
- **Post Default Path:** Sets the default location of **Render to Disk** clips.
- **Startup commands:** Saves commands that are executed on startup, but after the initial setup configuration and before the main render loop activates. Examples:

```
FEEDBACK*CLIENT ADD localhost 2001
FEEDBACK*COMMAND ADD localhost 2001 CLIPOUT
RENDERER*MAIN_LAYER SET_OBJECT Vizrt_RD/mra/Reference/TC/TC_DISPLAY
MAIN*DEBUG_CONTROL*RENDERINFO*PERFORMANCE SET 1
```

Copyright © 2026 Vizrt

Note: Log files are written to the *<viz data folder>* directory, normally *C:\ProgramData\Vizrt\VizEngine*. This directory is by default hidden in Windows, so to navigate to this directory in Windows Explorer specify the explicit path. For more information see Viz Log Files.

- **Network Drive Linking:** Enables to allow Viz Artist access to mapped network drives on certain systems where UAC is enabled. A machine reboot is required to apply the change.

8.14.1 Select Multiple Directories

More than one clip directory can be selected. When more than one directory path is selected in the **Clip Name** box, change the file path to *<clip_root>*. The included directories are parsed when executing the search, returning video clips with file names matching the search criteria.

Example: Search for a video clip named *next_tuesday.avi* in *D:\AEClips\AFL\Promos*. In the **Clip Name** box, change *D:\AEClips\AFL\Promos* to *<clip_root>*. The filename displayed in Viz Artist should be *<clip_root>\next_tuesday.avi*.

Note: The directory has to match the directory set when the Mediaftp service for video transfer from Viz One was installed.

Multiple directories can be used with:

- Dual Channel and Trio Box CG configurations, or
- With any Viz Artist/Engine configuration for the selection of a secondary video clip directory if a directory fails.

If the file path for a video clip is set to *<clip_root>* and the first directory fails, the same video file is searched for in the next directory in the Clip directory list. If the **Clip Name** box is used to search for a video when multiple Clip directories are selected, the search defaults to the last used directory.

8.14.2 SAM SDC01, SDC02, and SDC03 Protocols

To enable or disable a supported SAM protocol please change the `SAMSDC0x_enable`. By default those protocols are switched off. The listening is changed with `SAMSDC0x_address`. If one needs to configure multiple instances those listening addresses need to be unique among the Viz Engine instances.

SAM SDC01

```
## enable the SAM SDC01 protocol. This is disabled by default.
#* SAMSDC01_enable: Default=0
SAMSDC01_enable = 1
## the SAM SDC01 protocol listening endpoint.
#* SAMSDC01_address: Default=0.0.0.0:2055
# SAMSDC01_address = 0.0.0.0:2055
```

and

Copyright © 2026 Vizrt

SAM SDC02

```
## enable the SAM SDC02 protocol. This is disabled by default.
#* SAMSDC02_enable: Default=0
SAMSDC02_enable = 1
## the SAM SDC02 protocol listening endpoint.
#* SAMSDC02_address: Default=0.0.0.0:2056
# SAMSDC02_address = 0.0.0.0:2056
```

and

SAM SDC03

```
## enable the SAM SDC03 protocol. This is disabled by default.
#* SAMSDC03_enable: Default=0
SAMSDC03_enable = 1
## the SAM SDC03 protocol listening endpoint.
#* SAMSDC03_address: Default=0.0.0.0:2057
# SAMSDC03_address = 0.0.0.0:2057
```

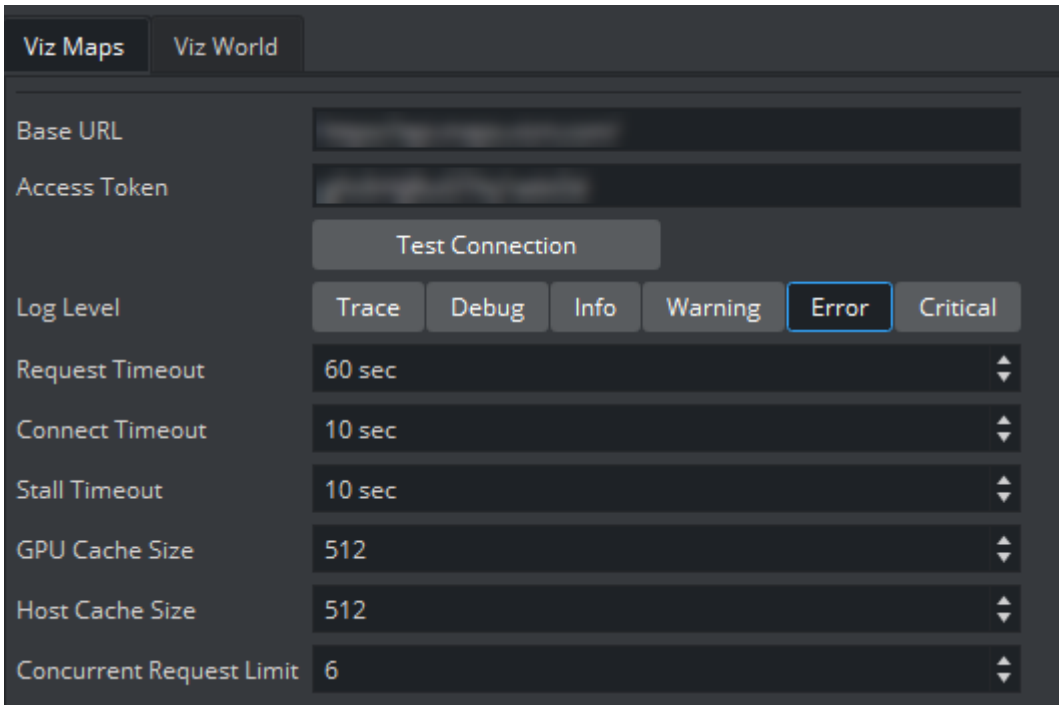
See Also

- **Clip Properties** in the Viz Artist User Guide

Copyright © 2026 Vizrt

8.15 Maps

8.15.1 Viz Maps



Configures the Viz Maps backend access.

Base URL and **Access Token** are usually provided by Vizrt Support. This si the backend access URL and a Token.

Log Level defines how many information is being logged.

Timeout settings:

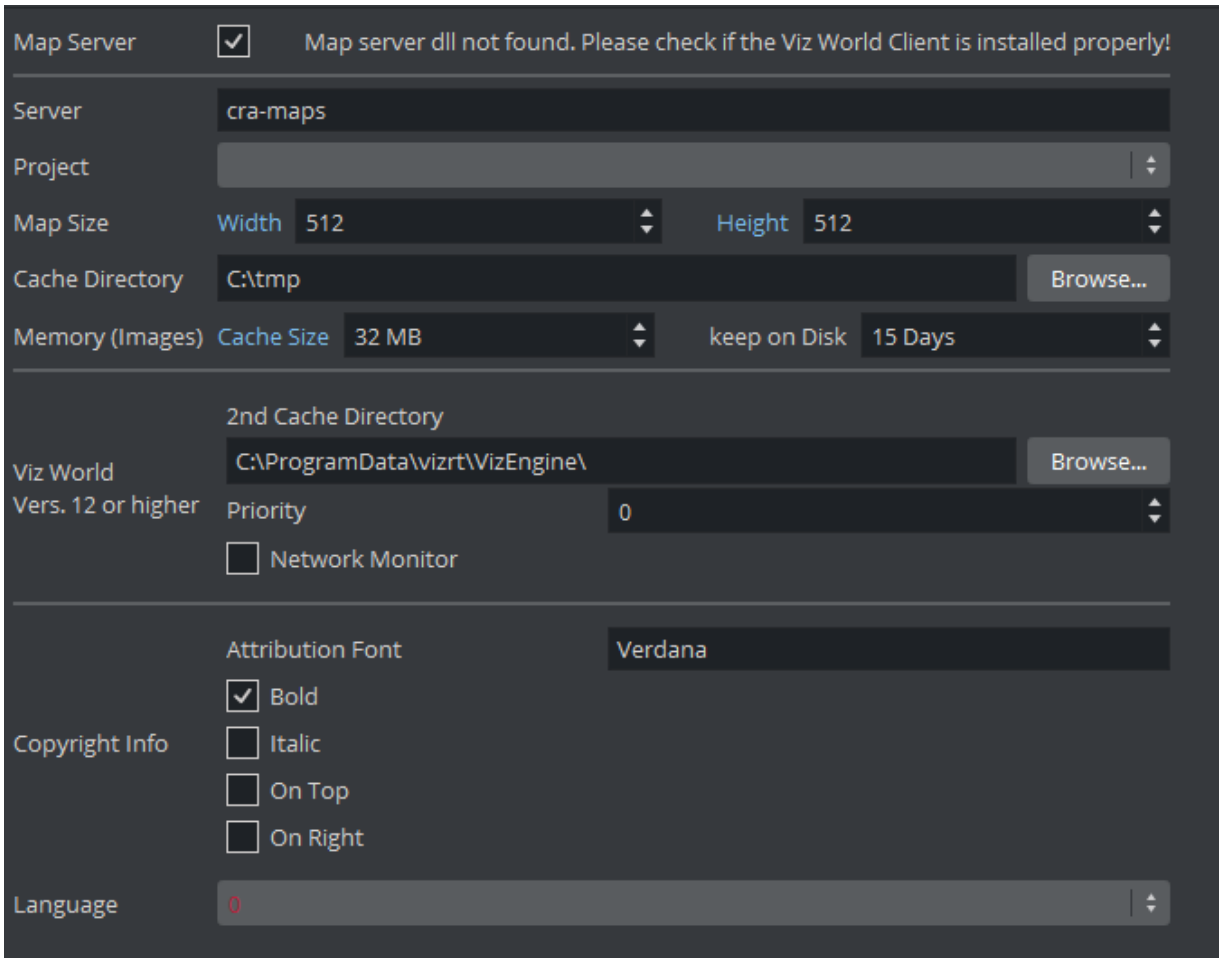
Defines the timeout in seconds how long to wait for requests, connections and stalls.

GPU Cache size defines how many MegaBytes of GPU memory are reserved to hold maps data. The same also applies to the **Host Cache Size**, which is the computers RAM memory.
Setting any of these to 0 means disabling the cache (not recommended).

Concurrent Request Limit defines how many active tasks are working on downloading data.

Copyright © 2026 Vizrt

8.15.2 Viz World



- **Map Server:** Enables or disables Viz World Server (WoS) connection for Viz World Client (WoC). If there is a problem with the Viz World installation, an error message is displayed next to the `Active/Inactive` button.
- **Server:** Sets the Viz World Server host.
- **Project:** Sets the default map project that opens with the client application.
- **Available:** Lists all available Viz World Server projects.
- **Map size:** Sets the default map size that is used with the client application.
- **Cache Directory:** Sets the cache directory for cached maps which can be a local drive, mapped drive or a Universal Naming Convention (UNC) path.

IMPORTANT! Make sure the Cache Directory folder is configured with read and write access rights.

- **Memory (Images):** Sets the number of images to keep in memory.
- **On disk (Days):** Sets the number of days to save images on disk.

Copyright © 2026 Vizrt

- **2nd Cache Directory:** Enables a second cache (see Cache above). The main purpose of the second cache is to enable redundancy in those cases where a main cache directory is on a different computer and for some reason fails. Another use case is to use it as a local cache to save loading time in more complex operations. To shorten load time, copy large static files to the correct local cache folder. In the *3D Map Setting* plug-in, there is a **Sync Local Cache Folder** button which copies all required files to the local cache. Note that the second cache directory settings can only be used by Viz World version 12.0 and later.
- **Priority:** Sets the machine's connection priority to the Viz World Server (Wos). Setting a number, where `1` is the lowest and `100` is the highest you may override connection priorities set by other machines. The configuration interface allows you to prioritize client connections from Viz Artist and On Air Viz Engine's used for preview and program output. Viz Engines must be in On Air mode for them to be prioritized. For Viz World Map Editor you can set it from its context menu. To enable this behavior on the server side you need to enable WoS to prioritize its connections/log ins. For more information, see the Viz World User Guide, *Server Launcher Configuration* section. If the configuration option is not visible, add the VizWorld.ini file and set the priority.
- **Network Monitor:** Enables you to monitor relevant network connections (server and cache folders). If you do not monitor the network and you try to connect over a disconnected network, connecting to a server or a UNC path, it takes time before the system reports back (for example, 30 seconds or more). Enabling network monitoring avoids such connection issues. Note that the network monitor only monitors a cache folder that uses a UNC path (not mounted/mapped drives). If the configuration option is not visible you need add the VizWorld.ini file.
- **Attribution Font:** Sets the font for the attribution.
- **Bold:** Sets the attribution font to **bold**.
- **Italic:** Sets the attribution font to *italic*.
- **On Top:** Places the attribution image to the top in the screen. Default is `bottom`.
- **On Right:** Places the attribution image to the right in the screen. Default is `left`.
- **Language:** The default language of Map Server.

8.15.3 To Add the VizWorld.ini File

1. Create and save a *VizWorld.ini* file to the following location: *<viz install folder>\plugin\data\maps*.
2. Open the file and enter the following:

```
Monitor=1 Priority=1 Language=[my Language ID]
```

3. **Save** the file and start, for example Viz Config, to see the configurable parameters.
   a. `Monitor` enables network monitoring. For more information see the **Network Monitor** setting under the Maps section.
   b. `Priority` sets Viz connection priority to the Viz World Server. For more information, see the Viz World User Guide.
   c. `my Language ID` refers to the order of languages in your list of languages (for example, English = `0`, Arabic = `1`, Hebrew = `2` and so on).

For more information see the **Language** setting under Maps.

Copyright © 2026 Vizrt

8.16 Matrox

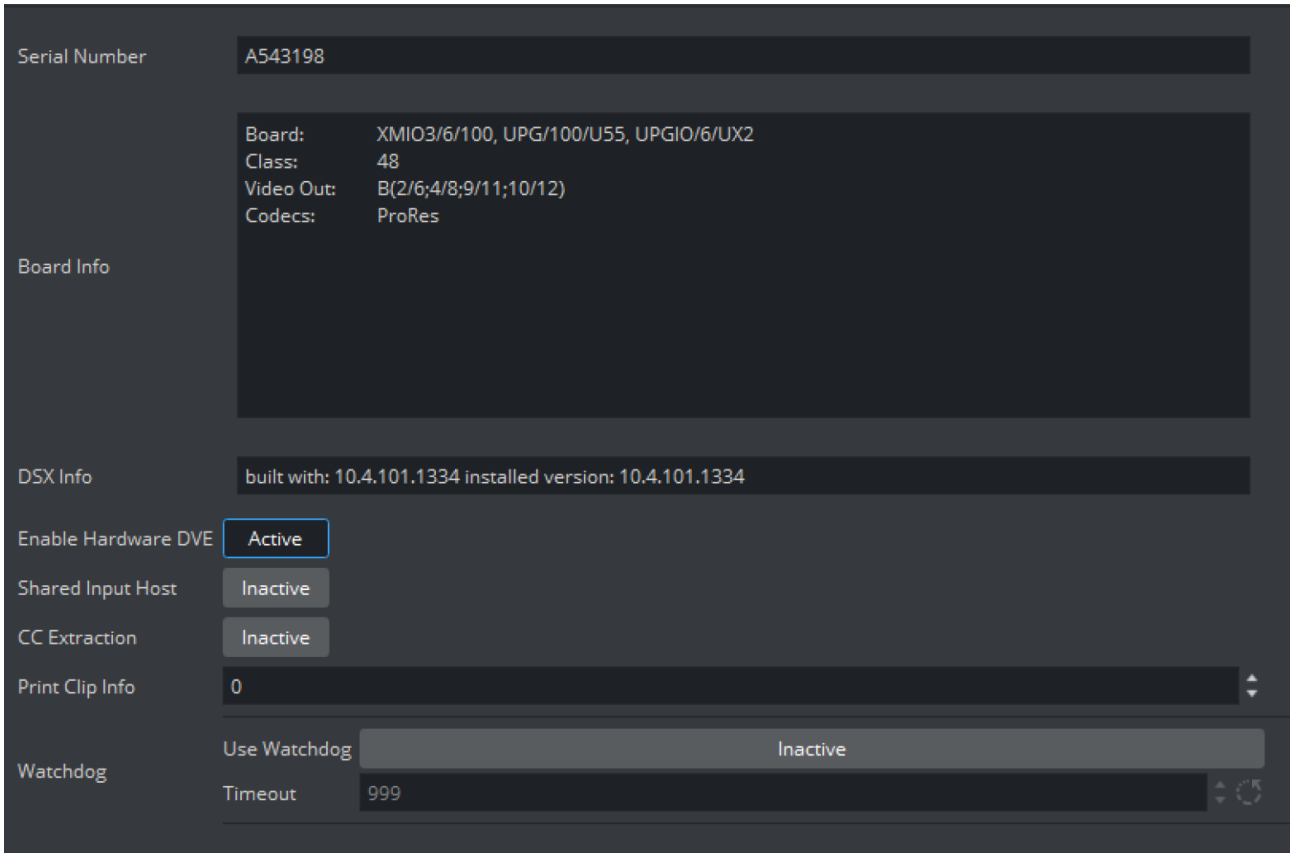
In the Matrox section, assign Matrox Input and Output channels to Viz Engine Input and Output channels. The GUI shows a drop down menu for the configurable parameters. The parameters available are dependent on the installed hardware.

This page provides information about the following:

- General Properties
  - Configuration File Only
- M264 Encoder/Decoder Boards

8.16.1 General Properties

The General Properties Panel shows information about the installed hardware.



- **Serial No.:** Shows the serial number of the installed Matrox board.
- **Board Info:** Shows the model and type of the Matrox board.
- **DSX Info:** Shows the software version and driver version.
- **Enable Hardware DVE (former Fast Texture Mode):** Disable to shorten the 'in out' delay in Texture Mode to a minimum.

Copyright © 2026 Vizrt

Note: If **Enable Hardware DVE** is set to `Inactive`, **DVE** does not work (see **Video Clip Playout Considerations** and **Video Playout** in the Viz Artist User Guide).

- **Shared Input Host:** Used to share the input with other Viz Engine instances.
- **CC Extraction:** Enables or disables the closed captioning extraction for Matrox X.mio3 - DSX LE 4.
- **Print Clip Info:** Enables printing of clip information to the console when activated. However, such information may cause the render loop to stall. Default mode is *Inactive*.
- **Watchdog:** Sets a timer that allows a system to continue video pass-through during an application crash or system failure (see also Watchdog (**Matrox X.mio3 Series**) and Video Board):
  - **Use Watchdog:** When set to **Active** enables the Matrox X.mio watchdog feature. It passes the input signal to a hardcoded output port as soon as Viz Engine is unresponsive. Default mode is `Inactive`.
  - **Timeout:** Sets the time, in milliseconds, until the watchdog takes over control. This value should not be smaller than the time of two fields/frames. Default value is `999 ms`.

Note: Use Watchdog and **Timeout** can also be set and changed in Video Board.

- **Genlock**
  - **Use Flywheel:** Adopts a tracking mode if the genlock signal is interrupted or lost that maintains the signal frequency until the source genlock signal is regained when activated. Default mode is *Active*.
  - **Max Recovery Time:** Represents the time in milliseconds (ms) provided to the flywheel to attempt to regain the genlock before an abrupt jump to the locked state is performed. Default value is `15`.
  - **Max Unclock Time:** Represents the time in milliseconds (ms) provided to the flywheel to remain in the unlocked state before switching to the free running state. Default value is `15`.

Configuration File Only

It is possible to increase the delay of the output to `1` frame to mitigate performance related frame drops that are indicated by *Buffer has been dropped, cancelled late* log messages. Increasing the delay to `2` or more frames does not have additional performance benefits.

This setting is only relevant with boards that do not have a hardware mixer (DSX LE series), or if using that mixer was disabled in the configuration. The most common way to disable the mixer is to **Enable Hardware DVE** mode.

Set *Matrox0.VideoOut1.NodeFrameDelay = 0* to values other than `0`.

Note: This value is set to at least `1` internally for UHD outputs on X.mio3 and DSX LE4 cards.

8.16.2 M264 Encoder/Decoder Boards

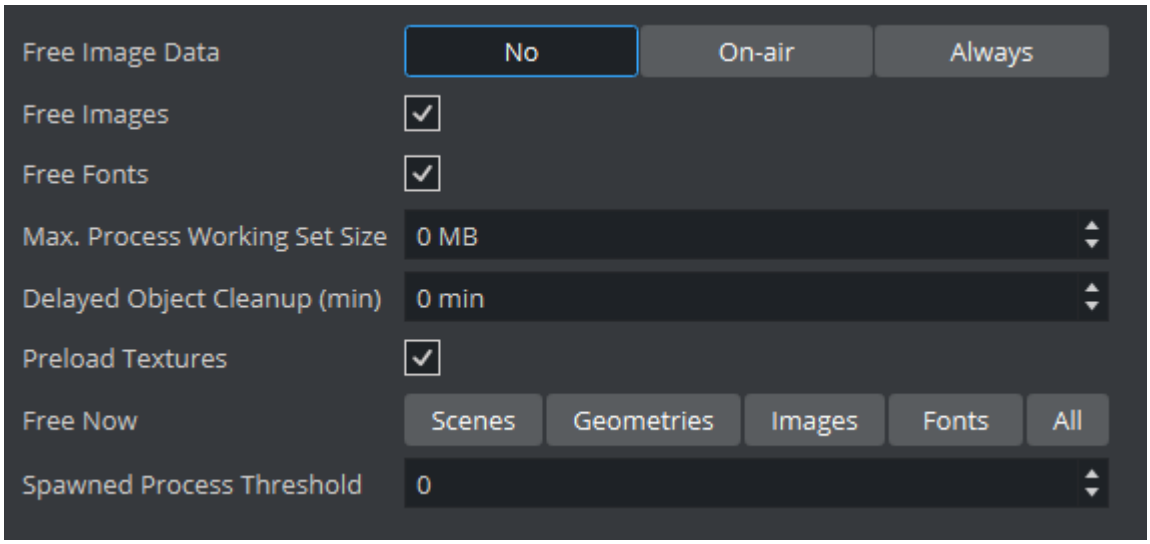
M264 encoder/decoder boards are usually configured automatically. If they do not appear in the configuration section, please double check the Matrox.Devices entry in the **MATROX_CONFIG** of the *viz.cfg* file.

It should look like *Matrox.Devices = <device1>,<device2>*.

Copyright © 2026 Vizrt

Depending on your model(for example, a M264 S3, which supports three streams), the entry would look like: *Matrox.Devices = <device1>,<m264device>,<m264device>,<m264device>*.

8.17 Memory Management



Application memory management involves supplying the memory (main memory and graphic card memory) needed for a program's objects and data structures used for in-memory objects such as images, fonts and so on from the limited resources available. Memory management also recycles memory for reuse when required and appropriate.

In the Memory Management section of the configuration, you can give detailed hints to the Viz Engine how memory should be handled:

- **Free Image Data:** Frees image data after texture creation when enabled (`On Air` or `Always`).
  - **No:** Disables the Free Image Data option. This option is faster, but requires a lot of memory.
  - **On Air:** Frees image data when in On Air mode, but not in Viz Artist mode.
  - **Always:** Frees image data every time after the texture was created. This option saves a lot of memory but is slower in case of texture rebuilds.

> ⚠ **Note:** If an image is modified, then its texture is rebuilt faster if the data already lies in the main memory (instead of re-loading it from the database).

- **Free Images:** Removes unused images (not referenced in a loaded scene) from the Image Pool (main memory and graphics card memory) when enabled (*On*).
- **Free Fonts:** Removes unused fonts from the Font Pool when enabled (*On*).
- **Max. Process Working Set Size (MB):** Attempts to automatically unload unused Pool objects once the threshold has been reached. For example, setting it to `6*1024` starts to cleanup as soon as Viz Engine uses more than 6 GB of memory. `0` means, no cleanup happens.
- **Delayed Object Cleanup (min):** Sets the delay to clean up pool objects. Default: `0` minutes which effectively disables the delayed clean up. If set to a value greater than `0` , objects do not immediately get deleted when unloaded from the renderer, and keeps objects in memory for subsequent use. While this improves performance for certain scenarios, it increases the memory footprint of Viz Engine.

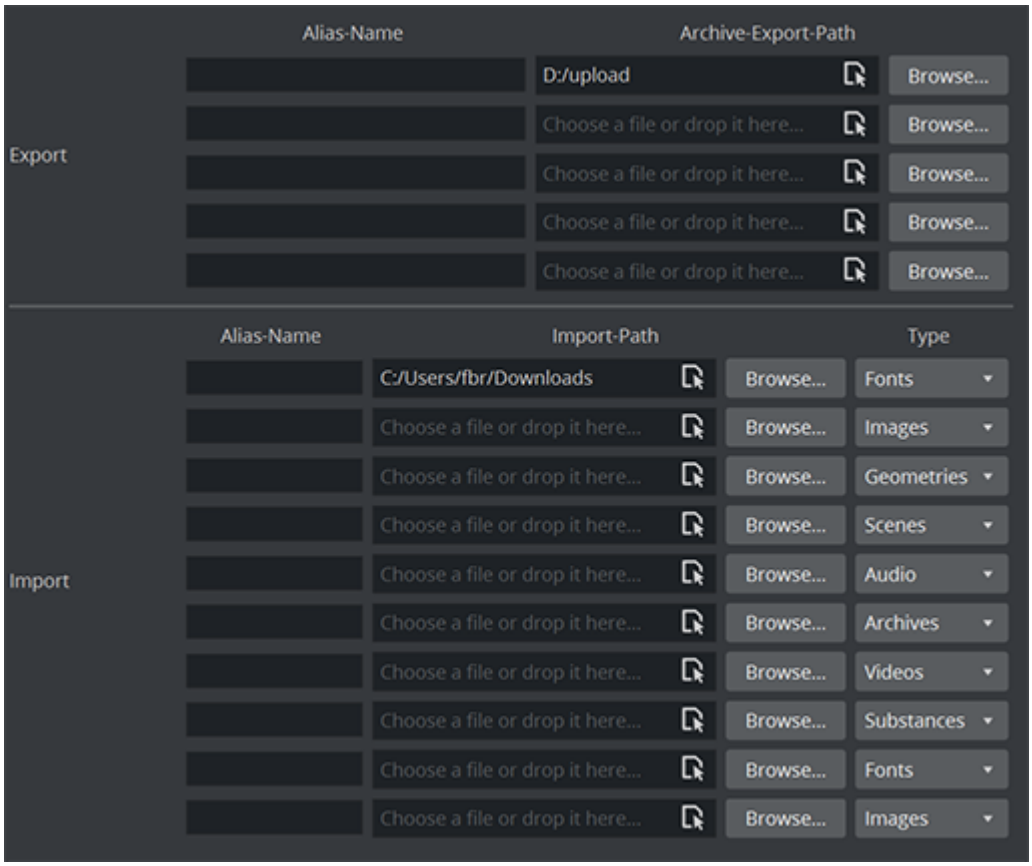
Copyright © 2026 Vizrt

- **Preload Textures:** Loads all images which are to be loaded with a Scene (they do not need to be rendered) as textures on the graphics card when enabled (`On`). This eliminates the texture creation time during rendering afterwards (for example, useful when initializing a show or a playlist). Default is enabled (`On`).
- **Free Now:** Frees the selected unused Pool objects (Scenes, Geometries, Images, Fonts or All) from the memory.
- **Spawned Process Threshold:** Sets the maximum number of child processes spawned by Viz. If the number of child processes exceed the set value, a pool cleanup is automatically triggered. When set to `0` , automatic cleanups are disabled. The default value is `0` .

Copyright © 2026 Vizrt

8.18 Path Aliases

In this section, up to five favorite archive and ten import paths can be set. These are accessible through Viz Artist's Archive and Import panes.



Name: Sets the path alias name for the archive or import path.

- **Path:** Sets the archive or import path which can be a local drive, mapped drive or a Universal Naming Convention (UNC) path.

IMPORTANT! Make sure the archive folder is configured with read and write access rights.

8.18.1 To Add a Path

Drag and drop a directory from Windows Explorer to the Paths or browse manually by using the **Browse** … button.

1. Enter a descriptive name in the **Name** field.
2. Assign a type. Options are:
   - Fonts
   - Images
   - Geometries

Copyright © 2026 Vizrt

- Scenes
- Audio
- Archives
- Videos
- Substances

In this way, when clicking on an alias before a file is imported, the import window automatically opens the designated folder and switches to the assigned type.

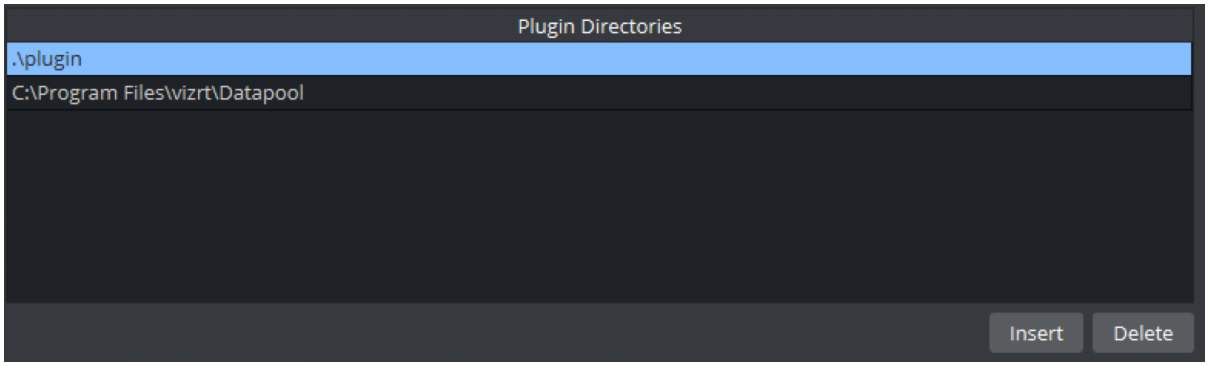
3. Click **Save**.

Copyright © 2026 Vizrt

8.19 Plug-ins

8.19.1 Plugin Directories

Since Viz Engine version 4.1, users can add additional directories for plug-ins. These directories are scanned at startup for available plug-ins and Viz Engine initializes them.

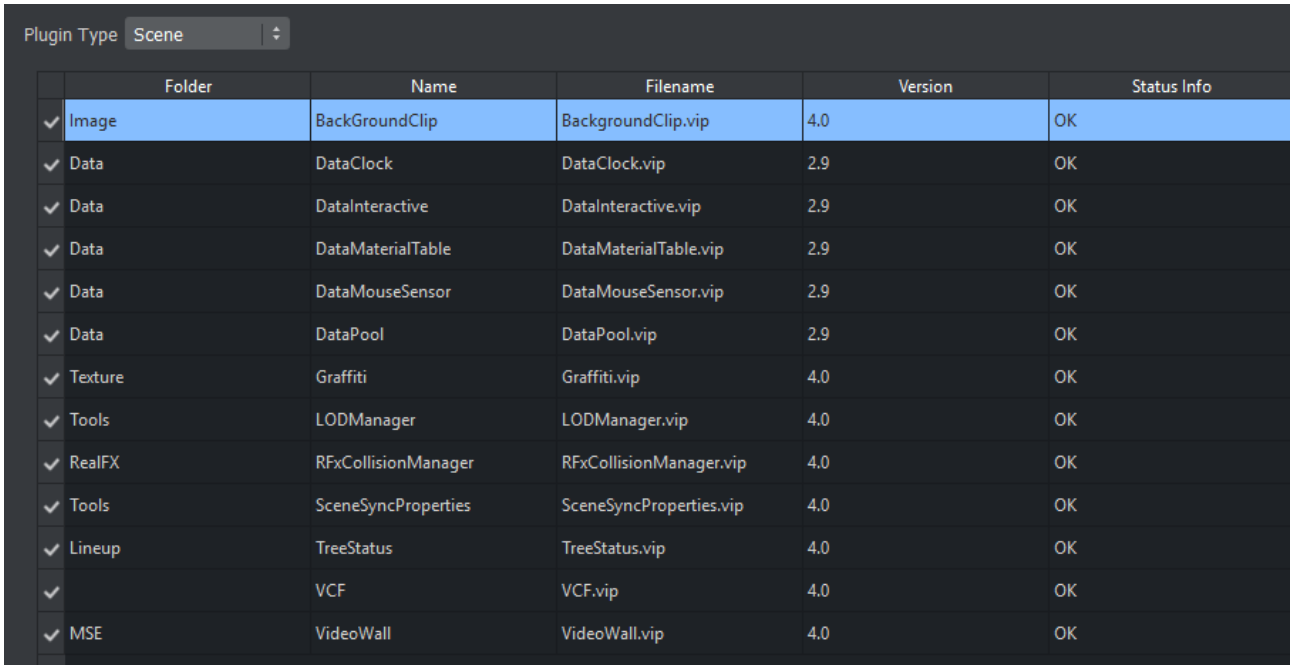


Use the buttons on the bottom to add or remove additional directories.

8.19.2 Plug-in Panel List

The Plug-ins panel lists all installed plug-ins recognized as valid. To display all information correctly, you must open Viz Config from Viz Artist, as the plug-ins are not actually loaded in the Viz Config standalone application. However, you can still enable or disable the loading state. Some unlicensed plug-ins do not load, while others do. For unlicensed plug-ins that do load, a watermark is shown. Viz Artist/Engine does not load inactive plug-ins at run-time. If a scene uses a plug-in that is deactivated, Viz Artist/Engine is unable to activate it without restarting.

Copyright © 2026 Vizrt



Plug-in categories are:

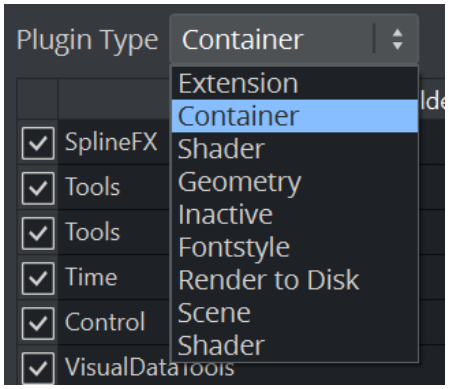
- **Extension**
- **Geometry**
- **Container**
- **Scene**
- **Shader**
- **Viz Engine Shader**
- **Fontstyle**
- **RenderToDisk**
- **Inactive**

For detailed information about the various available plug-ins, see the Viz Plug-ins User Guide in these sections:

- Geometry Plug-ins
- Container Plug-ins
- Scene Plug-ins
- Shader Plug-ins

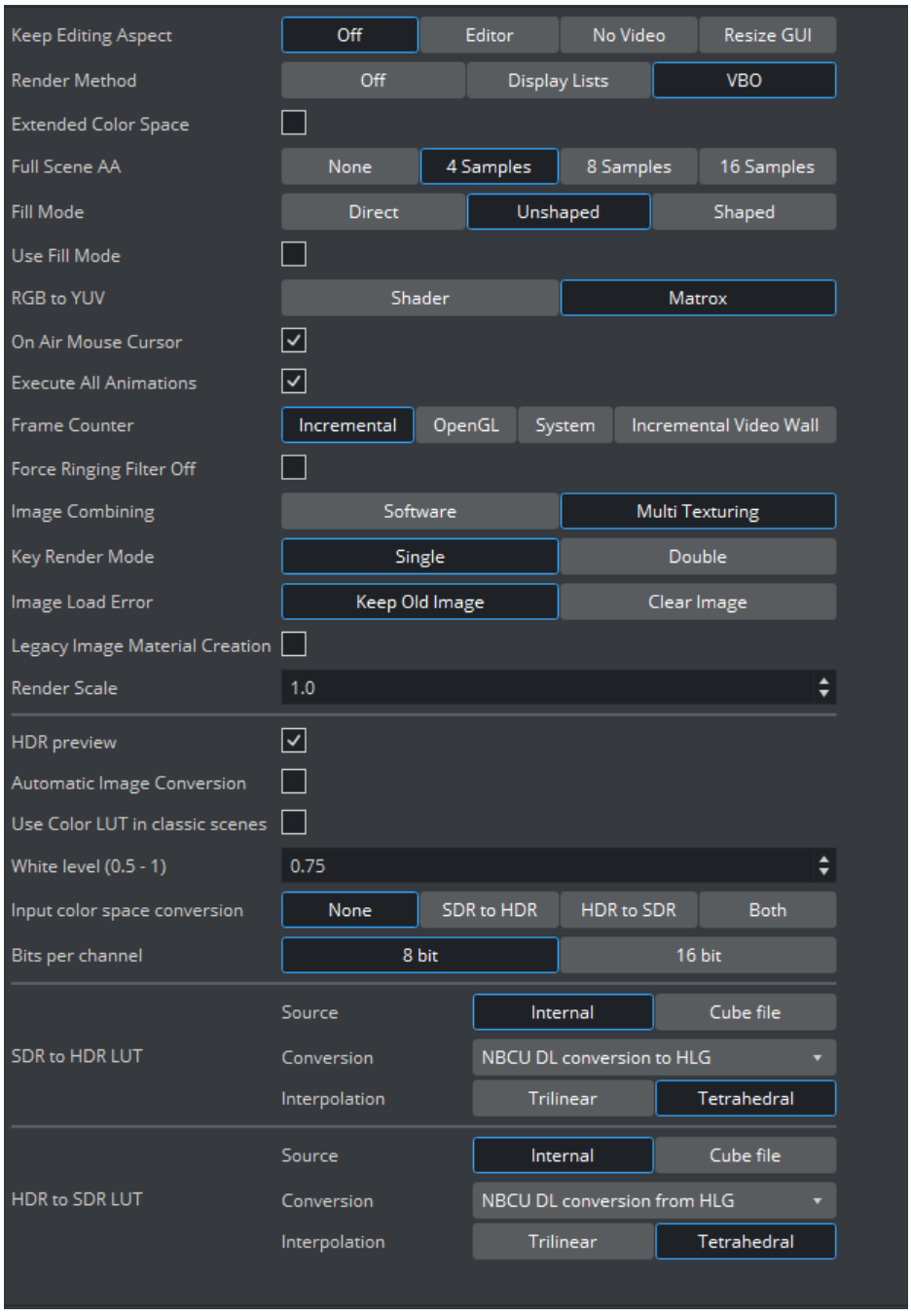
All plug-ins can individually be activated or deactivated. If a plug-in is inactive it is not loaded at startup. All inactive plug-ins are listed under the **Inactive** panel. Click the drop-down menu to select a plug-in category.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

8.20 Render Options



In this section, the following render options can be set:

Copyright © 2026 Vizrt

- **Force Sleep:** Forces the renderer to go to sleep on regular intervals. This allows the GUI to be more responsive. This setting should only be enabled on single processor systems, where the renderer task is using most of the CPU power.

Note: This option is only available for non video versions and might cause irregular frame rates.

- **Keep Editing Aspect:** Influences scene designs in Viz Artist mode. Options are:
  - **Off:** Shows scenes only in Anamorphic wide-screen in the 4:3 VGA render window.
  - **Editor:** Shows scenes using a letter-box format during scene editing giving designers the option to set a user defined camera aspect ratio (under **Scene Settings > Rendering**). On Air and Viz Engine modes are not affected.
  - **No Video:** Shows scenes using a letter-box format as long as the video out is inactive. If video out is active scenes are shown in Anamorphic wide-screen in Viz Artist mode.
  - **Resize GUI:** Increases the renderer window to 16:9 format when editing 16:9 scenes. On Air and Viz Engine modes are not affected.
- **Render Method:** Sets the use of `Display Lists`, `VBO` (Vertex Buffer Object), or `Off`:
  - **Off:** Requires geometries to be redefined in each render step.
  - **Display Lists:** Buffers the geometry definition, which can then be drawn faster. The display list only needs to be updated if the geometry or its parameter changes.
  - **VBO:** Vertex Buffer Object is the default Render Method. Filling a **VBO** is faster than creating a display list, which means VBO can give a performance boost if there are several geometry changes or rebuilds in a Scene design. **VBO** is a required setting for object background loading (see **Background Loading** in the Viz Artist User Guide).

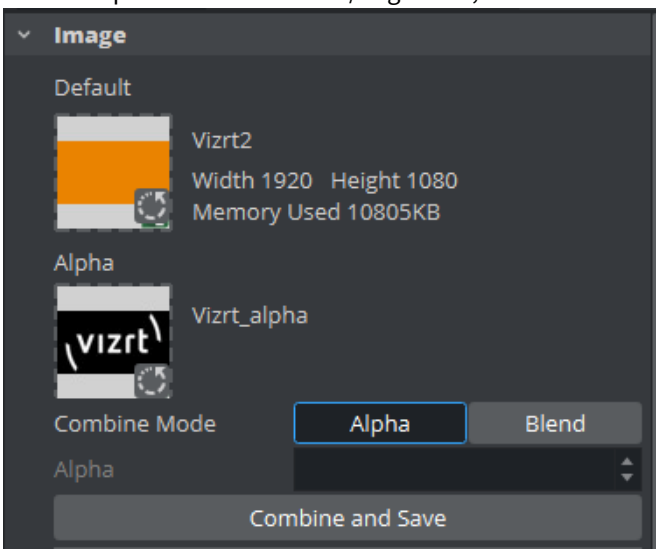
Information: Display Lists are legacy and should not be used anymore.

Note: When rendering transparencies, the output from VBOs and Display Lists may differ slightly. Because of this, the render method falls back to Display Lists in objects with transparency properties, for compatibility reasons.

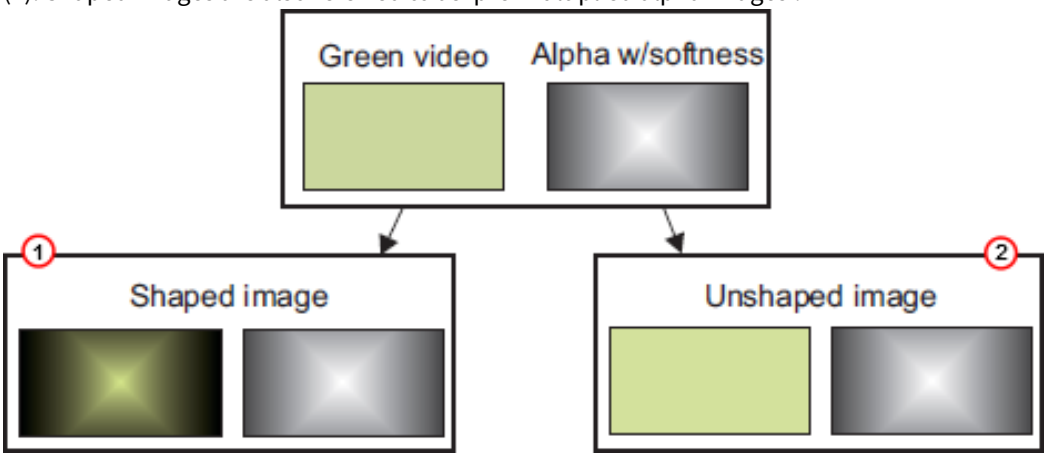
- **Extended Color Space:** Not in use.
- **Full Scene AA:** Sets the Hardware Anti-aliasing (provided by the graphics card). Options are:
  - **None**
  - **4 Samples**
  - **8 Samples**
  - **16 Samples**
- **Fill Mode** [(1)]
  - **Direct:** Does not modify fill output.
  - **Unshaped:** Divides fill by the key when AutoKey is enabled. (In case of a key < 100% this option brightens the fill in the output. The visually correct output can be seen after applying a linear Keyer.)
  - **Shaped:** Pre-multiplies fill with the key. (In case of a key < 100% this option darkens the fill in the output. The visually correct output can be seen after applying a linear Keyer.)
- **Use Fill Mode (for Post Production and NLE):**
  - **Active:** Uses Fill Mode for Post Production and NLE.
  - **Inactive:** Does not use Fill mode for Post Production and NLE.

Copyright © 2026 Vizrt

- **RGB to YUV[(1)]:** Enables color conversion either in the Shader or on the Matrox board. When alpha on the output is turned off on systems with either the Matrox X.mio3 in **Fast Texture Mode** or the Matrox DSX LE4**, RGB to YUV** must be set to `Shader`. In the case of HDR output, the color conversion has to be set to `Shader` as well. In all other cases, the recommended setting is **Matrox**.
- **On Air Mouse Cursor:** Enables a mouse cursor when in On Air mode and using interactive applications. Should be disabled for Video Wall and when DVI out is enabled.
- **Execute All Animations:** Enables Viz Engine to only animate visible objects when deactivated. Default is `Active`.
- **Frame Counter:** Selects the Frame Counter type for animations. This setting takes effect for Standard-PC versions, that use NVIDIA cards and drivers:
  - **Incremental:** Increases the field counter with every field (smooth animations).
  - **OpenGL:** Tries to requests the retrace counter through OpenGL. If not possible, due to driver or hardware problems, it falls back to the Incremental mode.
  - **System:** Uses the internal CPU clock.
  - **Incremental Video Wall:** Advances animation on Viz Engines simultaneously (by incrementing the frame counter based on the elapsed time between render steps) in a Video Wall environment with multiple Viz Engines. If a Viz Engine runs slower than real-time, the next frame (or frames) is skipped to catch up with the Viz Engines running real-time.
- **Force Ringing Filter Off:** Overrules ringing filter (forces it off) when set to `Active`. Forcing ringing filter off can enhance performance at the cost of possibly introducing visual artifacts, typically bands or edges near edges. It is advisable to keep the default value.
- **Image Combining:** Sets a second texture for image combining. In a Texture Editor (see the **Texture Editor** page in the **Scene Management** section of the Viz Artist User Guide), it is possible to set a second texture which is used for the image combining. The Texture Editor offers two possible modes: The first mode uses the second image as an alpha channel, whereas the second mode defines a blend between the two textures. Configuring Image Combining to Software enables the combination to be calculated entirely on the CPU. Configuring Image Combining to **Multi Texturing** enables the combination to be calculated on the graphics card for combining or blending the two images. In this case, the texture creation is faster and memory is saved as well. Default is Software. If there are performance or memory issues, especially with scenes imported from Viz Artist/Engine 2.x, it is recommended to change this setting to Multi Texturing.



- **Key Render Mode:** Determines how the key should be rendered. This configuration is used when the Key Render Mode (see **Global Settings** in Scene Settings in the Viz Artist User Guide) is set to **Config:**
  - **Double Pass:** Uses two rendering steps as in older 3.x versions. Double Pass should be used for old 3.x scenes for not breaking compatibility, for new scenes Single Pass should be used as it is faster.
  - **Single Pass:** Uses one rendering step as in 2.8 versions. A shaped video image has its video data multiplied by its alpha component (1) while the video data of an unshaped image remains untouched (2). Shaped images are also referred to as 'pre-multiplied alpha images'.

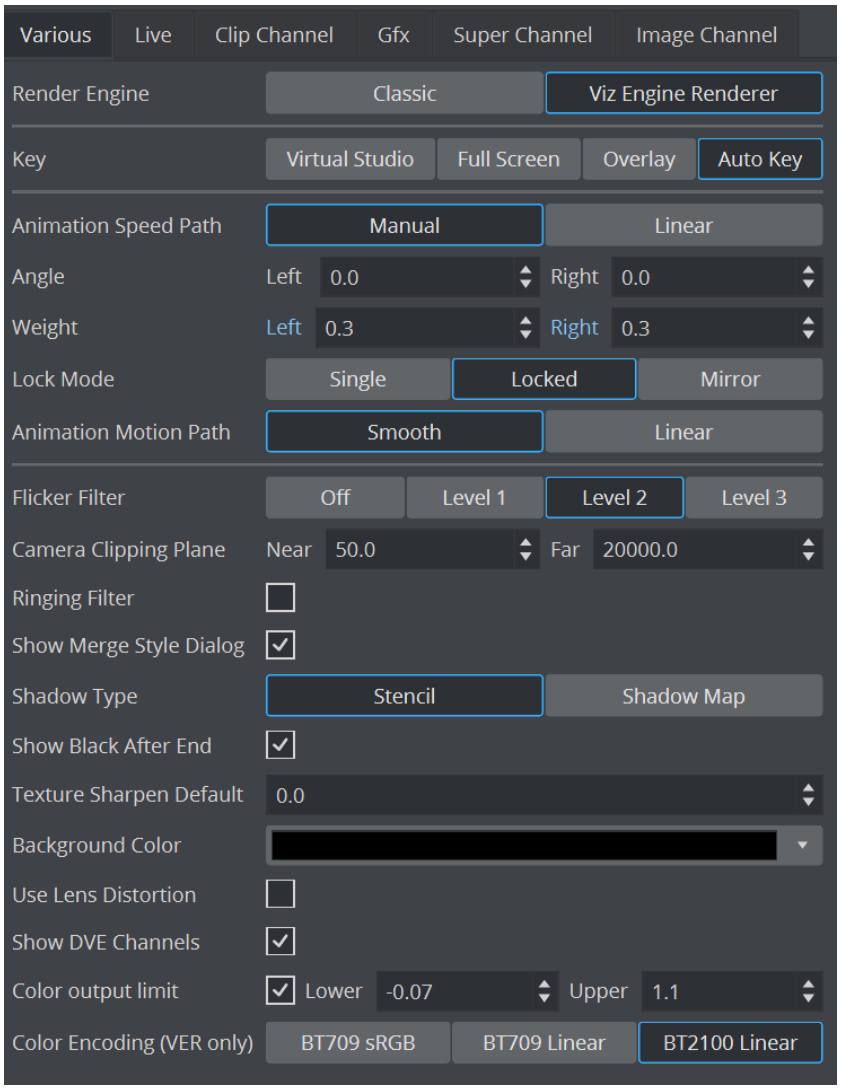


- **Image Load Error:** Configures Viz Engine to keep the old image or clear the image (not showing anything) if an image load error occurs.
- **Render Scale:** Changes the render scale when using a tracking system with lens distortion.

Note: Please see High Dynamic Range for details on all HDR configuration settings.

Copyright © 2026 Vizrt

8.21 Scene Default Values



This section configures the default values for new scenes.

- **Render Engine:** Sets the default Render Pipeline. For compatibility with previous versions, select *Classic*. To use the Viz Engine Render Pipeline, select *Viz Engine Renderer*.
- **Key:** Sets the key mode. Alternatives are `Virtual Studio`, `Full Screen`, `Overlay`, and `Auto Key`.
- **Animation Speed Path:** Sets the default animation speed either to **Manual** or **Linear,** and define the default **Angle** and **Weight**.
- **Animation Motion Path:** Defines the default motion path between position key frames. When set to `Smooth`, the motion path is calculated as a Bezier curve between position key frames. Handles are added to the key frame positions in the preview window to allow for path editing. When set to `Linear` the

Copyright © 2026 Vizrt

animated object follows a straight line between position key frames. This setting can be changed individually under **Path**, in the Viz Artist Channel Editor.

- **Flicker Filter:** Reduces interlaced flicker on high contrast objects when enabled. For example small lines and hard objects. Alternatives are `Off`, and `Level 1-3`.
- **Camera Clipping Plane:** Sets the range of the virtual camera. Near sets the close range while far defines the far range. Only objects within this range are rendered.
  - **Near:** Sets the Near value to clip unwanted objects from the foreground. Default value is `50`.
  - **Far:** Sets the Far value to clip unwanted objects from the background. Default value is `20000`.

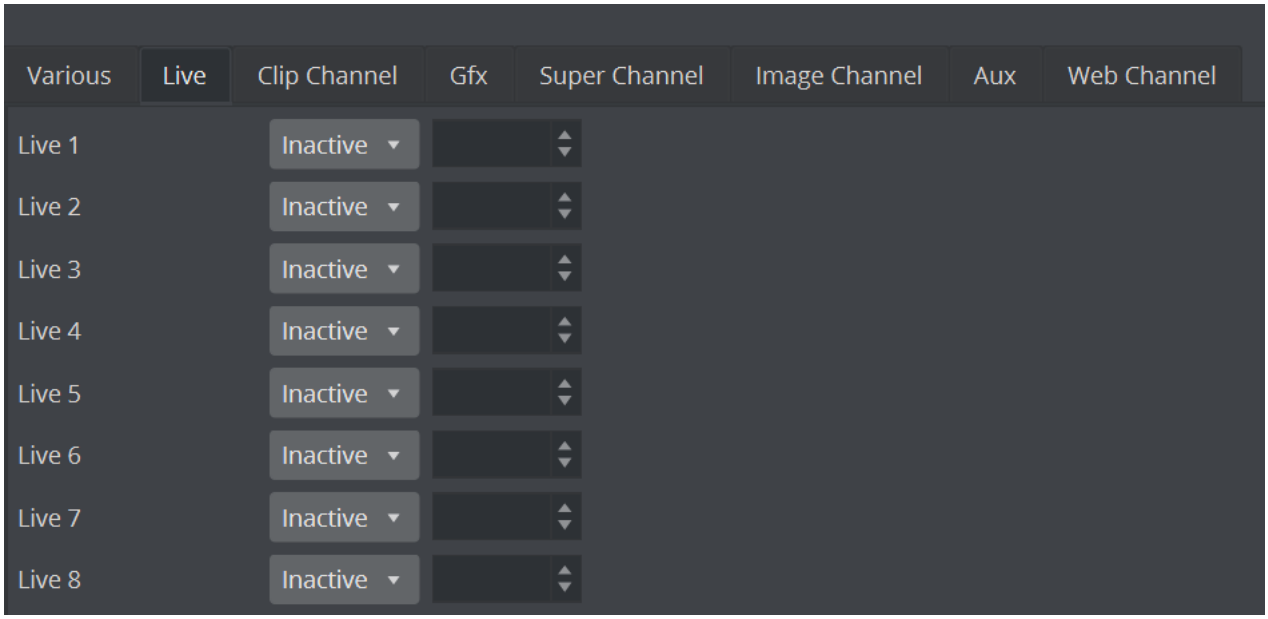
> ⚠ **Note:** The camera range is where the Z-buffer is within. So if Z-buffer problems arise, they may be solved by editing the camera clipping plane settings. The camera range setting is also responsible for the quality of your shadow maps.

- **Ringing Filter:** Sets the default value for the Ringing Filter. A ringing filter reduces high frequency values in the video signal created by high contrast and color changes in horizontal directions.
- **Show Merge Style Dialog:** Enables the user, when opening a scene in Viz Artist, to open old-style merged objects and expose containers within it.
  - This feature relates to scenes using old-style ordering of containers within merged objects, and solves the problem with auto-follow. When loading such scenes the dialog lets the user decide how to deal with them.
  - Users that are aware of this and decide to keep the old style can deactivate this dialog.
- **Shadow Type:** Selects how shadows are displayed. Choose between `Stencil` and `Shadow Map`.
- **Show Black After End:** Shows black after a clip has finished playing.
- **Texture Sharpen Default:** Sets the default sharpen value for textures.
- **Background Color:** Set default scene background color.
- **Video Input Layer Targets and Priority:** Sets the defaults for how the specific input channel should be used and its priority. Available options are `Inactive`, `Texture`, `DVE` and `Priority`.
- **Use Lens Distortion:** Activates the Lens Distortion. For detailed settings, see the **Advanced Lens Distortion** page in the **Cameras** section of the Viz Artist User Guide.
- **Show DVE Channels:** Shows the DVE Channels in the Scene Editor if activated.
- **Color output limit:** Defines the lower and upper limit if color limits are enabled in the color space. Defaults refer to the superwhite and superblack settings in HDR.
- **Color Encoding (VER only):** Determines the default color space for when scenes are created. Can be either SDR (BT709 sRGB (default) or Linear), or HDR (BT2100 Linear).

8.21.1 Media Asset Configuration

Here you can pre-configure all available media assets to the different modes of DVE or Texture. This works for Media Assets such as **Live Video**, **GFX Channels**, **Clip Channels**, and **Stream Channels**. **Super Channels** and **Image Channels** can only be pre-configured as `Inactive` or `DVE`.

Copyright © 2026 Vizrt

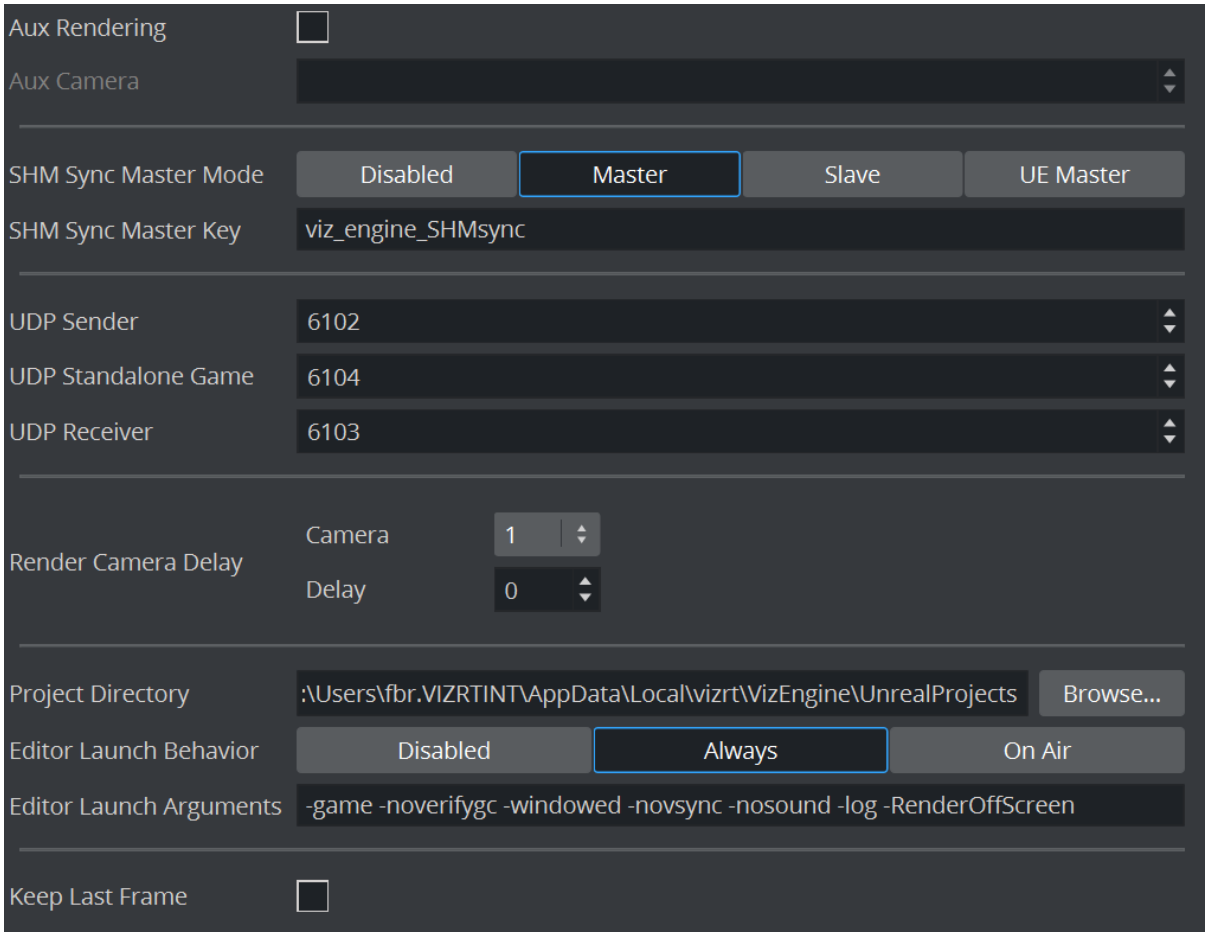


By setting the individual number and type of Media Asset to `Inactive`, `DVE` or `Texture`, the desired channel is automatically set up when dragged into the Scene Tree of new scenes. The priority option works only in DVE mode, and influences the drawing order of the defined channels.

Copyright © 2026 Vizrt

8.22 Unreal Engine

Settings to communicate with and control Unreal Engine.



- **Aux Rendering:** Needed for integration with Unreal Engine.
- **SHM Sync Master:** Defines how the synchronization between Viz Engine and Unreal Engine should work.
  - Disabled
  - Master
  - Slave
  - UE Master
- **SHM Master Key:** A unique identifier key.
- **UDP Sender**, **UDP Standalone Game**, and **UDP Receiver:** Defines which ports are being used to communicate with Unreal Engine.

A delay in frames can be set for each individual camera. This defines how much delay the Viz camera has to compensate the delay between requesting and retrieving the Unreal Rendering.

Copyright © 2026 Vizrt

Project Directory defines the location your unreal projects are located on your hard disk. This location is used when restoring projects from a file collection to a playout machine. **Editor** is the executable that is started if the **Editor Launch Behavior** is set to *Always* or *On Air*.

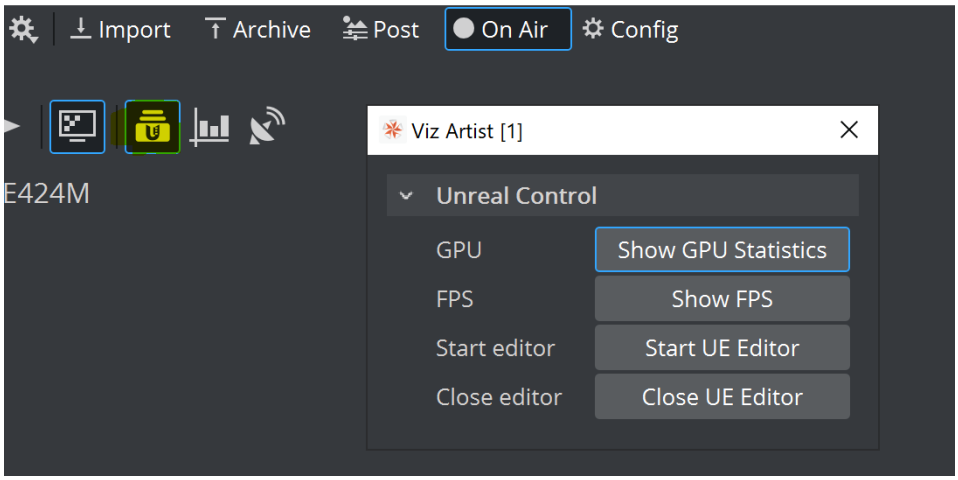
If a scene has a file collection associated, the unreal project is restored from Graphic Hub to the given **Project Directory**. The **Editor Launch Behavior** defines whether Unreal Editor is launched with the given launch Editor Launch arguments.

- *Always* means that the editor is launched regardless of Viz Engine being in On Air Mode or not (Viz Artist).
- *On Air* only launches the editor if Viz Engine is in playout mode.

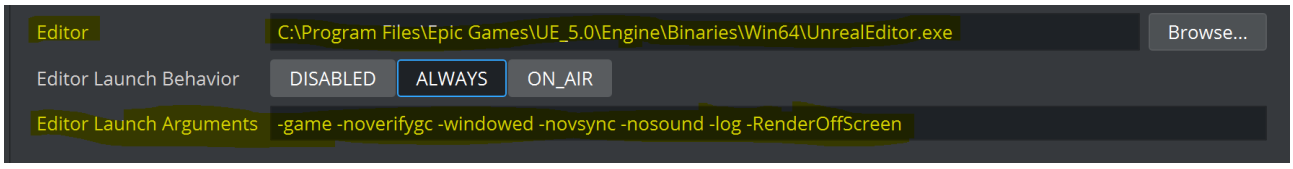
Keep Last Frame: Displays the last frame rendered instead of turning black (in case the connection is lost).

8.22.1 Start/Close Unreal Engine using Viz Engine in On Air Mode

You can start/close Unreal Engine, show FPS and GPU statistics by clicking the **Unreal Control** button.



The Start UE Editor button uses the highlighted settings.



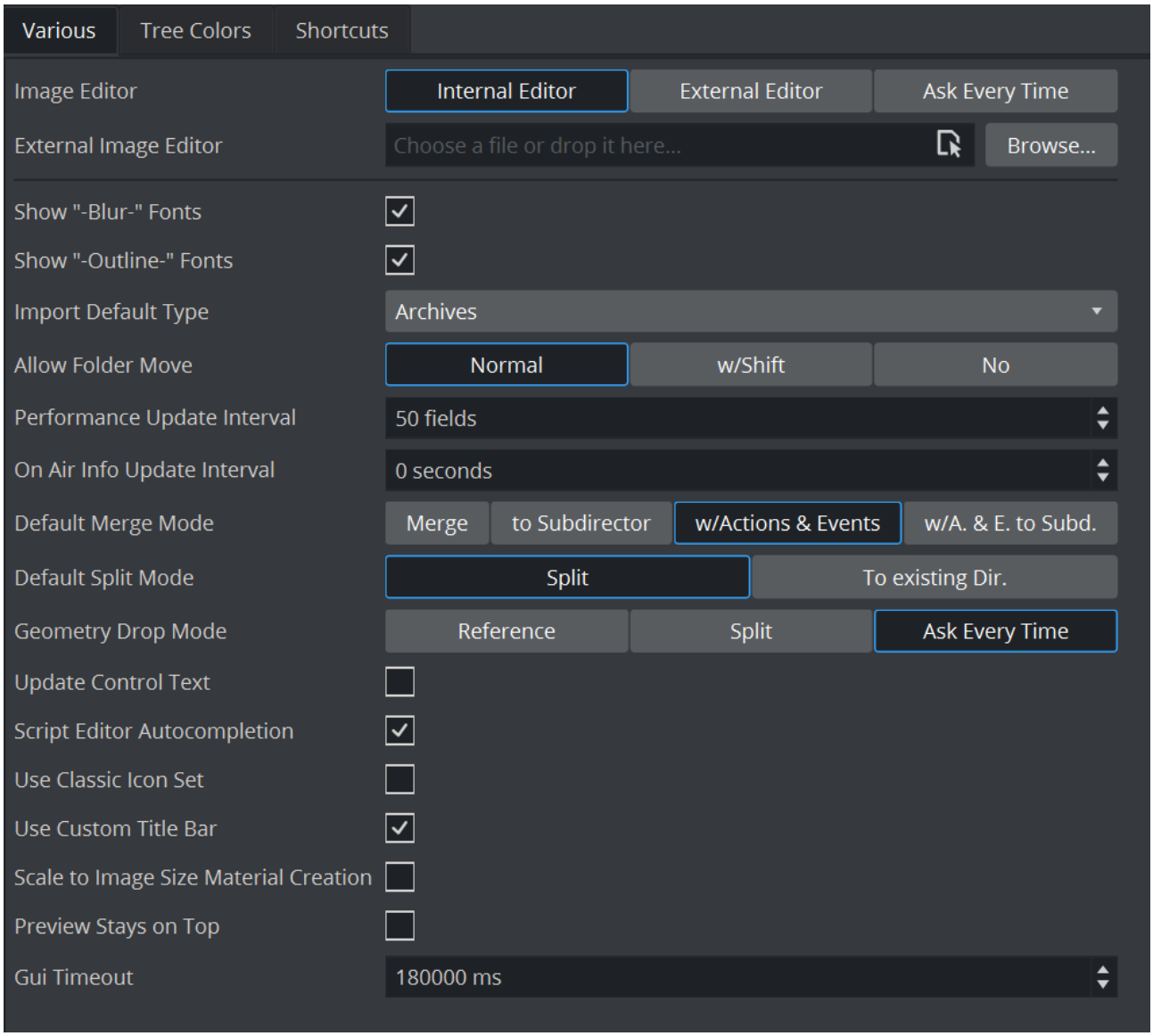
Copyright © 2026 Vizrt

8.23 User Interface

This section describes the user interface settings. Some of these settings are also available in Viz Artist.

- Various
- Tree Colors
  - To Change the Amount of Active Colors
- Shortcuts
  - To Edit a Shortcut

8.23.1 Various



Copyright © 2026 Vizrt

- **Image Editor:** Specifies whether to use the the built-in internal editor, an external editor or to always ask the user which one to use.
- **External Image Editor:** In case you prefer to edit your images not in the internal editor, provide the path to the external executable here.
- **Show "Blur" fonts:** Sets the default show or hide blur fonts in the Server area. The option can then be toggled with the **Item Context Menu** (see the Viz Artist User Guide).
- **Show "Outline" fonts:** Sets the default for showing or hiding the outline fonts in the Server area. The option can then be toggled with the **Item Context Menu** (see the Viz Artist User Guide).
- **Import Default Type:** Selects the default item type for imports, which is pre-selected in the GUI **Import Menu**. Options are:
    - **Fonts**
    - **Images**
    - **Geometries**
    - **Scenes**
    - **Audio**
    - **Archives**
- **Allow Folder Move:** Allows or restricts the user's ability to move/organize projects and folders in a Graphic Hub. Options:
    - **Normal:** Drag folders freely, as required to move (Default setting).
    - **w/Shift:** Press **SHIFT** and drag to move folder or folders.
    - **No:** No folder movement allowed.
- **Performance Update Interval:** Sets how often Viz Engine should update the Performance Bar when it is opened.
- **On Air Info Update Interval:** Sets the update interval for the On Air Information Panel window. Note that a shorter interval decreases render performance. Setting to `0` means that no update occurs.
- **Default Merge Mode:** Sets the default merge behavior available in the Viz Artist user interface. Options are:
    - Merge
    - [Merge] to sub director
    - [Merge] w/actions & events
    - [Merge] w/actions and events to sub director (*w/A. & E. to Subd.*)
- **Default Split Mode:** Sets the default split behavior available in the Viz Artist user interface. Options are:
    - Split
    - [Split] to existing director
- **Geometry Drop Mode:** How to handle geometries dragged into the scene:
    - Reference: Keep a reference to the geometry in the Graphic Hub
    - Split: Move the geometries data into the scene
    - Keep asking how to handle geometries.
- **Update Control Text:** Sends update message on every single change of Control Text (based on Keystrokes).
- **Script Editor Autocomplete:** Toggles the autocomplete suggestions inside the script editor.
- **Use Classic Icon Set:** Uses the legacy icon set.
- **Use Custom Title Bar:** Render a customized windows titlebar instead of the default one.
- **Scale to Image Size Material Creation:** In new Viz Engine Render pipeline, images dragged to the scene editor or scene tree are automatically scaled to match the image dimensions.

Copyright © 2026 Vizrt

- **Preview Stays on Top:** Shows Viz On Air preview as the top most window (by default). Disabling this allows other applications to be shown on top of it.
- **Gui Timeout (ms):** Defines the timeout threshold in milliseconds when awaiting command replies from Viz Engine. After the specified time, a timeout dialog is displayed. The minimum and default value `180000`, or 180 seconds.

Tip: Adjust the timeout value if timeouts occur when loading very large scenes in Viz Artist.

8.23.2 Tree Colors

The Colors tab gives the ability to change the colors for containers in the Scene Tree. A Container with a color code can be used as search criteria in the Scene Tree (see **Scene Tree Menu** in the Viz Artist User Guide).

Example: All text Containers can be colored gray and tagged Text, while all Containers that hold images can be colored green and tagged Image, and so on.

Four colors are configured and active by default with no text descriptions. Click **Set to Default** to set all color bars to their default setting. The color options are available in the GUI **Scene Settings** panel in the Tree Color Text setting, and available for use in the Scene Tree panel.

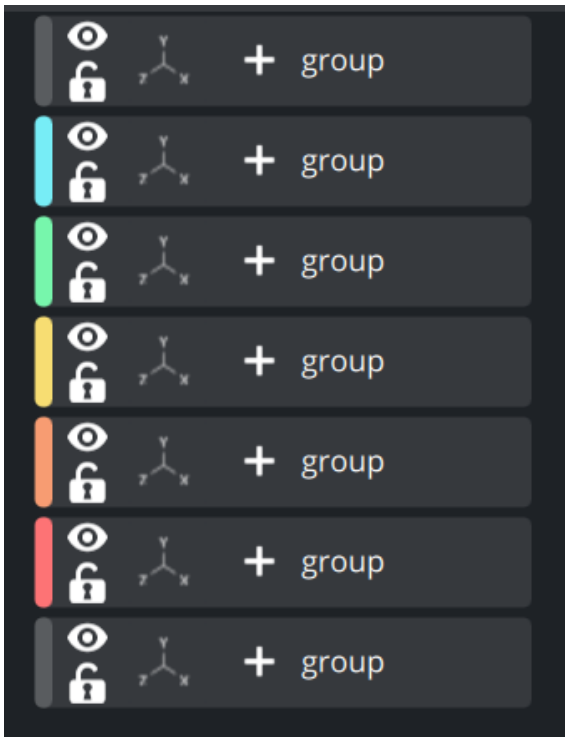
- **Show Hierarchy Lines:** Defines if the Scene Tree should also show lines to better separate hierarchy levels.
- **Highlight Complete Container:** Selects either the whole container or only a small stripe on the left side.

Highlight Container Setting	
Highlight Complete Container `ON`	Highlight Complete Container `OFF`

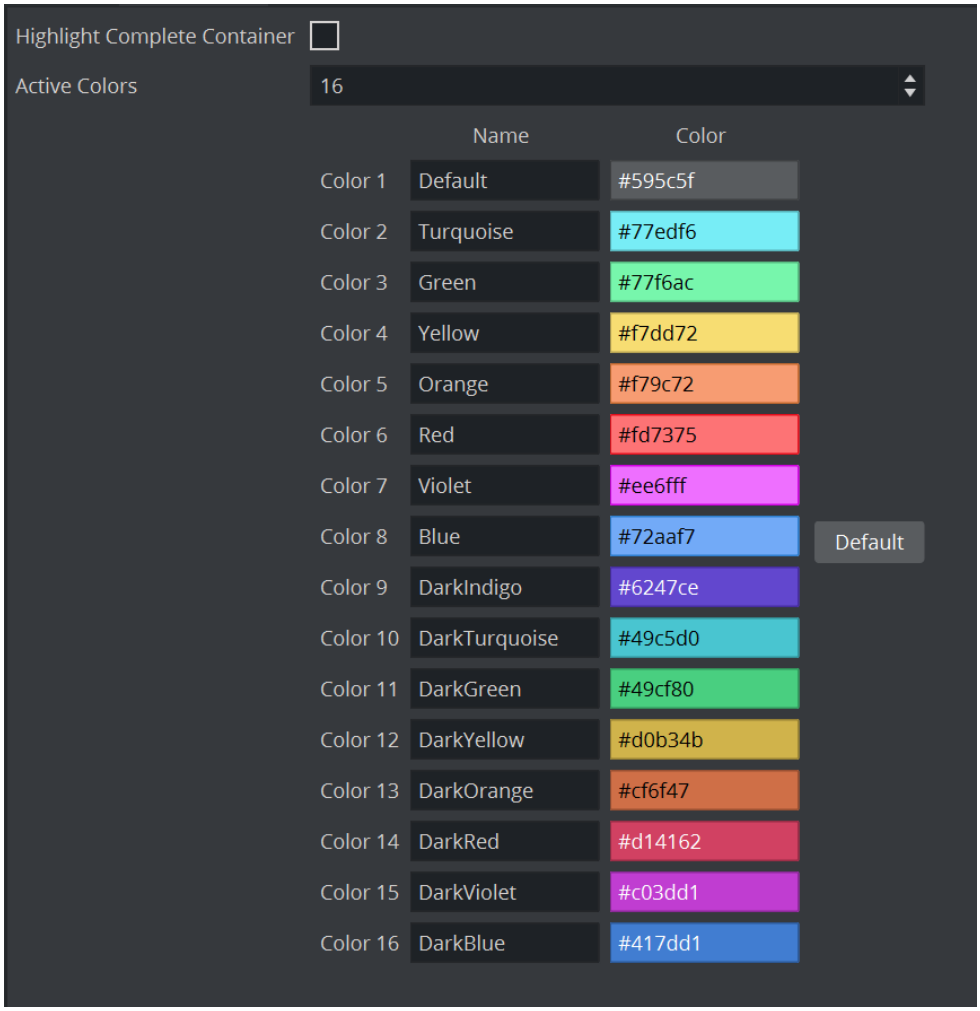
Copyright © 2026 Vizrt

Highlight Container Setting





Copyright © 2026 Vizrt



Click **Save** then restart Viz Artist for the changes to take effect.

To Change the Amount of Active Colors

In the **Active Colors** field enter the amount of colors to be active. Up to 16 colors can be configured and made active.

8.23.3 Shortcuts

The shortcuts view shows all available shortcuts and their current mappings. Shortcuts can be modified and exported for personalization.

> **Note:** Some shortcuts can not be modified, they are displayed in a slighter darker grey.

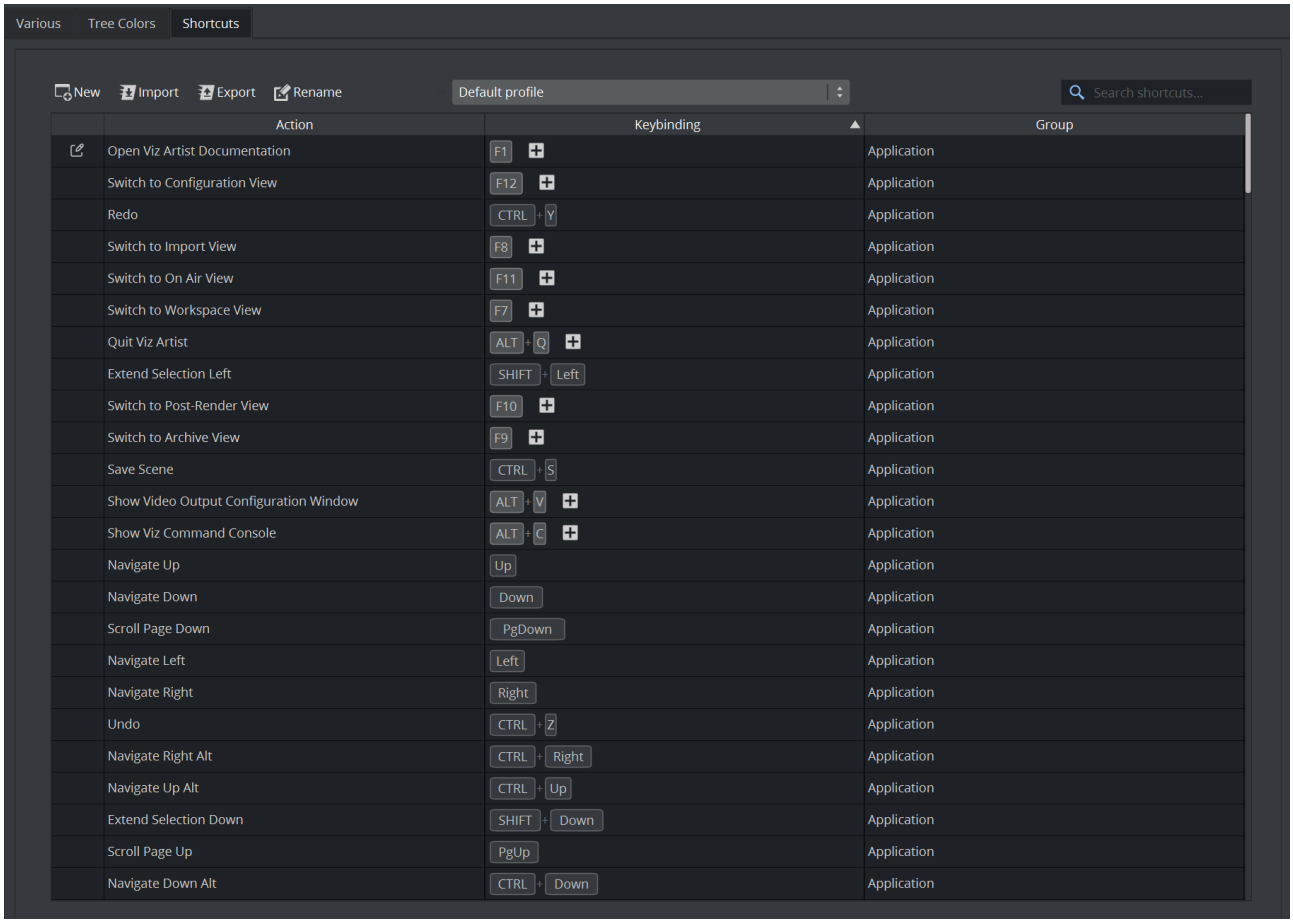
Shortcuts are grouped by their editors:

- General Application
- Archive View
- Asset View

Copyright © 2026 Vizrt

- Import
- On Air + Render Editor
- Plugin List View
- Scene Editor
- Scene Tree
- Script Editor
- Stage
- Stage Tree

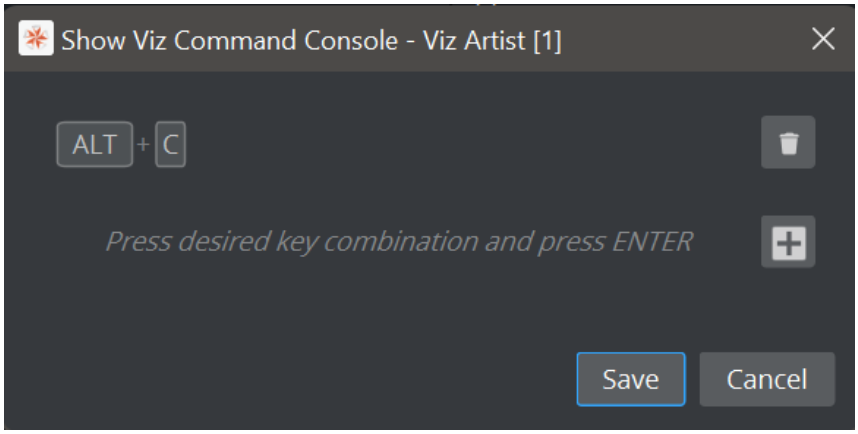
Note: On Air (Viz Artist) and Render Editor (Viz Engine) share the same functionality, however Render Editor shortcuts can be mapped different to achieve different behavior (for example, in videowall mode).



To Edit a Shortcut

If a shortcut can be modified, an icon ( ) is shown in the left row. Select it to show the dialog, or double click on the entry.

Copyright © 2026 Vizrt



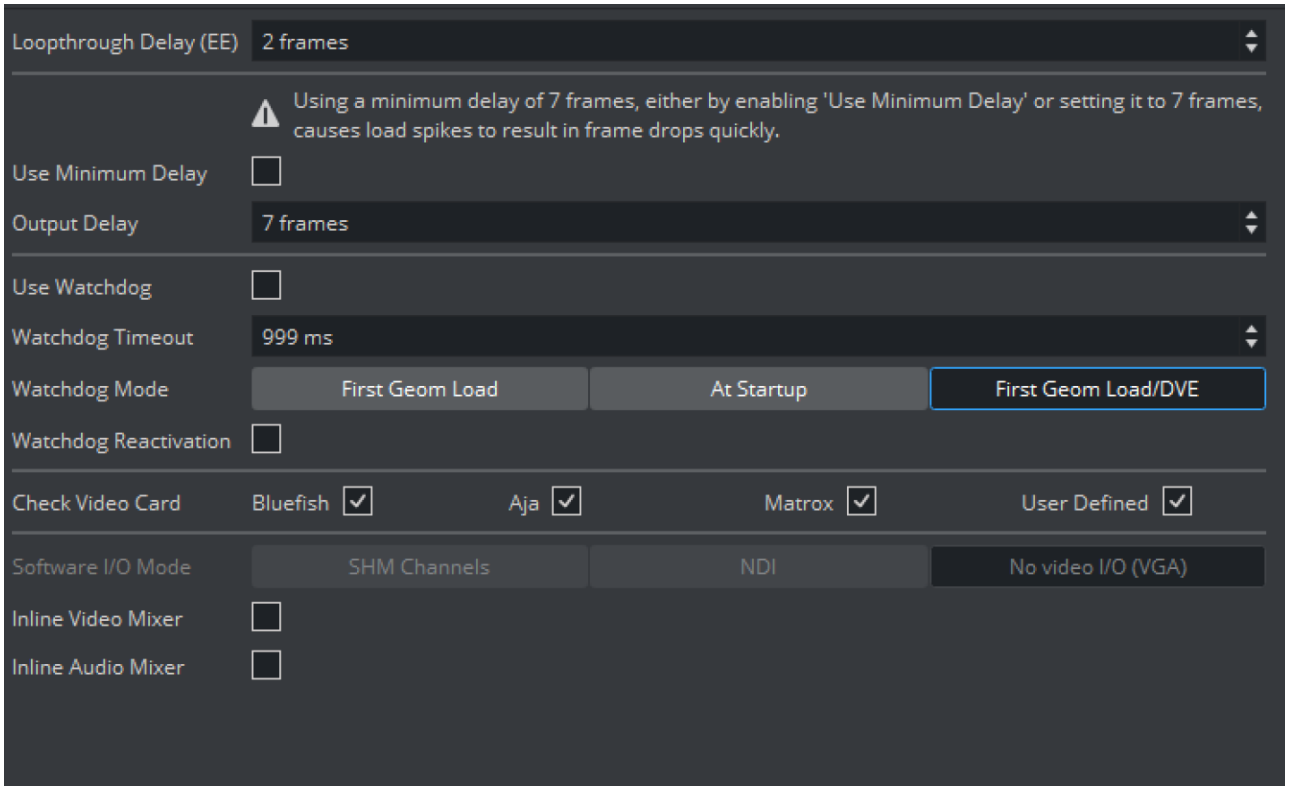
If you want to add an additional shortcut, press the desired combination and press the add (+) button. To remove a shortcut, press the trashcan (🗑) button.

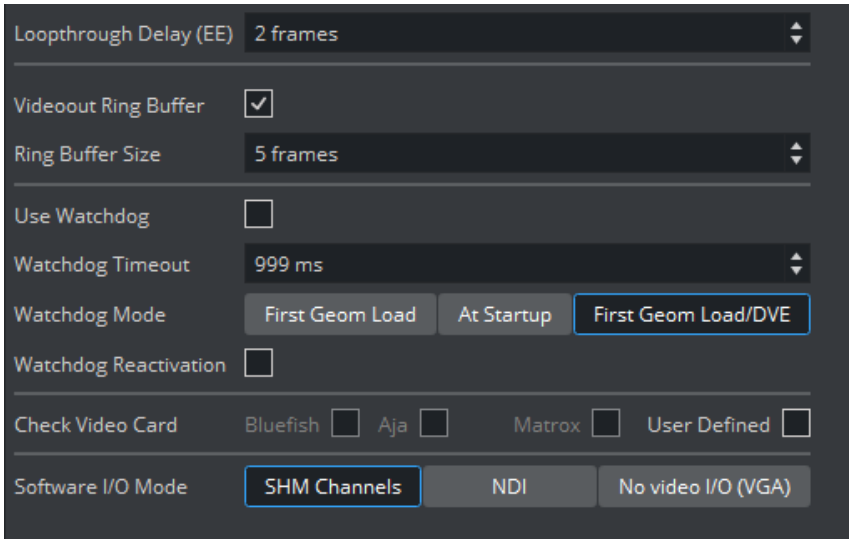
Copyright © 2026 Vizrt

8.24 Video Board

This section is used to configure video board related settings.

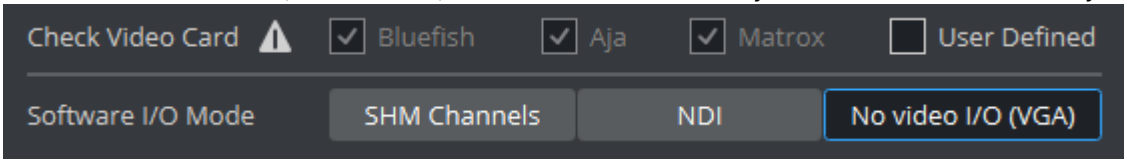
8.24.1 Video Board Properties





Copyright © 2026 Vizrt

- **Loopthrough Delay (EE):** Sets the delay for live video input (Bluefish boards only). This setting applies for all input channels. For Matrox, see Video Delay DVE in Video Properties.
- **Use Minimum Delay/Output delay:** Sets the number of frames between the earliest possible start of rendering and sending the frame. This mechanism is used with Matrox video boards, DSX Core, and software I/O mode NDI. The output delay includes the time for rendering, transfer of data from the GPU, and processing on the video board if required. The minimum output delay reserves one video unit for rendering. Using a higher output delay allocates the additional time to rendering. This can compensate for load spikes and prevent frame drops.
  The minimum delay depends on these factors:
  - **Matrox**: The hardware mixer of X.mio cards is the most important factor. If enabled, the minimum delay output delay is five frames in interlaced and six frames in progressive. This is necessary to synchronize DVE effects with graphics. If the hardware mixer is disabled or a DSX LE card is use, the minimum delay is two video units.
    ⚠ The following settings increase the minimum delay by one additional frame:
    - Using UHD resolutions on an X.mio3 or DSX LE4 card.
    - Enabling clean feed.
  - **DSX Core**: The minimum delay is seven frames in interlaced and eight frames in progressive.
  - **NDI**: The minimum delay is two video units.
- **Videoout Ring Buffer:** This mechanism is used by all video classes that do not use the output delay. It sets the render buffer for video output. Helps to prevent frame drops on the video output during the execution of commands or the loading of objects. When enabled, the engine renders a number of graphics frames in advance and provides them to the video hardware. The number of frames rendered forward is defined by the ring buffer Size. Large buffer sizes delay the output and increase the input-to-output delay for video textures.
- **Ring Buffer size:** Sets the size of the ring buffer in frames.
- **Use Watchdog:** Activates or deactivates the watchdog functionality.
  - **Matrox:** See **Video IO Related Configuration and Features > Matrox Watchdog Configuration** in the Viz Engine Administrator Guide.
  - **Bluefish:** See **Hardware Related Information > Video Boards > BlueFish444 > Special Configuration Options for Bluefish444** in the Viz Engine Administrator Guide.
- **Watchdog Timeout:** Sets the time, in milliseconds, until the watchdog takes over control. This value should not be smaller than the time of two fields/frames. The default value is `999 ms`.
- **Watchdog Mode:** Sets the Watchdog mode:
  - **First Geom Load:** Activates Video Out when the first geometry is loaded.
  - **At Startup:** Activates Video Out at startup.
  - **First Geom Load/DVE:** Activates Video Out when the first geometry is loaded or an input or clip channel is set to DVE.
- **Watchdog Reactivation:** Activates the watchdog again if scenes are unloaded from the renderer when set to `On`.
- **Check Video Card:** Selects which video cards to search for and use when Viz Artist/Engine is started. If a video card is not selected, it is not used, even if it is installed in the system. All cards are enabled by default.

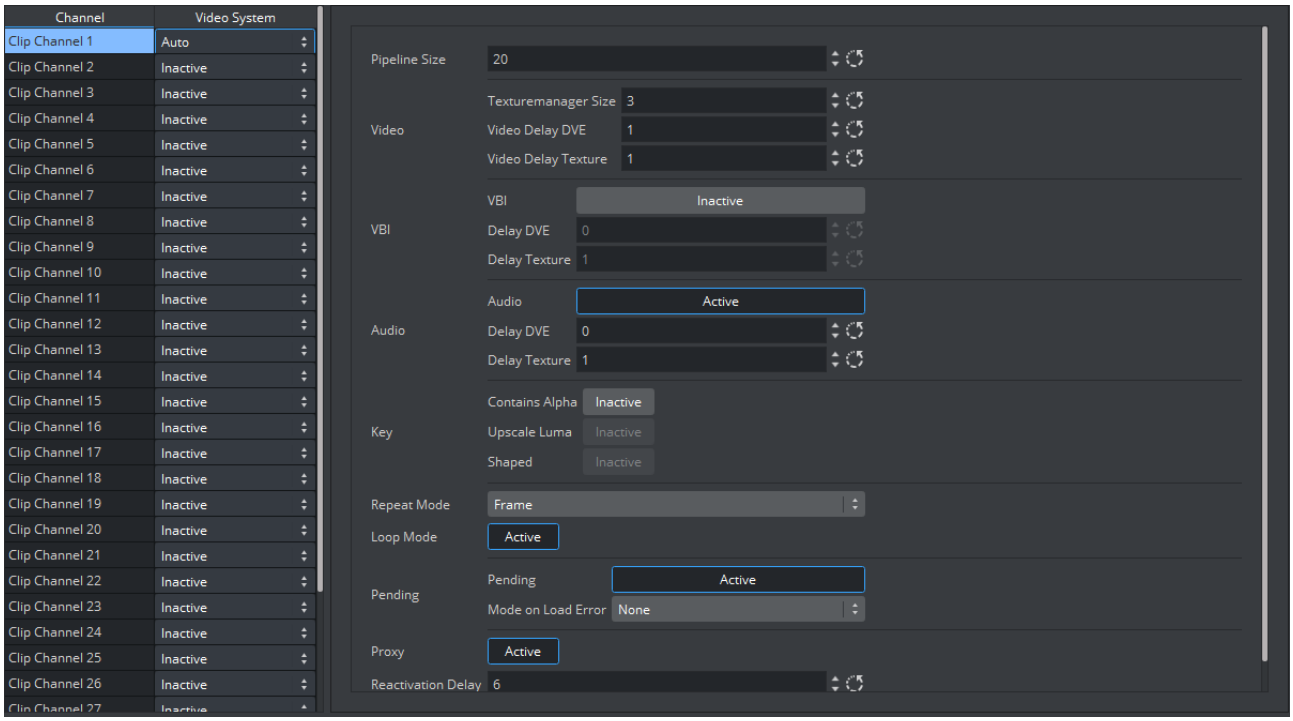


Copyright © 2026 Vizrt

    - **Select Individual Cards:** Selects/deselects video cards if **User Defined** is selected. Click on each listed card to select or deselect, as required.
    - **User Defined:** Selects and uses individual cards.
    - **None:** Makes all connected video cards unavailable for Viz Engine and Viz Artist when selected. This can be used to run a Viz Artist for a different video platform version, even if video cards are installed. Possible use-cases for this are:
        - Dual channel Viz Trio One Boxes, which have three Viz Artist/Engine instances running and where only the first two should use the video board. The third Engine instance is exclusively for Viz Trio preview.
        - With TriCaster implementation.
- **Software I/O Mode:** Sets a specific Video I/O mode for setups without video hardware or configurations where **Check Video Card** is set to `None`.
    - **SHM Channels:** Sets I/O mode for third party software, utilizing the Viz SHMLib software library.
    - **NDI:** Uses the NewTek NDI input and outputs. Read more about using NDI with Viz Engine and Artist in the Third Party Applications and Files chapter of the Viz Artist User Guide.
    - **No video I/O (VGA):** Disables all video version features, emulating the deprecated VGA version of Viz Engine.
- **Inline Video and Audio mixer:** Enables the zero frame delay mixer, which means that the surface (and audio) is copied directly from the input to the output.

Copyright © 2026 Vizrt

8.25 Video Input: Clip Input



Use the Video Input: Clip Input panel to configure available playback channels. The left hand pane defines the number of usable clip channels and their respective resolution. If the Video System is set to *Auto*, the configured output resolution will be used. In case of a video wall setup (full screen or custom resolution), *Auto* falls back to 1080i and must be reconfigured if a different resolution is required.

Caution: Depending on various factors like used codec, disk performance, current load of the system, and more, not all of the Clip Channels might be usable for production. This might be different from system to system. Please test the scene and the system's overall performance before taking a scene On Air.

8.25.1 Clip Input Properties

- **Pipeline Size:** Defines the number of frames that the Matrox internal clip reader buffer should buffer in advance. Default value is `20`.
- **Texturemanager Size:** Sets texture download buffer size.
- **Video Delay DVE:** Sets DVE Delay of video when used as DVE.
- **Video Delay Texture:** Sets texture Delay of video when used as Texture.
- **VBI:** Defines whether VBI should be used for this channel. Set to Active or Inactive. Default mode is `Inactive`.
- **Delay DVE:** Sets the number of frames VBI should be delayed, before the clip can be used, in DVE mode. Default value is `0` (Off).

Copyright © 2026 Vizrt

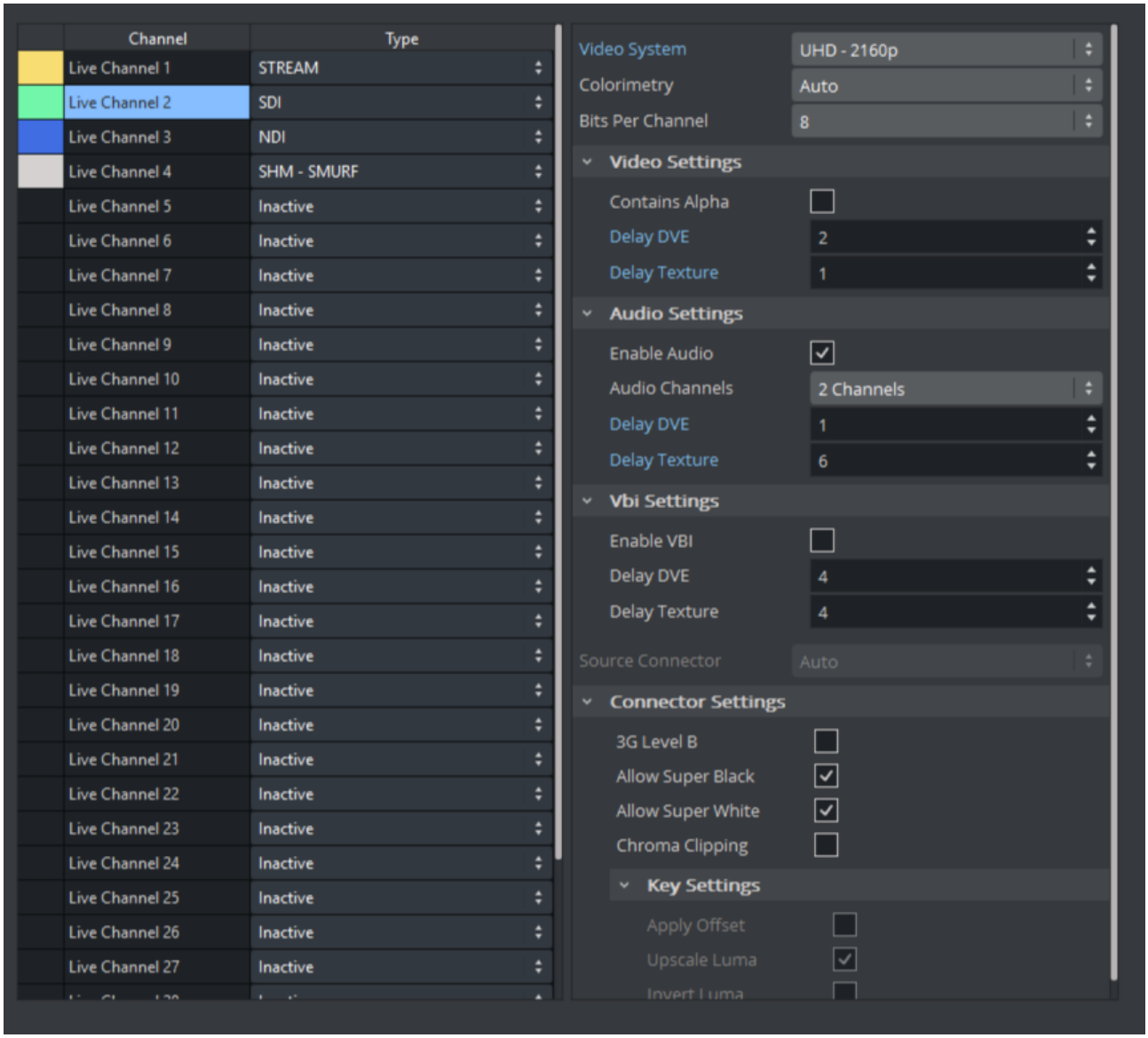
- **Delay Texture:** Sets the number of frames VBI should be delayed, before the clip can be used, in texture mode. Default value is `1`.
- **Audio:** Enables/disables audio for this channel. When set to `Inactive`, audio is disabled. Default value is `Active`.
- **Delay DVE:** Sets the number of frames the audio clip should be delayed in DVE mode before it can be mixed to the output. Default value is `4`.
- **Delay Texture:** Sets the number of frames the audio clip should be delayed in texture mode before it can be mixed to the output. Default value is `4`.
- **Contains Alpha:** Enables/disables playback of clips with alpha.
- **Upscale Luma:** Enables/disables the default for upscale luma. Per scene setting of this value is set per clip channel under Scene Settings and Video clip options.
- **Shaped:** Defines whether the fill from this channel, when the channel is used in DVE mode, should be interpreted as shaped video during DVE compositing. Default value is `Inactive`.
- **Repeat Mode:** Determines the behavior of the video input in case of capture drops. Options are:
  - **None:** Does not repeat. Input goes black.
  - **Field:** Repeats the last field.
  - **Frame:** Repeats the last frame.
- **Loop Mode:** Enables/disables default for loop mode. Per scene setting of this value is set per clip channel under Scene Settings and Video clip options.
- **Pending Enable:** Enables/disables pending clip player for this channel. The pending clip player allows clip loading of another clip while the clip channel is still using the current clip.
- **Mode on Load Error:** Determines the behavior of the current clip when loading of the pending clip fails. Options are:
  - **None:** Current clip mode is not changed.
  - **Stop:** Performs a Stop command on the current clip.
  - **Pause:** Current clip enters pause mode.
  - **Flush:** Unloads the current clip.
- **Proxy:** Allows playing video clips of a resolution different to the current configured resolution when set to `Active`. If set to `Inactive`, a video clip of a different resolution cannot be played.
- **Reactivation Delay:** Sets the minimum number of frames the texture contains black after the channel was activated to texture.
- **Delay After Input Graph:** Adds a surface delay to compensate the time needed for pose estimation.

See Also

- Matrox

Copyright © 2026 Vizrt

8.26 Video Input: Live Input



Use the Video Input: Live Input panel to configure channels utilizing external sources. The left hand pane defines the number of usable channels and their type. Depending on the video hardware you're using, the available types are:

- SDI
- IP (SMPTE ST 2110 or ST 2022-6)
- STREAM
- NDI
- Various Shared Memory inputs

Copyright © 2026 Vizrt

Information: The number of usable channels depends mainly on the installed video hardware.

8.26.1 Live Input Properties: Type SDI

- **Video System:** Either the required resolution or *Auto* to use the configured output resolution. In case of a video wall setup (full screen or custom resolution), *Auto* falls back to 1080i and must be reconfigured if a different resolution is required.
- **Colorimetry:** This setting is only relevant for HDR workflows and 1080p 50 / 59.94 / 60 and UHD. Available options are:
  - AUTO
  - *ITUR_BT_709*
  - *ITUR_BT_2100_HLG*
  - ITUR_BT_2100_SLOG3
- **Bits per channel:** Defines how many bits per channel are processed for an input.

Note: Bits per channel is not supported on SD channels.

- **Video Settings**
  - **Contains Alpha:** Enables alpha for this input. Only available on connectors that have an associated alpha connector.
  - **Delay DVE:** Sets DVE Delay of video when used as DVE.
  - **Delay Texture:** Sets texture Delay of video when used as Texture.

Note: Setting the texture delay of progressive inputs to `1` frame has a performance impact that scales with the aggregate bandwidth of the input signals.

- **Audio Settings**
  - **Enable Audio:** Toggles audio for this channel. When set to `Inactive`, audio is disabled. Default value is `Active`.
  - **Audio Channels:** Defines the number of Audio Tracks to be used on the channel. The number of available track configurations is dependent on the Video hardware being used.
  - **Delay DVE:** Sets the number of frames the audio clip should be delayed in DVE mode before it can be mixed to the output. Default value is `4`.
  - **Delay Texture:** Sets the number of frames the audio clip should be delayed in texture mode before it can be mixed to the output. Default value is `4`.
- **VBI Settings**
  - **Enable VBI:** Defines whether VBI should be used for this channel. Set to Active or Inactive. Default mode is `Inactive`.
  - **Delay DVE:** Sets the number of frames VBI should be delayed before the clip can be used in DVE mode. Default value is `0` (Off).
  - **Delay Texture:** Sets the number of frames VBI should be delayed before the clip can be used in texture mode. Default value is `1`.

Copyright © 2026 Vizrt

- **Source Connector:** Replaces *Map To Viz Channel* and assigns the selected physical connector to that channel.
- **Expert Settings**
  - **Delay After Input Graph:** Adds a surface delay to compensate the time needed for pose estimation.
- **Multiconnector Mode:** Defines which type of multiconnector mode is used when performing quad UHD or "None" in case a 12G channel is in use (A, B, C, D). Additionally, only channels A, E and I are allowed by Viz Engine to change this mode, although channels E and I can have only 2SI or Square Division set. Default value is *None* for channel A and *2SI* for E and I. Otherwise this control disabled. Only available when using an X.mio5 12G card and Video System is configured to an UHD resolution.
- **Connector Settings**
  - **3G Level B:** Toggles the Level B on that connector if configured to 1080p or higher.
  - **Allow Super Black:** Toggles super black on this connector.
  - **Allow Super White:** Toggles super white on this connector.
  - **Chroma Clipping:** Toggles chroma clipping on this connector.

8.26.2 Live Input Properties: Type IP

- **Video System:** Either the required resolution or *Auto* to use the configured output resolution. In case of a video wall setup (full screen or custom resolution), *Auto* falls back to 1080i and must be reconfigured if a different resolution is required.
- **Colorimetry:** This setting is only relevant for HDR workflows and 1080p 50 / 59.94 / 60 and UHD. Available options are:
  - AUTO
  - *ITUR_BT_709*
  - *ITUR_BT_2100_HLG*
  - ITUR_BT_2100_SLOG3
- **Bits per channel:** Defines how many bits per channel are processed for an input. This is not supported on SD channels!
- **Video Settings**
  - **Contains Alpha:** Enables alpha for this input. Only available on connectors that have an associated alpha connector.
  - **Delay DVE:** Sets DVE Delay of video when used as DVE.
  - **Delay Texture:** Sets texture Delay of video when used as Texture.

> ⚠ **Note:** Setting the texture delay of progressive inputs to `1` frame has a performance impact that scales with the aggregate bandwidth of the input signals.

- **Audio Settings**
  - **Enable Audio:** Toggles audio for this channel. When set to `Inactive`, audio is disabled. Default value is `Active`.
  - **Audio Channels:** Defines the number of Audio Tracks to be used on the channel. The number of available track configurations is dependent on the used Video hardware.
  - **Delay DVE:** Sets the number of frames the audio clip should be delayed in DVE mode before it can be mixed to the output. Default value is `4`.

Copyright © 2026 Vizrt

    - **Delay Texture:** Sets the number of frames the audio clip should be delayed in texture mode before it can be mixed to the output. Default value is `4`.
- **VBI Settings**
    - **Enable VBI:** Defines whether VBI should be used for this channel. Set to Active or Inactive. Default mode is `Inactive`.
    - **Delay DVE:** Sets the number of frames VBI should be delayed, before the clip can be used, in DVE mode. Default value is `0` (Off).
    - **Delay Texture:** Sets the number of frames VBI should be delayed, before the clip can be used, in texture mode. Default value is `1`.
- **Source Connector:** Replaces *Map To Viz Channel* and assigns the selected physical connector to that channel.
- **Expert Settings**
    - **Delay After Input Graph:** Adds a surface delay to compensate the time needed for pose estimation.
- **Connector Settings**
    - **Allow Super Black:** Toggles super black on this connector.
    - **Allow Super White:** Toggles super white on this connector.
    - **Chroma Clipping:** Toggles chroma clipping on this connector.

Warning: Although any Source Connector chosen here is correctly saved, when Viz Engine starts, it checks whether the available bandwidth in the SFP's allows this configuration to be used. In case an SFP doesn't have enough bandwidth, Viz Engine tries to use the first available connector in the other SFP (if it is available). If the other SFP doesn't have enough available bandwidth either, this channel is completely discarded.

Important: If the bandwidth is exceeded or to move an input explicitly to the second SFP pair, see Explicit Usage of the Second SFP Pair on Matrox X.mio5 Q25 for more details.

8.26.3 Live Input Properties: Type STREAM

- **Video System:** Either the required resolution or *Auto* to use the configured output resolution. In case of a video wall setup (full screen or custom resolution), *Auto* falls back to 1080i and must be reconfigured if a different resolution is required.
- **Stream Settings**
    - **Enable Audio:** If audio should be received or not
    - **URL:** URL of the stream in the format *[protocol]://@ip:port*. Possible protocol values are *RTP*, *RTSP*, *RTMP*, *UDP*, and *SRT* (for example, udp://@239.0.1.10:50202).
    - **NIC Address:** NIC that is receiving the stream.
- **RTSP and RTMP Settings:**
    - **SRC UDP Port:** Source port only used for RTSP. Must be different for every active RTPS stream.
- **SRT Settings:**
    - **Connection Mode**
    - **Remote URL** and **Passphrase**
    - **Seconds To Wait For Connection:** Seconds to wait until a connection approach is being aborted, Latency in Ms.

Copyright © 2026 Vizrt

- **Access Unit Aligned Input:** If `true`, the user application provides the decoder with data that is aligned on an access unit. Use if there are artifacts in input due to Open GOP.
- **Disable AV Sync:** Disables AV Sync. When there is no audio in the stream, this must be manually set to `true`.
- **Use SW Decoding:** Forces the use of Software Decoding even if M264 is available.
- **Expert Settings**
  - **Delay After Input Graph:** Adds a surface delay to compensate the time needed for pose estimation.

8.26.4 Live Input Properties: Type NDI

- **Video System:** Either the required resolution or *Auto* to use the configured output resolution. In case of a video wall setup (full screen or custom resolution), *Auto* falls back to 1080i and must be reconfigured if a different resolution is required.
- **NDI Settings**
  - **Enable Audio:** If audio should be received or not
  - **Source:** The URL of the NDI source to be used.
- **Expert Settings**
  - **Delay After Input Graph:** Adds a surface delay to compensate the time needed for pose estimation.

8.26.5 Live Input Properties: Type SHM - SMURF

- **Video System:** Either the required resolution or *Auto* to use the configured output resolution. In case of a video wall setup (full screen or custom resolution), *Auto* falls back to 1080i and must be reconfigured if a different resolution is required.
- **SHM Settings**
  - **Enable Audio:** If audio should be received or not
  - **SHM Key:** Shared Memory Key to be accessed.

8.26.6 Mapping of Legacy StreamIn Channels

When using a configuration file created by Viz Engine versions 3.14.1 and older (StreamIn channels are part of the configuration file), the corresponding stream input channels are mapped to live input channels 9 and upwards by default, to avoid conflicts with existing live input channels.

This behavior can be overwritten to start the re-mapping at live in channel 1 by setting the following configuration flag: *no_stream_channel_mapping_offset = 1*.

> ⚠ **Note:** To make the flag visible in the configuration file, Viz Engine needs to be started with *VerboseConfig = 1* once.

See Also

- Matrox Stream

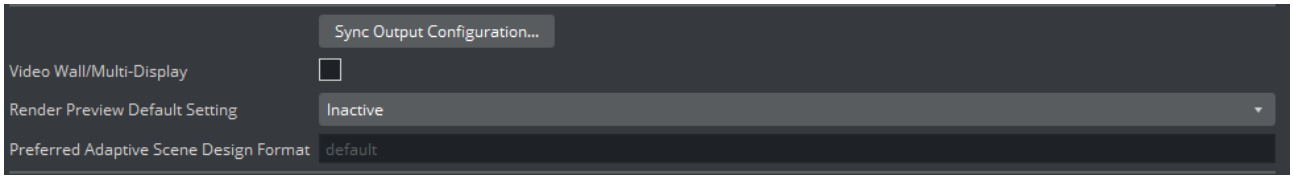
Copyright © 2026 Vizrt

8.27 Video Output

Use the Video Output section to configure special settings for video output, such as SPG settings and so on.

- Video Output Properties
- Output Format / Colorimetry / Bits Per Channel
- Video Output Editor
- Video Output Channel

8.27.1 Video Output Properties



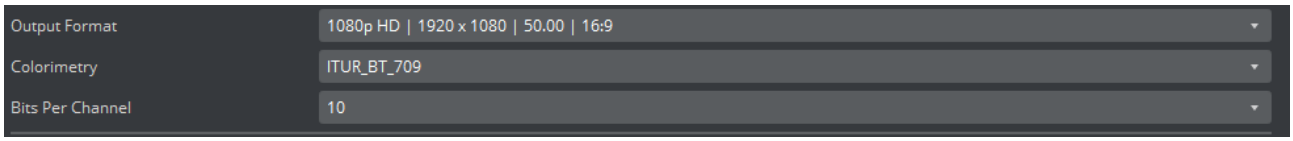
- **Sync Output Configuration:** Opens the Video Output Editor. You can also open the Video Output Editor by pressing **ALT + V**.
- **Video Wall/Multi-Display:** Sets the main output to the Digital Visual Interface (DVI).

Important: For video wall setups, this setting must be active and the output format must be set to *FULLSCREEN*.

- **Render Preview Default Setting:** Sets the default value for the **Preview** button (see Control Buttons), when Viz Engine is in *On Air* mode.
  - **Inactive:** Renders only video out signals. This increases performance, as the renderer does not have to render into an editor on-screen and into pixel buffer.
  - **Active:** Renders both video out signals and on-screen (this decreases performance).
  - **Fullscreen:** Sets the On Air window to screen size.
- **Preferred Adaptive Scene Design Format:** Defines which adaptive format to use. If, for example, a scene holds a *portrait* format, this format is used instead of the default format.

8.27.2 Output Format / Colorimetry / Bits Per Channel

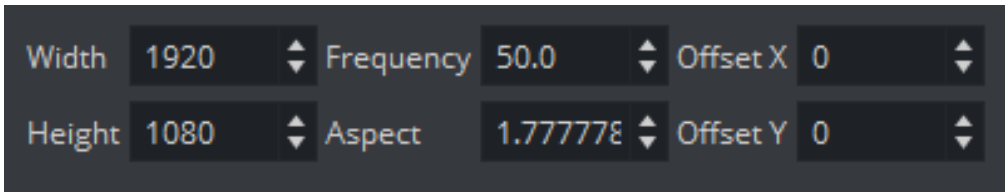
You set the output format, colorimetry, and bits per channel of the rendering engine using these combo boxes. *All* video hardware configurations are associated with the video standard set here as the output format. This setting also defines the frequency (frame rate) at which Viz Engine runs.



Copyright © 2026 Vizrt

For PAL and NTSC, you can set the aspect ratio to 4:3 (standard TV) and 16:9 (wide screen TV). **Fullscreen** sets the output format to the screen size of the current machine. Fullscreen also allows to modify the frame rate setting, but not other settings. Viz Engine supports UHDTV, 4K, UHDTV2, and 8K. UHDTV, 4K, UHDTV2, and 8K formats are only displayed when the system hardware supports them.

Choose *User Defined* to display a panel below the output format list. You can use user defined output format to meet the requirements of multi-pipe systems, such as a video wall. Configure the multi-pipe settings in the right part of the Editor.



- **Width:** Sets the width in pixels.
- **Height:** Sets the height in pixels.
- **Frame:** Sets the refresh rate/frequency per frame in hertz (Hz).
- **Aspect:** Sets the aspect ratio. For example, 1.778:1, which is 16:9 or 1.333:1, which is 4:3. The designer can set other aspect ratios for each scene, as described in the **Scene Settings** page in the *Scene Management* section of the Viz Artist User Guide.

Note: Make sure the physical refresh rate of the graphics hardware and the video hardware are configured with compatible settings.

- **Offset X:** Sets the horizontal alignment in pixels, on the screen. Value is calculated from the top left of the screen.
- **Offset Y:** Sets the vertical alignment in pixels, on the screen. Value is calculated from the top left of the screen.

There are three frequency groups/families: 50, 59,94, and 60 Hz. These define the output format and how fast Viz Engine operates. The frequency is the same as frames per second. This also defines the input format that is allowed, so an NTSC SD input cannot produce a PAL SD output, but an HD input with the same frequency as the SD output would work.

Information: NTSC and PAL are not supported by X.Mio5 boards (2110).

Fullscreen and *User Defined* hide the **Video Output Channel**, as these output formats do not support any kind of output channel.

Information: The **Colorimetry** option only appears for 1080p 50 / 59.94 / 60, UHD, 4K, 8K, Fullscreen and User Defined. At the other formats the output is BT.709 (SD: BT.601).

UHDTV Support

- UHDTV resolution is currently supported on Matrox X.mio3 - DSX LE 4 (SDI video boards only) and X.mio5 (recommended).

Copyright © 2026 Vizrt

- UHDTV2 resolution is currently supported on X.mio5 X2. Only 50 FPS is currently supported.

8.27.3 Video Output Editor

The Video Output Editor defines the synchronization standard and the output signal phases.

- **Freerun:** Locks Viz Engine to a clock signal on the video board.
- **Blackburst:** Locks Viz Engine to a Blackburst GenLock signal.
- **Tri-level:** Locks Viz Engine to a Tri-Level GenLock signal.
- **Auto:** Auto detects the genlock signal and locks to it.
- **Digital Input 1 and 2:** Locks Viz Engine to the signal on Input 1 or 2.
- **PTP:** Uses a Precision Time Protocol source for synchronizing. Mainly used in IP networks.
- **H-Phase and V-Phase:** Shifts the output signal with respect to the sync signal.

Note: The *Auto* option is only available on Matrox boards. On Matrox boards the h- and v-phase settings, are updated automatically with default values based on the output resolution (Except IP-based cards). Measuring the exact h- and v-phase is still required for each setup.

Make the V- and H-Phase Values Coincide

- Set the **V-phase** value.
  - The *V-granularity* is taken from the genlock.
  - The *V-delay* is calculated from *V-phase* * *V-granularity*.
  - The genlock is set with this *V-delay*.
- Set the **H-phase** value. Note that there is a distinction between whether the H-phase is a positive or a negative value.
  - If the **H-phase > 0**:
    - The genlock *H-delay* is set to `0`.
    - The *H-granularity* is taken from the video out channel.
    - The *H-delay* is calculated from *H-phase* * *H-granularity*.
    - The fill and key channels are set with this *H-delay*.
  - If the **H-phase <=0**:
    - The fill and key channel *H-delay* is set to `0`.
    - The *H-granularity* is taken from the genlock.
    - The *H-delay* is calculated from - *H-phase* * *H-granularity*.
    - The genlock is set with this *H-delay*.

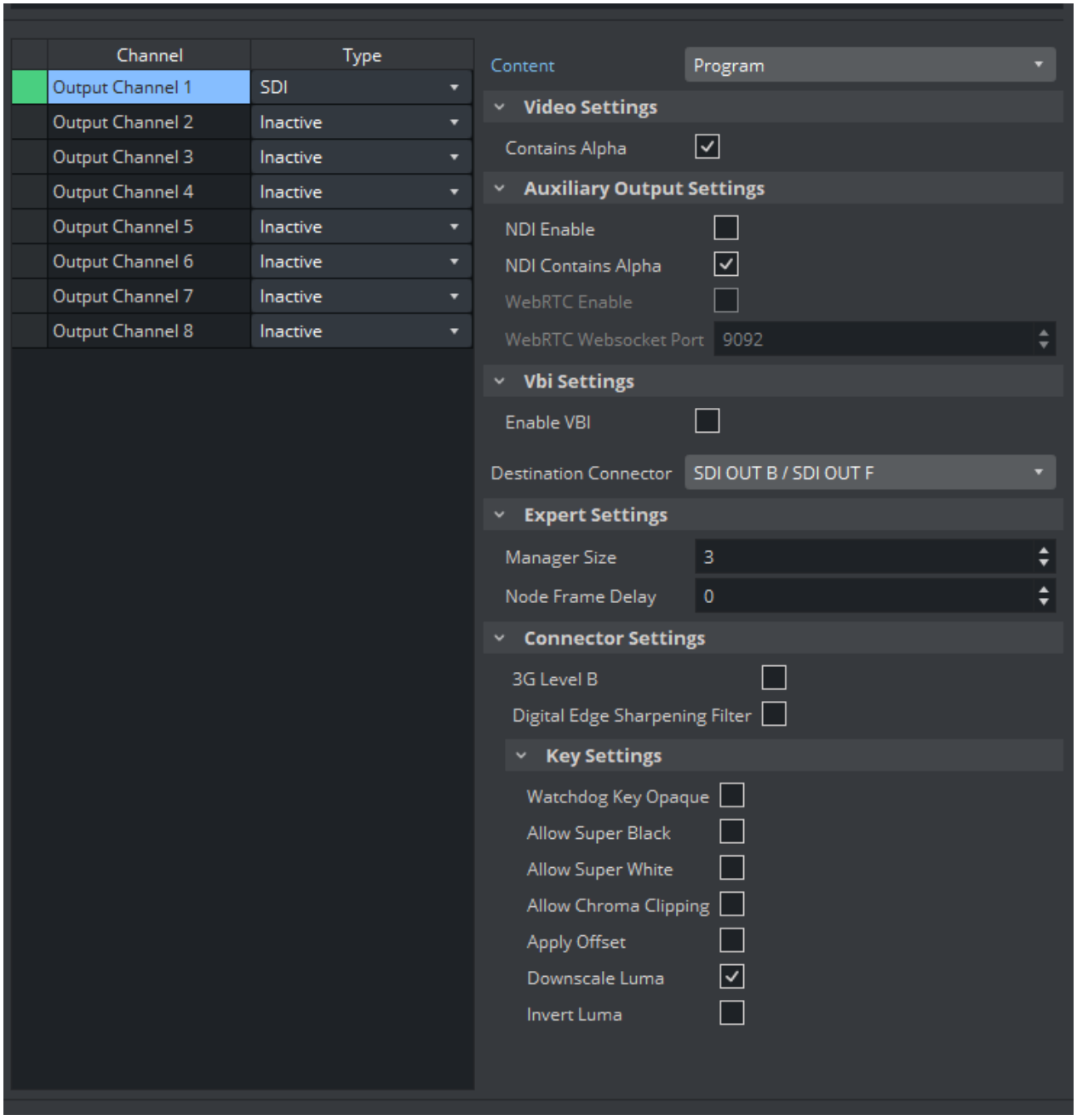
Note: The granularity and possible min/max values are printed to the Viz Artist/Engine console during startup.

It should be taken into account that when the genlock video format is different from the fill/key video format, the value of the *V-delay* matches the genlock lines and not the video output lines. The same applies to *negative H-phase* values.

Copyright © 2026 Vizrt

8.27.4 Video Output Channel

In the **Video Output Channel** panel, you select which Viz Artist/Viz Engine Output is mapped to the Matrox connector. The **Video Output Channel** panel shows the mapped Viz output channel and its editable parameters. The configuration has the same logic and looks familiar to the **Video Input** section of the configuration menu (see Video Input: Live Input). This panel is not displayed if Matrox hardware is not used.



Copyright © 2026 Vizrt

Important: If the configuration is done for a Dual Channel UHD setup using SMPTE ST 2110, the second Viz Engine needs to use the second SFP pair for output. See Explicit Usage of the Second SFP Pair on Matrox X.mio5 Q25 and DSX LE5 Q25 for more information.

To enable an Output Channel, select the row and choose the Type of channel: SDI, IP, or NDI (and, on DSX.core/M264 systems, the fixed Internal type on Output1). After that, choose the Content for the channel.

There is no longer a fixed Output1=Program / Output2=Preview pairing. Each of the up to 8 output channels is configured independently — you can have any number of Program outputs (not just one or two), in any order, and mix types freely (for example, an SDI program, a second SDI program at a different resolution, and an NDI output carrying a Matte Scene, all at once).

Resolution and node sharing: an additional output configured at the same resolution as the main program shares that program's rendering node/connector pipeline (the old single-node behavior). An additional output configured at a different resolution gets its own independent path instead. The rules for that independent path depend on Type:

- **SDI**: the resolution must be one of the standard broadcast resolutions (720p, 1080p, 1080i, or UHD if the board supports it).
- **IP**: the resolution can be arbitrary (see the custom resolution list below), but it must be progressive, and the underlying card node is sized up to the next multiple of 64 pixels in width / 2 lines in height to accommodate it.
- **NDI**: the resolution is fully arbitrary and independent — there's no shared card node to size around.

The possible content values are:

- **Unused**: Does not map this channel for output.
- **Program**: Maps a Program output to the selected video output. Any number of output channels can be set to Program — this is no longer capped at two.
- **Still Preview**: Maps the Preview output to the selected video output. Only available on SDI or IP channels.

Information: Still Preview requires an SDI or IP output channel. It no longer shares a slot with Program outputs — you can configure any number of Program outputs alongside a Still Preview, limited only by the number of output channels available.

- **Matte Scene**: Used for virtual studios, the effective matte scene.
- **KeyerAuxChannel**: Shows the key preview to adjust the Chroma Keyer.
- **Media Asset**: Outputs any media asset — Live, Clip, Web, SuperChannel, Image, Aux, etc. — chosen independently of the channel's own input configuration. This replaces the old Graphics content value; where Graphics only let you pick a GFX channel, Media Asset lets you monitor any asset type.
- **Clean and Downscaled Preview** are no longer separate top-level Content values. They are now child outputs of a Program output (set via a separate Child Output control), and are only offered on a Program output that is SDI or IP, is not the main program, and — for Downscaled Preview — only when the main program is configured at UHD or higher. Their behavior is otherwise unchanged: Clean outputs DVE video without overlay graphics (with its own audio mix, excluding stage audio), and requires Matrox X.mio3, X.mio3 IP, X.mio5, or X.mio5 IP (Watchdog functionality is not supported on X.mio5/X.mio5 IP, and still requires VideoOut A = Clean, VideoOut B = Program); Downscaled Preview outputs the Program content downscaled to HD, and is ignored unless the Program output is UHD.

Notes: Graphics output only work when the main scene is VER, but the nested scene can be *VER* or *Classic*.

Copyright © 2026 Vizrt

A clean feed increases the video In → Out delay, by *one* frame. At least one program always to be configured.
Clean feed and **Downscaled Preview** are only available on boards with a hardware compositor available (not DSX LE series).

Output Type

Each output channel (Output1…OutputN) has an independent Type: SDI, IP, NDI, or (DSX.core/M264 only) Internal. Type determines the underlying transport and, together with the board, what resolutions and content are available.

Type	Transport	Audio	Resolution source	Notes
SDI	Physical SDI connector	Yes	Fixed broadcast list only: 720p, 1080p, 1080i, plus UHD if the board reports UHD output capability	Card-bound; supports Clean/ DownscaledPreview child outputs and Still Preview
IP	2110 IP Stream	Yes	output-resolutions.xml custom list, plus the resolution can equal the main program's resolution (shared "card node")	Additional IP outputs at a different resolution than the main must be progressive; engine aligns width to a multiple of 64px and height to a multiple of 2 lines to fit the card node; frame rate must match the card node
NDI	NDI network stream	No (video only)	output-resolutions.xml custom list, fully arbitrary (no card-node alignment)	Not tied to a physical connector; independent from the legacy Program-only NDI and NDI Aux sender
Internal	None (software)	N/A	Fixed to the global output format	DSX.core/M264 only; always pinned to Output1; content is fixed to Program; cannot be reassigned

New: NDI as an output Type (5.6, Parallel Outputs)

Alongside SDI and IP, an output channel can now be set to Type = NDI. This is a genuinely new capability, distinct from the long-standing Program-only NDI and NDI Aux sender

	Program-only NDI	New Type = NDI output channel
Where configured	NDIENABLE/NDICONTAINSALPHA under the first SDI/IP (or Internal) Program output's Aux settings	TYPE dropdown on any output channel
Content	Always mirrors that channel's Program feed	Any supported content: Program, Matte Scene, Keyer Aux, Media Asset (not Still Preview, not a Clean/DownscaledPreview child)

Copyright © 2026 Vizrt

	Program-only NDI	New Type = NDI output channel
Resolution	Always the global output resolution	Independent — selectable from output-resolutions.xml, arbitrary
Instances	One global instance	One per configured output channel
Audio	Follows the program's audio config	None — video-only

Because it's video-only, don't use the new NDI output type as a replacement for an audio-carrying feed; use SDI/IP, or the legacy Aux NDI sender if you need the audio-following-program behavior.

Requirements: an NDI output license (binary or counted) and the Parallel Outputs license feature. Both a Content and a Type combobox must be exposed — if you don't see NDI in the Type list, the license/feature isn't active.

Alpha / bit depth: NDI output channels support alpha and 16-bit output depending on the configured pixel format (matches PA16/P216 handling used elsewhere), same as SDI/IP Contains Alpha.

Resolution cap: like every other sender in the engine, the NDI output is still capped by the license's maximum video resolution — an NDI channel cannot exceed what your Viz Engine license allows even if you add a bigger entry to output-resolutions.xml.

Custom output resolutions — output-resolutions.xml

Location: C:\ProgramData\vizrt\VizEngine\output-resolutions.xml

- Created automatically with defaults the first time any output resolution list is requested, if the file doesn't already exist.
- Loaded once per engine session (edits require a restart to take effect).
- Used to populate the Resolution dropdown for **IP and NDI** output channels only; **SDI** always uses the fixed broadcast list and ignores this file.

Default file contents:

Alias	Width	Height	Scan mode
540p	960	540	progressive
720p	1280	720	progressive
1080p	1920	1080	progressive
1080i	1920	1080	interlaced
1440p	2560	1440	progressive
UHD	3840	2160	progressive

Schema — add your own **<Resolution>** entries the same way:

Copyright © 2026 Vizrt

```
<?xml version="1.0" encoding="UTF-8"?>
<OutputResolutions>
  <Resolution width="960" height="540" scanmode="progressive" alias="540p"/>
  <Resolution width="1280" height="720" scanmode="progressive" alias="720p"/>
  <Resolution width="1920" height="1080" scanmode="progressive" alias="1080p"/>
  <Resolution width="1920" height="1080" scanmode="interlaced" alias="1080i"/>
  <Resolution width="2560" height="1440" scanmode="progressive" alias="1440p"/>
  <Resolution width="3840" height="2160" scanmode="progressive" alias="UHD"/>
  <!-- custom example -->
  <Resolution width="1600" height="900" scanmode="progressive" alias="900p"/>
</OutputResolutions>
```

- **scanmode** accepts **progressive**, **interlaced**, or **auto** (auto follows the engine's global system progressive/interlaced setting).
- **alias** is what appears in the Resolution dropdown and in the config file (**RESOLUTION** command); it must be unique and non-empty or the entry is silently skipped.
- Selecting a UHD-sized custom entry (≥3840 wide or ≥2160 tall) on an IP or NDI output still requires the board to report UHD output capability — otherwise the engine rejects it with "UHD resolution not supported on this output."
- Selecting "Auto" resets width/height/scan mode to 0/0/Auto, meaning "use the general output configuration" (the global system format).

Content and child outputs, by type

Content	SDI	IP	NDI	Internal
Program	Yes	Yes	Yes	Yes (fixed)
Still Preview	Yes	Yes	No	No
Matte Scene / Keyer Aux / Media Asset	Yes	Yes	Yes	No
Clean / DownscaledPreview (child output)	Yes	Yes	No	No

- Clean/DownscaledPreview child outputs require an SDI or IP main Program output — NDI output channels can't host or depend on them.
- The engine always requires at least one active SDI or IP Program output before any dependent child/Still-Preview output can exist; NDI/other content types can be configured without one.

Output Licensing

Several independent output paths exist that can send video over NDI, WebRTC, or SHM, each licensed differently. This section documents the current licensing behavior for each, as implemented in the engine today.

Note: **SDI_OUT_BINARY**, **NDI_OUT_BINARY/NDI_OUT_COUNTED**, and **PARALLEL_OUTPUTS** are the only license features involved. There is no dedicated WebRTC license feature and no dedicated SMURF/SHM license feature.

Copyright © 2026 Vizrt

Licensing by output path

Output path	Configured via	License(s) required	Resolution-capped
SDI/IP output channel — Content = Program or Still Preview	Type = SDI/IP, Content = Program/StillPreview	SDI Out	Yes
SDI/IP output channel — Content = Matte Scene, Keyer Aux, or Media Asset	Type = SDI/IP, Content = MatteScene/ KeyerAux/MediaAsset	SDI Out + Parallel Outputs	Yes
NDI output channel (any Content)	Type = NDI	NDI Out (binary or counted) + Parallel Outputs	Yes
NDI Aux — Program clone	Output1 Aux settings, "NDI Enable"	NDI Out (binary or counted)	Not enforced at creation
WebRTC Aux — Program clone	Output1 Aux settings, "WebRTC Enable"	Parallel Outputs	Not enforced at creation
NDI Aux — per GFX/SuperChannel	GFX/SuperChannel config, NDI send	Parallel Outputs	Yes, per frame
SMURF/SHM Aux — per GFX/ SuperChannel	GFX/SuperChannel config, SHM key	None	No
RTP/Mezzanine Out (legacy, unrelated to Parallel Outputs)	Legacy RTP Out aux	Mezzanine IP Out	n/a

Notes on SDI/IP output licensing

- SDI Out is the same license feature that has always covered SDI/IP output — it is unchanged for a single Program output.
- Parallel Outputs is only required on an SDI/IP channel when its Content is Matte Scene, Keyer Aux, or Media Asset. Additional Program or Still Preview outputs on SDI/IP do not require Parallel Outputs — only SDI Out, same as before.

Notes on NDI-shaped outputs

There are three distinct NDI-sending features, each licensed differently:

1. NDI output channel (Type = NDI) — requires both NDI Out and Parallel Outputs, regardless of Content.
2. NDI Aux (Program clone) — the legacy Output1 "NDI Enable" checkbox that always mirrors the main Program feed — requires only NDI Out.
3. NDI Aux (per GFX/SuperChannel) — the classic per-channel "send this channel over NDI" option — requires only Parallel Outputs, and does not require an NDI Out license.

Copyright © 2026 Vizrt

Notes on WebRTC and SMURF/SHM

- WebRTC has no dedicated license feature of its own; the Program-clone WebRTC Aux sender is gated solely by Parallel Outputs.
- SMURF/SHM Aux (per GFX/SuperChannel SHM output) has no license check and no resolution cap — it is unconditionally available whenever a channel supports it and has an SHM key configured.

Output Type IP/SDI Settings

Video Settings

- **Contains Alpha:** Defines if this output channel provides key information on the associated key output connector.

Auxiliary Output Settings

This option only appears if the output type is SDI/IP and output content is Program. Additional output content program will not show this option.

- **NDI Enable:** Enables NDI output as a clone output of the output program. Same resolution. Supports DVE.
- **NDI Contains Alpha:** Enabled alpha for the NDI auxiliary output.
- **WebRTC Enable:** Enables WebRTC output as a clone output of the output program. Same resolution. Supports DVE.
- **WebRTC Websocket Port:** Port used for the WebRTC connectiong

VBI Settings

- **Enable VBI:** Enabled VBI for the output channel.

Destination Connector

Chooses which connector to use

Expert Settings

- **Manager Size (frames):** Sets the number of frames available in the on-board memory for output. A too high value may cause memory problems on the Matrox board. Default value is `3`.
- **Node Frame Delay:** Specify the delay of the Matrox output node. This is relevant if the output node is intended to be used for recording by another application such as Channel Recorder.

Connector Settings

- **3G Level B:** Activates `Level B` for 3G mode in 1080p 50/60/60M (default mode is `Level A`).
- **Digital Edge Sharpening Filter:** Applies an edge sharpening filter to digital output video. Default mode is `Inactive`. SD configurations only.

Copyright © 2026 Vizrt

Key Settings

- **Watchdog Key Opaque:** Specifies if the output key must be opaque or transparent when the watchdog unit activates. Default mode is *Inactive*.
- **Allow Super Black:** Allows the luminance of the output signal to fall below the nominal SMPTE pixel values (7.5 IRE units) when enabled. Default mode is *Inactive*.
- **Allow Super White:** Allows the luminance of the output signal to exceed the nominal SMPTE pixel values (100 IRE units) when enabled. Default mode is *Inactive*.
- **Allow Chroma Clipping:** Determines whether to clip over-saturated chroma levels in the active portion of the output video signal. Default mode is *Inactive*.
- **Apply Offset:** Applies an offset to the luminance values such that the inverted result still falls within the 16-235 range. Default mode is *Inactive*.
- **Downscale Luma:** Compresses the luminance range of the output key signal from 0-255 to 16-235. Default mode is *Active*.
- **Invert Luma:** Inverts the luminance part of the output key signal (inverts the key). Default mode is *Inactive*.

Multiconnector Mode

Defines which multiconnector mode to use for 3G / 12G SDI channels. This setting is only available for Series 5 Matrox SDI cards (for example, X.mio5 12G SDI) when configured to UHD.

- **None:** 12G single connector mode.
- **2SI:** Quad-Link 2 sample interleave.
- **Square Division:** Quad-Link Square division.

Important: Multiconnector mode is only enabled for UHD resolutions and when at least four connectors are available.
If fewer than four connectors would be free for SQD or 2SI, this option is greyed-out and the connectors automatically run at their maximum bandwidth.

See Also

- Video Input Clip Input
- Video Input Live Input

Copyright © 2026 Vizrt

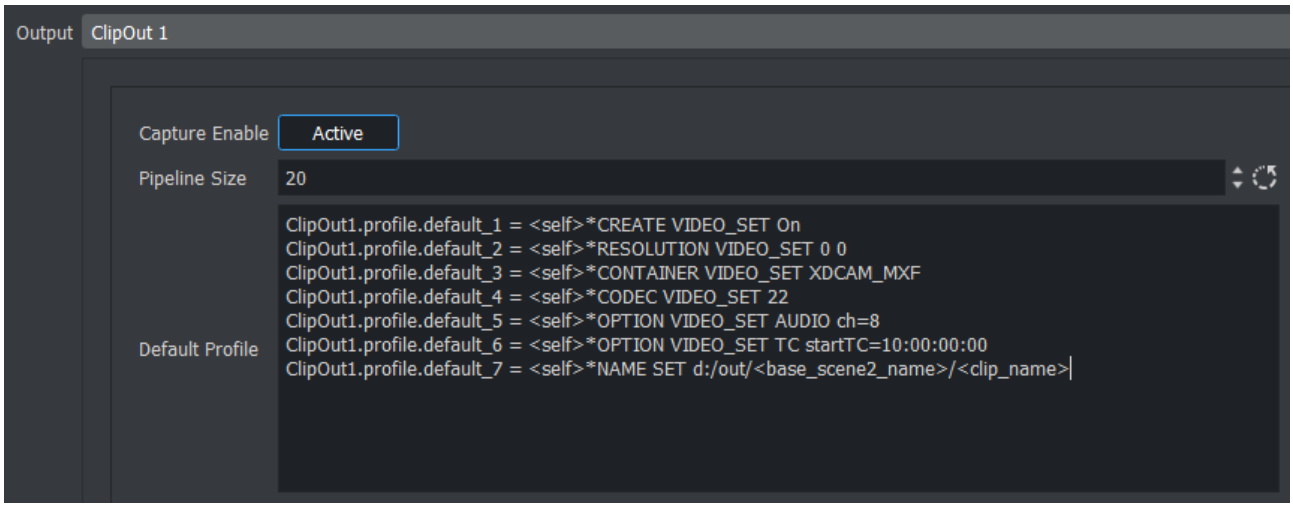
8.28 Video Output: Clip Output

Matrox Clip Out channel writes surfaces from your Matrox device into clip files. This can be used for example to record your output into a single clip file.

- To Configure Clip Output
- Clip-out Channel Placeholders
- Sample Commands

8.28.1 To Configure Clip Output

In the Video Output: Clip Output panel, configure the available Clip Channel outputs.



- **Capture Enable:** Enables or disables the clip writer functionality. The main use is to give control over host memory resources. When the clip writer functionality is not needed, the clip out channel does not need to be allocated.
- **Pipeline Size:** Controls the number of frames that the clip writer uses to handle a file. It is recommended to leave the default value.
- **Default Profile:** Contains a default profile that can be loaded on request, with the command `RENDERER*VIDEO*CLIPOUT*1*PROFILE APPLY`. Example:

```
# the <self> placeholder gets replaced with the proper RENDERER*VIDEO*CLIPOUT*1
ClipOut1.profile.default_1 = <self>*CREATE VIDEO_SET On
ClipOut1.profile.default_2 = <self>*RESOLUTION VIDEO_SET 0 0
ClipOut1.profile.default_3 = <self>*CONTAINER VIDEO_SET XDCAM_MXF
ClipOut1.profile.default_4 = <self>*CODEC VIDEO_SET 22
ClipOut1.profile.default_5 = <self>*OPTION VIDEO_SET AUDIO ch=8
ClipOut1.profile.default_6 = <self>*OPTION VIDEO_SET TC startTC=10:00:00:00
```

Copyright © 2026 Vizrt

```
ClipOut1.profile.default_7 = <self>*NAME SET d:/out/<base_scene2_name>/
<clip_name>
```

8.28.2 Clip-out Channel Placeholders

The available placeholders for clip-out channel are:

Variable	Definition
`<hostname>`	Expands to the host name.
`<if0>`	Expands to the IPV4 network address of the first network interface.
`<if1>`	Expands to the IPV4 network address of the second network interface.
`<if2>`	Expands to the IPV4 network address of the third network interface.
`<absolute_scene_name>`	Expands to the complete path of the loaded scene.
`<absolute_scene2_name>`	Expands to the complete path of the loaded scene2.

8.28.3 Sample Commands

1080i50 A MPEG-2 Codec / MpegIFrame422ProfileHighLevel_WithAlpha with a Bitrate of 150 and 16 Channel Audio

```
RENDERER*VIDEO*CLIPOUT*1*CREATE VIDEO_SET ON
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION VIDEO_SET 0 0
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET AVI
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET 36
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AUDIO ch=16 bits=24/32
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET IFRAME bitrate=150
RENDERER*VIDEO*CLIPOUT*1*NAME SET "D:
\clipout\1080i50_MPEG-2__MpegIFrame422ProfileHighLevel_WithAlpha[ch=16 bits=24-32]
[bitrate=150]"
RENDERER*VIDEO*CLIPOUT*1*CONTROL START
...
RENDERER*VIDEO*CLIPOUT*1*CONTROL PAUSE
```

Copyright © 2026 Vizrt

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
```

1080p50 Video AVC-Intra Encoder / AVCIntraClass50 with Timecode, No Audio

```
RENDERER*VIDEO*CLIPOUT*1*CREATE VIDEO_SET ON
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION VIDEO_SET 0 0
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET P2_MXF
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET 43
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AVCINTRA profile=1
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET TC startTC=00:10:00:00
RENDERER*VIDEO*CLIPOUT*1*NAME SET "D:\clipout\1080p50_Video_AVC-
Intra_Encoder_AVCIntraClass50[TimeCode]"
RENDERER*VIDEO*CLIPOUT*1*CONTROL START
...
RENDERER*VIDEO*CLIPOUT*1*CONTROL PAUSE
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
```

720p DVCPRO 100 Codec / D12_720_4to3_422_Fr1 Four Channel Audio

```
RENDERER*VIDEO*CLIPOUT*1*CREATE VIDEO_SET ON
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION VIDEO_SET 0 0
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET AVI
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET 17
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AUDIO ch=4 bits=24/32
RENDERER*VIDEO*CLIPOUT*1*NAME SET "D:
\clipout\720p60M_DVCPRO_100__D12_720_4to3_422_Fr1[ch=4 bits=24-32]"
...
RENDERER*VIDEO*CLIPOUT*1*CONTROL PAUSE
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
```

UHD 2160p XAVC Eight Channel Audio with M264 Encoder

```
RENDERER*VIDEO*CLIPOUT*1*CREATE VIDEO_SET On
RENDERER*VIDEO*CLIPOUT*1*CREATE AUDIO_SET Off
RENDERER*VIDEO*CLIPOUT*1*CREATE KEY_SET Off
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION VIDEO_SET 0 0
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET XAVC_MXF
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET 47
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AUDIO ch=8 bits=24/32
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET XAVC profile=HDIntra100CBG use_hw=1
RENDERER*VIDEO*CLIPOUT*1*NAME SET "C:\test.mxf"
RENDERER*VIDEO*CLIPOUT*1*CONTROL RECORD 1000
```

Copyright © 2026 Vizrt

Information: To record in XAVC UHD, the output color depth must be set to 10-bit (*output_bpc = 10*). Recording in XAVC UHD requires a M264 board (OPTION VIDEO_SET XAVC *profile=HDIntra100CBG use_hw=1*).

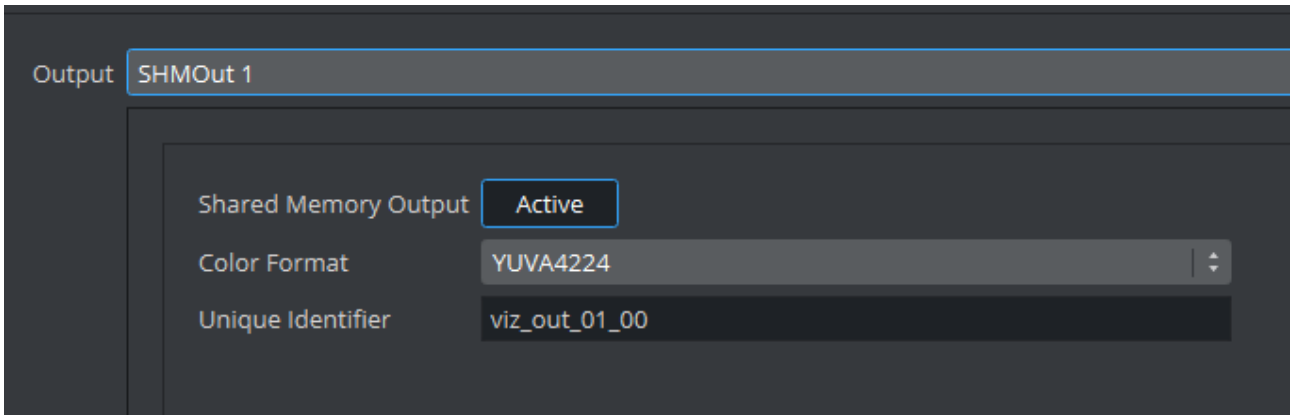
See Also

Clipout Channel Usage

Copyright © 2026 Vizrt

8.29 Video Output: SHM Output

Configure Shared Memory Output channels from this panel. The shared memory output is only available on systems with no video boards, or if **Check Video Card** is set to `None` in the Video Board section.

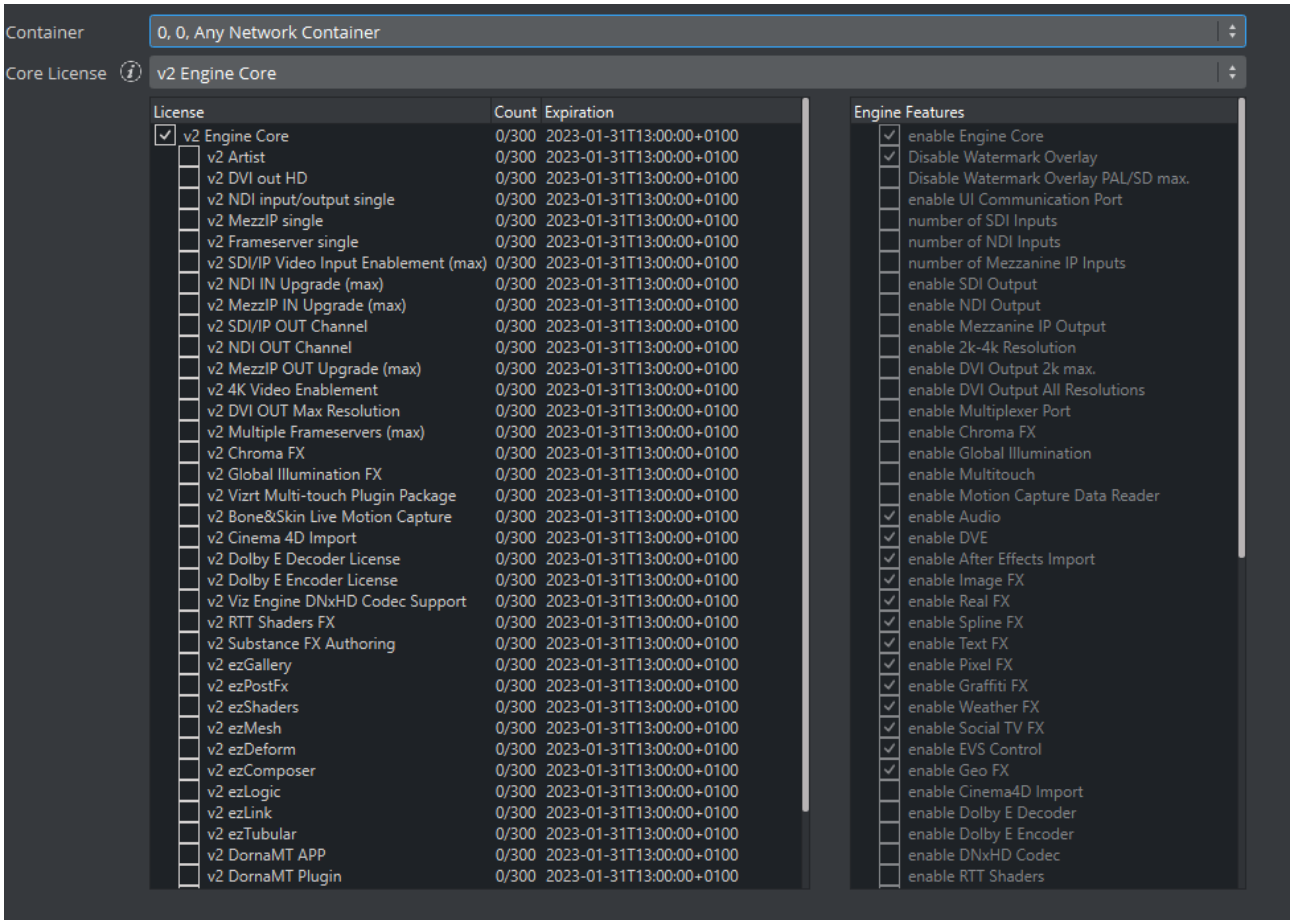


- **Shared Memory Output:** Sets the output mode for Shared Memory. Options are `Active` or `Inactive`.
- **Color Format:** The format the Shared memory texture is using: can be `YUVA4224` or `RGBA`.
- **Unique Identifier:** Changes the name of the Shared Memory if required. Used when connecting to services such as Viz Coder.

Copyright © 2026 Vizrt

8.30 Viz License Configuration

The Viz License Configuration allows users to configure their Viz Engine for WIBU Licensing.



The container specifies the location the system searches for a WIBU license. These are all available containers provided by the CodeMeter Runtime. If a license server is not listed here, be sure to check the Server Search List in your CodeMeter Runtime Configuration.

To quickly open the CodeMeter WebControl User Interface, press the **Open CodeMeter Administration** button.

> **Information:** Retrieving the license information from your network sources requires some time and currently blocks the dialog.

The license configuration is split into two columns:

The left column allows you to specify which licenses you want to acquire from your selected container.

- **Core License** selects the Core a user must acquire to start Viz Engine. The can be either:
  - Engine Core
  - Engine Core VER
  - Preview Core

Copyright © 2026 Vizrt

- Preview Core VER
- Artist for Learners
- Frameserver Core
- Viz Libero Core
- Viz Arena Camera Tracking Engine

Additional Licenses can be enabled by setting the checkmark next to it.

The right column shows which Viz Engine application features are enabled by your selected licenses. Several features are automatically enabled when you select a core license. To check which features are enabled or disabled by which license, hover over the feature to get a description.

For additional details related to each license, see WIBU Licensing System.

Important: Any change in the license configuration requires a restart of Viz Engine.

Copyright © 2026 Vizrt

8.31 Viz One

To enable data exchange between Viz One and Viz Engine, two services are required:

- A Viz One File System Monitor (Fsmon) Service (monitors files which are located in the Clip Data directory location (`--root=d:/`)).
- A File Transfer (Mediaftp) Service (transfers files to the Viz Engine).

Information: Both **Mediaftp** and **Fsmon** have individual *.msi* installers rather than being included in the Viz Engine *.msi* starting with Viz Engine version 4.0. They are included in the Viz Artist Bundle installer.

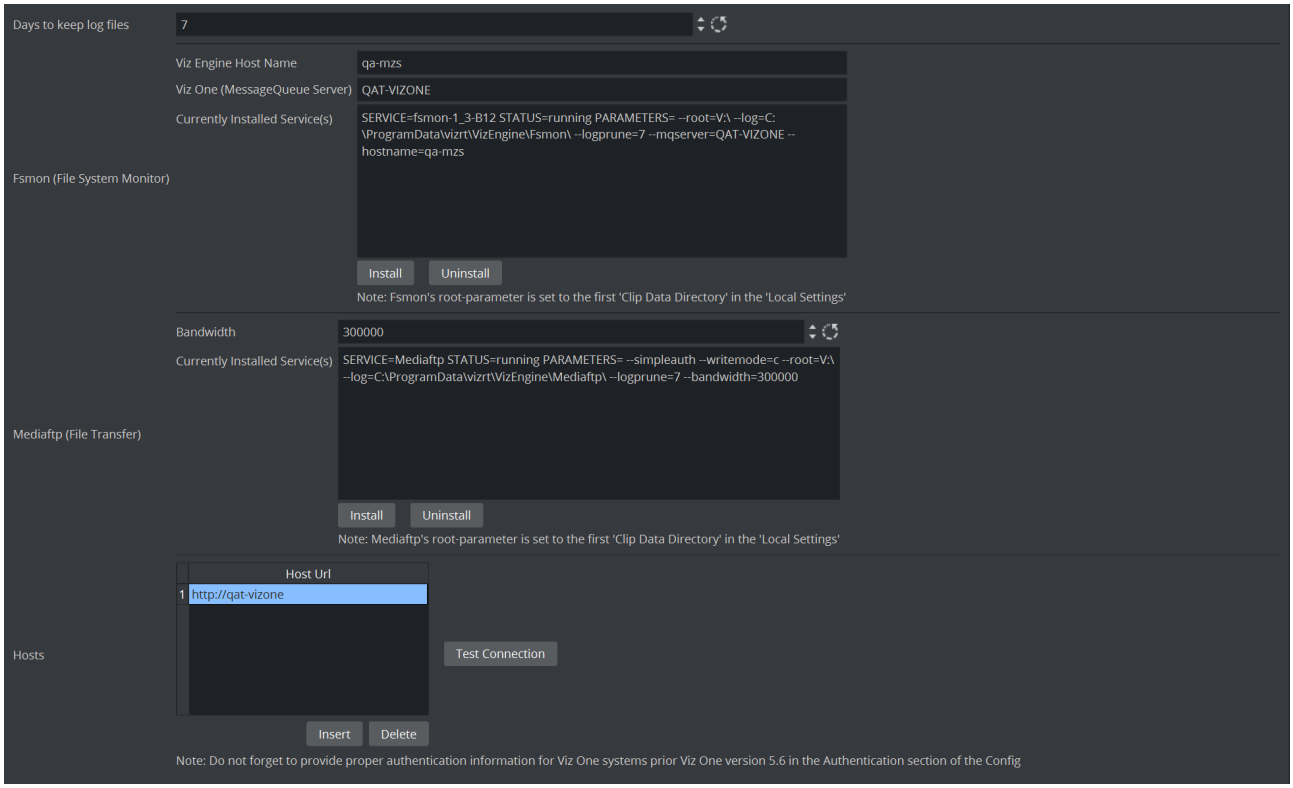
IMPORTANT! If the location of the Clip Data directory is changed after installation, remove and install the two services again. They automatically set to the new Clip Data Directory (see Local Settings).

To search, view and select video clips from Viz One, configure the Viz One Browser.

Note: Viz Artist features a *Viz One Transfer Status* pane that shows the progress of any active and finished clip transfers from Viz One. This pane requires the MediaFTP (File Transfer) and Fsmon (File System Monitor) services to be installed and running, and the correct configuration of the Viz One MessageQueue Server.

To use Viz One with Viz Artist, each Viz Artist/Viz One integration must be authenticated through the Authentication panel in the Viz Configuration. See Integration with Viz One for the Fsmon and Mediaftp install and remove procedures.

Copyright © 2026 Vizrt



8.31.1 Viz One Properties

This section details the properties of the Viz One configuration panel.

- **Days to keep log files:** Sets the number of days to keep log files (default is seven days). Log files older than the set number of days are deleted.

Fsmon (File System Monitor)

- **Viz Engine Host Name:** Sets the name of the local host (local host name entered by default). Make sure that the host name is exactly the same as the string specified in the Viz One Server Configuration
- **Viz One (MessageQueue Server):** Sets the host name of the active Message Queue server (do not use a protocol prefix, for example, *http://*)

IMPORTANT! To view and transfer files in Viz Artist, the host name entered in the **Viz One (MessageQueue Server)** must also be entered in the Viz One Browser panel.

- **Currently Installed Service(s):** Shows the currently installed Fsmon service, with its parameters.
- **Install/Uninstall:** Installs and removes Fsmon service.

Note: Any currently installed service must be removed before a new service can be installed.

Copyright © 2026 Vizrt

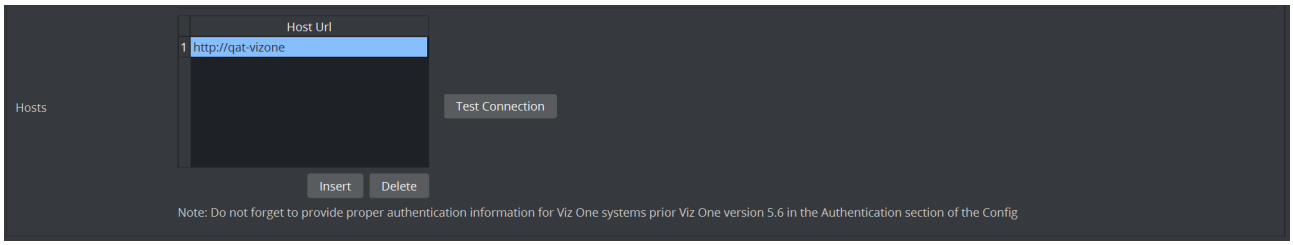
Mediaftp (File Transfer)

- **Band Width:** Sets the transfer bandwidth, in Kbits per second.
- **Currently Installed Service(s):** Shows the currently installed Mediaftp service with its parameters.
- **Install/Uninstall:** Installs and removes Mediaftp service.

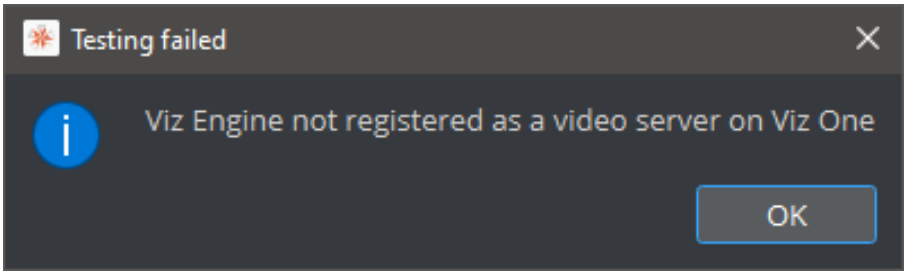
> **Note:** Any currently installed service must be removed before a new service can be installed.

Viz One Browser

This is the configuration for the Viz One Browser in the Media Asset tab (see the Media Asset section of the Viz Artist User Guide).

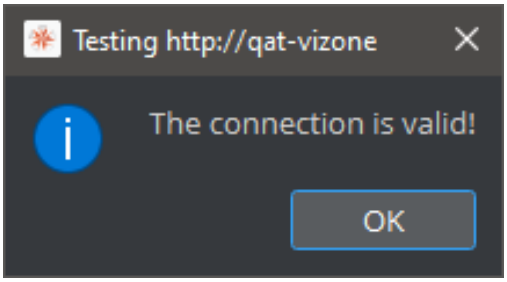


- **Hosts:** Opens a dialog to provide Viz Engine with the host-name for the desired Viz One instance when the **Add** button is clicked. The host name must include the protocol prefix (for example, *http://vme57-sia*). To remove a previously configured Viz One instance, select it from the list of hosts and click the **Delete** button.
- **Test Connection:** Selects the desired host and click the **Test Connection** button to check if the configured Viz One connection is working correctly.
  - If Test Connection returns that the Viz Engine is not registered as a Video Server, the Engine must be added as a server in the Viz One Studio Administration. Please refer to the Viz One documentation for further information.



  - If Test Connection returns that the connection is valid, the Viz One instance has been configured correctly and is ready for use with Viz Engine.

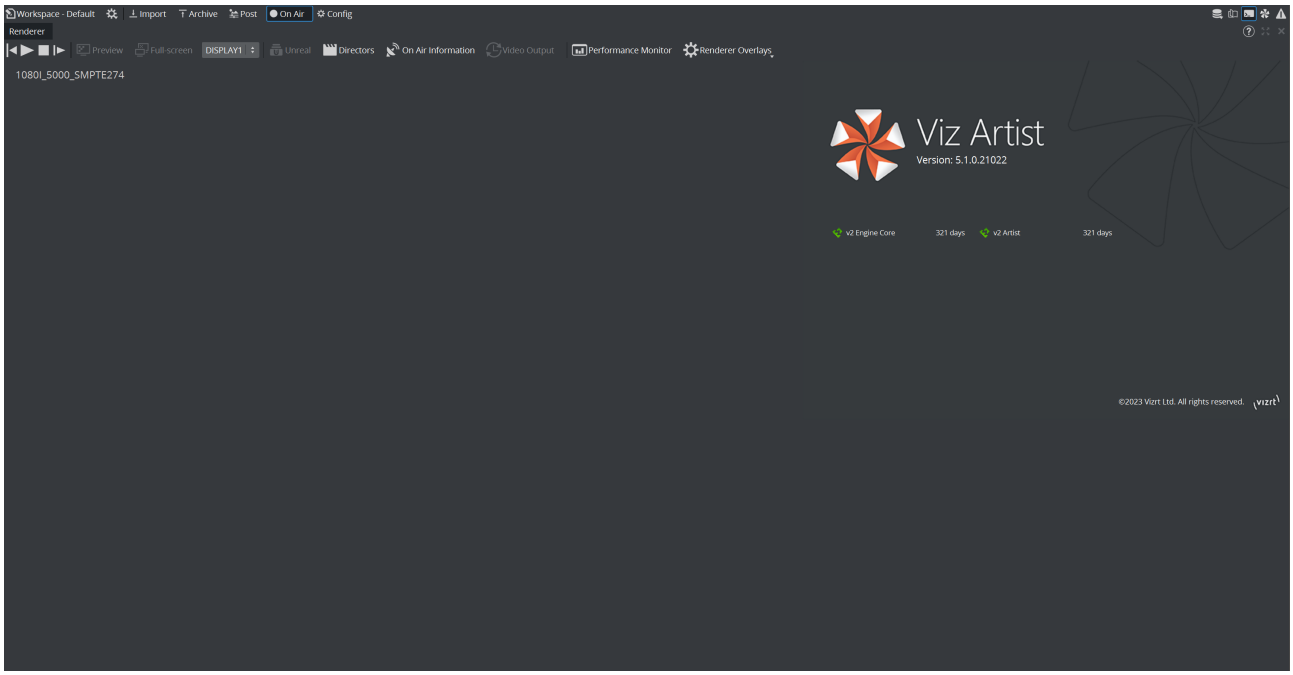
Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

9 On Air Mode

The On Air interface may vary, depending on the software and hardware configuration used. In Viz Artist, designers can click the On Air button on the main menu to switch Viz Artist from a modeling tool to a render engine. The application then waits for control commands; however, scene animations can also be rendered by the use of the Control Buttons (top-left corner).



The top left of the On Air screen shows a set of Control Buttons, as well as a Performance bar button. Depending on the software and hardware settings, additional buttons and information are available.

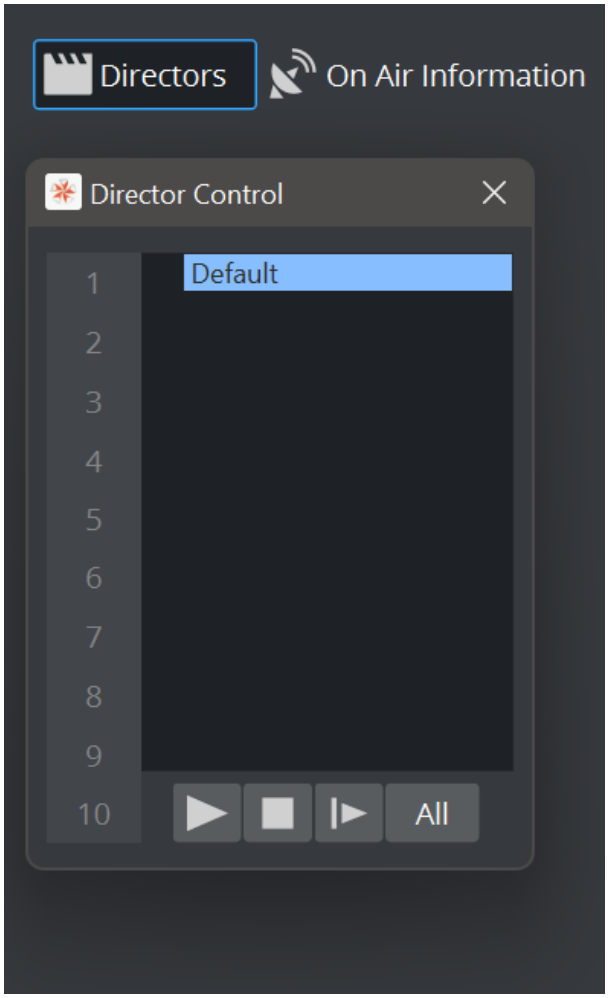
This section contains information on the following topics:

- Director Control Pane
- Control Buttons
- Performance
- On Air Information Panel
- License Information
- Tracking Timing Monitor
- Camera Stats Overlay

Copyright © 2026 Vizrt

9.1 Director Control Pane

When in On Air mode, the clapper board button opens the Director Control Panel window.



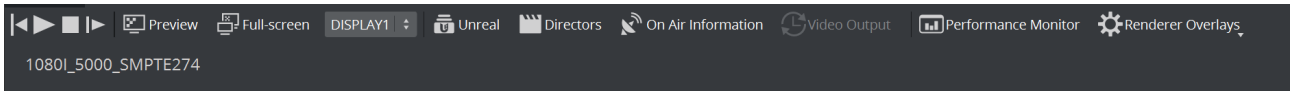
The Director Control Panel window can be used to select and animate one, multiple or all directors in the front, main or back layer. In addition, it can be used to set slots and to animate a combination of director(s).

See the **Director Control Panel** page in **The Stage for Animation** section of the Viz Artist User Guide for a detailed description of the Director Control Panel.

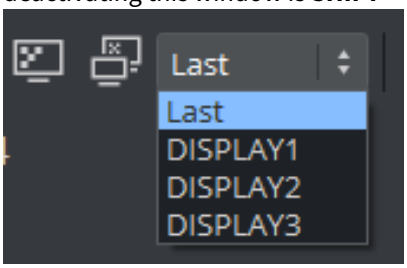
Copyright © 2026 Vizrt

9.2 Control Buttons

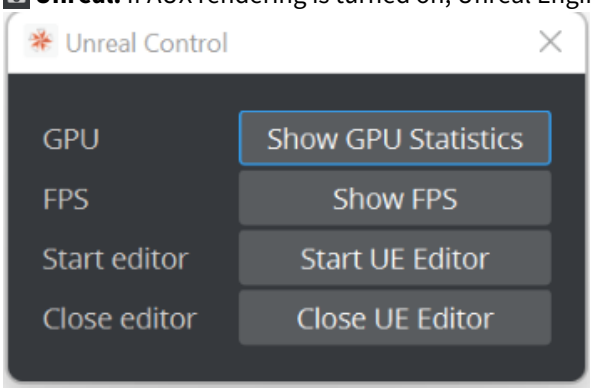
This section contains information on the Controls Buttons, which include the Play and the On Air buttons.



- **Back:** Jumps to the beginning of the animation in the scene.
- **Play:** Starts the animation of the scene.
- **Stop:** Stops the animation of the scene.
- **Continue:** Continues the animation after it stopped at a stop point.
- **Render Preview:** Shows or hides the VGA Preview window. The keyboard shortcut for activating or deactivating this window is **SHIFT + BACKSPACE**.

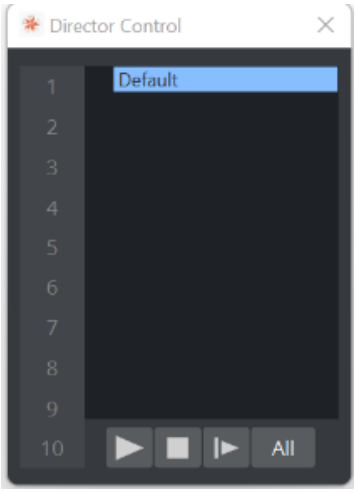
  

  The state of the VGA Preview window depends on how the **Render preview default setting** in the **Video Output** section of Viz Config is configured:
  - **Inactive:** Does not display by default.
  - **Active:** Displays by default.
  - **Fullscreen:** Sets the On Air window to screen size.
- **Unreal:** If AUX rendering is turned on, Unreal Engine Control can be enabled:

- **Directors:** Shows or hides the Director Control Pane window.

Copyright © 2026 Vizrt



- **On Air Information:** Shows or hides the On Air Information Panel window.
- **Video Output:** Shows the video sync settings.
- **Performance Monitor:** Shows or hides the Performance Bar.
- **Renderer Overlays:** Allows to enable on of the following overlays:
    - Performance Bar
    - Camera Stats
    - Center Shift Monitor
    - Audio Levels

Copyright © 2026 Vizrt

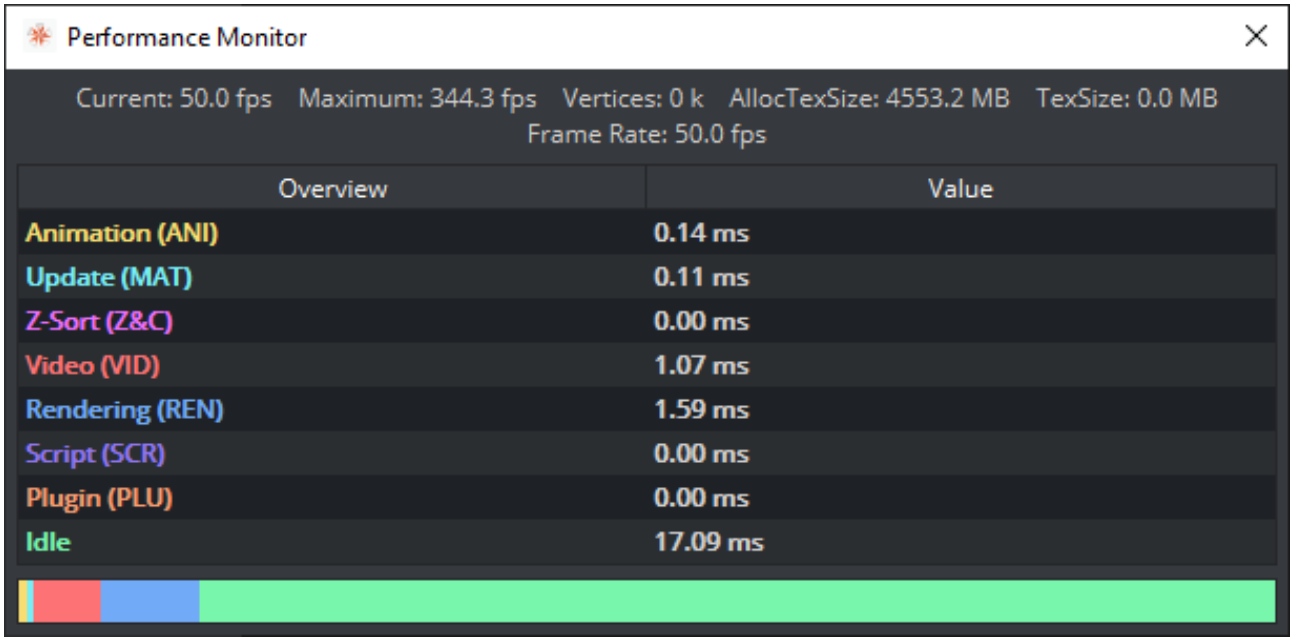
9.3 Performance

Analyzing the performance of Viz Artist/ Viz Engine can be done with:

- **Performance Monitor:** Closely monitors a range of parameters for analyzing real-time performance.
- **Performance Analyzer:** Monitors key performance and camera parameters, as a head-up display in the renderer window.
- **Performance Tracing:** Captures a detailed trace report over multiple frames.

9.3.1 Performance Monitor

To open the Performance monitor click on the Performance Monitor toolbar button.



The performance bar gives an idea of the current scene rendering performance (frames per second).

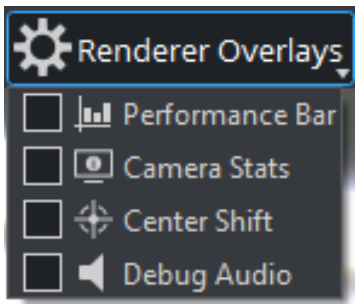
- **Current (CUR):** Shows how many frames per second the scene renders in On Air mode. The number should be above 50 (PAL) or 60 (NTSC), according to the rate that has been specified in the **Output Format** section of the Viz Engine Administrator Guide.
- **Maximum (MAX):** Shows how many frames per second the scene can render at without waiting for vertical retrace. The higher the maximum value, the more performance is left. If the maximum value is reduced to below `50` or `60`, the scene is not rendering in real-time.
- **Vertices (VER):** Shows the number of vectors in the scene.
- **AllocTexSize (TET):** Shows the total allocated size of texture memory.
- **TexSize (TEC):** Shows the size of the currently used texture memory.
- **Animation (ANI):** Shows how many microseconds all active directors and animation channels take. This indicator is linked to the yellow bar.

Copyright © 2026 Vizrt

- **Update (MAT):** Transforms each container in the scene into world coordinate space. This indicator is linked to the cyan bar.
- **Z-Sort (Z&C):** Refers to Z-sort and Culling, and sorts all containers for correct transparency drawing and determines if containers are visible in the current camera view. This indicator is linked to the pink bar.
- **Video (VID):** Shows how many microseconds video input (live video texture) and video output take. De-interlaced video inputs take longer time than progressive and interlaced. The only way to improve this value is to use a faster system. This indicator is linked to the red bar.
- **Rendering (REN):** Shows how many microseconds it takes to render all objects on the screen. A faster graphics card improves this value. This indicator is linked to the blue bar.
- **Script (SCR):** Shows the consumed time in microseconds from all active scripts. This indicator is linked to the dark green bar.
- **Plugin (PLU):** Indicates how much time in microseconds all active plug-ins spend in each render cycle. This indicator is linked to the orange bar.
- **Idle:** Shows available resources in microseconds the renderer has available. This indicator is linked to the light green bar.

9.3.2 Renderer Overlays

To select which performance data to show in the renderer window, click on the **Render Overlays** tool button then select the options to show:



- **Performance Bar:** Shows the a detailed Performance analyzer.
- **Camera Stats:** Enables the head-up display (HUD) showing the following parameters in the renderer view:
  - **Camera 1-n:** Shows the currently selected camera.
  - **Position:** Shows the camera's X, Y and Z position.
  - **Pan/Tilt/Twist:** Shows the camera's pan, tilt and twist parameters.
  - **FovX/FovY:** Shows the camera's field of view (FOV) for the horizontal (X) and vertical (Y) plane.
  - **Center Shift:** Shows the X and Y position of the camera's center shift.
- **Center Shift:** Shows the center shift marker as a cross hair in the renderer.
- **Debug Audio:** Turns on/off a visual representation of the audio mixing.

Performance Bar (Classic)

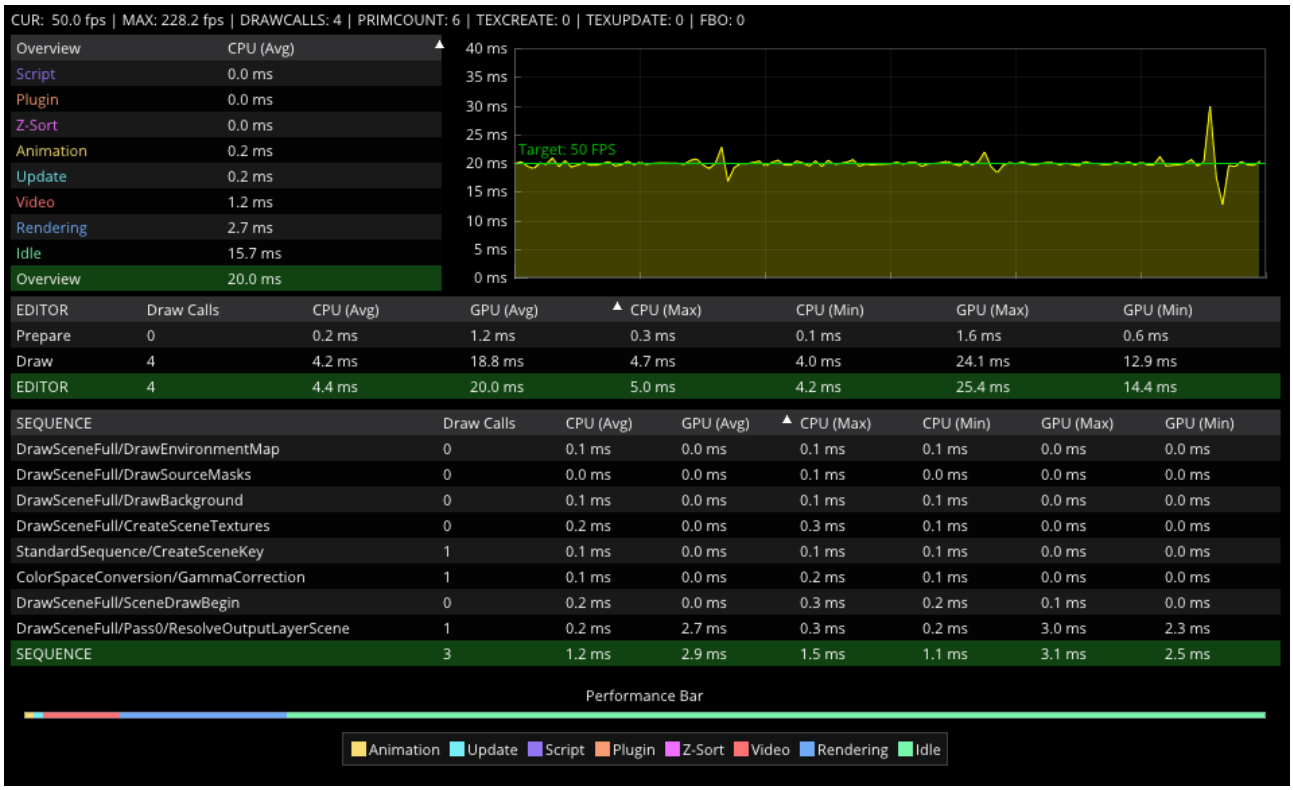


Copyright © 2026 Vizrt

The performance bar gives an idea of the current scene rendering performance (frames per second).

The values shown are the same as described in the Performance Monitor section above.

Performance Bar and Analyzer (Viz Engine Render Pipeline)



The Viz Engine Render Pipeline shows more detailed information about:

- Current and Maximum Framerate.
- Segmented render time and historic data.
- Timing information for all used Sequence pipelines.
- Performance Bar.

Performance Tracing

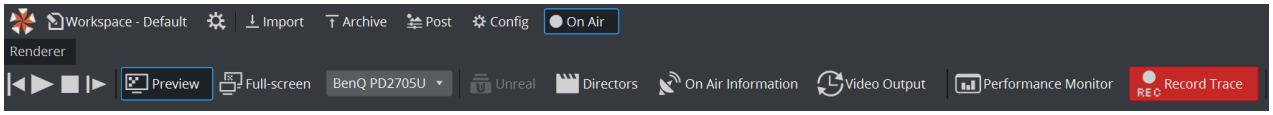
Performance Tracing records exactly what Viz Engine does in every frame, for example rendering, scripts, plug-ins and video. The recording is saved as a trace file (`.pftrace inside C:\ProgramData\vizrt\VizEngine\trace`) in the open Perfetto trace format.

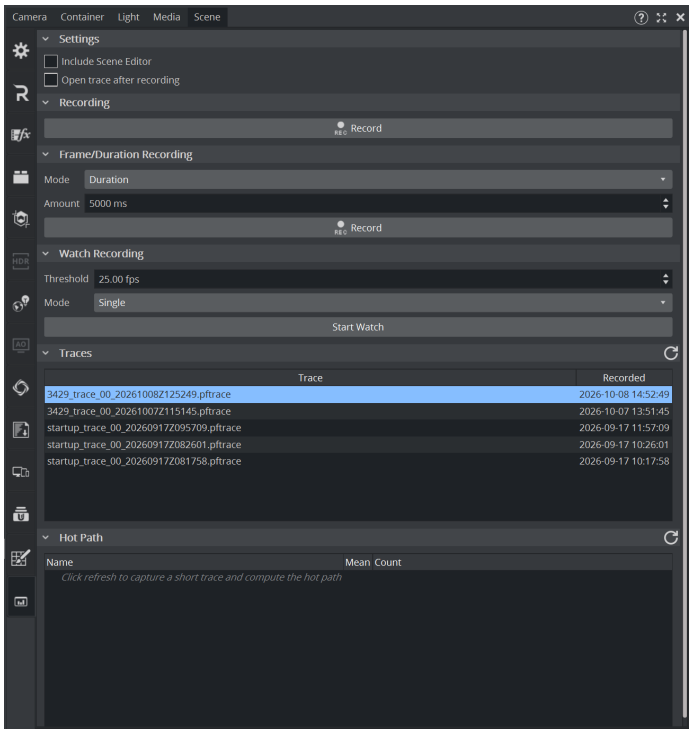
If you see performance problems such as dropped frames, stutter or a low frame rate, record a trace while the problem happens and attach the file to your support ticket.

Copyright © 2026 Vizrt

Where to Find Tracing

- **Record Trace toolbar button:** The record button in the On Air toolbar starts and stops a recording with one click. The button turns red while recording.
- **Performance Monitor > Tracing tab:** Open the Performance Monitor and select the **Tracing** tab. All tracing options are here.
- **Scene Settings > Performance:** The same tracing options are also in the Scene Settings panel (not available in Engine mode).





Recording a Trace

1. Prepare the system so the problem can happen, for example by loading the affected scene and taking it On Air.
2. Click **Record Trace** in the On Air toolbar, or **Record** in the Tracing tab.
3. Reproduce the problem. Keep the recording short: a few seconds that cover the problem are enough.
4. Click the button again to stop the recording. The trace file is saved automatically.
5. Open **Performance Monitor > Tracing** and find the new trace at the top of the **Traces** list.
6. Drag the trace from the list straight into your support ticket, an email or a folder.

Tip: *If the problem is hard to reproduce on demand, use* ***Watch Recording*** *(see below). It waits for the frame rate to drop and saves a trace for you automatically.*

Copyright © 2026 Vizrt

Recording Options

Settings

- **Include Scene Editor:** Also records what the scene editor does, not only the renderer. Turn this on if the problem happens while you work in Viz Artist rather than On Air.
- **Open trace after recording:** Opens each finished recording in the trace viewer straight away. Off by default: new traces only appear in the Traces list.

Recording

- **Record / Stop:** Starts a recording and stops it when you click again. This is the same as the Record Trace toolbar button.

Frame/Duration Recording

- **Mode:** Records either for a set **Duration** (milliseconds) or for a set number of **Frames**.
- **Amount:** The duration (default 5000 ms) or the number of frames (default 300).
- **Record:** Starts the recording. It stops by itself when the duration or frame count is reached.

Watch Recording

Watch Recording runs in the background and saves a trace only when the frame rate falls below a threshold. Use it for problems that happen only now and then.

- **Threshold:** The frame rate (fps) that triggers a trace. Set it to your output rate (for example 50 or 59.94), or slightly below it.
- **Mode:**
    - **Single:** Saves one trace at the first frame-rate drop, then stops watching.
    - **Continuous:** Keeps watching and saves a trace at every drop.
- **Start Watch / Stop Watch:** Starts or stops watching.

Note: *Only one kind of recording can run at a time. While a recording or watch is running, the other record buttons are unavailable.*

Traces

The Traces list shows all recorded traces, newest first.

- **Trace:** The file name. Hover over it to see the full path on disk.
- **Recorded:** When the trace was recorded, in local time. The time in the file name is UTC.
- **Double-click:** Opens the trace in the trace viewer.
- **Drag and drop:** Drag a trace into a ticket, an email or a folder to copy the file.
- **Refresh:** Reloads the list.

Hot Path

Hot Path gives a quick view of where time is spent right now, without opening a trace file.

- **Refresh:** Records a short 3-second trace and shows it as a call tree. The time of that capture is shown next to the button.
- The most expensive chain of calls (the hot path) is expanded and highlighted.
- **Name:** The track or function.
- **Mean:** Average time per call.

Copyright © 2026 Vizrt

- **Count:** Number of calls during the capture.
- Hover over an entry to see its total time and self time, as a time value and as a share of the track.

Copyright © 2026 Vizrt

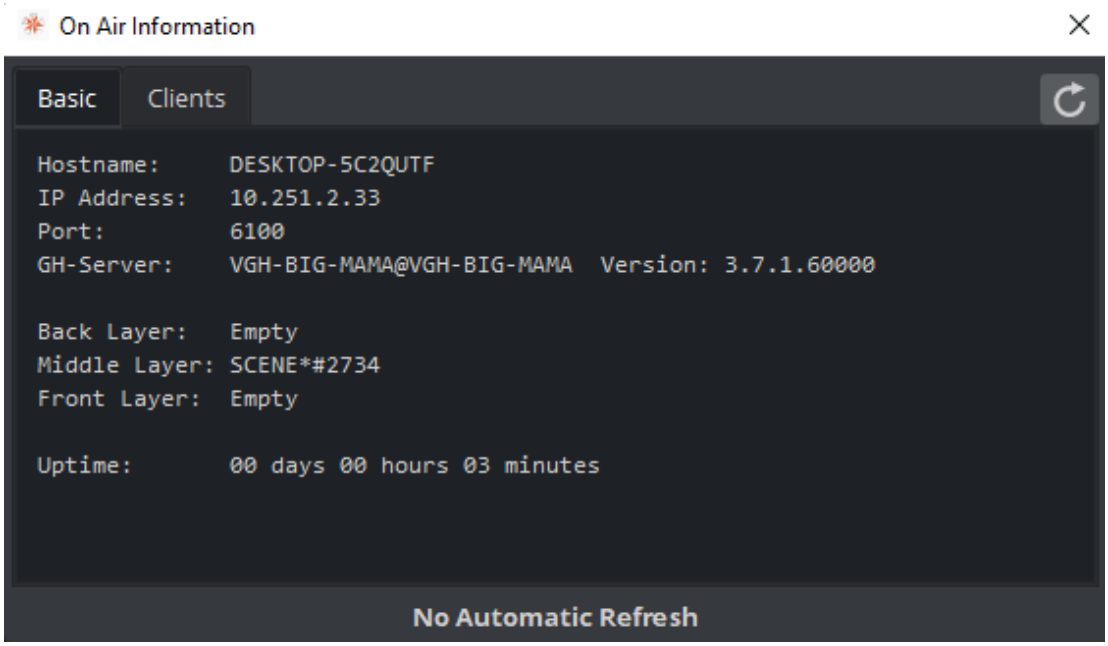
9.4 On Air Information Panel

The On Air Information panel shows the required parameters to send external control commands and all connected clients, with the IP address, host name and Viz Port.

> **Note:** Polling for On Air information can decrease the performance. For information on how to adjust the On Air Update Interval, see the User Interface page of Viz Configuration.

9.4.1 Basic Tab

In the Basic tab, the parameters required to send external control commands are shown:



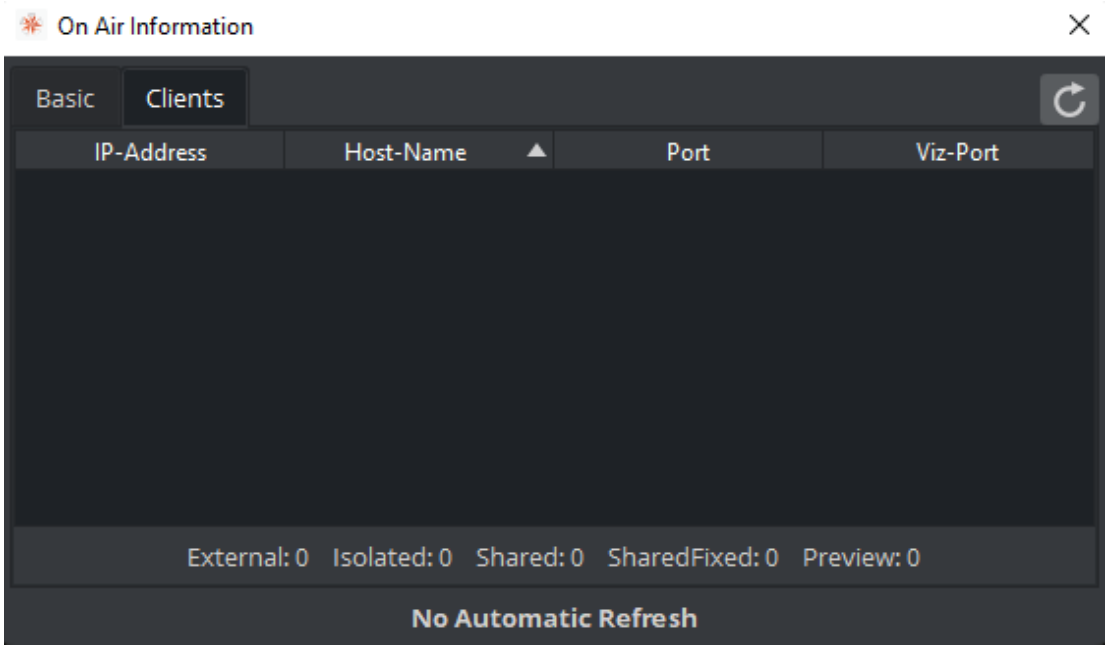
- **Refresh button:** Refreshes the status information.
- **Hostname:** Shows the name external control programs can use to communicate with Viz Artist.
- **IP Address:** Shows the IP address external control commands can communicate with Viz Artist.
- **Port:** Shows the port Viz Artist is using. Default port is `6100`, but may be changed in the **Communication** section of Viz Configuration.
- **GH-Server:** Shows the Graphic Hub Manager server Viz Artist is connected to.
- **Back Layer:** Shows the name of the scene that is defined to run in the background of the middle and front layer scene(s).
- **Middle Layer:** Shows the name of the scene that is defined to run in the middle between the back and front layer scene(s).
- **Front Layer:** Shows the name of the scene that is defined to run in the foreground of the back and middle layer scene(s).

Copyright © 2026 Vizrt

- **Uptime:** Shows the time elapsed since Viz was started.

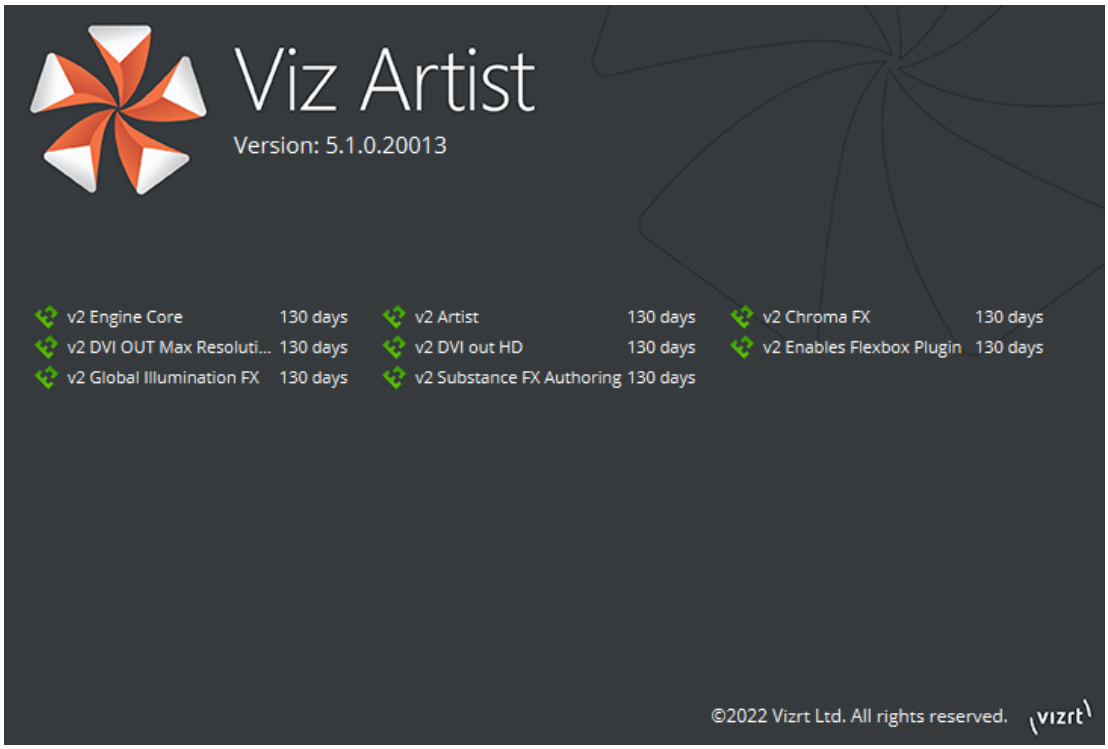
9.4.2 Clients Tab

In the Clients tab, all connected clients are shown with the IP address, host name and Viz Port.



Copyright © 2026 Vizrt

9.5 License Information



The license information listing shows licensed features and how many days a given license has left before it must be renewed.

9.6 Tracking Timing Monitor

9.6.1 What is it?

When you use a tracked camera, the tracking system keeps sending camera position data to Viz Engine. If that data arrives late, too early or not at all, virtual graphics can shake, jump or drift against the real camera image.

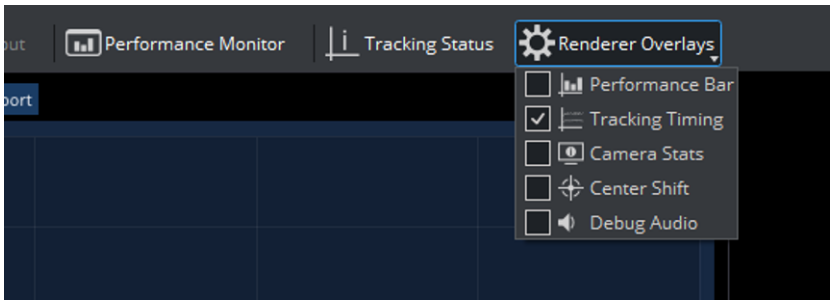
The **Tracking Timing Monitor** shows you whether tracking data is arriving smoothly. You can use it to answer questions like:

- Is my tracking system sending data steadily?
- When did a problem happen, and how long did it last?
- Which camera is affected?

9.6.2 Opening the monitor

In Viz Artist, go to the **On-Air** toolbar:

- **Tracking Status** button: opens the **Camera Status** window, which shows all cameras at a glance.
- **Overlays > Tracking Timing**: shows the timing graph on top of the render preview. This option is only available while the preview is enabled.



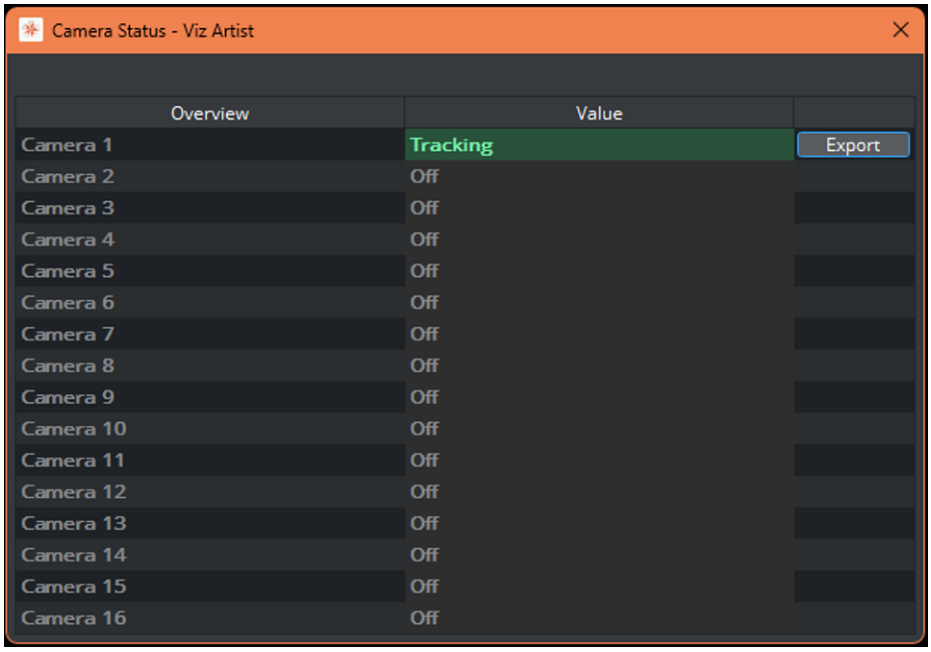
9.6.3 Camera Status window

The window has one row for each camera (up to 16). The color tells you the state of each camera:

Status	Color	Meaning
Tracking	Green	Data is arriving normally.
Underrun	Orange	Data was missing when the engine needed it.
Overrun	Orange	Data is arriving faster than the engine can use it.
No Data	Red	No tracking data is arriving at all.

Copyright © 2026 Vizrt

Status	Color	Meaning
Off	Grey	Tracking isn't active for this camera.



9.6.4 Reading the timing graph

The overlay has two views. You switch between them with the **Live** and **Blocks** buttons.

Live view: the last minute

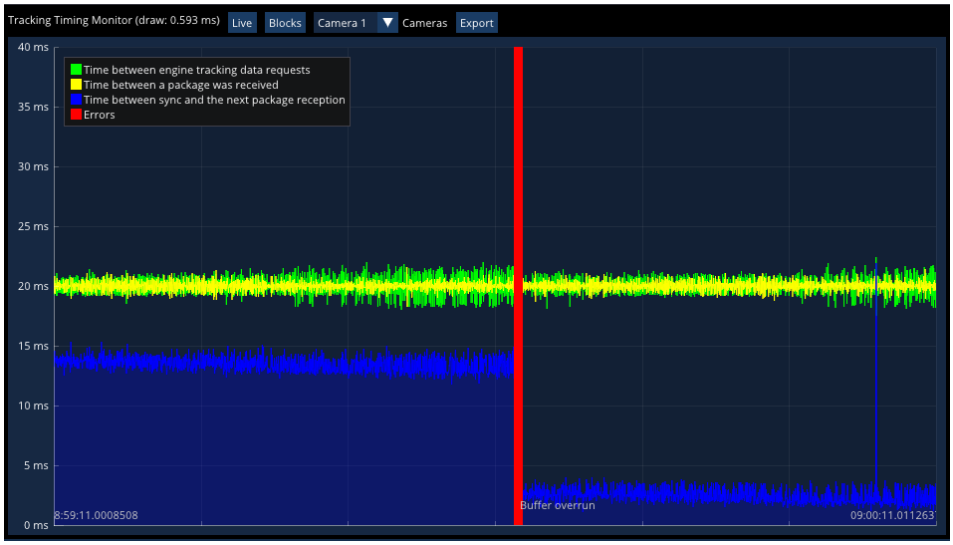
The Live view shows three lines:

- **Green:** how often the engine asks for tracking data.
- **Yellow:** how often tracking data actually arrives.
- **Blue area:** how long the engine waits for data after each frame starts.

What good looks like: flat, steady lines.
What a problem looks like: spikes, gaps, or a line that keeps slowly rising or falling.

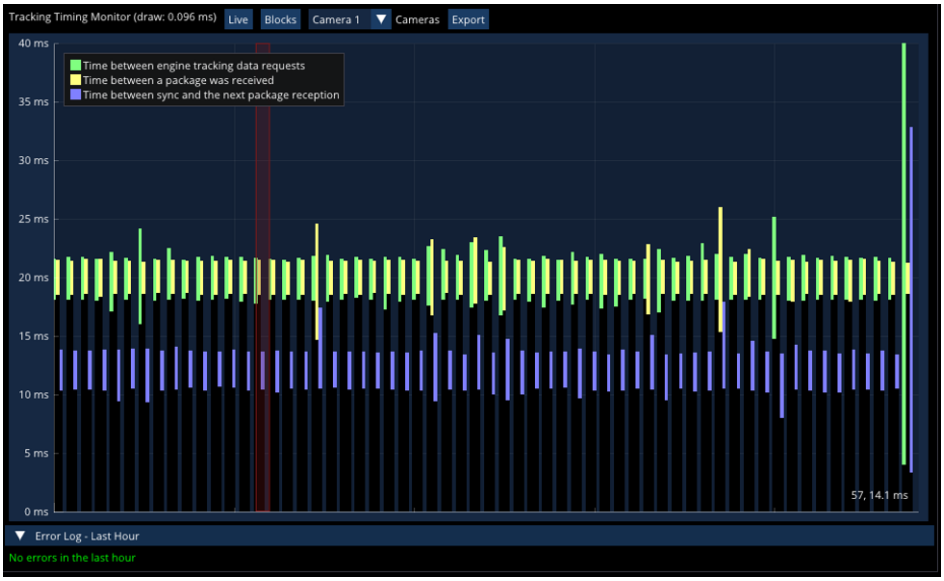
Problems show up as **red markers** along the bottom of the graph. When a problem has just happened, a **blinking warning** appears at the top.

Copyright © 2026 Vizrt



Blocks view: the last hour

The Blocks view shows one bar for each minute of the last hour. Tall or uneven bars mean timing was unstable during that minute. **Click a bar** to see that minute in detail in the Live view.



Error Log

The Error Log lists the problems of the last hour: when each one started and ended, what kind of problem it was, and how long it lasted. **Click an entry** to jump to that moment in the Live view.

Copyright © 2026 Vizrt



9.6.5 Saving the data for support

If you need to report a tracking problem, click **Export** next to the affected camera in the Camera Status window and choose where to save the file. This saves the camera's timing history as a CSV file, which you can open in Excel or send to your tracking vendor or to Vizrt Support.

You can also start the export from the overlay itself.

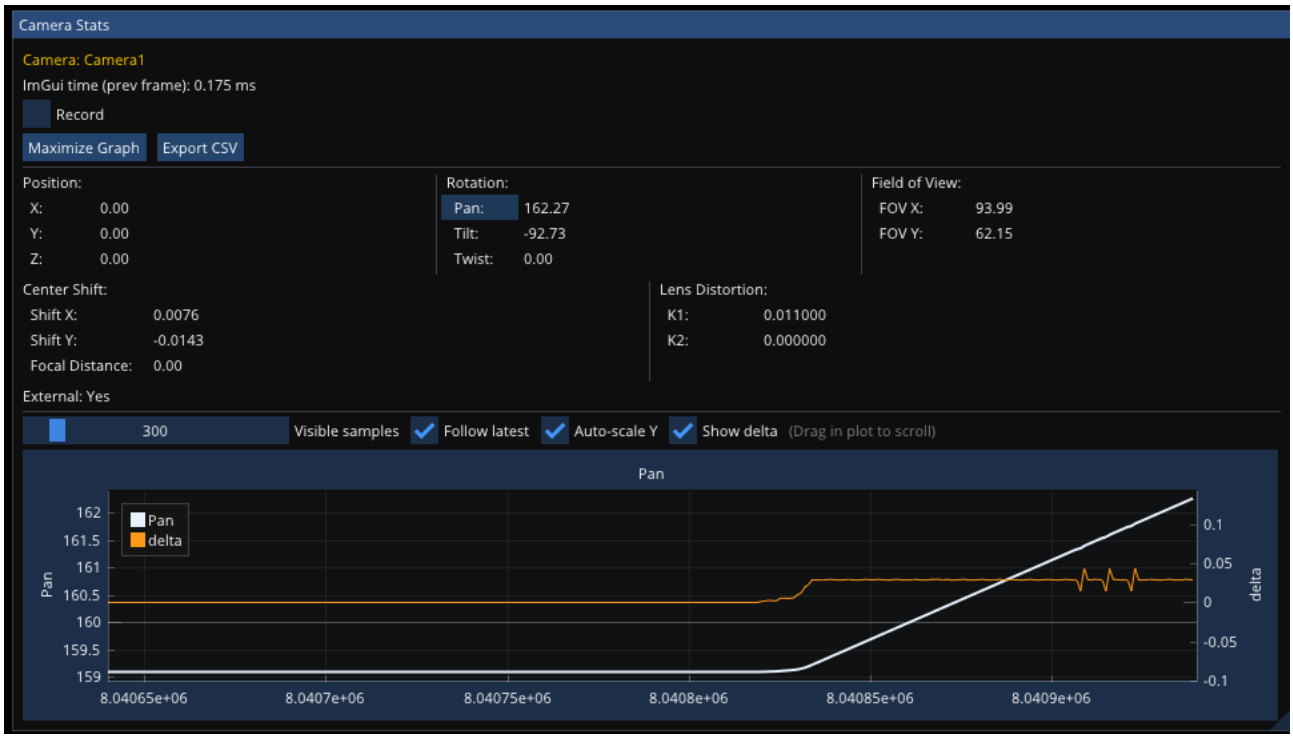
9.6.6 Tips

- **Red "No Data":** check the cable or network connection and the tracking system's output settings.
- **Repeated orange warnings:** the tracking system and Viz Engine may not be running in sync. Check the genlock and frame-rate settings on both sides.
- **Slowly rising or falling line:** this often means a clock drift between the tracking system and the engine.

Copyright © 2026 Vizrt

9.7 Camera Stats Overlay

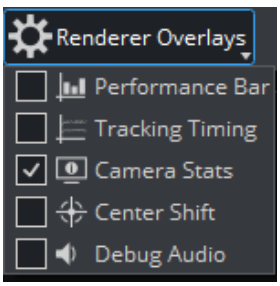
9.7.1 1. What it is for



The Camera Stats overlay lets you watch the live values of the current camera in the Render Preview and follow how they change over time. Use it to check camera position, rotation, field of view, optical centre shift, focal distance and lens-distortion values, to compare a few of them side by side, and to save a recording for later analysis in a spreadsheet.

It appears only in the Render Preview and does not affect the captured or on-air output.

9.7.2 2. Quick start



- Turn on “Camera Stats” in the Renderer Overlays menu (see section 3).
- Click the value(s) you want to watch -- up to three (see section 5).
- Switch on “Record” if you want to keep a long history (see section 7).
- Click “Export CSV” to save the recording (see section 8).

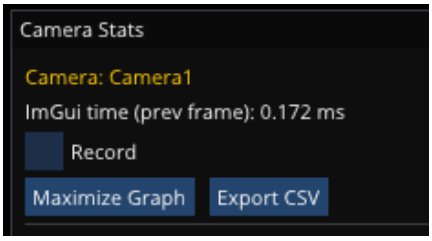
9.7.3 3. Turning the overlay on

Open the Renderer Overlays menu -- the settings (gear) button in the Render Preview toolbar -- and tick Camera Stats. The Camera Stats window then appears in the preview. Untick the entry to hide it again.

The same menu also contains the other preview overlays (Renderer Performance, Tracking Timing, Center Shift Marker, Audio); you can turn Camera Stats on or off independently of those.

9.7.4 4. The window at a glance

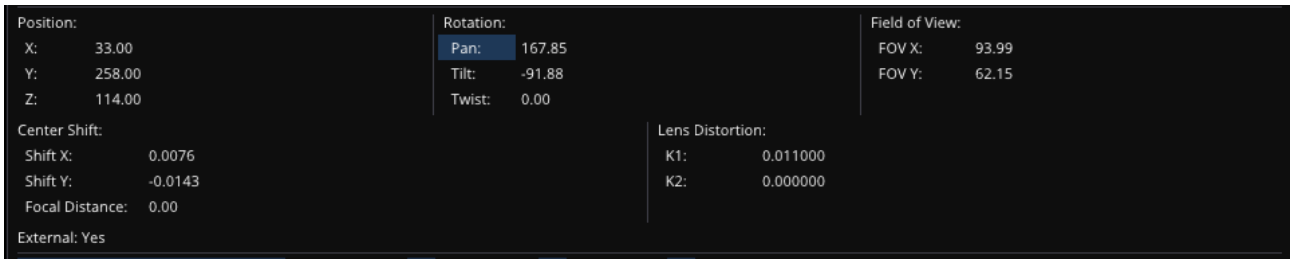
The Camera Stats window opens on the left of the preview. At the top is a row of buttons, below it the current camera values, and at the bottom the graphs.



Button	What it does
Record	Keeps a long history so you can scroll back and export it. See section 7.
Maximize graph / Restore	Hides the value list so the graphs fill the whole window. Restore brings the values back.
Export CSV	Saves the recording to a file. See section 8.

Copyright © 2026 Vizrt

9.7.5 5. Choosing which values to plot



The current camera values are listed in groups. Click a value to add it to the graphs; click it again to remove it. You can plot up to three values at the same time -- once three are selected, clicking another has no effect until you remove one. The value list is hidden while the graph is maximized.

Group	Values	Shows
Position	X, Y, Z	where the camera is in the scene
Rotation	Pan, Tilt, Twist	which way the camera is pointing
Field of View	FOV X, FOV Y	horizontal and vertical field of view
Center Shift	Shift X, Shift Y	optical centre shift of the lens
Focal Distance	Focal Dist	the camera's focus distance
Lens Distortion	K1, K2, Focus, Zoom	lens distortion values (listed only while a lens is active)

9.7.6 6. Working with the graphs

Each value you pick gets its own graph, stacked one below the other. All graphs share the same time axis so they line up for easy comparison, and each one scales its height to its own values. When a value changes, its line updates live.

Use the controls above the graphs to change what you see:



Copyright © 2026 Vizrt

Control	What it does
Visible samples	Sets how much history is shown at once -- drag left to zoom in, right to zoom out.
Follow latest	Keeps the newest data in view. It turns off by itself as soon as you scroll by hand.
Drag in a graph	Drag left or right inside a graph to scroll back and forth through the history.
Auto-scale Y	Fits each graph's height to the values on screen. Turn it off to set the range yourself with the slider that appears.
Show delta	Adds a second, orange line showing how much the value changes from one moment to the next, on its own scale.

9.7.7 7. Recording the history

The graphs always run live, showing the most recent activity. The Record button controls how far back the history reaches:

Record	History kept	When to use
Off (default)	the last couple of minutes	Quick glance at what the camera is doing right now.
On	a long recording	Capturing a take you want to review or export afterwards. A red REC marker is shown while recording.

Turn Record on before the take you want to keep. If you turn it off again, only the recent live view is kept -- the longer recording is not retained.

9.7.8 8. Saving a recording (CSV)

Click Export CSV to save the recording to a file. A short message shows where the file was written; it disappears after a few seconds. The file has one row per recorded moment and opens directly in Excel, Google Sheets or any spreadsheet. It always uses a dot as the decimal separator, so it opens correctly whatever your regional settings are.

The file contains these columns, in order:

Time_ms, PosX, PosY, PosZ, Pan, Tilt, Twist, FovX, FovY,

CenterShiftX, CenterShiftY, FocalDistance,

LensK1, LensK2, LensFocus, LensZoom

The first column is the time in milliseconds; the rest are the recorded values. The lens columns are filled only for moments when a lens was active.

Copyright © 2026 Vizrt

10 Video IO Configuration and Features

These chapters list some common Video IO related configurations which mainly apply to Matrox boards unless stated differently.

This section contains information on the following topics:

- Basic IO Configuration
- High Dynamic Range
- Mixed Mode Video Support
- Frame Accurate Output
- Shared Usage of Input Channels
- Dynamic Channel Allocation
- Matrox Supported Codecs
- DVE Performance
- Watchdog
- NDI Output from GFX Channel
- Clip Player Pro Features
- Extraction of VANC Data
- Multiple SDI/IP Outputs
- Clipout Channel Usage

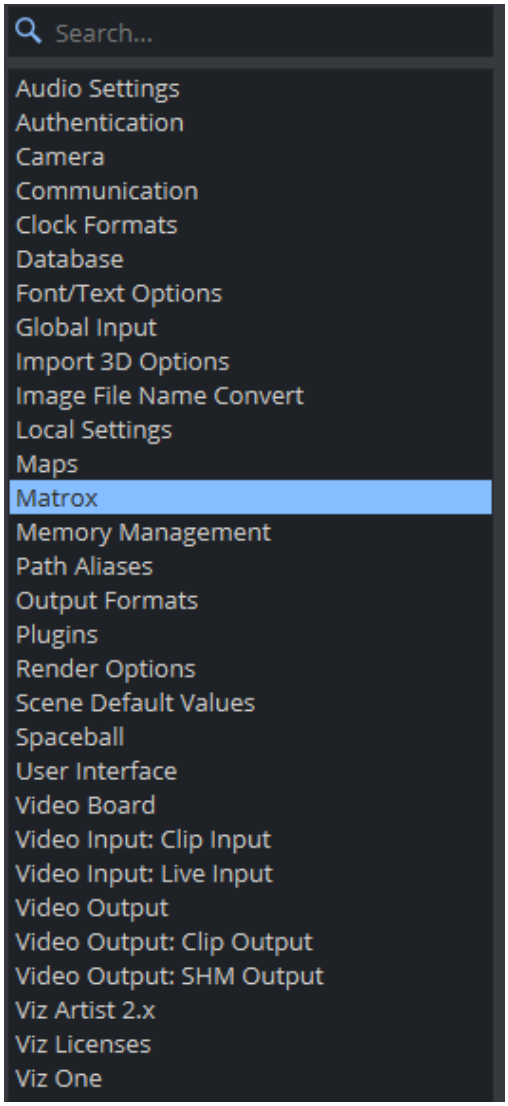
Copyright © 2026 Vizrt

10.1 Basic IO Configuration

A new way to configure and map your input sources to Viz Engine Live Sources, allowing proper migration from existing workflows to new standards like 2110 or NDI.

10.1.1 General Information

The section for enabling/disabling inputs was removed completely, and all other Video Input sections have been concentrated in the two sections for Clip Input and Live Input. Configuration for Output can still be found in the Matrox section.

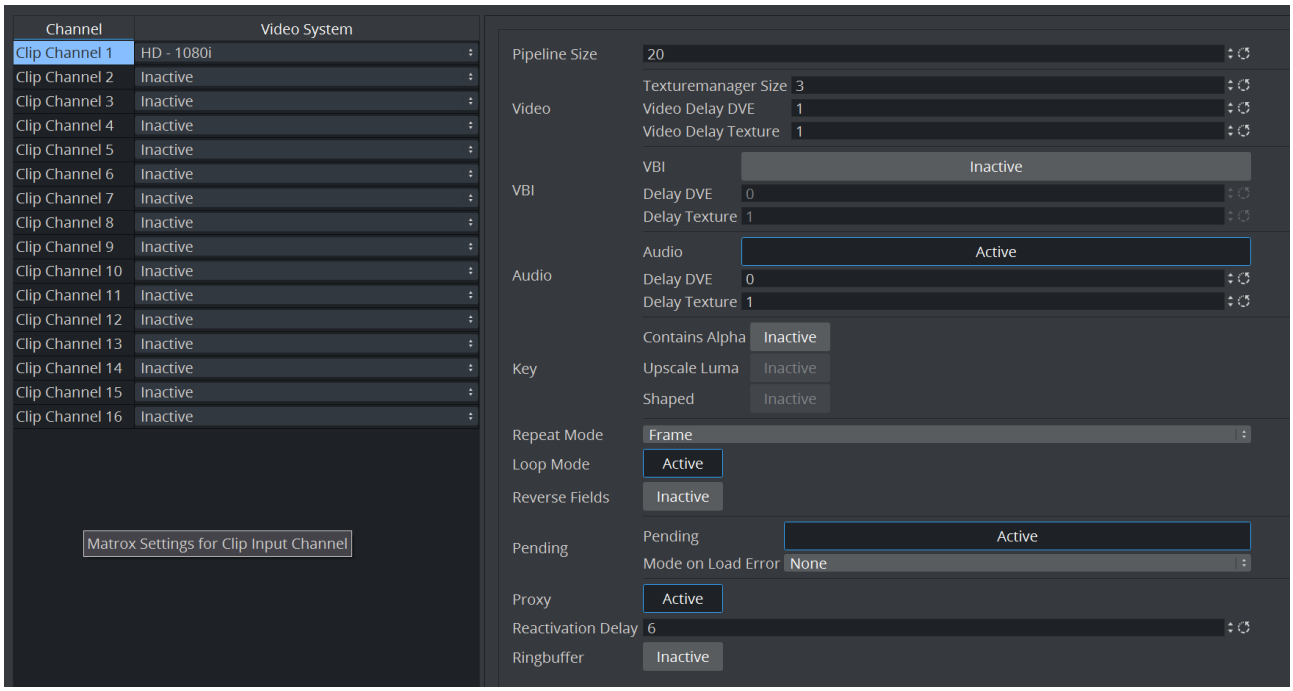


Copyright © 2026 Vizrt

10.1.2 Clip Input Channels

The only change for configuring the Clip Input channels is the location to enable single channels, which can be done now directly in the Video Input: Clip Input section of Viz Config. The Video System can either be set to the desired resolution family or to the new system *Auto*, which configures the system equivalent to the selected Output Format. Explicit selection of anything other than *Auto* is only necessary in mixed setups (for example, 1080p clip channel in a 1080i environment) or for video wall and full screen setups, where the resolution doesn't match standard broadcast formats.

> **Information:** The fallback resolution is 1080i.



10.1.3 Live Input Channels

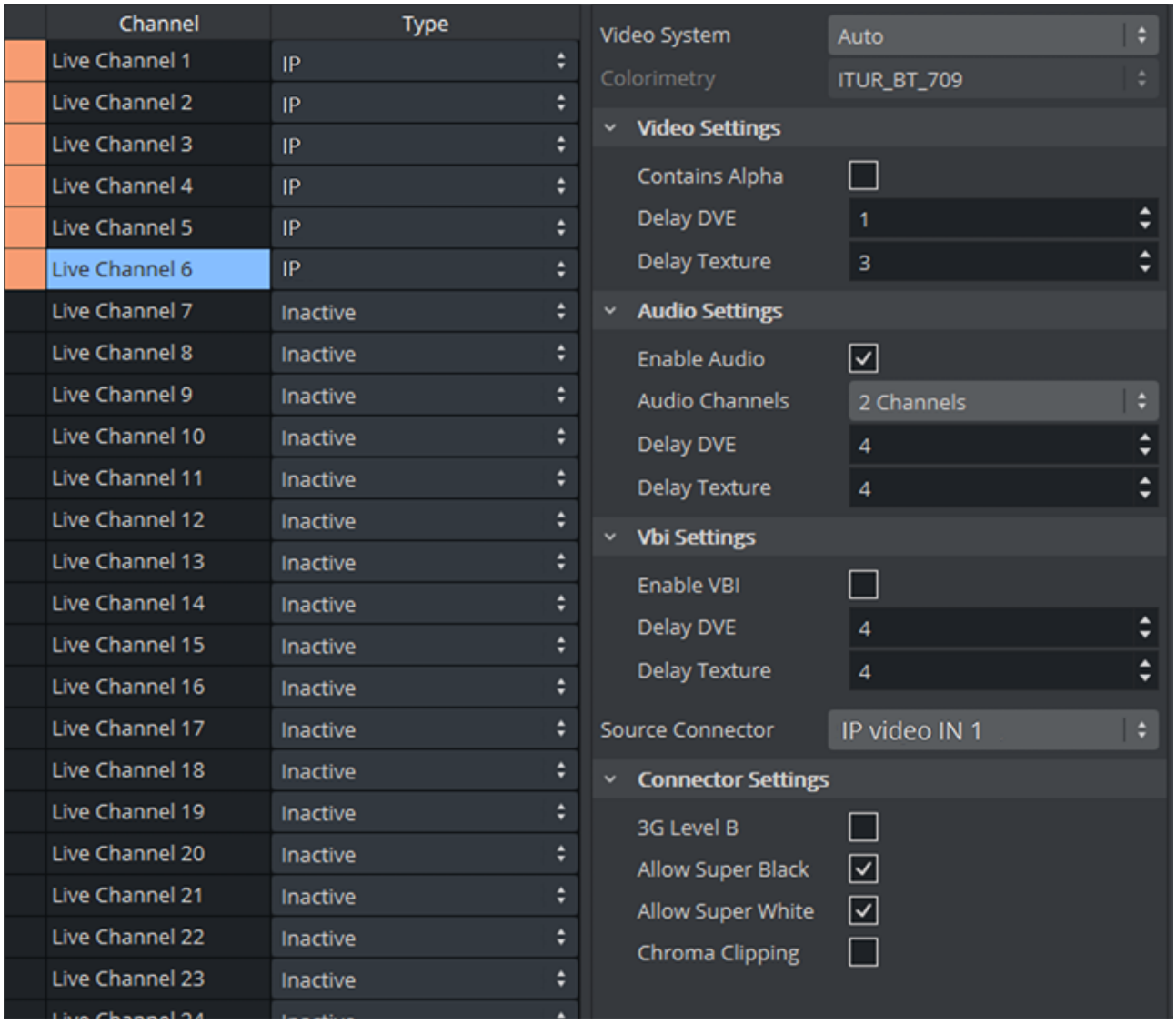
This section now includes all inputs (except clip channels) and allows a mix of different resources, like SDI or IP inputs, and various streaming and shared memory sources. With the correct mapping, it is possible to use the same scene containing Live Inputs on both SDI and IP systems, without the need to adapt the scene for IP (previously StreamIn) input channels.

Just like Clip Input Channels, the Live Input Channels can be set to use the Video System *Auto*. This configures the input resolution based on the Output System. Explicit selection of a resolution is only necessary for video wall or full screen setups, if the resolution doesn't match any broadcast standard.

> **Information:** The fallback resolution is 1080i.

Map To VizChannel was replaced by the setting *Source Connector*. For SDI and IP channel types the physical connector, which should be used as input on the video board, needs to be selected.

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

10.2 High Dynamic Range

High Dynamic Range (HDR), often in combination with wide color gamut (WCG), is a way to improve the colors shown on screens. This page explains which environments support HDR, and which Viz Engine configuration settings are required.

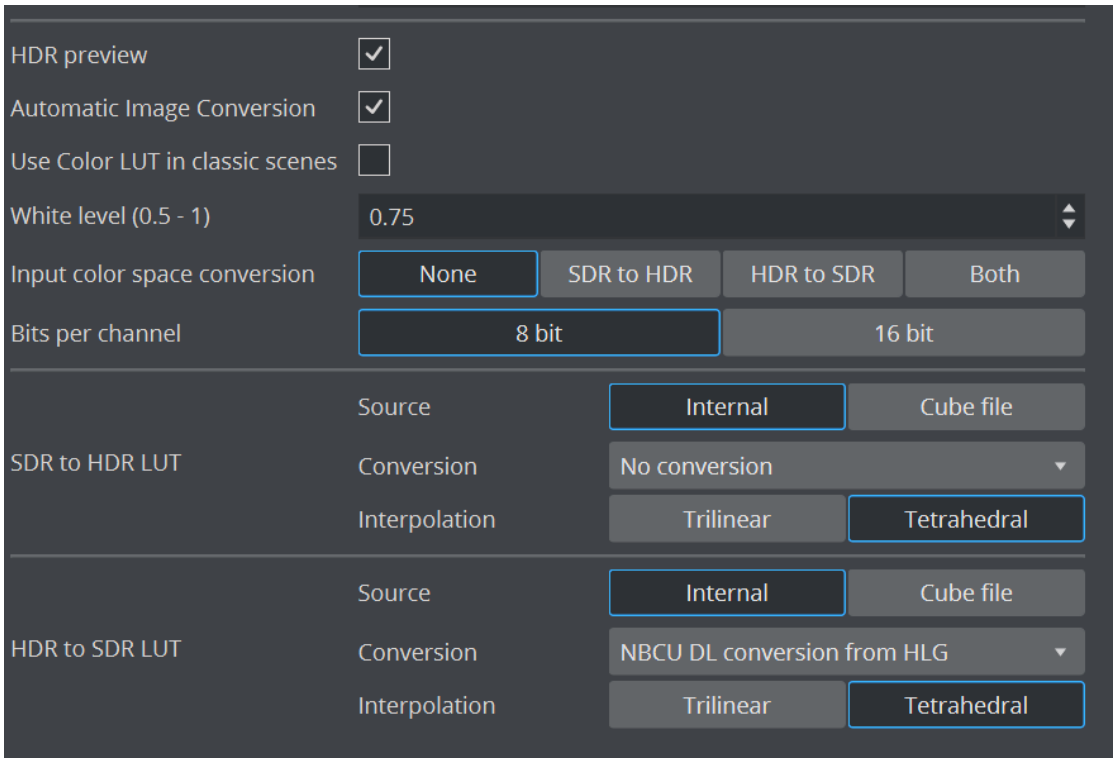
> **Note:** Most settings and functionalities described, assume that HLG is used as an HDR format. Check the support for S-Log3.

HDR is supported with Matrox SDI or IP boards, and with software I/O mode NDI. The HDR preview on desktop monitors can be used with VGA engine versions as well. Please refer to the Viz Artist User Guide for additional information on HDR previews, and an overview of HDR use cases.

10.2.1 How to Configure HDR

HDR affects multiple aspects of Viz Engine configuration. Important settings can be found in the Render Options, Video Output, Video Input, and Clip Input.

Render Options



Copyright © 2026 Vizrt

General Settings

- **HDR preview:** Enables the OS support for HDR preview windows.
- **Automatic Image Conversion:** Viz Engine tries to adjust the color space for any image source.
- **Use Color LUT in classic scenes:** Enable this setting if the engine output is HDR, and classic scenes are being used. Viz Engine pipeline automatically handles this.
- **White Level:** Used to scale between the linear scene color space, and the output HLG signal. The scene light value 1.0 is scaled to the white level on the HLG output. The default value is 75%. 100% white selected in sRGB in the color picker, is scaled to the same value.
- **Input color space conversion:** If enabled, Live and Clip input textures are converted between SDR and HDR. The renderer then picks the appropriate version of the texture, depending on where it is used. It is recommended to use **Both**, for any setup with HDR.
- **Bits per channel:** Using 16 bits per channel avoids banding.

Color Conversion LUTs

Viz Engine uses lookup tables (LUTs) to convert between SDR and HDR. Some LUTs are built-in, while LUTs from third parties can be imported from cube files. For most workflows two LUTs are required, one to convert from SDR to HDR, and another to convert from HDR to SDR. Using a matching pair of LUTs reduces round trip conversion losses.

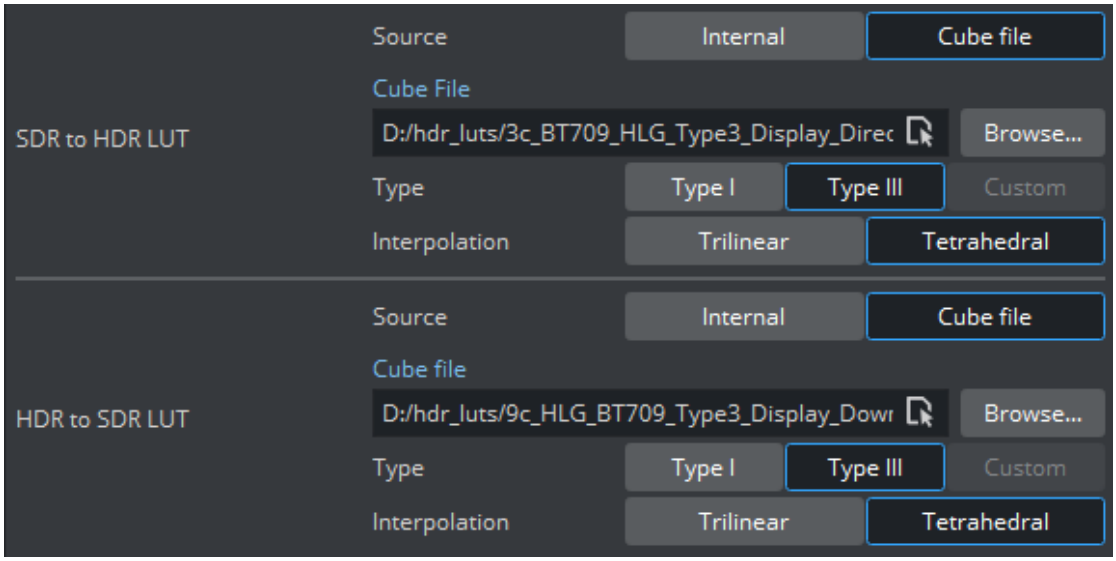
Interpolation: Lookup tables contain the converted colors for a relatively small set of input colors. For other colors, the output is interpolated. Trilinear and tetrahedral interpolation are two different mathematical approaches. Tetrahedral interpolation is generally recommended, especially for HDR to SDR conversions.

Built-in LUTs

There are three built-in LUTs: An SDR to HDR conversion based on ITU BT.2048-8, and a pair of LUTs developed by NBCUniversal Media, LLC. All three encode display light conversions between SDR and HLG, and support extended video range.

Copyright © 2026 Vizrt

External LUTs



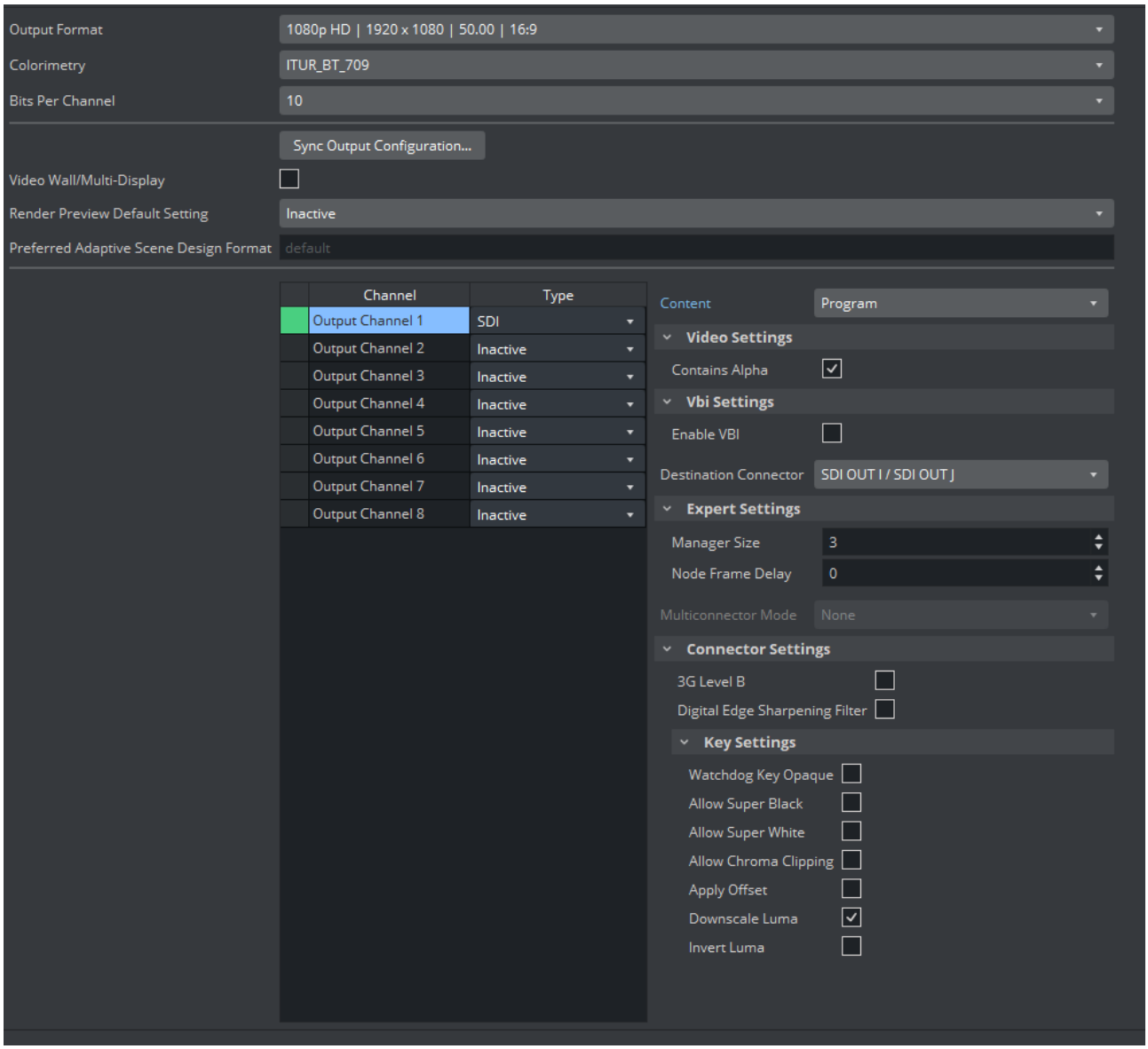
HDR conversion LUTs are available from several third parties, notably the BBC and NBCU. To import an external LUT, select **Cube file** as the **Source**, and set the file path.

The BBC established a nomenclature of different LUT types, depending on the processing range. Viz Engine supports Type I LUTs (which are limited to nominal range), and Type III LUTs (which support extended range). Configure the type that matches the used cube file.

Copyright © 2026 Vizrt

Video Configuration

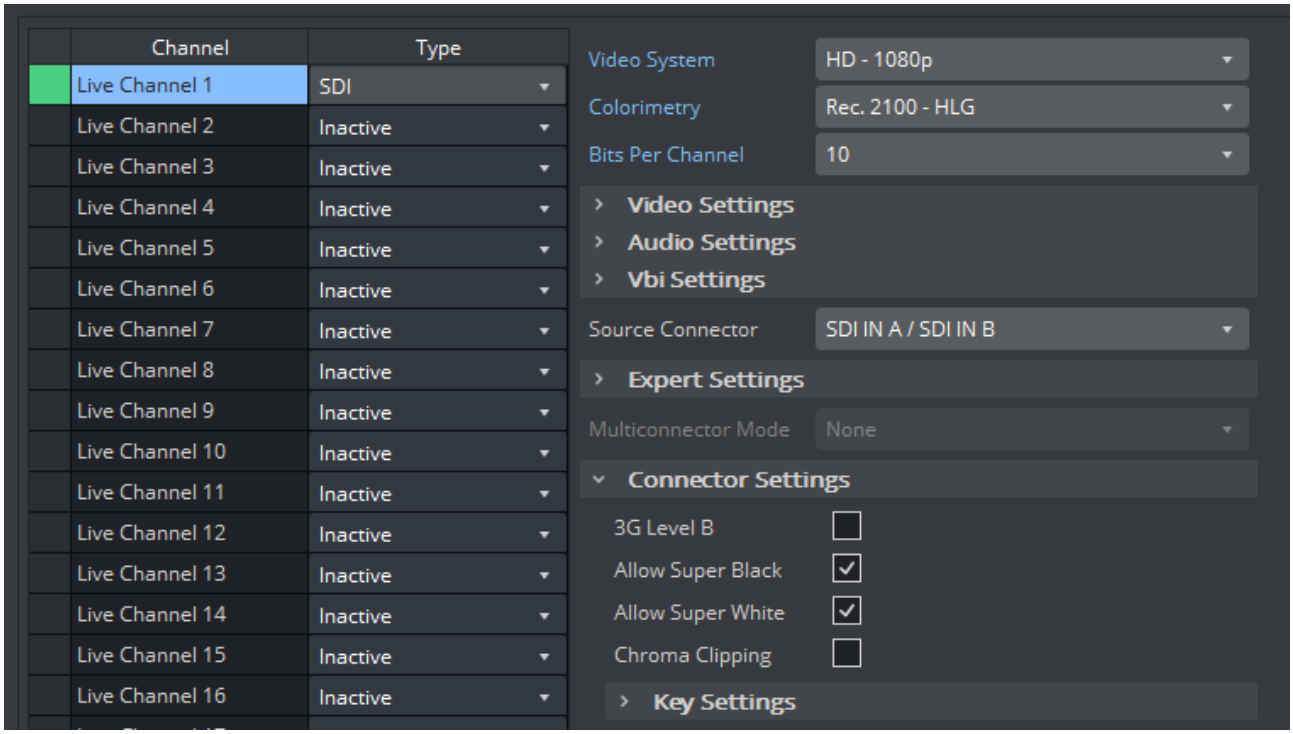
Video Output



- HDR is supported in 1080p or higher.
- Set the output colorimetry to HLG or S-Log3.
- Use 10 bits per channel to avoid banding.
- Enable **Allow Super Black** and **Super White**, to enable extended range video signals.

Copyright © 2026 Vizrt

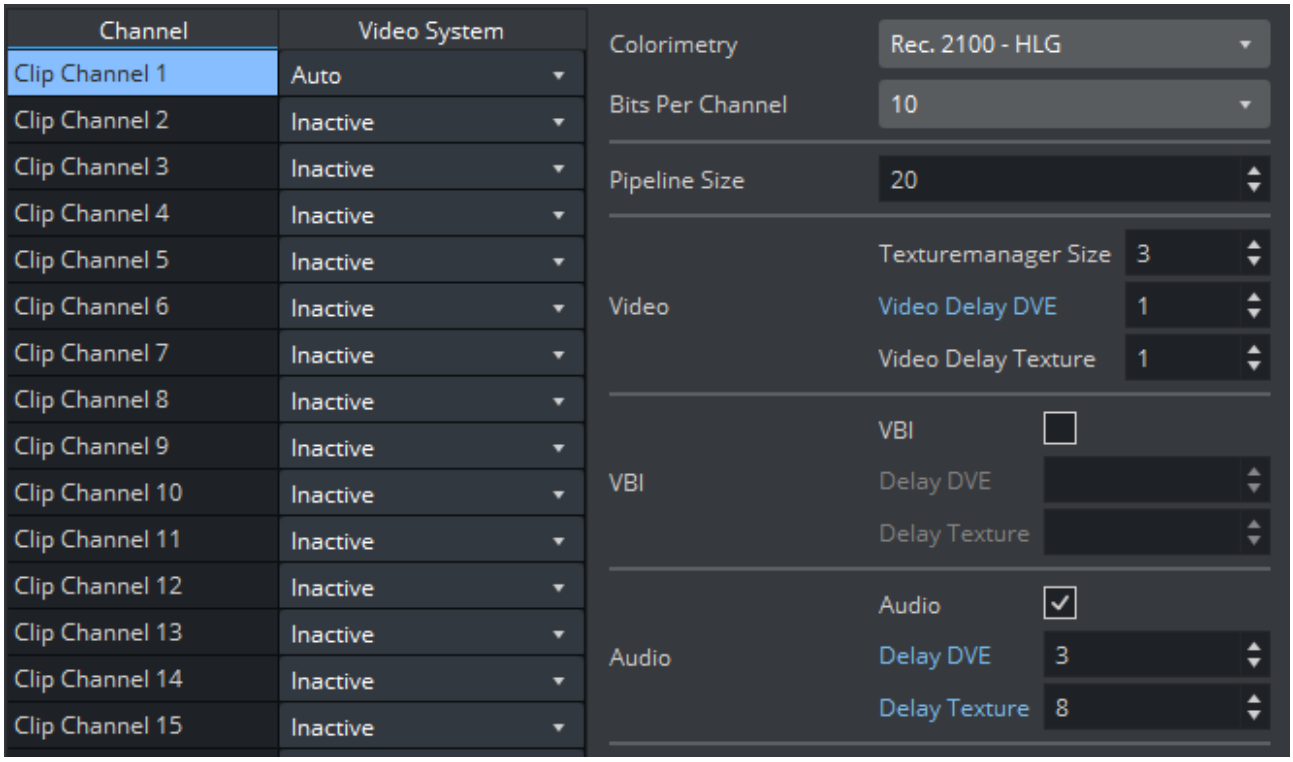
Live Inputs



- Configure the resolution, frame rate, and colorimetry of the input signal.
- It is recommended to use at least ten bits per channel.

Copyright © 2026 Vizrt

Clip inputs



Similar to live configuration, select the format and colorimetry of the clips that should be used in the channel. The Matrox clip player assumes that any loaded clip has the configured colorimetry, while VML uses the metadata of the clip. Using 10 bits per channel is recommended.

10.2.2 NDI Additional Information

HDR is fully supported with NDI, but there are a few details to consider:

- Only the program NDI output in software I/O Mode NDI supports HDR.
- NDI HDR inputs are supported with both Software I/O Mode NDI and Matrox boards.
- The colorimetry of inputs is automatically determined based on metadata in the NDI stream. Streams with missing or incorrect colorimetry metadata are interpreted as SDR.

10.2.3 S-Log3 Support in Viz Engine

Using S-Log3 requires a Matrox board. The only use case supported with S-Log3 is to render SDR overlays on top of S-Log3 videos or clips in the scene background. A proper SDR to S-Log3 LUT needs to be provided. These LUTs might not match Type I or Type III as described above. For a fine grained control, you can configure the settings `sdr_to_hdr_lut_input_range` and `sdr_to_hdr_lut_output_range` separately, in the configuration file.

Copyright © 2026 Vizrt

10.3 Mixed Mode Video Support

This section gives an overview of the Matrox mixed mode video support. The tables are valid for both genlock families in Viz Engine. One is 25/50 (for example, PAL/720p50/1080i25/1080i50) and the other is 30M/60M (for example, NTSC/720p60M/1080i30M/1080i60M).

The genlock family refers to the used house signal (for example, black burst) frequency, typically PAL for the first and NTSC for the latter. Please observe that genlock families can not be mixed.

The following contain information on the following topics:

- Source: PAL or NTSC
  - Video In to DVE and Texture
  - Clip In to DVE and Texture
- Source: 720p
  - Video In to DVE and Texture
  - Clip In to DVE and Texture
- Source: 1080i
  - Video In to DVE and Texture
  - Clip In to DVE and Texture

10.3.1 Source: PAL or NTSC

The first two tables show video in to DVE and texture output capabilities, while the next two tables show clip in to DVE and texture output capabilities. Please observe that genlock families can not be mixed.

Video In to DVE and Texture

Output DVE	Video In		
	PAL/NTSC	720p	1080i
PAL/NTSC	OK	-	-
720p	-	-	-
1080i	OK	-	-
Output Texture	**Video In**		
	PAL/NTSC	720p	1080i
PAL/NTSC	OK	-	-
720p	OK	-	-

Copyright © 2026 Vizrt

Output Texture	Video In		
1080i	OK	-	-

Clip In to DVE and Texture

Output DVE	Clip In		
	PAL/NTSC	720p	1080i
PAL/NTSC	OK	-	-
720p	-	OK	-
1080i	OK	OK	-

Output Texture	Clip In		
	PAL/NTSC	720p	1080i
PAL/NTSC	OK	-	-
720p	OK	OK	-
1080i	OK	OK	OK

10.3.2 Source: 720p

The first two tables show video in to DVE and texture output capabilities, while the next two tables show clip in to DVE and texture output capabilities. Please observe that genlock families can not be mixed.

Video In to DVE and Texture

Output DVE	Video In		
	PAL/NTSC	720p	1080i
PAL/NTSC	-	-	-
720p	-	-	OK
1080i	-	-	OK

Copyright © 2026 Vizrt

Output Texture	Video In		
	PAL/NTSC	720p	1080i
PAL/NTSC	-	OK	-
720p	-	OK	-
1080i	-	OK	-

Clip In to DVE and Texture

Output DVE	Clip In		
	PAL/NTSC	720p	1080i
PAL/NTSC	OK	-	-
720p	-	OK	-
1080i	OK	OK	OK

Output Texture	Clip In		
	PAL/NTSC	720p	1080i
PAL/NTSC	OK	OK	-
720p	OK	OK	-
1080i	OK	OK	OK

10.3.3 Source: 1080i

The first two tables show video in to DVE and texture output capabilities, while the next two tables show the clip in to DVE and texture output capabilities. Please observe that genlock families can not be mixed.

Copyright © 2026 Vizrt

Video In to DVE and Texture

Output DVE	Video In		
	PAL/NTSC	720p	1080i
PAL/NTSC	-	-	-
720p	-	-	-
1080i	-	-	OK

Output Texture	Video In		
	PAL/NTSC	720p	1080i
PAL/NTSC	-	-	OK
720p	-	-	OK
1080i	-	-	OK

Clip In to DVE and Texture

Output DVE	Clip In		
	PAL/NTSC	720p	1080i
PAL/NTSC	OK	-	-
720p	-	OK	-
1080i	OK	OK	OK

Output Texture	Clip In		
	PAL/NTSC	720p	1080i
PAL/NTSC	OK	-	OK
720p	OK	OK	OK

Copyright © 2026 Vizrt

Output Texture	Clip In		
1080i	OK	OK	OK

10.4 Frame Accurate Output

This section contains information on the following topics:

- FAO Prerequisites
- Commands
- General Purpose IO Commands
- Timed Command Execution

Copyright © 2026 Vizrt

10.4.1 FAO Prerequisites

Frame accurate playout can be achieved by using a Timecode Reader card (for example, Plura AVPCL) and sending timed commands from any control application (like Media Sequencer). Scenes need to be loaded and cued some time ahead and then taken On Air at a given timecode. To make sure clip handling is fast, the system has to be configured to use either a RAID-0 hard disc configuration or SSDs.

Plura AVPCL boards can be used by Viz Engine if the driver is installed correctly. Please note that the config file needs to be modified to enable the Timecode sources within Viz Engine:

- The Timecode Reader Card for Viz Engine needs to be enabled in the Viz Config File by setting `TCReaderUsage = 2`.
- The correct timecode source must be set by using **TCReaderSource**. Viz Engine supports *LTC*, *VITC*, *ATC_LTC*, *ATC_VITC*, *HANC_LTC*, *HANC_VITC*, and *FIELDS*. To specify multiple sources the format would be: `TCReaderSource = LTC | VITC`.

Frame accuracy is only supported on Matrox boards.

Currently the following is supported:

Source	Notes	Supported
Live Input as Texture	Inputs need to be enabled in the scene and activated.	✅
Live Input as DVE		✅
Clip Playback as DVE	Currently not supported. Will be fixed in one of the next versions.	❌
Clip Playback as Container Texture	Currently only ClipTexturePlayer Renderer is supported.	✅

⚠ **Note:** Frame accurate Clip Playback using the Matrox player as DVE target is currently not supported.

⚠ **Note:** Initializing any Clip player can take a while. Therefore it is required to increase the output delay to prevent any framedrops.

Copyright © 2026 Vizrt

10.4.2 Commands

To enable frame accurate handling of clips, it is necessary to preload the next scene by using the CUE command:

```
SCENE*<Scene Name> CUE
```

Because the loading time of clips is affected by disc speed and the used codec, allow at least a one second head start for the CUE command. The earlier you cue the next scene the more likely all of the clips are prepared when needed.

After the initial CUE command it is/could be necessary to send the following commands. If the scene was saved at another position than 0, send:

```
SCENE*<Scene Name>*STAGE SHOW 0.0
```

The following is always needed to finally set and activate the scene:

```
RENDERER SET_OBJECT SCENE*<Scene Name>
```

If the director is not configured to auto start you should use:

```
MAIN_SCENE*STAGE START
```

Note: The first and last commands are optional.

To Use the ClipTexturePlayer

The ClipTexturePlayer introduced in Viz Engine 5.3 allows to play a various range of broadcast and non broadcast clip formats (including transparency) without the need to utilize a certain clip channel. They can be used for promotion clips or similar.

The following prerequisites are needed:

- Add the clip to the stage by dragging the ClipTexturePlayer to a director within the stage.
- **Disable Liveplay** and animate the frames within the director.
- The Frames-per-second need to match the output frequency.

Limitations

- ClipTexurePlayer is only available in Viz Engine Render Pipeline scenes, Classic Render Pipeline scenes are not supported.
- Audio is not supported (yet).

Copyright © 2026 Vizrt

10.4.3 General Purpose IO Commands

Viz Engine can send and receive frame accurate commands via General Purpose input/output (GPIO), provided the following requirements are met:

- PCI or PCI Express Sealevel I/O device with eight, 16 or 32 digital inputs installed. Devices connected via Ethernet or USB cannot be used.
- Viz Engine 3.3 (rev 8394) or later to receive commands.
- Viz Engine 3.8.2 to send commands.
- A Matrox X.mio-series video board is required for getting the actual field which is played out.

This section contains information on the following topics:

- Input Functionality
- Flow of the GPI signal
- Output Functionality
- Commands
  - Pin Command - Set
  - Command - Clear
  - Information - Get
  - Pin Command - Test
  - Enable - Set
  - GPO Command - Set

Input Functionality

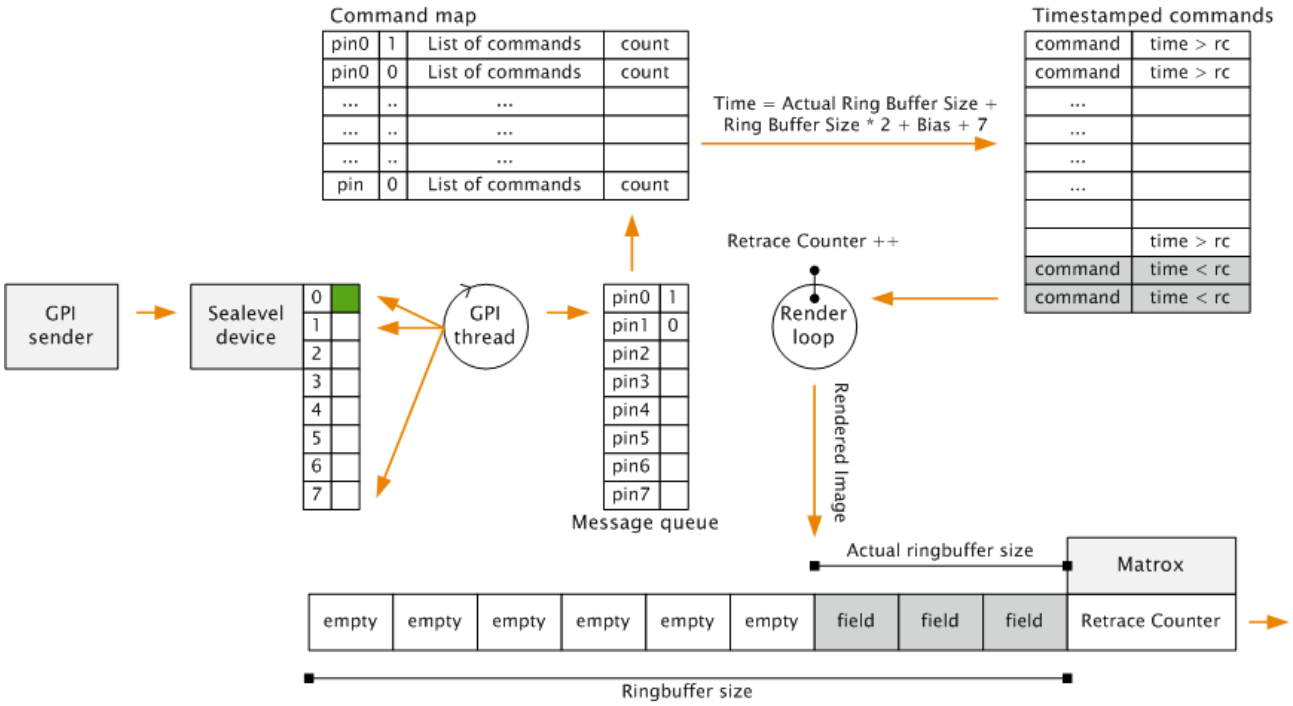
Currently, there are five commands available which enable you to queue commands for execution when a pin on the Sealevel board shows a raising or falling edge. For every Pin, an arbitrary amount of commands can be queued for the raising and falling event. Every command can be armed with a counter which tells Viz Engine how often the command should be executed before it is removed from the queue. A counter of `0` tells Viz Engine that the command should never be removed from the queue.

The following diagram illustrates the Flow of the GPI signal from the sender until the consequences of the executed command are rendered into the correct position in the Matrox ringbuffer. As soon as a **GPI sender** changes the status of a pin, connected to the **Sealevel device**, the change is reflected in an internal register of the card. In Viz Engine, a **thread** polls this register every millisecond. As soon as a change is found it calculates the timestamp for when the command should be executed.

The thread looks for the pin command in the **Command map** and queues the command into the **Timestamped commands** queue. As the actual depth of the Matrox ringbuffer, known the render loop, checks every field if it is time to execute a command from the queue. This guarantees that the command is executed at the correct field, no matter how large or full the ringbuffer actually is.

Copyright © 2026 Vizrt

Flow of the GPI signal



Output Functionality

Since version 3.8.2, Viz Engine is able to send external triggers via GPIO on systems with a supported Sealevel GPI/O card installed, using the GPO Command - Set. Depending on the GPI/O card configuration in the Sealevel driver, the first configured output channel could reside in a higher bank. For example, bank 2 -> first channel = 16.

Note: Only one channel can be sent at a time.

Commands

The following are the available GPIO commands:

Pin Command - Set

```
VIZ_COMMUNICATION*GPI_PIN_COMMAND SET ”<Command ID> <Command>” <PIN> <UP_DOWN>
<COUNT>
```

Adds a command to the queue.

Copyright © 2026 Vizrt

Example: `VIZ_COMMUNICATION*GPI_PIN_COMMAND SET "-1 RENDERER*MAIN_LAYER*STAGE START" 0 1 0`

- **<Command ID>:** The command ID specifier.
- **<Command>:** Command string which should be queued.
- **<PIN>:** Input pin number (valid from `0-31`) where the first Pin is `0`.
- **<UP_DOWN>:** `0` means the command should be executed on a falling edge, `1` means the command should be executed on a raising edge.
- **<COUNT>:** Executes the command <Count> times. The command executes once per event and NOT <Count> times per event. A value lower or equal to `0` means that the command is never removed from the queue.
- **Remarks:** After the GPI event occurs, the execution time of the command is calculated. The execution time is calculated in the following way: `Ring_Buffer_Size*2+7+Delayed_Command_Bias`.
  The `<Delayed_Command_Bias>` can be set in the Viz Config file.
  Therefore, `Ring_Buffer_Size*2+7` is the minimum delay for GPI triggered commands.

Command - Clear

```
VIZ_COMMUNICATION*GPI_PIN_COMMAND CLEAR <PIN> <UP_DOWN>
```

Clears all commands from the queue.

- **<PIN>:** Input Pin number (valid from `0-31`) where the first Pin is `0`.
- **<UP_DOWN>:** `0` simulates a command executed on a falling edge, `1` simulates a command executed on a raising edge.

Information - Get

```
VIZ_COMMUNICATION*GPI_INFO GET
```

Sends information on the installed GPI device. If a valid device is present, the command returns the number of available GPI Banks. Therefore a value of `1` means that one bank (eight inputs) is available. If the command returns `0` no valid GPI device is present.

Pin Command - Test

```
VIZ_COMMUNICATION*GPI_PIN_COMMAND TEST <PIN> <UP_DOWN>
```

Sends all queued commands for the pin and signal to the Viz Engine. The counter for the commands is not decremented.

- **<PIN>:** Input Pin number (valid from `0-31`) where the first Pin is `0`.

Copyright © 2026 Vizrt

- **<UP_DOWN>:** `0` means the command should be executed on a falling edge, `1` means the command should be executed on a raising edge.

Enable - Set

```
GLOBAL*GPI_ENABLE SET <1 or 0>
```

Enables or disables GPI commands for the Viz Engine. All functionality is available except of sending the commands on a GPI signal. If disabled, the command is shown in the console and a warning is shown, that GPI is disabled. The TEST command is working as usual, even if GPI is disabled.

- `1` enables GPI execution
- `0` disables GPI execution

GPO Command - Set

```
VIZ_COMMUNICATION*GPO_BIT SET <bit> <state>
```

Sets the specified output channel to either `On` or `Off`:

- `1` is `On`.
- `0` is `Off`.

The command can be sent directly via the Viz Engine command line, via an action keyframe, or DataPool plug-ins.

Example: `send VIZ_COMMUNICATION*GPO_BIT SET 17 1`

IMPORTANT! The bit count starts at `0`.

Copyright © 2026 Vizrt

10.4.4 Timed Command Execution

To achieve frame accurate output at a certain time, Viz Engine utilizes a time code reader board and special commands. Viz Engine records certain commands and executes them at the exact time that the time code reader board hits the timestamp.

This section contains the following information:

- Time Code Related Commands
- Prerequisites
- Configuration

Note: Viz Engine can only guarantee a frame accurate output, not a field accurate output.

Time Code Related Commands

Time Code	Description
`TC*TC_INFO GET`	Gives you all available time codes in the system. The return is a list of indexes and strings. Every time code source is identified by a unique index.
`TC*TC_ACTIVE GET`	Gives all time codes which have been updated in the last two fields.
`TC*TC GET <index>`	Returns the time code for the index.
`TC*USED_TC GET`	Returns the current used timecode by index
`TC*USED_TC SET`	Sets the timecode source by index.

Copyright © 2026 Vizrt

Time Code	Description
`TC*TC_COMMAND SET_TC <index> <HH>:<MM>:<SS>:<FF> -1 COMMAND`	Queues a Command for execution on a specific time. **<index>:** Index of the timecode. **<HH>:<MM>:<SS>:<FF>:** The timecode in Hours, Minutes, Seconds and Fields. **<#>:** Command answer identifier (currently only a value of `-1` is supported). **COMMAND:** The Command to be executed. **Example:** `TC*TC_COMMAND SET_TC 1 12:07:10:00 -1 RENDERER SET_OBJECT SCENE*<9D1F7526-CFEF-C844-848279FE076E9104>` Activates scene <9D1F7526-CFEF-C844-848279FE076E9104> at exactly 12:07:10:00.
`TC*TC CLEAR <index>`	Clears all queued commands for the time code source with index <index>.
`TC*DELETION_THRESHOLD SET <TC>`	Defines a time frame after which a command is removed when it has not been executed. The threshold needs to be defined as `hh:mm:ss:ff`. <TC> is a valid timecode. **Example:** `TC*DELETION_THRESHOLD SET 00:00:10:00`
`TC*ACTIVATION_THRESHOLD SET <TC>`	Defines a time frame until when a command has to be executed (even if the given timecode has already passed). The threshold needs to be defined as `hh:mm:ss:ff`. <TC> is a valid timecode.
`TC*INSERT_THRESHOLD SET <TC>`	Defines a time frame into the future during which command are allowed to be executed when inserted. The threshold needs to be defined as `hh:mm:ss:ff`. <TC> is a valid timecode. **Example:** `TC*INSERT_THRESHOLD SET 01:00:00:00`

Prerequisites

The following prerequisites are required:

Copyright © 2026 Vizrt

- The system must be equipped with a Plura TC Reader board. This needs to be enabled in the Viz Config File by setting `TCReaderUsage = 2`.
- The correct TC source must be set by using TCReaderSource. Viz Engine supports LTC, VITC, ATC_LTC, ATC_VITC, HANC_LTC, HANC_VITC. To specify multiple sources the format would be: `TCReaderSource = LTC | VITC`.
- Commands need to be sent with a timecode.
  For example: `TC*TC_COMMAND SET_TC 1 12:00:00:00 -1 RENDERER*MAIN_LAYER SET_OBJECT SCENE*<9968FF26-35B0-4C01-AE5A4F8D11D1543B>`. This ensures that the mentioned scene is shown on the output exactly at 12:00:00:00.
- The Output delay should be set to a certain number of frames to prevent any frame drops (for example, to initialize clip codec, etc.).

> **Note:** To verify this, make sure the final output device (Recorder, Monitor) can display the same timecode as attached to the Viz Engine.

Configuration

To configure correct field accurate timeout, a few parameters need to be set. Three important values are responsible for the proper playout of Graphics, DVE Live sources and DVE Clips:

Timed Command Bias

```
Command: TC*TC_BIAS SET n (in fields)
Config Setting: delayed_command_bias = n
```

This represents the overall delay in fields, the commands are sent if the time code hits the timestamp. For instance, if your output appear two frames too late, you need to *use delayed_command_bias*= `-2`.

> **Note:** The Command `TC*TC_BIAS SET n` immediately takes effect for all TC commands sent after this one, without restarting Viz Engine.

The correct sequence to bring a scene On Air exactly at 12:00:00:00 with clips prepared would then be:

```
SCENE*<uuid> LOAD
SCENE*<uuid> CUE

TC*TC_COMMAND SET_TC 1 12:00:00:00 -1 RENDERER*MAIN_LAYER SET_OBJECT SCENE*<uuid>
TC*TC_COMMAND SET_TC 1 12:00:00:00 -1 RENDERER*STAGE START
```

Copyright © 2026 Vizrt

10.5 Shared Usage of Input Channels

This feature allows two or more instances of Viz Engine on one system to show the same content by using the same physical input without the need for a distribution amplifier.

10.5.1 Configuration

To share the input, all instances must have their corresponding input channels configured the same to be used.

Example: `live_type1 = SDI1`

The Viz Engine instance hosting the physical input needs the following value set in the config file:

```
Matrox.InputHost = 1
```

Important: Using shared inputs and setting this configuration value increases the overall in-to-out delay by 1 frame.

10.5.2 Important

The first instance of Viz Engine that is started owns the physical connector of the input. All other instances started after share the content on the video board. If the first instance (in control of the input connector) is closed, all other instances referring to that connector lose their content as well, and need to be closed and restarted again. There is no automatic recovery when the owning instance of Viz Engine crashes.

Note: It is imperative that the first engine fully boots before the others can allocate the shared resource.

Watchdog configuration is also not supported in this case.

Info: This feature is available on system topology SDI boards only (x.mio3, DSX.LE4 and successors).

Info: This feature also works if the requested connector is physically allocated by a Channel Recorder instance.

10.5.3 Sharing Inputs on X.mio5 ST 2110 Boards

This feature works differently on ST 2110 boards. It is possible to subscribe to the same multicast address on multiple connectors. Network switches won't send the same IP packet twice, so no bandwidth is lost. In an NMOS environment, this is straight forward by connecting the same sender to different receivers. For static (and potentially some SDN) setups, it is necessary to configure the same multicast address on the inputs in *ipconfig.xml*.

The important benefit of this approach compared to the dual channel SDI setup is that the inputs work individually, and the shutdown or stopping of one Viz Engine instance doesn't influence the other.

Copyright © 2026 Vizrt

10.6 Dynamic Channel Allocation

As of now, whenever you need an additional input or clip channel, you need to open the configuration, set the channel to the desired resolution on the Video Input page and restart Viz Engine. Only after that, the channel becomes available.

The dynamic allocation allows to dragging the resource from the Media Assets to your scene for immediate use. It uses the default values and the output resolution for configuration. If you need a different resolution for that specific channel, enter Video Input configuration, and select the resolution. The Media Asset can be used afterwards, without the need to restart Viz Engine.

10.6.1 Limitations

- Only dynamic adding of channels is supported at the moment. Removal is possible only by setting the channel to `Invalid` in the configuration, then restarting Viz Engine.
- The maximum number of channels has not changed.
- Dynamic allocation is only supported for inputs. Outputs cannot be acquired on demand.

Note: This feature is meant to be used during design mode only!

Copyright © 2026 Vizrt

10.7 Matrox Supported Codecs

The Matrox X.mio series cards all support a selection of different codecs for both SD and HD. Certain codecs require certain card classes. Depending on the card class, a license upgrade may be required to extend the range of codecs supported by the installed video card. The Matrox video cards have a built-in license dongle used by the Matrox codecs to determine licensing rights.

- For the Matrox X.mio and DSX LE video cards, the following classes apply:
  - **100 class:** No clip playback.
  - **500 class:** SD and HD clip playback.
  - **550 class:** SD and HD clip playback, including support for Apple ProRes codecs.

IMPORTANT! The Codec DNxHD requires its own license, issued by Vizrt.

All codecs are implemented in the software. For all full detailed overview of supported codecs, please refer the **Supported Codecs** page in the **Media Assets** section of the Viz Artist User Guide.

Copyright © 2026 Vizrt

10.8 DVE Performance

This section contains information on the following topics:

- General Information
- X.mio5
- X.mio3
    - Single Channel Performance (1080i50)
    - Dual Channel Performance (1080i50)

10.8.1 General Information

The number of DVE channels that can be used in Viz Engine depends on the video hardware. The limiting factor is the amount of memory on the card, but also the available bandwidth for transfers between the FPGA and the on-board memory. For example, on a Matrox X.mio3 card, you can scale eight 1080i surfaces to 99.9% and compose those eight surfaces at the same time. If the scaling factor is reduced, the overall DVE performance increases, due to the lowered amount of bandwidth needed. At a certain point, it might even be possible to add an additional DVE input. Looking at a single channel system fit with an X.mio3, nine inputs can be used as DVE, as one is always consumed by the background layer. However, setting the system up as dual channel, reduces the amount of DVE layers to three in each channel, considering that two background layers are used.

10.8.2 X.mio5

Input mapping used : 11in1out

Rate	Scaling	Foreground/ Background	Inputs possible without significant performance issues
UHD 50 Hz	100%	1	4 UHD + 2 FullHD
UHD 50 Hz	100%	0	5 UHD + 1 FUllHD / 4 UHD + 5 FullHD
UHD 50 Hz	99%	1	4 UHD + 2 FullHD
UHD 50 Hz	99%	0	5 UHD + 1 FUllHD / 4 UHD + 5 FullHD
UHD 50 Hz	50%	1	5 UHD
UHD 50 Hz	50%	0	5 UHD + 1 FullHD
UHD 50 Hz	33%	1	5 UHD + 1 FullHD
UHD 50 Hz	33%	0	5 UHD + 1 FullHD
UHD 60 Hz	100%	1	3 UHD + 3 FullHD

Copyright © 2026 Vizrt

Rate	Scaling	Foreground/ Background	Inputs possible without significant performance issues
UHD 60 Hz	100%	0	4 UHD + 1 FullHD
UHD 60 Hz	99%	1	3 UHD + 3 FullHD
UHD 60 Hz	99%	0	4 UHD + 1 FullHD
UHD 60 Hz	50%	1	4 UHD + 1 FullHD
UHD 60 Hz	50%	0	4 UHD + 1 FullHD
UHD 60 Hz	33%	1	4 UHD + 1 FullHD
UHD 60 Hz	33%	0	4 UHD + 2 FullHD

10.8.3 X.mio3

Single Channel Performance (1080i50)

The maximum number of DVE Assets (excluding Fore- and Background layer) on **X.mio3**:

Size	Maximum
100%	9
99%	9
33%	10 (Break even point ~95%)

Dual Channel Performance (1080i50)

The maximum number of DVE Assets (excluding Fore- and Background layer) on **X.mio3** for each single channel:

Size	Maximum
100%	4
99%	4
33%	4

Copyright © 2026 Vizrt

Note: The asterisk (*) denotes measurements are done with DSX Utils 9.8.0.22112 on Viz Engine 3.9.0.66266.

Copyright © 2026 Vizrt

10.9 Watchdog

A watchdog is essentially a timer that allows a system to continue video pass-through when an application has crashed or there is a system failure. Matrox X.mio3 boards have a built-in watchdog capability; however, the bypass only works for video (including optional embedded audio), while AES audio is not bypassed.

Note: The watchdog feature is only available on the X.mio3 board.

This section contains the following topics and procedures:

- Mechanical Bypass
- Hardware Bypass
- Transition from Watchdog to Video

10.9.1 Mechanical Bypass

A relay (input to output) that works in case of a power loss.

10.9.2 Hardware Bypass

Operates in a powered machine state (input to the board's output). In hardware bypass mode the incoming video and reference signals must be compliant to provide the correct watchdog functionality.

Tip: The hardware bypass introduces an h-phase on the output. The value of this phase depends on the video output format.

10.9.3 Transition from Watchdog to Video

There are three possible settings which control when to deactivate the watchdog after Viz Engine is restarted.

- **0:** Deactivated as soon as the first geometry is loaded to Viz Engine.
- **1:** Deactivated as soon as Viz Engine finished the startup process. In this case, black plays out as no scene is loaded yet.
- **2:** Deactivated as soon as either the first geometry is loaded to Viz Engine or a live input as DVE is used in the scene, or both.

When watchdog is deactivated (with `video_loopthrough_mode=2`) and a scene with DVE input is loaded, a few black frames are shown. The watchdog should be activated after the input channel is ready. To provide a glitch free transition from watchdog to video, configure the watchdog's deactivation delay by setting the delay in fields for the watchdog to wait before deactivation.

Copyright © 2026 Vizrt

10.10 NDI Output from GFX Channel

This allows to send content from any GFX Channel in a scene to a separate NDI output stream. The resolution of the NDI output stream is equal to the resolution of the GFX Channel itself, with the width rounded down to the nearest even number. The scan mode of the NDI stream is always progressive, and it does not include audio. The possible number of concurrent outputs depends on resolution, CPU performance and available network bandwidth.

Important: The Parallel Outputs license is necessary to use this feature.

Important: Parallel Outputs supports GFX channels only in Texture mode. If the resolution of the scene in the GFX channel exceeds 2048 pixels in either direction, the 4K Video Enablement license is required in addition. Exceeding 4096 pixels is not supported.

10.10.1 Configuration

The name of the NDI sender can be configured per GFX Channel in the Viz Engine configuration file and defaults to *# Gfx1.NDIAuxName = VizEngine-0 GFX-1* for GFX Channel 1 on Viz Engine instance 0.

10.10.2 Operation

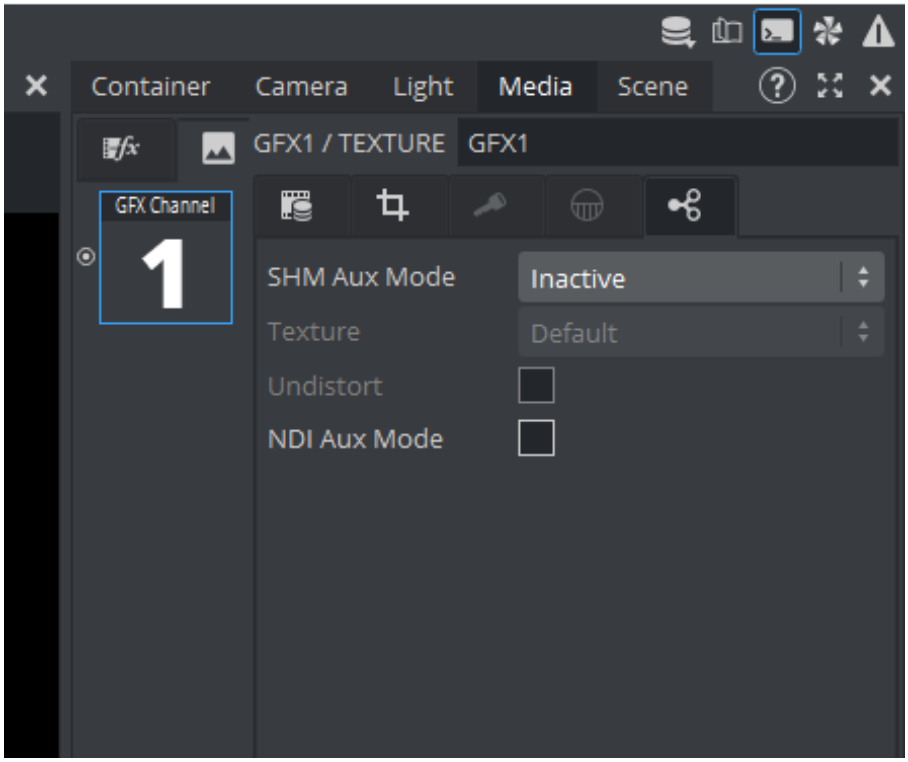
To enable the NDI output stream, either click the checkbox in the pane shown (screenshot below) or send the following command (with *n* being the number of the GFX Channel):

```
MAIN_SCENE*GFX*n*NDI_AUX*MODE SET SEND
```

To stop sending untick the checkbox again or send the command (with *n* being the number of the GFX Channel):

```
MAIN_SCENE*GFX*n*NDI_AUX*MODE SET INACTIVE
```

Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

10.11 Clip Player Pro Features

Viz Engine supports new codecs using the VML clip player.

Important: The Clip Player Pro license is required to use these codecs.

10.11.1 H.265

HEVC/H.265 encoded clips can be played using the VML clip player. The clips are decoded on the CPU, which might result in a high CPU load depending on the clip resolution. If hardware-accelerated playback is enabled, it is possible to decode it using GPU; however, not all GPU supports decoding of H.265 encoded clips. It is recommended that the duration of a GOP does not exceed 0.5s.

10.11.2 HAP

The VML clip player in Viz Engine currently supports HAP 1, HAP 5 and HAP Y encoded clips. Clips using these codecs can directly be used by the GPU, which results in minimal CPU performance impact. However, due to the size of the clips, disk performance and general data throughput of the system are affected.

Note: The chunk size must be set high enough to ensure smooth playback of HAP clips. A good rule is to match the chunk size to the number of CPU cores available on the machine. Note that when encoding with FFmpeg, the default chunk size is `1`, so you need to explicitly specify a higher value.

10.11.3 GPU Hardware Acceleration / NVDEC

VML clip player supports NVDEC hardware acceleration for supported codecs like H.264 or H.265. A Clip Player Pro license is required to enable the GPU clip decoding. Without a license, decoding is done on the CPU.

GPU Requirements

Tested on:

1. NVIDIA GA104 [GeForce RTX 3070 Ti] (Ampere Architecture)
2. NVIDIA RTX 4000 SFF Ada generation (Ada Lovelace)
3. NVIDIA RTX A2000 (Ampere Architecture)

Details of architecture/family can be found here: https://docs.nvidia.com/video-technologies/video-codec-sdk/12.1/nvdec-video-decoder-api-prog-guide/index.html.

- **Supported Codecs:** H.264, H.265, AV1
- **Supported Pixel Formats:** NV12, YUV420p, P016, YUV420p10le
- **Supported Containers:** MP4
- **Bit Depth:** 8-bit and 10-bit
- **Audio Codec:** AAC, PCM

Copyright © 2026 Vizrt

Enable Hardware Accelerated Playback

```
## Allow using NVDEC to decode clips
#* clip_use_NVDEC: Default=0
# clip_use_NVDEC = 0
clip_use_NVDEC = 1
```

Limitations

1. 12-bit bit depth not tested or supported.
2. Interlaced is not supported for hardware-accelerated video decoding.

> ⚠ **Note:** The clip channel must be configured manually to progressive to be able to play with NVDEC.

3. It is not possible to choose which GPU to use (for decoding clips) in a multi-GPU setup.

Copyright © 2026 Vizrt

10.12 Extraction of VANC Data

10.12.1 Types of VANC Data

Two types of VANC data can be extracted and written to shared memory pipes for further use.

- **Closed Captions:** CC data embedded in the VANC data as per SMPTE ST 291M if they are written which the specified DID 0x61.
- **Custom data:** Extraction of data within the user-defined areas as defined in the SMPTE ST 291M - Type1 DID 0xC0 - 0xCF and Type2 DID 0x50 - 0x5F.
  The current implementation is limited to the extraction of data based on the NBA Tissot Timer (NTT) protocol by Swiss Timing LTD.

All data is updated once per video unit (field or frame).

10.12.2 Configuration

For data to be extracted, `Matrox.CCExtraction = 1` must be set either directly in the configuration file or by setting the value in the Matrox General Section of the Configuration GUI. Additionally, VBI must be enabled on all related input and output channels. As VBI can only be handled from one input, the correct one must be selected in the Scene Settings.

10.12.3 Shared Memory Locations

Closed Caption

CC data is stored in five different shared memory keys:

- `Cea708_String1`
- `Cea708_String2`
- `Cea708_String3`
- `Cea708_String4`
- `Cea708_NumLines`

The `_String` values hold the closed captions itself and `_NumLines` defines how many of those are actually valid.

NTT Data

The valid data is stored in the shared memory key `NTT_Message` as plain string and needs to be interpreted according to the protocol specification.

Copyright © 2026 Vizrt

10.13 Multiple SDI/IP Outputs

10.13.1 Prerequisites

- Matrox SDI or IP board with enough connectors available
- Parallel Output license.

Important:
- Parallel Outputs support GFX channels only in Texture mode.
- If the resolution of the scene in the GFX channel exceeds 2048 pixels in either direction, the 4K Video Enablement license is required in addition.
- Exceeding 4096 pixels is not supported.

10.13.2 Types of Output Signal

Viz Engine is capable of sending multiple signals out of one Viz Engine instance. Given enough available connectors (in SDI), next to a Fill and Key signal, one Viz Engine instance is able to deliver (one or more) output formats:

Preview purposes	Keyer purposes	Adaptive and Multioutput purposes
Still Preview (by sending a Viz Engine command).	**Matte Scene**: To setup a virtual studio and its matte areas.	Any number of the available output channels, each with its own **Content** and resolution.
Clean Feed: the incoming video without any compositing effects (graphics on top).	**KeyerAux**: To finetune the Precision Keyer.	
Downscaled Preview (for example, for UHD workflows) the output is 1080p.		

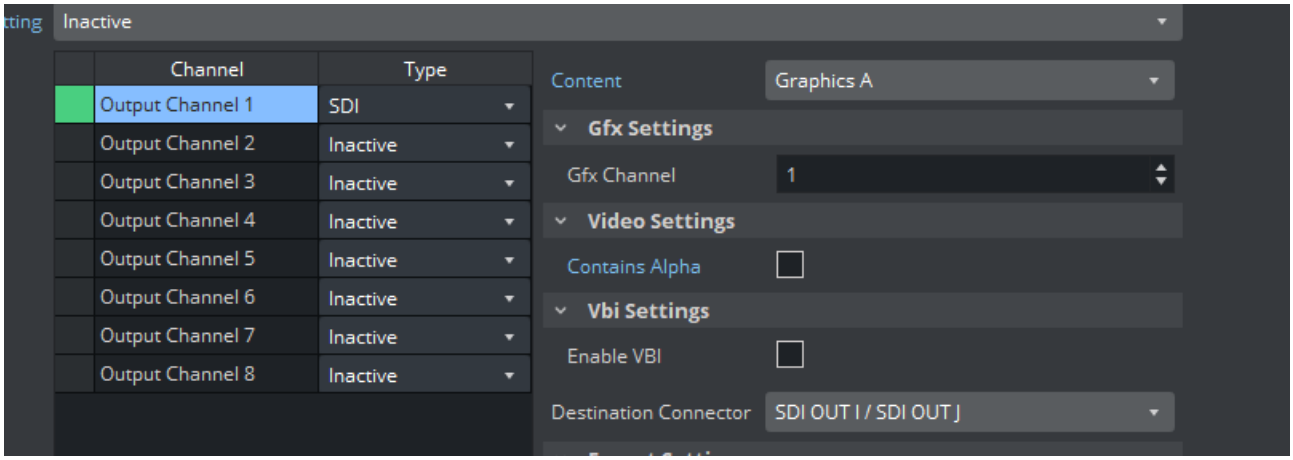
From Viz Engine 5.6, additional outputs are no longer limited to two GFX channels and can run at their own resolutions. See Video Output for the current list of output types and their licence requirements.

To Configure the Output Signal

1. Go to configuration setting **Video Output.**
2. On the left side, choose your output channel (SDI/IP) (Hardware dependent).
3. Select the **Content** (see Types of Output Signal).
   - For **Media Asset**, choose the asset type and the channel you want to map. This replaces the *Graphics* content value of earlier versions.

Copyright © 2026 Vizrt

Information: The assignment can also be changed on the fly during runtime.



See Also

- NDI Output from GFX Channel

10.14 Clipout Channel Usage

10.14.1 Introduction

The clip output channel allows capturing of the rendered video and audio as it appears on the output of the video card into one or more files. Creation of interleaved and atomic files is possible. Depending on the resolution/ container/codec combination video, as fill and key, audio, VBI, and time code information can be saved.

The generation of a proxy clip with video and optional audio is also supported. It is similar to the post-render plug-ins, except that does realtime capturing of audio, video and VBI of the actual output as it appears on the video screen. The clip out channel is available on Matrox X.mio hardware or DSX Core systems.
Before it can be used it needs to be enabled in the configuration (Config/Matrox/ClipOut1/Capture Enable). Once this is done it may be configured and controlled via the external command interface. The basic command structure is:

```
RENDERER*VIDEO*CLIPOUT*n*command
```

where *n* is the index of the clip channel, starting at one and command is a placeholder for the various commands. The configuration is state aware (the possible next configuration command depends on the current state of the clip out channel). For example, the codec that can be set depends on the container that is configured and so on. Hence, a certain sequence of the commands must be obeyed.

10.14.2 Available File Types

The first step is to configure the type of file to create. It is a multiple choice of the following types:

- Video
- Key
- Audio
- VBI
- Proxy

Video File

This file contains video, as fill and key, or fill only. If supported by the container and codec, it may be interleaved with audio and VBI. It can carry time code. For some container/codec combinations, audio and VBI are required (whereas for others they may be optional).

Enable or disable creation of the video file with the following command:

```
RENDERER*VIDEO*CLIPOUT*n*CREATE VIDEO_SET arg
```

where *arg* is a boolean argument and can be enabled with `On` or `1` or disabled with `Off` or `0`.

Key File

This file contains only the alpha component of the video signal. This is beneficial if the requested codec does not support an alpha channel. If enabled then the video signal is split into fill and key. The fill part is written into the video file and the key part into this file. Enable or disable creation of the key file with the following command:

Copyright © 2026 Vizrt

```
RENDERER*VIDEO*CLIPOUT*n*CREATE KEY_SET 1
```

Audio File

This file contains only the audio signal. This is beneficial if the container/codec combination does not allow audio or does not utilize the current audio configuration. Enable or disable creation of the audio file with the following command:

```
RENDERER*VIDEO*CLIPOUT*n*CREATE AUDIO_SET 1
```

VBI File

This file contains only the VBI signal. A separate VBI file is useful when the container/codec does not allow interleaved VBI or does not utilize the current VBI configuration. Enable or disable creation of the *.vbi* file with the following commands:

```
RENDERER*VIDEO*CLIPOUT*n*CREATE VBI_SET 1
```

Proxy File

If enabled, a proxy clip is created along the full resolution clip. The proxy clip also supports interleaved audio. Enable or disable creation of the audio file with the following command:

```
RENDERER*VIDEO*CLIPOUT*1*CREATE PROXY_SET 1
```

Copyright © 2026 Vizrt

10.14.3 Video, Audio, VBI Settings

- Video Resolution
- Video Container
- Video Codec
- Audio Container
- VBI Container
- VBI Codec
- Options
    - Audio Configuration
    - Mpeg IFrame Configuration
    - VBI Config
    - Time Code Configuration
    - High Quality XDCAM HD
    - ASF Audio Configuration
    - ASF Video Configuration
    - QT Codec Configuration
    - QT Video Configuration
- Proxy
    - Proxy Resolution
    - Proxy Container
    - Proxy Codec
    - Proxy Options
    - Proxy Audio Configuration
    - Proxy ASF Audio Configuration
    - Proxy ASF Video Configuration
    - Proxy QT Codec
    - Proxy QT Video
    - Profile

Video Resolution

After setting the file types the resolution is next. Get or set the resolution with the following command:

```
RENDERER*VIDEO*CLIPOUT*n*RESOLUTION VIDEO_GET
RENDERER*VIDEO*CLIPOUT*n*RESOLUTION VIDEO_SET w h
```

w stands for width and *h* for height in pixels of the full frame. Width and height can be arbitrary numbers but need to smaller or equal to the resolution of the program output channel.

Frame rate, scan mode and aspect ratio are copied from the program output channel.

Copyright © 2026 Vizrt

Video Container

The next step is to select the container for the video file. If a key file is requested then it applies for the key file, too. The following commands show how to query, get and set the container type:

```
RENDERER*VIDEO*CLIPOUT*n*CONTAINER VIDEO_ENUMERATE
RENDERER*VIDEO*CLIPOUT*n*CONTAINER VIDEO_GET
RENDERER*VIDEO*CLIPOUT*n*CONTAINER VIDEO_SET container
```

- `VIDEO_ENUMERATE` lists the possible containers. The available containers depend on the resolution.
- `VIDEO_GET` shows the current container.
- `VIDEO_SET` sets the container to container.

The argument may be given either as string or integral value. Available containers:

Container	Description
AVI	avi container
P2_MXF	P2 mxf container
XDCAM_MXF	xdcam mxf container
DVCPRO_MXF	dvcpro mxf container
MATROX_MOV	mov container with matrox native file writer
QT_MOV	mov container with quicktime framework file writer
ASF	windows asf container
WMV	windows wmv container

Example: Set Container Type To XDCAM_MXF

```
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET XDCAM_MXF
```

Video Codec

Once the container is set, the video codec can be selected. The following commands show how to query, get and set the codec type:

Copyright © 2026 Vizrt

```
RENDERER*VIDEO*CLIPOUT*n*CODEC VIDEO_ENUMERATE
RENDERER*VIDEO*CLIPOUT*n*CODEC VIDEO_GET
RENDERER*VIDEO*CLIPOUT*n*CODEC VIDEO_SET codec
```

- `VIDEO_ENUMERATE` lists the possible codecs. The available codecs depend on the container type and the resolution.
- `VIDEO_GET` shows the current codec.
- `VIDEO_SET` sets the codec to codec. The argument may be given either as string or integral value.

Example: Set Codec Type To Mpeg IFrame 4224 In SD

```
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET 35
```

Audio Container

If a separate audio file should be generated, then the desired audio container must be set.

The following commands show how to query, get and set the container type:

```
RENDERER*VIDEO*CLIPOUT*n*CONTAINER AUDIO_ENUMERATE
RENDERER*VIDEO*CLIPOUT*n*CONTAINER AUDIO_GET
RENDERER*VIDEO*CLIPOUT*n*CONTAINER AUDIO_SET container
```

- `AUDIO_ENUMERATE` lists the possible containers.
- `AUDIO_GET` shows the current container.
- `AUDIO_SET` sets the container type to container. The argument may be given either as string or as integral value.

Available containers:

Container	Description
WAV	wav container
W64	wave64 container
AIFF	aiff container
MXF	mxf audio container

Copyright © 2026 Vizrt

Example: Set Audio Container To wave64

```
RENDERER*VIDEO*CLIPOUT*1*CONTAINER AUDIO_SET W64
```

VBI Container

If a separate *.vbi* file should be generated, then the VBI container must be specified. The following commands show how to query, get and set the container type:

```
RENDERER*VIDEO*CLIPOUT*n*CONTAINER VBI_ENUMERATE
RENDERER*VIDEO*CLIPOUT*n*CONTAINER VBI_GET
RENDERER*VIDEO*CLIPOUT*n*CONTAINER VBI_SET container
```

- `VBI_ENUMERATE` lists the possible containers are being listed.
- `VBI_GET` shows the current container.
- `VBI_SET` sets the container type to container. The container argument may be given either as string or as integral value.

Available containers:

Container	Description
AVI	avi container

Example: Set VBI Container To AVI

```
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VBI_SET AVI
```

VBI Codec

The following commands show how to query, get and set the codec type:

```
RENDERER*VIDEO*CLIPOUT*n*CODEC VBI_ENUMERATE
RENDERER*VIDEO*CLIPOUT*n*CODEC VBI_GET
RENDERER*VIDEO*CLIPOUT*n*CODEC VBI_SET codec
```

- `VBI_ENUMERATE` lists the possible codecs.
- `VBI_SET` sets the codec to codec. The codec argument may be given either as string or integral value.
- `VBI_GET` shows the current codec.

Copyright © 2026 Vizrt

Example: Set VBI Codec To YUYV422

```
RENDERER*VIDEO*CLIPOUT*1*CODEC VBI_SET YUYV422
```

Options

Some resolution/container/codec combinations permit or require additional configuration. The following commands show how to query for options and how to get the current options:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE group
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET group
```

- `VIDEO_ENUMERATE` lists all possible and mandatory options for the specified group. A special group *ALL* lists all option groups with their possible configurations.
- `VIDEO_GET` shows the current configuration for the specified group.

Example: Enumerate All Available Video Options

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_ENUMERATE ALL
```

Audio Configuration

The following commands query, get, and set the audio configuration:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE AUDIO
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET AUDIO
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET AUDIO options
```

- `VIDEO_ENUMERATE AUDIO` lists the available options.
- `VIDEO_GET AUDIO` shows the current audio configuration
- `VIDEO_SET AUDIO` specifies the audio options.

The options argument can be:

- **Off:** Does not capture audio.
- **On:** Uses the current audio configuration from Viz Engine.
- **ch=n:** Uses *n* audio channels.
- **ch=n bits=v:** Uses *n* audio channels with *v* valid and total bits.
- **ch=n bits=v/t:** Uses *n* audio channels with *v* valid bits and *t* total bits.

Some codecs require a specific set of audio configuration. In this case, the applied audio option must meet the codec requirements (for example, MXF-XDCAM). Otherwise, the clip fails to open.

Copyright © 2026 Vizrt

Example: Set Audio Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AUDIO ON
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AUDIO ch =4 bits =24/32
```

Mpeg IFrame Configuration

Mpeg IFrame codecs require the configuration of the IFRAME parameters. The following commands show how to enumerate, get the current configuration and set new values:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE IFRAME
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET IFRAME
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET IFRAME options
```

- `VIDEO_ENUMERATE IFRAME` lists the avaiable mpeg iframe options.
- `VIDEO_GET IFRAME` lists the current configuration and with VIDEO_SET IFRAME new values can be set.

The options argument can have the following values:

- **OFF:** Does not configure mpeg iframe.
- **bitrate=r:** Sets the bitrate (*r,* an integral value) in Mbits per second.
- **forceDCTFrame=b:** Switches the forced DCTFrame off or on (*b* is either `0` or `1`).
- **DCPrecision:** Set to a fixed value: `DCPrecision8bits`.
- **zigzag:** Set to a fixed value: `ZigZagTypeAlternate`.
- **rounding:** Set to a fixed value: `RoundingTypeMatroxCustom`.

For a detailed list of available values use the `VIDEO_ENUMERATE IFRAME` command. All arguments must be given in one command. They do not accumulate.

Example: Mpeg IFrame Configuration For SD

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET IFRAME bitrate =30 forceDCTFrame =1
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET IFRAME OFF
```

VBI Config

Some codecs, like D10, require VBI information to be written. In that case, this options must be set (even if Viz Engine is not configured for VBI). The following commands show how to enumerate, get the current configuration and set new values:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE VBI
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET VBI
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET VBI options
```

Copyright © 2026 Vizrt

The options argument is the number of VBI lines to write into the file. It must be an even number for interlaced resolutions. A value of ON can be used as shortcut for setting the Viz Engine configured VBI lines. OFF switches VBI configuration off. Some codecs require a specific set of VBI configuration. In this case, the applied VBI option must meet the codec requirements. Otherwise the clip fails to open.

Example: VBI Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET VBI ON
RENDERER*VIDEO*CLIPOUT*}1*OPTION VIDEO_SET VBI 34
```

Time Code Configuration

If supported by the container then time code can be written.

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE TC
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET TC
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET TC [ options ]
```

The options argument can be omitted to switch on the default time code. Other values can be:

- **startTC=hh:mm:ss:ff:** Sets hh:mm:ss:ff as start time code.
- **OFF:** Switches off time code option.

Example: Time Code Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET TC
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET TC startTC =12:34:56:00
```

High Quality XDCAM HD

For XDCAM HD, this flag specifies to use the high quality settings in the codec with the compromise to use more CPU and time to generate a compressed image.

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE HQ_XDCAM
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET HQ_XDCAM
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET HQ_XDCAM arg
```

The *arg* argument can be ON or OFF.

- **ON:** Enables high quality writing.
- **OFF:** Uses fast encoding.

Copyright © 2026 Vizrt

Example: High Quality XDCAMHD Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET HQ_XDCAM ON
```

ASF Audio Configuration

If the container is of type ASF or WMV, ASF audio can be configured. The following commands show how to enumerate, get the current setting and set new values:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE ASFAUDIO
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET ASFAUDIO
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET ASFAUDIO options
```

The options argument in `VIDEO_SET ASFAUDIO` can have the following values:

- **codec_type=t:** Sets the ASF codec type to *t*.
- **codec_format=f:** Sets the ASF codec format to *f*.

For a detailed list of the available options use the `VIDEO_ENUMERATE ASFAUDIO` command. All arguments must be given in one command. They do not accumulate.

Example: ASF Audio Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET ASFAUDIO OFF
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET ASFAUDIO codec_type =2 codec_format =4
```

ASF Video Configuration

If the container is of type ASF or WMV, ASF video can be configured. The following commands show how to enumerate, get the current setting and set new values:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE ASFVIDEO
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET ASFVIDEO
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET ASFVIDEO options
```

The options argument in `VIDEO_SET ASFVIDEO` can have the following values:

- **codec_type=t:** Sets the ASF codec type to *t*.
- **codec_mode=m:** Sets the ASF codec mode to *m*.
- **bitrate=r:** Sets the bitrate to *r*.
- **maxbitrate=r:** Sets the maximum bitrate to *r*.
- **bufferwindow=w:** Sets the bufferwindow to *w*.
- **maxbufferwindow=w:** Sets the maximum buffer window to *w*.
- **secondsperkeyframe=s:** Sets the seconds between keyframes to *s*.

Copyright © 2026 Vizrt

- **quality=q:** Sets the quality to *q*.
- **vbrquality=q:** Sets the variable bitrate quality to *q*.

For a detailed list of the available options use the `VIDEO_ENUMERATE ASFVIDEO` command. All arguments must be given in one command. They do not accumulate.

Example: ASF Video Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET ASFVIDEO On
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET ASFVIDEO codec_type=3 codec_mode=2
quality=100
```

QT Codec Configuration

If the container is of type QT_MOV then the quicktime framework can be configured. This options is available **after** the clip has been opened by a successfully `NAME SET` command. The following command shows how to enumerate the available QT codecs, get the current setting and how to set a new codec.

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE QTCODEC
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET QTCODEC
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET QTCODEC options
```

The options argument for the `VIDEO_SET QTCODEC` command can have the following values:

- **codec_type=c:** Sets the QT codec to *c*. It can be given as 4cc character value or as little endian decoded integer.

Please take a look at the `VIDEO_ENUMERATE QTCODEC` output.

Example: QT Codec Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET QTCODEC codec_type=avc1
```

QT Video Configuration

If the container is of type QT_MOV, the quicktime framework can be configured. This option is available **after** the clip has been opened by a successfully `NAME SET` command.The following command shows how to enumerate the available QT video options, get the current setting and how to set new options:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_ENUMERATE QTVIDEO
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_GET QTVIDEO
RENDERER*VIDEO*CLIPOUT*n*OPTION VIDEO_SET QTVIDEO options
```

The options argument for the `VIDEO_SET QTVIDEO` command can have the following values:

Copyright © 2026 Vizrt

- **allow_frame_reordering=0|1:** Enables/disables frame reordering when compression video data. If set to true, the decode order of the frames are different from the display order. This operation may produce a better compression.

> ⚠ **Note:** Not all compression formats support this option.

- **allow_temporal_compression=0|1 :** Enables/disablestemporal compression when compressing video data. If set to true, frames are compressed based on the previous frames.

> ⚠ **Note:** Not all compression formats support this option.

- **allow_time_frame_change=0|1:** Enables/disables frame time changes by the compressor. If set to true, the compressor combines similar adjacent frames into one frame with a duration equal to the sum of the duration of the two frames.
- **average_data_rate=r:** Average data rate of the resulting QuickTime movie file. The value is expressed in bytes per second. Setting a data rate of zero indicates that the quality setting determines the size of compressed data.
- **max_frame_delay_count=n:** Maximum number of frames that can be kept into memory before emitting one frame. Set to `-1` for no limit.
- **max_key_frame_interval=n:** Maximum interval between each key frame. If set to `0`, all frames are key frames.
- **max_partial_sync_frame_interval=n:** Maximum interval between each partial key frame.
- **cpu_time_budget=n:** CPU time budget in microseconds for each frame sent to the codec. This option may be ignored by the codec. If set to `0`, the codec attempts to compress the data as fast as possible. If set to `0xFFFFFFFF`, no limit is set.
- **quality=n:** Quality of the video data compression. The value is expressed as an integer going between 0 and 1024 (inclusive) where 0 is the minimum quality and 1024 is lossless compression.
- **limit_data_rate=n:** Total size of compressed data in bytes per second that the codec output must not exceed.

For a detailed list of available options use the `VIDEO_ENUMERATE QTVIDEO` command. The options for `QTVIDEO` accumulate. They may be given on several consecutive command lines.

Example Quicktime Video Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET QTVIDEO allow_frame_reordering =0
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET QTVIDEO allow_temporal_comression =1
 average_data_rate =10000000 max_file_update_interval =5
```

Proxy

If the generation of a proxy file was enabled, the proxy needs to be configured. The procedure is similar to the one from the full resolution above.

Copyright © 2026 Vizrt

Proxy Resolution

First step is to set the resolution. Get or set the resolution with the following commands:

```
RENDERER*VIDEO*CLIPOUT*n*RESOLUTION PROXY_GET
RENDERER*VIDEO*CLIPOUT*n*RESOLUTION PROXY_SET w h
```

w is width and *h* is height in pixels of the full frame. Width and height can be arbitrary numbers, but need to be smaller or equal to the resolution of the program output channel. Frame rate, scan mode and aspect ratio are copied from the program output channel. There are some shortcuts available.

- **w=0 h=0:** Sets the resolution to the same values as for the program output channel.
- **w=0 h=n:** Sets the resolution to *n* percent of the program output channel.

Example: Proxy Set Resolution To 50%

```
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION PROXY_SET 0 50
```

Proxy Container

The next step is to select the container for the video file. The following commands show how to query, get and set the container type.:

```
RENDERER*VIDEO*CLIPOUT*n*CONTAINER PROXY_ENUMERATE
RENDERER*VIDEO*CLIPOUT*n*CONTAINER PROXY_GET
RENDERER*VIDEO*CLIPOUT*n*CONTAINER PROXY_SET container
```

- `PROXY_ENUMERATE` lists the possible containers. The available containers depend on the resolution. The container argument may be given either as string or integral value.
- `PROXY_GET` shows the current container.

Example: Proxy Set Container Type To ASF

```
RENDERER*VIDEO*CLIPOUT*1*CONTAINER PROXY_SET AS
```

Proxy Codec

Once the container is set the video codec can be selected. The following commands show how to query, get and set the codec type:

```
RENDERER*VIDEO*CLIPOUT*n*CODEC PROXY_ENUMERATE
RENDERER*VIDEO*CLIPOUT*n*CODEC PROXY_GET
```

Copyright © 2026 Vizrt

```
RENDERER*VIDEO*CLIPOUT*n*CODEC PROXY_SET codec
```

- `PROXY_ENUMERATE` lists the possible codecs. The available codecs depend on the container type and the resolution. The codec argument may be given either as string or integral value.
- `PROXY_GET` shows the current codec.

Example: Proxy Set Codec Type To YUYV422

```
RENDERER*VIDEO*CLIPOUT*1*CODEC PROXY_SET YUYV422
```

Proxy Options

Some resolution/container/codec combinations require or permit additional configuration. The following commands show how to query for options and how to get the current options.

```
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_ENUMERATE group
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_GET group
```

- `PROXY_ENUMERATE` lists all possible and mandatory options for the specified group. A special group *ALL* lists all option groups with their possible configurations.
- `PROXY_GET` shows the current configuration for the specified group.

Example: Proxy Enumerate All Available Options

```
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_ENUMERATE ALL
```

Proxy Audio Configuration

The following commands query, get, and set the audio configuration:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_ENUMERATE AUDIO
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_GET AUDIO
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_SET AUDIO options
```

- `PROXY_ENUMERATE AUDIO` lists the available options.
- `PROXY_GET AUDIO` shows the current audio configuration.
- `PROXY_SET AUDIO` specifies audio options.

The options argument can be:

- **Off:** Does not capture audio.
- **On:** Uses the current audio configuration from the Viz Engine.

Copyright © 2026 Vizrt

Examples: Proxy Set Audio Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET AUDIO ON
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET AUDIO OFF
```

Proxy ASF Audio Configuration

If the container is of type ASF or WMV, ASF audio can be configured. The following commands show how to enumerate, get the current setting and set new values:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_ENUMERATE ASFAUDIO
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_GET ASFAUDIO
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_SET ASFAUDIO options
```

The options argument in `PROXY_SET ASFAUDIO` can have the following values:

- **codec_type=t:** Sets the ASF codec type to *t*.
- **codec_format=f:** Sets the ASF codec format to *f*.

For a detailed list of the available options use the `PROXY_ENUMERATE ASFAUDIO` command. All arguments must be given in one command. They do not accumulate.

Example: Proxy ASF Audio Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET ASFAUDIO OFF
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET NAME SETASFAUDIO codec_type=2
 codec_format=4
```

Proxy ASF Video Configuration

If the container is of type ASF or WMV, ASF video can be configured. The following commands show how to enumerate, get the current setting and set new values:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_ENUMERATE ASFVIDEO
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_GET ASFVIDEO
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_SET ASFVIDEO options
```

The options argument in `PROXY_SET ASFVIDEO` can have the following values:

- **codec_type=t:** Sets the ASF codec type to *t*.
- **codec_mode=m:** Sets the ASF codec mode to *m*.
- **bitrate=r:** Sets the bitrate to *r*.
- **maxbitrate=r:** Sets the maximum bitrate to *r*.
- **bufferwindow=w:** Sets the bufferwindow to *w*.

Copyright © 2026 Vizrt

- **maxbufferwindow=w:** Sets the maximum buffer window to *w*.
- **secondsperkeyframe=s:** Sets the seconds between keyframes to *s*.
- **quality=q:** Sets the quality to q.
- **vbrquality=q:** Sets the variable bitrate quality to *q*.

For a detailed list of the available options use the `PROXY_ENUMERATE ASFVIDEO` command. All arguments must be given in one command. They do not accumulate.

Example: Proxy ASF Video Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET ASFVIDEO On
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET ASFVIDEO codec_type =3 codec_mode =2
 quality =100
```

Proxy QT Codec

If the container is of type QT_MOV, the quicktime framework can be configured. This options is available **after** the clip has been opened by a successfully `NAME SET` command. The following command shows how to enumerate the available QT codecs, get the current setting and how to set a new codec:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_ENUMERATE QTCODEC
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_GET QTCODEC
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_SET QTCODEC options
```

The options argument for the `PROXY_SET QTCODEC` command can have the following values:

- **codec_type=c:** Sets the QT codec to *c*. It can be given as 4cc character value or as little endian decoded integer. Please take a look at the `PROXY_ENUMERATE QTCODEC` output.

Example: Proxy Quicktime Codec Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET QTCODEC codec_type = mp4v
```

Proxy QT Video

If the container is of type QT_MOV, the quicktime framework can be configured. This options is available **after** the clip has been opened by a successfully `NAME SET` command. The following command shows how to enumerate the available QT video options, get the current setting and how to set new options:

```
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_ENUMERATE QTVIDEO
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_GET QTVIDEO
RENDERER*VIDEO*CLIPOUT*n*OPTION PROXY_SET QTVIDEO options
```

Copyright © 2026 Vizrt

The options argument for the `VIDEO_SET QTVIDEO` command can have the following values:

- **allow_frame_reordering=0|1:** Enables/disables frame reordering when compressing video data. If set to true, the decode order of the frames is different from the display order. This operation may produce better compression.

> ⚠ **Note:** Not all compression formats support this option.

- **allow_temporal_compression=0|1:** Enables/disables temporal compression when compressing video data. If set to true, frames are compressed based on the previous frames.

> ⚠ **Note:** Not all compression formats support this option.

- **allow_time_frame_change=0|1:** Enables/disables frame time changes by the compressor. If set to true, the compressor combines similar adjacent frames into one frame with a duration equal to the sum of the duration of the two frames.
- **average_data_rate=r:** Average data rate of the resulting QuickTime movie file. The value is expressed in bytes per second. Setting a data rate of zero indicates that the quality setting determines the size of compressed data.
- **max_frame_delay_count=n:** Maximum number of frames that can be kept into memory before emitting one frame. Set to `-1` for no limit.
- **max_key_frame_interval=n:** Maximum interval between each key frame. If set to `0`, all frames are key frames.
- **max_partial_sync_frame_interval=n:** Maximum interval between each partial key frame.
- **cpu_time_budget=n:** CPU time budget in microseconds for each frame sent to the codec. This option may be ignored by the codec. If set to `0`, the codec attempts to compress the data as fast as possible. If set to `0xFFFFFFFF`, no limit is set.
- **quality=n:** Quality of the video data compression. The value is expressed as an integer between 0 and 1024 (inclusive) where 0 is the minimum quality and 1024 is lossless compression.
- **limit_data_rate=n:** Total size of compressed data in bytes per second that the codec output must not exceed.

For a detailed list of available options use the `PROXY_ENUMERATE QTVIDEO` command. The options for `QTVIDEO` accumulate. They may be given on several consecutive command lines.

Example: Proxy Quicktime Video Configuration

```
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET QTVIDEO average_data_rate=0
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET QTVIDEO quality=256
```

Profile

It is possible to store all commands that make up the configuration of the clip out channel as a profile in the configuration file. The profile is then applied with the `PROFILE APPLY` command.

Copyright © 2026 Vizrt

Example Apply Stored Profile

```
RENDERER*VIDEO*CLIPOUT*1*OPTION PROFILE APPLY
```

The profile can be edited in the configuration settings of Viz Artist under the section `CHANNELS_CONFIG`.

Copyright © 2026 Vizrt

10.14.4 Clipout Control

For frame accuracy reasons some clip control commands need to be delayed according to the prevailing ringbuffer size. Other commands needed for configuration are executed immediately. Hence one needs to be careful with the timing when sending commands otherwise commands may not be executed in the order they were sent it. For example, a flush followed by configuration and open commands may result in having the flush execute after open command due to a ringbuffer.
Commands that have delayed execution are:

- `CONTROL RECORD`
- `CONTROL CHANGE_DURATION`
- `CONTROL PAUSE`
- `CONTROL FLUSH`

Open the Clip(s)

After configuring (but before the Quicktime options), the clip(s) can be opened for writing. The following commands show how to set and get the clip name:

```
RENDERER*VIDEO*CLIPOUT*n*NAME GET
RENDERER*VIDEO*CLIPOUT*n*NAME SET "clipname" [ TDIR Interval ]
RENDERER*VIDEO*CLIPOUT*n*NAME PLACEHOLDER_SET "clip_base_name"
```

The clipname argument for the `SET` command must be given as absolute path name. The directory separators must be given as forward slashes. The extension should be omitted since it is added automatically, depending on the resolution, container and codec types.

> ⚠ **Note:** Quotes around clipname are mandatory and literal if the TDIR argument is used. The TDIR argument is the interval with which the file is updated regarding the clip length. Using the TDIR feature allows for so called “growing files”. It is currently supported on *.avi*, *.mxf*, *.mov* and *.wav* containers.

- `MATROX_MOV` containers create a separate reference file that is updated according to the given interval. See below for file naming convention.
- `QT_MOV` containers update the quicktime file itself. The interval in seconds must be larger than 1.0s and less than 60.0 seconds. A value of 1.0s or less disables this feature.

The clipname can consist of placeholders as well. The following placeholders are available:

- **<clip_root>:** The clip data directory as set in the configuration file.
- **<clip_name>:** The value that has been set with PLACEHOLDER_SET.
- **<absolute_scene_name>:** The complete name of the scene currently loaded in the render editor.
- **<base_scene_name>:** The last part of the name of the scene currently loaded in the render editor.
- **<absolut_scene2_name>:** The complete name of the scene that is to be loaded into the render editor.
- **<base_scene2_name>:** The last part of the name of the scene that is to be loaded into the render editor.

The resulting filenames, depending on the type of the file, are as follows:

- **video:** Clipname argument + automatic extension.

Copyright © 2026 Vizrt

- **key:** Clipname argument + “_key” + automatic extension.
- **audio:** Clipname argument + “_audio” + automatic extension.
- **vbi:** Clipname argument + “_vbi” + automatic extension.
- **proxy:** Clipname argument + “_proxy” + automatic extension.
- **MATROX_MOV:** Reference file clipname argument + “.Ref” + automatic extension.

Example Open a Clip

```
RENDERER*VIDEO*CLIPOUT*1*NAME SET v:/clips/testclip
```

Record And Close Commands

```
RENDERER*VIDEO*CLIPOUT*n*CONTROL RECORD frames
RENDERER*VIDEO*CLIPOUT*n*CONTROL CHANGE_DURATION frames
RENDERER*VIDEO*CLIPOUT*n*CONTROL PAUSE
RENDERER*VIDEO*CLIPOUT*n*CONTROL FLUSH
```

The `RECORD` command starts recording up to the given number of frames. If the argument is zero it records until a `PAUSE` or `FLUSH` command is sent.

Once the recording has finished, additional `RECORD` commands can be issued. The number of frames to be recorded can be changed during a recording operation with `CHANGE_DURATION`.

In Interlaced resolutions, `RECORD` can only start at a first field. If the command is issued at a second field then this field it is dropped and the recording starts at the next first field.

`FLUSH` closes the clip and resets the clip channel. This command can also be used before a clip has been opened to reset the clip out channel.

Please remember that the flush command is a delayed command that needs to travel through the ringbuffer. Therefore it shall be waited until the command has executed before sending in new commands to set up a new recording.

Example: Record 500 Frames

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL RECORD 500
```

Example: Change the Requested Number of Frames to 300

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL CHANGE_DURATION 300
```

Copyright © 2026 Vizrt

Example: Pause an Ongoing Recording

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL PAUSE
```

Example: Finish and Close

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
```

Query Current Clip Values

```
RENDERER*VIDEO*CLIPOUT*n*CONTROL GET_REQUESTED_DURATION
RENDERER*VIDEO*CLIPOUT*n*CONTROL GET_RECORDED_DURATION
RENDERER*VIDEO*CLIPOUT*n*CONTROL GET_TOTAL_DURATION
```

- `GET_REQUESTED_DURATION` returns the frame argument from the last `RECORD` command.
- `GET_RECORDED_DURATION` returns the number of already recorded frames since the last `RECORD` command.
- `GET_TOTAL_DURATION` returns the total number of recorded frames so far.

Copyright © 2026 Vizrt

10.14.5 Clipout Feedback Channel

The clip out channel has integration into the feedback channel. The keyword is `CLIPOUT` and it has no further arguments.

```
FEEDBACK*COMMAND ADD hostname portnumber CLIPOUT
```

where *hostname* and *portnumber* are to be substituted with your values.

The following example sends feedback on port number `2001` on localhost for clip out events:

```
FEEDBACK*CLIENT ADD localhost 2001
FEEDBACK*COMMAND ADD localhost 2001 CLIPOUT
```

Message Layout Sent over the Feedback Channel

The general layout of a message is `CLIPOUT[n]` followed by a text string. The *n* in the square brackets is the number of the clip out channel. Since at the moment there is only one channel possible, this number is set to zero.

Open Feedback

This message is sent as a reaction to the `NAME SET` command.

Successful Open Feedback Message Layout

```
CLIPOUT [n] OPEN CLIP = 'arg1' VIDEO = arg2 (video_file,key_file,proxy_file)
AUDIO=arg3 (audio_file [ , ...]) VBI = arg4 (vbi_file) TC = arg5
```

Failed Open Feedback Message Layout

```
CLIPOUT [n] OPEN CLIP = 'arg1' VIDEO = arg2 AUDIO = arg3 VBI = arg4 TC = arg5 ERR =
arg
```

- **arg1:** Filename of the clip to be written. It is the base filename (without file extension).
  arg2: Shows whether the clip contains video. Either `0` or `1` followed by the filenames containing video in parenthesis.
  arg3: Shows whether the clip contains audio. Either `0` or `1` followed by the filenames containing audio in parenthesis.
  arg4: Shows whether the clip contains vbi. Either `0` or `1` followed by the filename containing vbi in parenthesis.

Copyright © 2026 Vizrt

arg5: Shows whether the clip contains timecode. Either `0` or `1`.

arg6: Contains the error description in case of an error.

Example: Successful Open Feedback Message Layout

```
CLIPOUT [0] OPEN CLIP = ’ d :/ sd_1 ’
            VIDEO =1( d :/ sd_1 . mxf , d :/ sd_1_key . mxf , )
            AUDIO =1()
            VBI =0()
            TC =0
```

Start Recording Feedback

This message is sent as a reaction to the `CONTROL RECORD` command.

```
CLIPOUT [n] START REQ=arg1
```

- **arg1:** Number of frames that are requested to be recorded.

Example: Record Feedback Message

```
CLIPOUT [0] START REQ=0
```

Change Duration Feedback

This message is sent as a reaction to the `CONTROL CHANGE_DURATION` command.

```
CLIPOUT [n] CHANGE REQ=arg1
```

- **arg1:** Number of frames that are requested to be recorded.

Example: Change Duration Feedback Message

```
CLIPOUT [0] CHANGE REQ=300
```

Ongoing Recording Feedback

This message is sent throughout an ongoing recording after every 64 frames.

Copyright © 2026 Vizrt

Successful Write Feedback Message Layout

```
CLIPOUT [n] WRITE REC=arg1 TOT=arg2
```

Failed Write Feedback Message Layout

```
CLIPOUT [n] WRITE REC=arg1 TOT=arg2 ERR=arg3
```

- **arg1:** Number of frames that are recorded since the last record command.
- **arg2:** Number of total frames recorded.
- **arg3:** Contains the error description in case of an error.

Pause Feedback

This message is sent as a reaction to the `CONTROL PAUSE` command.

```
CLIPOUT [n] STOP CLIP='arg1' REC=arg2 TOT=arg3
```

- **arg1:** Filename of the clip to be written. It is the base filename (without file extension).
- **arg2:** Number of frames that are recorded since the last record command.
- **arg3:** Total frames recorded.

Example

```
CLIPOUT [0] STOP CLIP = 'd:/sd_1' REC=5000 TOT=12000
```

Flush Feedback

This message is sent as a reaction to the `CONTROL FLUSH` command.

```
CLIPOUT [n] FLUSH CLIP = 'arg1' TOT=arg2
```

- **arg1:** Filename of the clip to be written. It is the base filename (without file extension).
- **arg2:** The number of total frames recorded.

Copyright © 2026 Vizrt

10.14.6 Clipout Examples

- Recording
    - Advanced Recording
- Create Clips with Stage Events Example
    - Enable Feedback
    - Cleanup Clip Out and Clip In Channels
    - Cleanup Renderer
    - Set Up Stage Start Event
    - Load Scenes And Set Audio And Video Clips
    - Cue Scenes
    - Load Scenes Into Renderer
    - Reset Animation Timelines
    - Configure Clip Out Channel
    - Start The Animation
    - Close Clip
- Render a Playlist
    - Default Profile
    - Scene Load Interceptor
    - Stage Interceptors
    - Startup Commands
    - Ready to Render The Playlist

Recording

Create a clip with just video. Viz program output resolution is set to 576i25.

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
--> wait until the flush has been executed
RENDERER*VIDEO*CLIPOUT*1*CREATE VIDEO_SET 1
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION VIDEO_SET 0 0
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET AVI
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET DvCPro_411
RENDERER*VIDEO*CLIPOUT*1*NAME SET v:/clips/sample
RENDERER*VIDEO*CLIPOUT*1*CONTROL RECORD 100
 --> wait until resording has finished 10 RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
```

- We start with flushing the channel, `CONTROL FLUSH`, to have a clean state.
- Then we tell the clip channel to create the video file, `VIDEO_SET 1`.
- The clip resolution should match the output resolution, `RESOLUTION VIDEO_SET 0 0`.
- Container is .avi, `CONTAINER VIDEO_SET AVI`.
- And codec is dvcpro411, `CODEC VIDEO_SET DvCPro_411` the codec requires no further options .
- Open the clip with `NAME SET v:/clips/sample`.
- And record 100 frames, `CONTROL RECORD 100` wait until all frames have been recorded.

Copyright © 2026 Vizrt

- Close the clip, `CONTROL FLUSH`.
- The resulting is a clip *v:\clips\sample.avi*.

Advanced Recording

Create a downscaled clip with video, separate key, audio, VBI and proxy. Viz Engine program output resolution is set to 1080i25.

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
--> wait until the flush has been executed
RENDERER*VIDEO*CLIPOUT*1*CREATE VIDEO_SET 1
RENDERER*VIDEO*CLIPOUT*1*CREATE KEY_SET 1
RENDERER*VIDEO*CLIPOUT*1*CREATE AUDIO_SET 1
RENDERER*VIDEO*CLIPOUT*1*CREATE VBI_SET 1
RENDERER*VIDEO*CLIPOUT*1*CREATE PROXY_SET 1
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION VIDEO_SET 2 0
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET XDCAM_MXF
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET 27
RENDERER*VIDEO*CLIPOUT*1*CONTAINER AUDIO_SET W64
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VBI_SET AVI
RENDERER*VIDEO*CLIPOUT*1*CODEC VBI_SET YUYV422
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET IFRAME bitrate=30 forceDCTFrame=0
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AUDIO ch=8 bits=24/32
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET VBI ON
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SETTCstartTC=11:00:00:00
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION PROXY_SET 0 50
RENDERER*VIDEO*CLIPOUT*1*CONTAINER PROXY_SET QT_MOV
RENDERER*VIDEO*CLIPOUT*1*CODEC PROXY_SET YUYV422
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET AUDIO ON
RENDERER*VIDEO*CLIPOUT*1*NAME SET v:\clips\advanced
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET QTCODEC SVQ3
RENDERER*VIDEO*CLIPOUT*1*OPTION PROXY_SET QTVIDEO average_data_rate=0 quality=512
 RENDERER*VIDEO*CLIPOUT*1*CONTROL RECORD 1000
--> wait until finished
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
```

- Start with flushing the channel, `CONTROL FLUSH`, to have a clean state.
- Tell the clip channel to create the video, key, audio, VBI and proxy files:
  - `VIDEO_SET 1`
  - `KEY_SET 1`
  - `AUDIO_SET 1`
  - `VBI_SET 1`
  - `PROXY_SET 1`
- The resolution should be SD 16:9, `RESOLUTION VIDEO_SET 2 0`.
- The video and key files should be D10 files.
  - `CONTAINER VIDEO_SET XDCAM_MXF`
  - `CODEC VIDEO_SET 27`

Copyright © 2026 Vizrt

- Since we want separate audio we need to specify the audio container, `CONTAINER AUDIO_SET W64`.
- Same applies for VBI:
  - `CONTAINER VBI_SET AVI`
  - `CODEC VBI_SET YUYV422`
- Set the D10 Iframe options, `OPTION VIDEO_SET IFRAME bitrate=30 forceDCTFrame=0`.
- Audio configuration, `OPTION VIDEO_SET AUDIO ch=8 bits=24/32`.
- And VBI configuration, `OPTION VIDEO_SET VBI ON`.
- The MXF file allows to carry time code information, `OPTION VIDEO_SETTCstartTC=11:00:00:00`.

This finishes setup of the hires part, next is the proxy configuration.

- The proxy should be scaled to 50%, `RESOLUTION PROXY_SET 0 50`.
- Container is QT mov, `CONTAINER PROXY_SET QT_MOV`.
- Codec needs to be set to uncompressed, `CODEC PROXY_SET YUYV422`.
- Audio should be included in the proxy, too, `OPTION PROXY_SET AUDIO ON`.

Before we can setup the proxy QT options we need to open the clips:

- Open the clip, `NAME SET v:/clips/advanced`.
- Then set the proxy QT codec, `OPTION PROXY_SET QTCODEC SVQ3`.
- And the proxy QT options, `OPTION PROXY_SET QTVIDEO average_data_rate=0 quality=512`.
- Record some frames, `CONTROL RECORD 1000` and wait until they are recorded.
- Close the clip, `CONTROL FLUSH`.
  The resulting clips are:
  - *v:\clips\advanced.d10.mxf* (SD 576i with fill)
  - *v:\clips\advanced_key.d10.mxf* (SD 576i with key)
  - *v:\clips\advanced_vbi.avi* (HD1080i with VBI)
  - *v:\clips\advanced_proxy.mov* (proxy with video and audio)

Create Clips with Stage Events Example

This example creates a reference clip and uses stage events to start the recording. It contains various counters and a timecode. It has the capability for audio and video as well. Let's take a look at each step involved first.

Enable Feedback

```
FEEDBACK*CLIENT ADD localhost 2001
FEEDBACK*COMMAND ADD localhost 2001 CLIPOUT
```

At first we enable the feedback channel to send UDP packets to localhost. Now we are ready to initialize the clip channels and the renderer:

Copyright © 2026 Vizrt

Cleanup Clip Out and Clip In Channels

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
RENDERER*BACK_LAYER*STAGE STOP
RENDERER*MAIN_LAYER*STAGE STOP
RENDERER*FRONT_LAYER*STAGE STOP
RENDERER*VIDEO*CLIPIN*1*CONTROL FLUSH
```

Flush any clips in the clip in and clip out channel. Since the flush command needs to travel through the ringbuffer we need to wait a few frames until they have been executed.

Cleanup Renderer

```
RENDERER*BACK_LAYER SET_OBJECT
RENDERER*MAIN_LAYER SET_OBJECT
RENDERER*FRONT_LAYER SET_OBJECT
POOLS CLEANUP
```

Unload scenes from the renderer and cleanup all pools. We want to start with a fresh environment here.

Set Up Stage Start Event

```
STAGE_EVENTS*STAGE_START CLEAR
STAGE_EVENTS*STAGE_START ADD "RENDERER*VIDEO*CLIPOUT*1*CONTROL RECORD 5000" -1
STAGE_EVENTS*STAGE_START GET
```

Enable the Stage Start event to trigger the record command on the clip out channel.

Load Scenes And Set Audio And Video Clips

```
<! - - load background scene -->
SCENE*Vizrt_Research_Developement/Reference/TC/bg LOAD
SCENE*Vizrt_Research_Developement/Reference/TC/
bg*STAGE*DIRECTOR*Director*KEY*$audioclip*CLIP SET "<11EBC981-11A3-45C7-
A78545119738FC68>"
SCENE*Vizrt_Research_Developement/Reference/TC/bg*VIDEO*CLIPIN*1*TARGET SET DVE
SCENE*Vizrt_Research_Developement/Reference/TC/
bg*STAGE*DIRECTOR*Director*KEY*$videoclip*CLIPNAME SET "e:/out/1080_bgclip.m4v"
SCENE*Vizrt_Research_Developement/Reference/TC/
bg*STAGE*DIRECTOR*Director*KEY*$videoclip*START SET "00:00:00:00"
SCENE*Vizrt_Research_Developement/Reference/TC/TC_DISPLAY LOAD
```

This loads the background scene, and sets the audio clip and the video clip. The scene for the main layer is also loaded here.

Copyright © 2026 Vizrt

Cue Scenes

```
SCENE*Vizrt_Research_Developement/Reference/TC/bg CUE
SCENE*Vizrt_Research_Developement/Reference/TC/TC_DISPLAY CUE
```

Both scenes are cued. This is required for frame accurate clip playback.

Load Scenes Into Renderer

```
RENDERER*BACK_LAYER SET_OBJECT Vizrt_Research_Developement/Reference/TC/bg
RENDERER*MAIN_LAYER SET_OBJECT Vizrt_Research_Developement/Reference/TC/TC_DISPLAY
```

The scenes are loaded into the back layer and main layer.

Reset Animation Timelines

```
RENDERER*BACK_LAYER*STAGE SHOW 0.0
RENDERER*MAIN_LAYER*STAGE SHOW 0.0
```

The timelines for back and main layers are reset to zero. Now we are ready to configure the clip out channel.

Configure Clip Out Channel

```
RENDERER*MAIN_LAYER*TREE*$UserData$Label*GEOM*TEXT SET 1080i XDCAM_MXF422, Audio=8 ch
RENDERER*VIDEO*CLIPOUT*1*CREATE VIDEO_SET On
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION VIDEO_SET 0 0
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET XDCAM_MXF
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET 22
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AUDIO ch=8
RENDERER*VIDEO*CLIPOUT*1*NAME SET "d:/1080i_3in" 5.0
```

The first line sets a text container in the main layer with some descriptive information. The following lines configure the clip out channel and on the last line open the clip.

Start The Animation

```
RENDERER*BACK_LAYER*STAGE START
RENDERER*MAIN_LAYER*STAGE START
```

This starts the animation on both layers. The stage start event trigger fires and the recording starts.

Copyright © 2026 Vizrt

Close Clip

After the recording of the predetermined number of frames has been finished (can be observed on the feedback channel) we close the clip:

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
```

We have now recorded a clip with 5000 frames in XDCAMHD422 format in an MXF container.

Render a Playlist

Recent additions to Viz Artist ease rendering a complete playlist into separate clips. This example guides through those new features. First we need to prepare some steps in the configuration file.

Default Profile

First, we create default profile that we can load with a single command. In This profile we make use of the placeholders for the name of the clip. This is under the section `CHANNELS_CONFIG` in the configuration file.

```
# the <self> placeholder gets replaced with the proper RENDERER*VIDEO*CLIPOUT*1
ClipOut1.profile.default_1=<self>*CREATE VIDEO_SET On
ClipOut1.profile.default_2=<self>*RESOLUTION VIDEO_SET 0 0
ClipOut1.profile.default_3=<self>*CONTAINER VIDEO_SET XDCAM_MXF
ClipOut1.profile.default_4=<self>*CODEC VIDEO_SET 22*
ClipOut1.profile.default_5=<self>*OPTION VIDEO_SET AUDIO ch=8
ClipOut1.profile.default_6=<self>*OPTION VIDEO_SET TC startTC=10:00:00:00
# we make use of three placeholders.one is the clip root.the others are
# for the scene name to be set and clip placeholder name itself
# that gets set with RENDERER*VIDEO*CLIPOUT*1*NAME PLACEHOLDER_SET
ClipOut1.profile.default_7=<self>*NAME SET <clip_root> playlist/<base_scene2_name>/
<clip_name>
```

Scene Load Interceptor

The profile that we created in the previous section should always be applied when we change the scene. To be more specific we want it to be applied right before when a scene gets loaded into the render editor but before it is actually loaded. We make use of the command injector on the scene load interceptor on the `MAIN_LAYER`.

```
RENDERER*MAIN_LAYER CLEAR_INJECT_BEFORE
RENDERER*MAIN_LAYER ADD_INJECT_BEFORE "RENDERER*VIDEO*CLIPOUT*1*PROFILE APPLY" -1
```

Copyright © 2026 Vizrt

Stage Interceptors

The Stage events have been already introduced above. Here we make use of them again but this time bound only onto the `MAIN_LAYER`. When the animation starts then an infinite record command is set to the clip out channel. When the animation stops the clip is flushed and hence properly closed.

```
STAGE_EVENTS*LAYER_STAGE_START CLEAR MAIN_LAYER
STAGE_EVENTS*LAYER_STAGE_STOP CLEAR MAIN_LAYER
STAGE_EVENTS*LAYER_STAGE_START ADD MAIN_LAYER "RENDERER*VIDEO*CLIPOUT*1*CONTROL
RECORD 0" -1
STAGE_EVENTS*LAYER_STAGE_STOP ADD MAIN_LAYER "RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH
" -1
```

Startup Commands

The Interceptors are not part of a scene or the configuration file. They need to be applied once. So we just bundle them together in the startup commands section together with the activation of the feedback channel. The feedback channel commands make only sense if the controlling application resides on the localhost. Otherwise, replace localhost with the name of the host where the UDP feedback should be directed at or leave them out of the startup commands.

```
# scene load interceptor
StartupCommands_1= RENDERER*MAIN_LAYER CLEAR_INJECT_BEFORE
StartupCommands_2= RENDERER*MAIN_LAYER ADD_INJECT_BEFORE
"RENDERER*VIDEO*CLIPOUT*1*PROFILE APPLY" -1
# stage main layer interceptor
StartupCommands_3= STAGE_EVENTS*LAYER_STAGE_START CLEAR MAIN_LAYER
StartupCommands_4= STAGE_EVENTS*LAYER_STAGE_STOP CLEAR MAIN_LAYER
StartupCommands_5= STAGE_EVENTS*LAYER_STAGE_START ADD MAIN_LAYER
"RENDERER*VIDEO*CLIPOUT*1*CONTROL RECORD 0" -1
StartupCommands_6= STAGE_EVENTS*LAYER_STAGE_STOP ADD MAIN_LAYER
"RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH " -1
# feedback channel
StartupCommands_7= FEEDBACK*CLIENT ADD localhost 2001
StartupCommands_8= FEEDBACK*COMMAND ADD localhost 2001 CLIPOUT
```

Ready to Render The Playlist

Load the scenes into Viz Engine and cue them.

```
SCENE*Vizrt_RD*/tmp/n1 LOAD
SCENE*Vizrt_RD*/tmp/n1 CUE
SCENE*Vizrt_RD*/tmp/n1*STAGE SHOW 0.0
SCENE*Vizrt_RD*/tmp/n2 LOAD
SCENE*Vizrt_RD*/tmp/n2 CUE
SCENE*Vizrt_RD*/tmp/n2*STAGE SHOW 0.0
SCENE*Vizrt_RD*/tmp/n3 LOAD
```

Copyright © 2026 Vizrt

```
SCENE*Vizrt_RD*/tmp/n3 CUE
SCENE*Vizrt_RD*/tmp/n3*STAGE SHOW 0.0
```

After that the scenes are ready to be put into the render editor. Now let's set the placeholder for the clipname and render the first clip.

```
RENDERER*VIDEO*CLIPOUT*1*NAME PLACEHOLDER_SET "heidi_1"
SCENE*Vizrt_RD/tmp/n1*STAGE SHOW 0.0
RENDERER*MAIN_LAYER SET_OBJECT Vizrt_RD/tmp/n1
```

This triggers the scene load command injector that executes the `RENDERER*VIDEO*CLIPOUT*1*PROFILE APPLY` which in turn loads the profile and applies it to the clip out channel.

```
RENDERER*VIDEO*CLIPOUT*1*CREATE VIDEO_SET On
RENDERER*VIDEO*CLIPOUT*1*RESOLUTION VIDEO_SET 0 0
RENDERER*VIDEO*CLIPOUT*1*CONTAINER VIDEO_SET XDCAM_MXF
RENDERER*VIDEO*CLIPOUT*1*CODEC VIDEO_SET 22
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET AUDIO ch=8
RENDERER*VIDEO*CLIPOUT*1*OPTION VIDEO_SET TC startTC=10:00:00:00
RENDERER*VIDEO*CLIPOUT*1*NAME SET <clip_root>playlist/<base_scene2_name>/<clip_name>
```

assuming that is set *v:/clips/out/*, then the clip name is expanded to *v:/clips/out/playlist/n1/heidi_1.xdcamHD422.mxf*. The next step is to start the animation. This triggers the stage start command injector.

```
RENDERER*MAIN_LAYER*STAGE START
```

Which, in turn, triggers the stage start command injector.

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL RECORD 0
```

Now the controlling application watches the feedback channel it subscribed to before:

```
CLIPOUT [0] OPEN CLIP='v:/clips/out/playlist/n1/heidi_1' VIDEO=1 (v:/clips/out/
playlist/n1/heidi_1.xdcamHD422.mxf,,) AUDIO=1 () VBI=0 () TC=0
CLIPOUT [0] START REQ=0
CLIPOUT [0] WRITE REC=64 TOT=64
CLIPOUT [0] WRITE REC=128 TOT=128
CLIPOUT [0] WRITE REC=192 TOT=192
CLIPOUT [0] WRITE REC=256 TOT=256
CLIPOUT [0] WRITE REC=320 TOT=320
CLIPOUT [0] WRITE REC=384 TOT=384
CLIPOUT [0] WRITE REC=448 TOT=448
CLIPOUT [0] WRITE REC=512 TOT=512
CLIPOUT [0] STOP CLIP= 'v:/clips/out/playlist/n1/heidi_1' REC=550 TOT=550 CLIPOUT [0]
FLUSH CLIP='v:/clips/out/playlist/n1/heidi_1' TOT=550
```

Copyright © 2026 Vizrt

as we can see in the feedback messages there are also `STOP` and `FLUSH` commands. Once the animation stops the stage stop command injector kicks in.

```
RENDERER*VIDEO*CLIPOUT*1*CONTROL FLUSH 0
```

This command stops and flushes the clip. Now we can repeat with rendering the next clip until the playlist is completed.

11 SMPTE 2110 Information

The following chapters include an introduction into SMPTE 2110 standard as well as some common practices, guides and troubleshooting sections:

- General Information and Introduction
- Configuring a Static Network
- Working with NMOS
- Troubleshooting SMPTE ST2110 and NMOS
- Explicit Usage of the Second SFP Pair on Matrox X.mio5 Q25 and DSX LE5 Q25

Copyright © 2026 Vizrt

11.1 General Information and Introduction

11.1.1 General Description

From a pure Viz Engine viewpoint, the main standards we have to deal with are SMPTE ST 2110-20 (video), -30 (audio) and -40 (ancillary data) as well as SMPTE ST 2059-2 (synchronization profile). Those elements are transported in the essence network. If NMOS is also part of the infrastructure, a separate management network might be available. In addition to the standards mentioned before, other IT infrastructure elements, like DNS and DHCP, could be available.

A typical SMPTE ST 2110 network consists of various endpoint devices connected to one or more switches. The switches might be arranged in a spine-leaf architecture, with high bandwidth backbone switches and lower bandwidth leaf switches where the devices connect to. Additionally, one or more PTP clocks are required for synchronization. Having DHCP and/or DNS servers running in your network could increase manageability.

In NMOS controlled networks, an RDS and a Network Orchestrator/Controller might be part of the infrastructure as well, although not all NMOS Controllers need a registration server, as they are sending RESTful calls directly to the devices.

Unlike SDI (Serial Digital Interface), which carries video, audio,
and ancillary data as a single multiplexed signal over coaxial cable, ST 2110
separates each media essence into independent IP flows:

- **Video** — transported as uncompressed RTP streams
- **Audio** — transported independently from video
- **Ancillary data** — metadata, subtitles, timecode, etc.

This separation allows each essence to be routed, processed, and switched independently, enabling flexible and scalable broadcast infrastructures.

A typical ST 2110 network consists of:

- **Endpoint devices** (Viz Engine, cameras, switchers, etc.) connected to switches
- **Switches** — often arranged in a **spine-leaf architecture**:
    - Leaf switches: where devices connect (lower bandwidth)
    - Spine switches: backbone interconnects (high bandwidth)
- **PTP grandmaster clocks** for synchronization
- Optionally: DHCP, DNS, and NMOS servers for management

There are typically two distinct network segments:

Network	Purpose
Essence network	Carries the actual media flows (video, audio, ancillary)
Management network	Carries NMOS control traffic, PTP (if out-of-band), DNS, DHCP

Copyright © 2026 Vizrt

Multicast and IGMP

ST 2110 flows use **IP multicast**. Receivers subscribe to multicast groups using **IGMP** (Internet Group Management Protocol). The network switch must be configured to support IGMP snooping to avoid flooding all ports with all flows.

- **IGMPv2** — basic join/leave
- **IGMPv3** — adds source-specific multicast (SSM) filtering

Synchronization (PTP)

All devices in a ST 2110 network must be synchronized to a common clock using **PTP (Precision Time Protocol)**, specifically the **SMPTE ST 2059-2** profile.

Key concepts:

- **PTP Grandmaster** — the authoritative clock source
- **BMCA** (Best Master Clock Algorithm) — determines which clock becomes grandmaster
- **Clock Domain** — a logical grouping; all devices in the same domain sync together
- **In-band PTP** — PTP traffic shares the essence network
- **Out-of-band PTP** — PTP traffic uses a separate management network

The Matrox video boards in Viz Engine connect to PTP via their SFP ports. The PTP settings (domain number, delay mechanism, IP mode) are configured in `ipconfig.xml`.

11.1.2 Standards

SMPTE ST 2110

Standard	Definition
2110-20	Uncompressed video flows.
2110-21	Defines traffic shaping and delivery timing. Defines terms like Narrow Sender/Receiver and Wide Sender/Receiver.
2110-22	Constant Bit-Rate Compressed video flows.
2110-30	Uncompressed audio flows.
2110-31	Compressed audio flows according to AES3.

Copyright © 2026 Vizrt

Standard	Definition
2110-40	Ancillary data flows.
2110-41	Fast metadata flows (WIP).
2059-2	PTP synchronization profile for broadcast media

Redundancy (ST 2022-7)

ST 2110 supports seamless redundancy via ST 2022-7, where each flow is sent on two independent paths. If one path fails, the receiver reconstructs from the other.
Enabled per-output in `ipconfig.xml` via the `Redundancy` setting.

AMWA NMOS

Id	Name	Description
IS-04	Discovery & Registration	Allows control and monitoring applications to find the resources on a network.
IS-05	Device Connection Management	Provides a transport-independent way of connecting Media Nodes.
IS-06	Network Control	Lets broadcast control applications manage what happens on the network itself.
IS-07	Event & Tally	Provides an IP-friendly mechanism to carry time-sensitive information, like tally or event data. Is similar to GPIO in a traditional broadcast environment.
IS-08	Audio Channel Mapping	Allows channel-level operations within NMOS environments like channel muting, language swapping,...
IS-09	System Parameters	Allows an NMOS Node to obtain global configuration parameters that are common across the system.
IS-10	Authorization	Allows an API server to accept or reject requests depending on what a client is authorized to do.

Copyright © 2026 Vizrt

Id	Name	Description
IS-11	Sink Metadata Processing	Provides information about physical devices called Sinks associated with Receivers. Allows to configure media parameters of Senders based on information about Sinks.
MS-04	ID & Timing Model	Documents a model for identity and timing that applies to AMWA NMOS specifications that apply to content
MS-05-01	NMOS Control Architecture Framework	Defines a system for modelling various types of devices which interrelates with existing NMOS Specifications and the JT-NM Reference Architecture.
BCP-002-01	Natural Grouping	Documents best practice and recommendations for how to indicate and handle “Natural Groups” of Resources in AMWA NMOS APIs.
BCP-003-01	Secure Communications in NMOS Systems	Documents best practice for using secure transport for NMOS API communications.
BCP-003-02	Authorization in NMOS Systems	Documents best practice for an API server to accept or reject requests depending on what a client is authorized to do.
BCP-003-03	Certificate Provisioning in NMOS Systems	Documents best practice for automated provisioning of TLS Server Certificates to NMOS APIs.
BCP-004-01	Receiver Capabilities	Allows an IS-04 Receiver to express parametric constraints on the types of streams that it is capable of consuming.
BCP-005-01	EDID to Receiver Capabilities Mapping	Provides a scheme and guidelines for expressing EDID information via Receiver Capabilities.

See Networked Media Open Specifications for more details.

Viz Engine Configuration

All IP-related settings are stored in:

```
%ProgramData%\\Vizrt\\VizEngine\\ipconfig.xml
```

⚠ Never delete XML tags — only edit values between tags.
Never change `name` attributes (e.g. `<SFP name="SFP A">`).

Copyright © 2026 Vizrt

SFP Settings (per port)

- `IPv4Address`, `IPv4Netmask`, `IPv4Gateway` — static PTP address
- `DhcpEnable` — X.mio5 only; enables DHCP for PTP
- `MasterClockDomainNumber` — must match your PTP grandmaster
- `JoinType` — IGMP version (2=IGMPv2, 3=IGMPv3)
- `DelayMechanism` — EndToEnd (1) or PeerToPeer (2)

Flow Settings (per input/output)

- `EnableFlow` — enables the flow at startup
- `DstAddress` / `DstUdpPort` — multicast destination
- `SrcUdpPort` — source port (outputs only)
- `AudioPacketDuration` — 125µs (1) or 1ms (4)

> On X.mio5, each SFP has **two separate MAC addresses** and requires **two IP addresses**: one for PTP (in ipconfig.xml) and one for the Windows NIC.

11.1.3 Definition of Terms

Term	Definition/Explanation
SMPTE ST 2110	A series of specifications covering various aspects of transporting media content over IP, as well as describing the related infrastructure topics.
AMWA NMOS	Advanced Media Workflow Association Networked Media Open Specification A family of specifications that enable interoperability between media devices on an IP infrastructure.
IGMP	The Internet Group Management Protocol is a communication protocol used by hosts and adjacent routers on IPv4 networks to establish multicast group memberships. It is an integral part of IP multicast and allows the network to direct multicast transmissions only to hosts that have requested them.
SDP	The Session Description Protocol is a format for describing multimedia communication sessions. NMOS IS-05 uses the SDP to describe the content and format of the sender flow, to which a receiver should connect to.

Copyright © 2026 Vizrt

Term	Definition/Explanation
SFP	A Small Form-Factor Pluggable is a compact, hot-pluggable network interface module used together with a Fiber-Optics cable to connect SMPTE ST 2110 devices to a switch. 
AOC	Active Optical Cable have the optical electronics already connected, eliminating the connectors between the cable and the optical module. 
DAC	Direct Attach Copper use twin-axial copper cables directly attached to the SFP.
FEC	Forward Error Correction is a technique used for controlling errors in data transmission.
PTP	Precision Time Protocol is a protocol used to synchronize clocks throughout a computer network. For broadcast media systems the PTP profile defined in SMPTE 2059-2 is required.
BMCA	The Best Master Clock Algorithm is a method to determine the best clock based on a series of properties.
In-Band	Describes the method to send management data on the same network as the media data.
Out-of-Band	Describes the method to send management data over a network separated from the media data network.
DHCP	The Dynamic Host Configuration Protocol is a network management protocol used for automatically assigning IP addresses.

Copyright © 2026 Vizrt

Term	Definition/Explanation
DNS	The Domain Name System is a hierarchical and decentralized naming system for computers, services, or other resources connected to the network.
DNS-SD	DNS-based Service Discovery allows clients to discover a named list of service instances and to resolve those services to hostnames using standard DNS queries.
RDS	A Registration & Discovery Server is used to register Nodes, Devices, Senders and Receivers in a centralized place using NMOS IS-04.
SDN	Software-defined networking technology is an approach to network management that enables dynamic and efficient configuration.
REST	Representational state transfer defines a set of constraints for how the architecture of a big network infrastructure should behave. A web API that obeys the REST constraints is informally described as RESTful and is typically loosely based on HTTP methods to access resources via URL-encoded parameters and the use of JSON or XML to transmit data.
NIC	A Network Interface Controller is a computer hardware component that connects a computer to a computer network.
MAC address	A Media Access Control address is a unique identifier assigned to a NIC for use as a network address in communications within a network segment.

SFP and Cabling

Type	Description
SFP	Small Form-Factor Pluggable — hot-pluggable optical transceiver
AOC	Active Optical Cable — integrated optics, no separate SFP
DAC	Direct Attach Copper — twin-axial copper, short reach

Both sides of a connection must use SFPs with the same nominal transfer rate and range classification (Short Reach / Long Reach).

Copyright © 2026 Vizrt

11.2 Configuring a Static Network

When talking about static IP networks for SMPTE ST 2110, we need to distinguish between static TCP/IP addresses for the ports and static multicast address assignment for the single flows.

Network	Purpose
Essence network	Carries the actual media flows (video, audio, ancillary)
Management network	Carries NMOS control traffic, PTP (if out-of-band), DNS, DHCP

Caution: Plan your network segments carefully. Static environments have a high risk of IP address conflicts.

11.2.1 Static TCP/IP Address Assignment

If no DHCP server is available for your essence and management networks, IP address, subnet mask and gateway address need to be assigned statically in the various network settings. Proper planning of the network segments is essential to minimize the risk of conflicts. However, the danger of incorrectly assigned addresses is generally high in such an environment.

Static Configuration of Matrox IP Video Boards

Matrox X.mio3 IP does not support DHCP and IP address, subnet mask and gateway for the PTP network segment need to be configured in *ipconfig.xml* located in *\%ProgramData%\vizrt\VizEngine*.

Matrox X.mio5 IP supports DHCP for both the PTP network (*ipconfig.xml*) and the Windows network stack. In case of static configuration, *DhcpEnable* needs to be set to false and IP address, subnet mask and gateway must be set manually for every used SFP section in *ipconfig.xml*. Additionally, every Matrox related NIC must be configured statically in the Windows network configuration section.

Important: Every port (SFP) of a Matrox X.mio5 has two different MAC addresses and thus **must** have two separate IP addresses for each, Windows NIC and PTP settings.

11.2.2 Static Multicast Address Assignment

Some of the routers/switches available for SMPTE ST 2110 infrastructure still work like a classical SDI router, and rely on static assignment of the multicast addresses of the different senders and receivers. The switching is done by the router and not by subscribing to different sender multicast addresses.

In such an environment, the multicast addresses and ports for every sender and receiver must be configured in the *ipconfig.xml*. The *EnableFlow* flag must also be set to true in that case.

It is recommended to turn on AutoRecovery also, to make sure all changes made during run-time are written back to *ipconfig.xml*.

Copyright © 2026 Vizrt

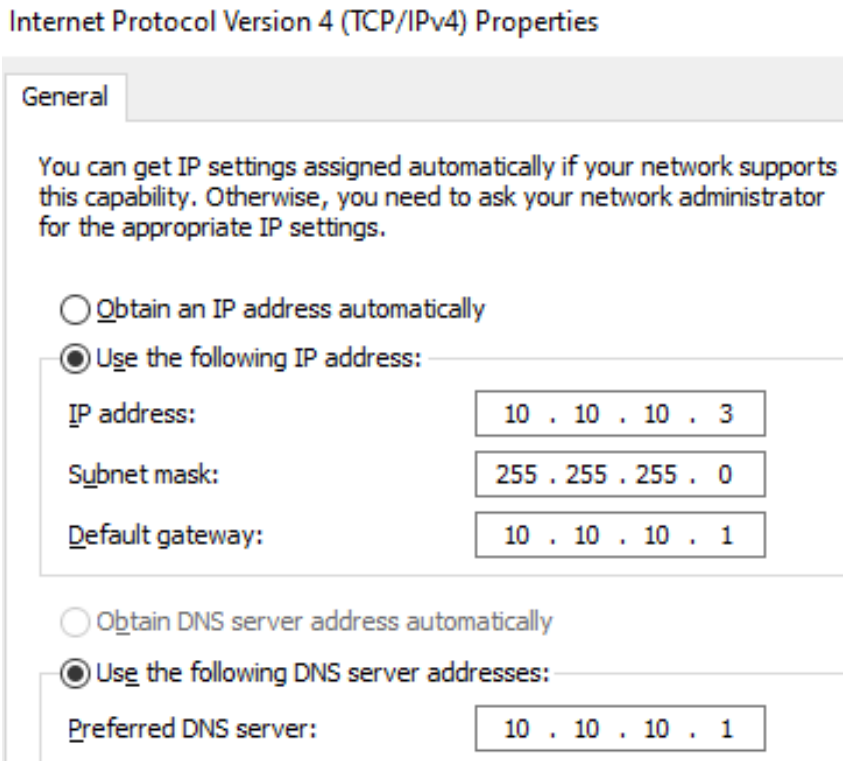
Examples

Static config in *ipconfig.xml:*

```
<SFP name="SFP A">
    <IPv4Address>10.10.10.2</IPv4Address>
    <IPV4Gateway>10.10.10.1</IPV4Gateway>
    <IPv4Netmask>255.255.255.0</IPv4Netmask>
    <Smpte2059SfpSettings>
        <DhcpEnable>false</DhcpEnable>

<Video name="IP video IN 1">
    <EnableFlow>true</EnableFlow>
    <RtpPayloadId>96</RtpPayloadId>
    <GroupHint>IP In Channel 1</GroupHint>
    <Label>IP Video In 1</Label>
    <Description>IP Video In 1</Description>
    <Primary>
        <DstAddress>239.9.20.1</DstAddress>
        <DstUdpPort>50020</DstUdpPort>
```

Static config in Windows network settings:



Copyright © 2026 Vizrt

11.3 Working with NMOS

11.3.1 What is NMOS

The Networked Media Open Specifications provide an open and simple to use control - plane solution that enables interoperability and management of IP connected audio and video devices. They let you find, connect and configure media devices to enable video and audio on your IP network. NMOS is open, industry developed, free of charge and available to everyone. Initially, it was designed to support SMPTE ST 2110, with the possibility for other transport standards as well. NMOS is a fundamental part of the JT - NM technical recommendation, JT - NM TR - 1001, enabling operational efficiency through automation and by reduction of manual overhead in setting up networked systems.

11.3.2 A Typical NMOS Environment

The central part in a typical NMOS setup is the Registration & Discovery Server. Every node, device, sender and receiver is registered there and can be discovered using NMOS IS-04. NMOS controllers/orchestrators can then be used to create NMOS IS-05 requests by use of SDPs to tell a receiver to subscribe to a certain sender's multicast address. The NMOS controller can also be used to assign multicast addresses to the senders, which allows centralized management of the multicast address/port assignment.

11.3.3 How to Configure NMOS on Viz Engine Systems

Settings Related to the Matrox Driver

By default, NMOS control is turned off. To enable it, a file which is part of the Matrox driver installation, needs to be edited. The file can be found in *%PROGRAMFILES%\Matrox DSX-TopologyUtils\System64*. Its name is the serial number of your Matrox board followed by the extension *.json* (for example, *A654321.json*).

The following value must be set to true as shown:

```
"enabled": true,
```

Additionally, the registration server must be configured in case DNS-SD is not enabled (example):

```
"fallback registration server": {
"api version": "auto",
"host name": "10.1.1.127",
"port": 50000
},
```

DNS service discovery can be enabled or disabled using that flag:

```
"use service discovery": true,
```

Copyright © 2026 Vizrt

All other values are either not relevant for Viz Engine setups yet, or are set by Viz Engine when started for the first time.

Settings Related to Viz Engine

Certain settings in *ipconfig.xml* are related to NMOS, and can be edited directly in that file. If they are not changed, default values are being used. The following values are used for registering the node and device (examples):

```
<!--Name of the NMOS IS-04 node to be used.-->
<NmosNode>Vizrt Node on hostname</NmosNode>
<!--Name of the NMOS IS-04 device to be used.-->
<NmosDevice>Vizrt Device on hostname</NmosDevice>
```

Every sender and receiver now uses two values to identify it at the registration server (examples):

```
<Label>IP Video Out 1</Label>
<Description>IP Video Out 1</Description>
```

json etc..

Copyright © 2026 Vizrt

11.4 Troubleshooting SMPTE ST2110 and NMOS

11.4.1 Known Issues

Issue	Solution
Audio does not work anymore after upgrade to Viz Engine 4.3.0 (or higher)	Viz Engine 4.3.0 introduces Group Hints to group together video, audio and ancillary data flows. Flows with the same group hint create a channel based on the hint of the video flow. If no audio flow uses the same group hint, no audio connector will be created. **Advance notice:** The next major version of Viz Engine will introduce Group Hints which are conformant with NMOS specifications. Free of choice group hints can still be used, but they cannot be sent to registration servers.
NMOS does not work anymore after a Matrox driver upgrade	Matrox changed the structure of the *serial.json* in their latest driver versions. Please verify all the settings in this file, especially the *enabled* setting.
My static network settings are changed after a driver upgrade	Unfortunately uninstalling the Matrox driver deletes all settings of the NICs created for X.mio5. This is expected, as Windows is removing all related devices when the drivers are removed. Please check and reapply the IP settings for the Matrox ethernet adapters after installing the new driver. **Advance notice:** Future Matrox drivers will try to preserve the Windows network settings if a static configuration is used.

11.4.2 SMPTE ST 2110

Issue	Possible Solution
Matrox board doesn't lock to Blackburst	• X.mio3: make sure the cable is properly connected and that a stable sync signal is available. • X.mio5: although early X.mio5 boards still have a genlock connector, Blackburst should not be used with this boards, as the driver doesn't support proper locking anymore.

Copyright © 2026 Vizrt

Issue	Possible Solution
Matrox board doesn't lock to PTP	• Make sure SFPs and fiber optics are properly installed and attached. The status of the SFPs can be checked in X.info. • Make sure that you have a valid link. Green LEDs on the board and on the switch show a correct link. At least SFP A needs to be used. • Make sure that both sides of the connection use SFPs with the same nominal transfer rate, and the same range classification. • Make sure you use the correct combination of cable and SFPs - Short Reach or Long Reach. • Make sure the correct IGMP join type is set in the IP config file. This needs to match the settings on the IP switch. • Make sure that the correct IP address, gateway if necessary, and subnet mask is set in the IP config file, if you don't use DHCP, or that DHCP is enabled in the IP config file. • Make sure the master clock domain number is set correctly in the IP config file. • Make sure the ports on the IP switch are configured to route PTP, or that the switch has PTP enabled at all.
Matrox board doesn't receive flows on the inputs	• Make sure the flow is enabled. You can use MAIN*CONFIGURATION*CHANNELS*LIVEIN_n*ENABLEDSTATE GET to verify. • Make sure destination address and UDP port are set correctly and match the sender. You can use MAIN*CONFIGURATION*CHANNELS*LIVEIN_n*DSTADDRESS GET and ...*DSTUDPPORT GET to verify. • On X.mio3 make sure the sender is a narrow sender. • For audio on X.mio5 make sure the configuration matches exactly the flow settings - number of channels, audio packet duration. • Make sure the correct IGMP join type is set in the IP config file. This needs to match the settings on the IP switch. • Make sure SFPs and fiber optics are properly installed and attached. The status of the SFPs can be checked in X.info. • Make sure that you have a valid link. Green LEDs on the board and on the switch show a correct link. At least SFP A needs to be used. • Make sure that both sides of the connection use SFPs with the same nominal transfer rate, and the same range classification. • Make sure you use the correct combination of cable and SFPs - Short Reach or Long Reach. • Make sure you have a valid SDI/IP input license.

Copyright © 2026 Vizrt

Issue	Possible Solution
Output of Matrox board cannot be received on other devices	• Make sure the flow is enabled. You can use MAIN*CONFIGURATION*MATROX*VIDEOOUT_n*ENABLEDSTATE GET to verify. • Make sure you have configured a valid source address on your SFPs and UDP port on your outputs. You can use MAIN*CONFIGURATION*MATROX*VIDEOOUT_n*SRCADDRESS GET and ...*SRCUDPPORT GET to verify. • Make sure you have configured valid destination multicast addresses and ports for your outputs. You can use MAIN*CONFIGURATION*MATROX*VIDEOOUT_n*DSTADDRESS GET and ...*DSTUDPPORT GET to verify. • Make sure SFPs and fiber optics are properly installed and attached. The status of the SFPs can be checked in X.info. • Make sure that you have a valid link. Green LEDs on the board and on the switch show a correct link. At least SFP A needs to be used. • Make sure that both sides of the connection use SFPs with the same nominal transfer rate, and the same range classification. • Make sure you use the correct combination of cable and SFPs - Short Reach or Long Reach. • Make sure you have a valid SDI/IP output license.
Matrox board doesn't work at all, when using UHD resolution	• X.mio3: UHD is not supported on X.mio3 • X.mio5: Make sure to use SFPs with a nominal transfer rate of 25GbE on both your switch and the video board. The switch must support 25GbE transfer rates as well.
Matrox has trouble detecting SFP when switching between 10GbE and 25GbE SFPs	• Make sure you do a restart or a cold reboot.

Copyright © 2026 Vizrt

11.4.3 NMOS

Issue	Possible Solution
Neither the node nor the device shows up in the registry	• Check, if NMOS is enabled in the JSON file associated with that device. The file *serialnumber*.json can be found in *%ProgramFiles%\Matrox DSX-TopologyUtils\System64*. • Make sure to set either *use service discovery* to true or to enter a valid *registration server IP* and *registration server port*. • Verify that your device can be registered by opening http://localhost:port/x-nmos/node/v1.3/devices/. The *port* is also defined in the JSON with *control port*.
Neither senders nor receivers show up in the registry	• Senders and receivers are only registered when they are used. Start Viz Engine to register all used outputs and inputs. Senders and receivers will be unregistered when Viz Engine is closed. • Verify that your senders/outputs can be registered by opening *http://localhost:port/x-nmos/node/v1.3/senders/*. *port* is also defined in the JSON with *control port*. • Verify that your receivers/inputs can be registered by opening *http://localhost:port/x-nmos/node/v1.3/receivers/*. *port* is also defined in the JSON with *control port*.
NMOS IS-05 connection requests are ignored	• Check Viz Engine console for error messages. • Requests are only accepted, if the incoming flow matches the configuration in regard to resolution, frame rate and scan mode. • IS-05 requests can be used to schedule a change for a later point in time. Make sure to wait until this time is reached.

Copyright © 2026 Vizrt

11.5 Explicit Usage of the Second SFP Pair on Matrox X.mio5 Q25 and DSX LE5 Q25

This applies to the Matrox X.mio5 Q25 and the Matrox DSX LE5 Q25. If the total bandwidth of all connections exceeds the available bandwidth or if an input or output needs to be moved explicitly to the second SFP pair, the following applies:

Connectors on the second SFP pair start at the values listed below:

- Video connectors on SFP C start at 17.
- Audio connectors on SFP C start at 129.
- Ancillary data connectors on SFP C start at 33.

Information: For inputs this happens automatically during configuration based on the nominal bit rate of the selected resolution.

11.5.1 Examples

Example 1

Single Channel setup with ten 1080p inputs, input configuration:

- Input channel 1 - 8 mapped to IP input 1 - 8.
- Input channel 9, 10 mapped to IP input 17, 18.

Example 2

Dual Channel setup in UHD with fill/key, output configuration:

- Viz Engine 1 program output mapped to IP output 1, IP output 2 is used by the key.
- Viz Engine 2 program output mapped to IP output 17, IP output 18 is used by the key.

Copyright © 2026 Vizrt

12 Audio in Viz Engine

This section contains the technical description of the Viz Engine audio system, and contains the following information:

- Overview
- Device Recognition and Selection
- Timing Behavior and Delay Settings
- Audio Plug-in
- Clip Formats
- Speaker Names
- Emergency Alert System
- Shared Memory Integration of Channel Volume Levels
- Dolby E Configuration
- Channel Setup and Clip Channel Routing
- Matrox Audio

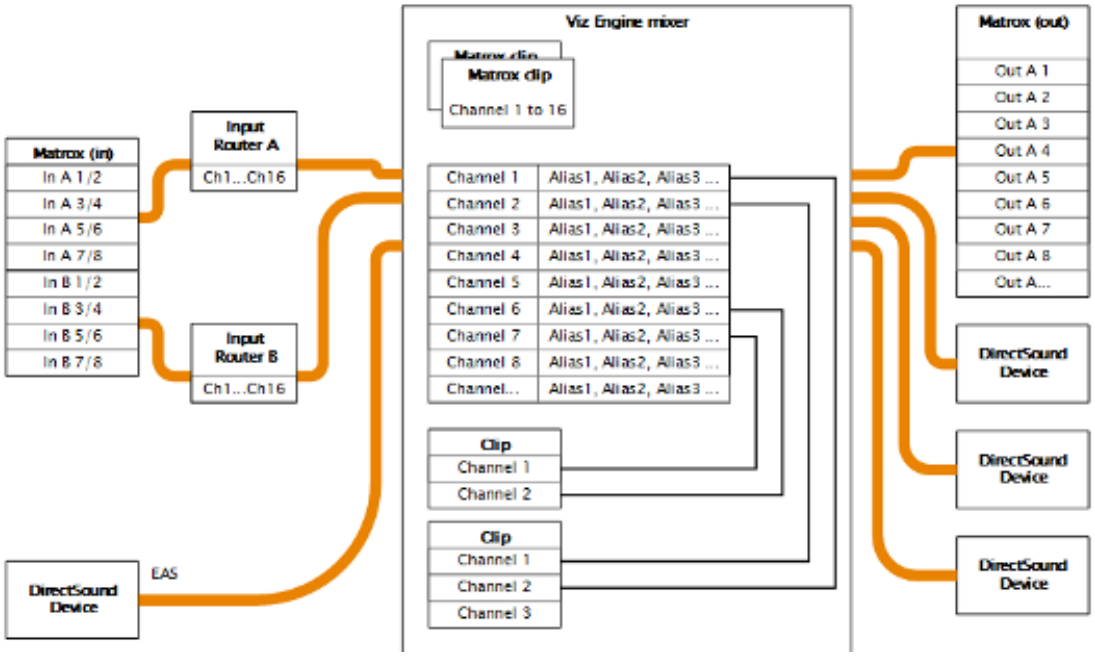
Copyright © 2026 Vizrt

12.1 Overview

Viz Engine receives and plays out audio through the video board, as embedded or AES/EBU audio. It can also play its audio mix through any audio playback device installed in Windows, such as a sound card.

This page contains information on the following topics:

- Channels
  - Audio Channels
  - Matrox Input Channels
  - Output Channels
- Matrox Routing
  - Matrox Live Input Routing
  - Matrox Clip Routing
- Audio Clip Formats
- Sound Card Output
  - Timing
- Audio Input for Emergency Alerts



12.1.1 Channels

Audio Channels

Viz Engine handles up to 16 audio channels for both input and output. This corresponds to the maximum number of embedded audio channels on an HD-SDI video source. Every channel can be given one or more user-defined aliases.

Matrox Input Channels

If using Matrox video hardware, the 16 input channels are available as AES/EBU input or embedded audio.

Output Channels

After mixing, the Viz Engine writes the audio data to the available output devices.

> **Note:** Matrox are only able to output digital audio.

12.1.2 Matrox Routing

Matrox Live Input Routing

On Matrox cards, it is possible to route live input channels to any internal Viz Engine output channel. Several channels can be routed to a single internal channel, but it is not possible to duplicate input channels.

Matrox Clip Routing

Audio from Matrox clips are mapped one by one to the internal Viz Engine channels; hence, no routing is possible. Audio from audio clips played through the stage is routed to the internal audio channel. This can be done automatic or manually.

12.1.3 Audio Clip Formats

Viz Engine decodes audio clips with its built-in decoder. The supported formats are:

- WAV
- MP3
- Ogg Vorbis

Codecs and DirectShow filters installed in Windows are not used, so installing them does not add support for other formats. Convert other audio formats to one of the formats above before you import them. Clips are converted to 48 kHz when they are decoded.

Audio in video clips is decoded by the clip player. See VML Clip Player.

12.1.4 Sound Card Output

Viz Engine can play its audio mix through any playback device that Windows reports, such as a sound card, a USB audio interface, or the HDMI or DisplayPort audio of a monitor. Sound card output uses the standard Windows audio system through the built-in MiniAudio library. It does not require DirectX or DirectSound.

Sound card output is intended for audio monitoring. Use it:

- on a laptop or workstation without a video board, for example when designing scenes in Viz Artist
- to monitor the audio locally
- when analog audio is needed.

Copyright © 2026 Vizrt

Note: Matrox boards only output digital audio (embedded or AES/EBU).

To choose which device each channel plays on, see Device Recognition and Selection.

Timing

When a video board is installed, the video timing of the board drives the audio mix, and the sound card plays the same mix. Without a video board, Viz Engine paces the audio mix with the system clock at 48 kHz.

12.1.5 Audio Input for Emergency Alerts

The only audio Viz Engine records from a Windows recording device is alert audio for the Emergency Alert System (EAS). See Emergency Alert System. Use the video board inputs for all other live audio.

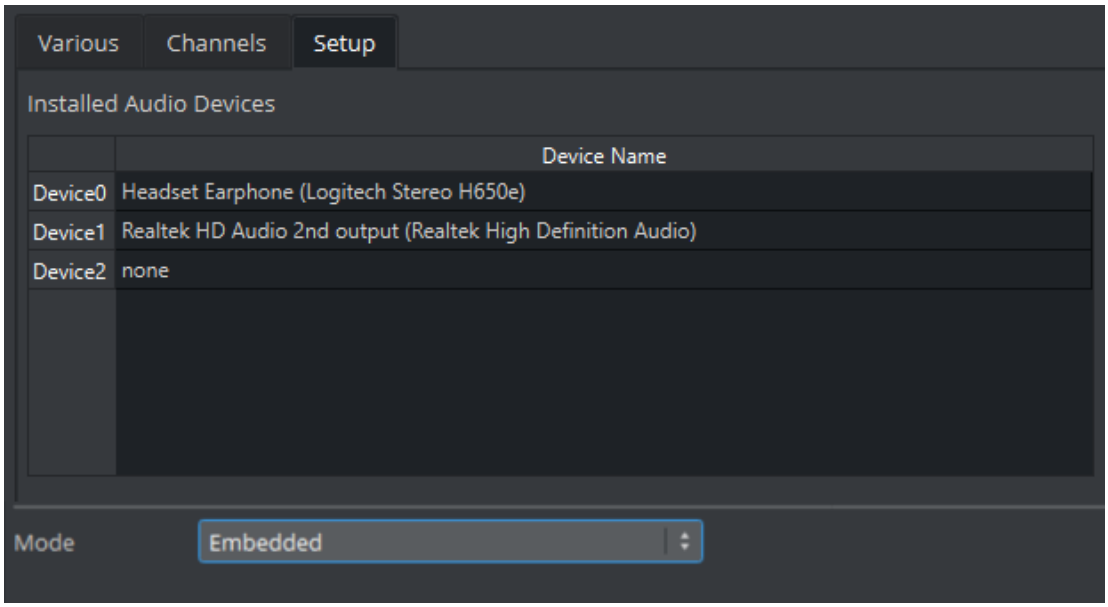
See Also

- Channel Setup and Clip Channel Routing
- Device Recognition and Selection
- Emergency Alert System

Copyright © 2026 Vizrt

12.2 Device Recognition and Selection

Viz Engine can play audio through any playback device that Windows reports, such as a sound card, a USB audio interface or HDMI audio. Each of the internal audio channels 0 to 7 can be assigned to a playback device, and several channels can share one device.



The audio system is able to use any installed DirectSound capable audio device. On every device, up to 16 channels can be used.

- **Mode:** Refers to the audio mode of the Matrox board. Options are:
  - **Embedded:** Captures audio from the Live video input connectors and is made available to the Viz Engine audio mixer, to mix it with other audio sources. Then output as embedded audio on the live video output connectors.
  - **AES:** Captutes audio from the AES input connectors and is made available to the Viz Engine audio mixer, to mix it with other audio sources. Then output on the AES output connectors.
  - **Embedded -> AES:** Captures embedded audio from the live video input connectors and is made available to the Viz Engine audio mixer, to mix it with other audio sources. Then output on the AES output connectors.
  - **AES -> Embedded:** Captures audio from the AES input connectors and is made available to the Viz Engine audio mixer, to mix it with other audio sources. Then output as embedded audio on the live video output connectors.
  - **Loop:** Loops audio through. No audio is mixed.
  - **Default:** Captures audio, but no output on the Matrox card.

During the startup process Viz Engine tests all available audio cards installed on the system. Manual activation of audio devices is done in **SECTION AUDIO_CONFIG** of the Viz Config file. By default a one to one channel assignment from the first audio device is done when a new device is selected.

Copyright © 2026 Vizrt

12.2.1 Device Recognition

At startup, Viz Engine lists the playback devices that Windows reports and writes each device name to the console. The names are also listed in **SECTION AUDIO_CONFIG** of the Viz Config file as `Available0`, `Available1` and so on. This list is for reference only. Viz Engine builds it again at every startup, so editing it has no effect.

12.2.2 Device Selection

Each internal channel is assigned to a device by name with the `VIZChannelDevice0` to `VIZChannelDevice7` settings in **SECTION AUDIO_CONFIG** of the Viz Config file. Enter the device name exactly as it appears in the `Available` list:

```
VIZChannelDevice0 = Speakers (Realtek(R) Audio)
VIZChannelDevice1 = Speakers (Realtek(R) Audio)
```

- A channel set to `none`, or with no device, is assigned the Windows default playback device at startup. The default device is set under **Sound** in the Windows settings.
- If the configured device is not present at startup, the channel plays on the Windows default playback device instead, and Viz Engine writes a warning to the console.
- If a device is disconnected while Viz Engine is running, the channels assigned to it go silent. Viz Engine reconnects to the same device automatically when it is available again. It never moves these channels to a different device.

Note: Only `VIZChannelDevice0` to `VIZChannelDevice7` are evaluated for sound card output.

See Also

- Overview
- Audio Settings

Copyright © 2026 Vizrt

12.3 Timing Behavior and Delay Settings

Sound card output has no per-device timing or latency settings. The DirectSound settings `BlockReadForward`, `SyncTargetPosition` and `SyncTargetWindowSize` are no longer used. If an older Viz Config file still contains them, Viz Engine ignores them.

12.3.1 Timing

When a video board is installed, the video timing of the board drives the audio mix, and every assigned sound card plays the same mix. Without a video board, Viz Engine paces the audio mix with the system clock at 48 kHz.

Sound card output is intended for audio monitoring. For broadcast output, use the embedded or AES/EBU audio of the video board.

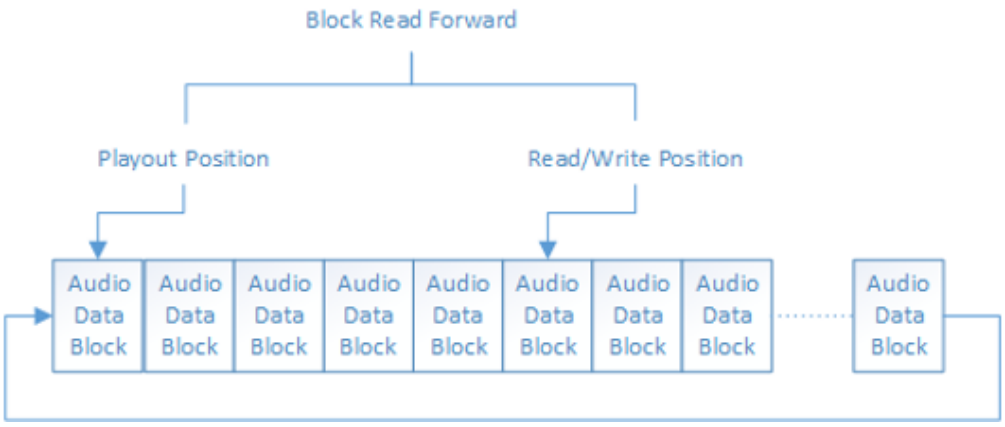
Timing behavior can only be set for each activated DirectSound. The default values should work for most devices; however, differences may occur between devices.

12.3.2 Channel Device and Channel Track Settings

Each of the internal channels 0 to 7 can be assigned to its own playback device with the `VIZChannelDevice<n>` settings in **SECTION AUDIO_CONFIG**, so the channels can be spread over several devices. See Device Recognition and Selection.

The `VIZChannelTrack<n>` settings are still written to the Viz Config file, but they no longer affect sound card output. Each internal channel is sent to the matching channel number on its device.

Timing on Matrox boards

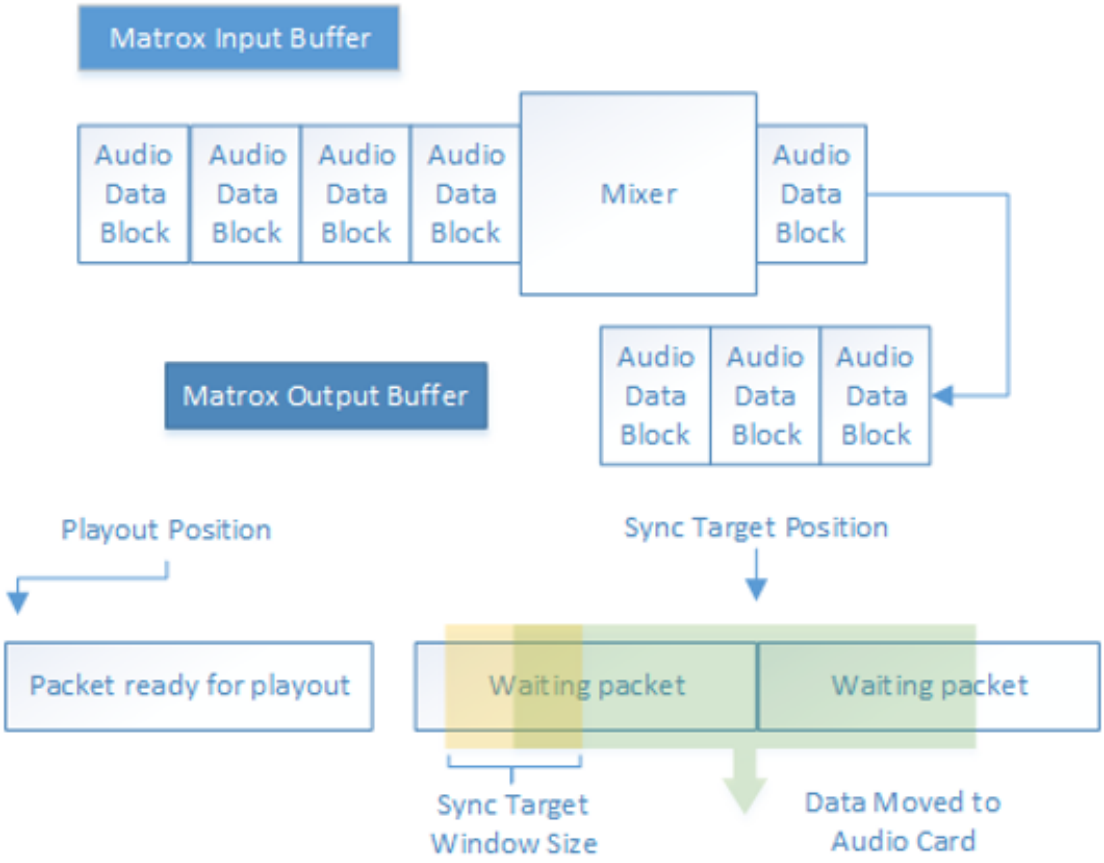


All sound devices use a ringbuffer that stores data until it is sent out to the audio channels, and this ringbuffer is organized in blocks of data. All sound hardware defines a distance in blocks (or bytes) that needs to be maintained. For almost all devices on the market a distance of six blocks is sufficient to have a clean output without artifacts.

Copyright © 2026 Vizrt

Tip: Try to decrease the *Block Read Forward* value down to `3` to accomplish less delay for the output. Viz Engine creates a ringbuffer of one second which is split in 60 blocks. This gives an output delay of ten milliseconds when the *BlockReadForward* value is `6` (see the Viz Config file's SECTION AUDIO_CONFIG).

Note: An audio card which is not synchronized always runs faster or than a synchronized video or audio card. There is a mechanism needed to keep all audio cards synchronized with each other. The mechanism, shown below, is used by Viz Engine to fulfill this condition:



After mixing the packages received from the Matrox board, the blocks of audio data are moved to the Matrox output buffer. This buffer is organized as a ringbuffer and holds one second of data and this is the maximum delay that can be achieved with the described mechanism.

The `SyncTargetPosition` parameter, set in `SECTION AUDIO_CONFIG`, defines the position relative to the playout position of the Matrox board where the audio data for the direct sound cards are branched. If the card is running slower than the reference card, the synchronized position moves away from the playout position. If it is faster, `SyncTargetPosition` moves to the playout position. The `SyncTargetWindowSize` parameter defines the border, when Viz Engine starts to re-sample the DirectSound data to bring `SyncTargetPosition` back in place. The predefined value of 250 samples is a good compromise between performance and quality. If a

Copyright © 2026 Vizrt

cheap audio card is used and small artifacts can be heard, try to increase this value. Good ranges are from `250` up to `600`.

`SyncTargetPosition` is used to synchronize the different audio cards to each other. Every audio card shows a specific delay behavior. Increase or decrease this value if one card is faster than the other. If the value is too small, artifacts occur; however, most audio cards work fine with the predefined values. `SyncTargetPosition` and `SyncTargetWindowsSize` are configurable settings that can be set separately for every activated audio card.

Channel Device and Channel Track settings

In Viz Engine, it is possible to combine two or more devices for playout of the Viz Engine's internal audio channels. Note that Viz Engine internally can use up to 16 channels. On many professional multichannel cards the channels are organized in virtual devices with two channels. A good example is the following configuration:

```
Available2 = M-Audio Delta 66 1+2
Available3 = M-Audio Delta 66 3+4
```

In Viz Engine it is possible to combine these two devices and create a quad speaker configuration as shown below:

```
VIZChannelDevice0 = M-Audio Delta 66 1+2
VIZChannelDevice1 = M-Audio Delta 66 1+2
VIZChannelDevice2 = M-Audio Delta 66 3+4
VIZChannelDevice3 = M-Audio Delta 66 3+4
VIZChannelDevice4 = Realtek HD Audio output
VIZChannelDevice5 = Realtek HD Audio output
VIZChannelDevice6 = Realtek HD Audio output
VIZChannelDevice7 = Realtek HD Audio output
VIZChannelDevice8 = none
VIZChannelDevice9 = none
VIZChannelDevice10 = none
VIZChannelDevice11 = none
VIZChannelDevice12 = none
VIZChannelDevice13 = none
VIZChannelDevice14 = none
VIZChannelDevice15 = none
VIZChannelTrack0 = 0
VIZChannelTrack1 = 1
VIZChannelTrack2 = 0
VIZChannelTrack3 = 1
VIZChannelTrack4 = 4
VIZChannelTrack5 = 5
VIZChannelTrack6 = 6
VIZChannelTrack7 = 7
VIZChannelTrack8 = 0
VIZChannelTrack9 = 0
VIZChannelTrack10 = 0
VIZChannelTrack11 = 0
VIZChannelTrack12 = 0
```

Copyright © 2026 Vizrt

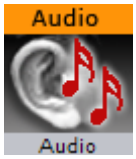
```
VIZChannelTrack13 = 0
VIZChannelTrack14 = 0
VIZChannelTrack15 = 0
```

See Also

- The Viz Config file, **SECTION AUDIO_CONFIG**.
- Device Recognition and Selection

Copyright © 2026 Vizrt

12.4 Audio Plug-in



The Audio plug-in allows a designer to configure audio channels.

Go to **Audio** in Container Plugins (see the Container Plugins section of the Viz Artist User Guide), for more information on the Audio plug-in.

This plug-in is located in Viz Artist (**Built-ins > Container Plugins > Global**) and can be applied to any container.

Copyright © 2026 Vizrt

12.5 Clip Formats

The recommended audio format is WAVE, as it gives the least decoding time and the best performance. Additionally, it is the only format that matches the Viz Engine support for 16 channels.

Video clips can have interleaved audio in it. The format is limited to 24-bit and 48 khz. There needs to be at least two channels in it, as mono is not supported. Again, the maximum channels are 16.

SDI in, break-out box (BOB) out is supported as well as BOB in and SDI out. It can be controlled by the video/clip channels controls.

Viz Engine is able to import and play the following Formats:

- **WAVE:** Up to 96 kHz, 24-bit and 16 Channels.
- **MP3:** All Formats (Stereo only).
- **OggVorbis:** All Formats, up to 16 Channels.

See Also

- Audio Settings
- Matrox configuration

Copyright © 2026 Vizrt

12.6 Speaker Names

Viz Engine understands the following default speaker names:

- *FRONT_LEFT*, *FRONT_RIGHT* and *FRONT_CENTER*
- *LOW_FREQUENCY*
- *BACK_LEFT*, *BACK_RIGHT*, and *BACK_CENTER*
- *FRONT_LEFT_OF_CENTER* and *FRONT_RIGHT_OF_CENTER*
- SIDE_LEFT and *SIDE_RIGHT*
- *TOP_CENTER*, *TOP_FRONT_LEFT*, *TOP_FRONT_CENTER*, *TOP_FRONT_RIGHT*, *TOP_BACK_LEFT*, *TOP_BACK_CENTER* and *TOP_BACK_RIGHT*
- *SPEAKER_RESERVED*

See Also

- Audio Settings
- Matrox configuration interface

Copyright © 2026 Vizrt

12.7 Emergency Alert System

The Emergency Alert System (EAS) is a national warning system used in the United States that supersedes the Emergency Broadcast System (EBS) and the CONELRAD System. EAS is jointly coordinated by the Federal Communications Commission (FCC), Federal Emergency Management Agency (FEMA) and the National Weather Service (NWS). The official EAS is designed to enable the President of the United States to speak to the citizens of the United States within ten minutes.

In Viz Engine, alert audio can be inserted live from a Windows recording device, such as a microphone or a line input. Viz Engine records from the Windows default recording device. When the alert starts, its audio fades in and is mixed into the EAS output channels. When the alert stops, the audio fades out and Viz Engine releases the recording device.

Starting the alert does not mute the program audio. The alert audio is mixed on top of it, after master volume and mute are applied, so the alert is heard even while the program audio is muted. Mute other audio from the control application if it must not be heard during the alert.

If the recording device is lost during an alert, the alert audio stops. Viz Engine does not switch to a different recording device.

The behavior is supported with the Text-to-Speech plug-in, as well as audio clips which are recorded prior to sending. These clips can be added dynamically. Viz Engine then adds them directly into predefined audio channels and also mute the other channels.

12.7.1 To Specify Output Channels for EAS

1. Open Audio Settings and select the **Channels** tab.
2. In the **Out Channel** list, enter the keywords `EAS0` and `EAS1` for the left and right channels, respectively. If either keyword is used in any channel, explicit mapping is used.

Several output channels may be pointed to by the same EAS channel. If no explicit mapping is set, Viz Engine defaults to output channels zero and one. Empty mapping is not allowed for the EAS system.

12.7.2 EAS Commands

- `MAIN*AUDIO*EAS_SYSTEM START` : Opens the recording device and fades the alert audio in. Returns an error if no recording device is available. Does nothing if the alert is already running.
- `MAIN*AUDIO*EAS_SYSTEM STOP` : Fades the alert audio out and releases the recording device.
- `MAIN*AUDIO*EAS_SYSTEM DELAY <frames>` : Sets how many audio sample frames the alert audio stays behind the recording device (48000 frames is one second). By default, the alert starts half a second (24000 frames) behind the recording device. Values are limited to 960 to 46080 frames. Negative values are ignored.

Copyright © 2026 Vizrt

12.8 Shared Memory Integration of Channel Volume Levels

The volume level of Live Input Channels like SDI, IP or NDI as well as clip channels can be exposed to a Shared Memory Variable. This could be used, for example, to visualize a level meter of an input source.

Any change to the input level is written into a shared memory variable in the System.Map. The naming convention is the following:

- *AudioLevel.LiveIn0_0* for the **first Live Input** and the **first channel**.
- *AudioLevel.LiveIn0_1* for the **first Live Input** and the **second channel**.
- *AudioLevel.LiveIn1_0* for the **second Live Input** and the **first channel**.
- *AudioLevel.LiveIn1_1* for the **second Live Input** and the **second channel**.
- *AudioLevel.LiveIn5_63* for the **sixth Live Input** and the **sixty fourth channel**.
- *AudioLevel.ClipIn0_0* for the **first Clip** and the **first channel**.
- etc....

To specify how often these values are updated, you can either specify a fixed value in the Viz configuration file:

```
############################
SECTION AUDIO_CONFIG
############################

## 0: No calculation of VU meter values at all (default), > 0: frequency of frames
when calculation happens (every X frame(s)).
AudioLevelMeterUpdateInterval = 2
```

or you can send a command to change it during runtime:

```
MAIN*CONFIGURATION*AUDIO*LEVEL_METER_UPDATE_INTERVAL GET / SET
```

Information: A setting of `0` means to not send values anymore.

Note: That the value you receive is in decibel, which is a logarithmic value between -0 and -120.

A sample script to generate a volume meter:

```
dim mySrc as string
Sub OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)
   dim tmp as array[string]
   dim normal_val as double
   If mapKey.StartsWith("AudioLevel") Then
      dim val = Cdbl(map[mapKey])
      normal_val = calculate(val)
      mapKey.Split("_",tmp)
      dim idx = cInt(tmp[-1])
      if ChildContainerCount >= idx then
```

Copyright © 2026 Vizrt

```
            GetChildContainerByIndex(idx).scaling.y = normal_val
      end If
    End If
End Sub

function calculate(db as double) as double
    dim myval as double
    calculate = 10 ^(db/20.00)
end function

sub OnInitParameters()
    RegisterParameterString("source","Source","LIVE1",20,20,"")
end sub

Sub OnParameterChanged(parameterName As String)
    if parameterName = "source" Then
       if mySrc<>"" Then
          system.map.UnregisterChangedCallback("AudioLevel."&mySrc&"_0")
          system.map.UnregisterChangedCallback("AudioLevel."&mySrc&"_1")
       end If
    mySrc = GetParameterString("source")
    system.map.RegisterChangedCallback("AudioLevel."&mySrc&"_0")
    system.map.RegisterChangedCallback("AudioLevel."&mySrc&"_1")
    end If
end Sub
```

This script generates an input field, that specifies the source (for example, ClipIn0). Any change in the shared memory map for AudioLevel.Clip0_0 or AudioLevel.Clip0_1 (Stereo) is applied to y-scaling of two subcontainers:





Copyright © 2026 Vizrt

12.9 Dolby E Configuration

Important: Currently there is no verified Dolby E support in Viz Engine.

This section explains how to consistently set up a Viz Engine for use with the DolbyE audio codec.

- Enable DolbyE
- Prepare for DolbyE Decoding
- Prepare for DolbyE Encoding
- Decode Plus Encode
- Pass-through Mode

Note: We use zero-based channel numbering in this configuration (channels zero through 15).

One most important prerequisite is a dongle containing a license from Minnetonka, the firm which makes the Surcode encoder/decoder SDK. You must have audio setup for 16-channel embedded mode in Viz. The inputs containing a DolbyE stream should also be configured for 16-channel, or at least 8-channel, depending on how many other audio channels are in the SDI audio track.

12.9.1 Enable DolbyE

There is one global flag for this in the Viz config file, this must be set for decoding or encoding.

```
DolbyEEnabled = 1
```

12.9.2 Prepare for DolbyE Decoding

DolbyE uses two audio channels on input. These channels must be a contiguous pair (i.e. 0/1, 2/3, 4/5, or 6/7). On the transmit side SDI audio does work mainly with audio pairs, as well as groups.
The decoded data is always eight channels, and is always written to the upper eight channels, eight through 15.

You must specify two pieces of information:

- The video channel (which live input to use).
- The first audio channel in the pair.

For the first:

```
ChannelDolbyEnabled__0 = 1 // video live input 1 contains the DolbyE data, or
ChannelDolbyEnabled__3 = 1 // video live input 4 contains the DolbyE data
```

For the second, you specify the first channel in an audio pair:

```
ChannelDolbyPos__1 = 4 // video live input 2 contains the DolbyE data on channels 4/5
```

Copyright © 2026 Vizrt

The channels containing the DolbyE stream are muted on output. For example, if your input consists of PCM audio on channels 0-5, and DolbyE data on channel 6-7, you get 14 channels of audio on output (given no other audio routing settings):

```
0 = input 0
1 = input 1
2 = input 2
3 = input 3
4 = input 4
5 = input 5
6 = silence - dolby data is muted
7 = silence - dolby data is muted
8 = decoded 0
9 = decoded 1
10 = decoded 2
11 = decoded 3
12 = decoded 4
13 = decoded 5
14 = decoded 6
15 = decoded 7
```

12.9.3 Prepare for DolbyE Encoding

In this case, we go from eight channels of PCM audio to two channels for the DolbyE stream. For input to the Dolby code, you can choose from two banks of eight channels: the lower eight (zero through seven) or the upper eight (eight through 15). The config field:

```
DolbyEOutput = 0 // No encoding
DolbyEOutput = 1 // Encode the lower eight, 0..7
DolbyEOutput = 2 // Encode the upper eight, 8..15
```

For output, you must give the first audio channel in a pair, which contains the DolbyE stream, for example:

```
DolbyEMixInPosition = 4 // goes out on channels 4/5
```

12.9.4 Decode Plus Encode

It is possible to perform a scenario which uses both encoding and decoding.

12.9.5 Pass-through Mode

There are settings which enables a DolbyE data stream to go through the Viz audio mixer untouched. Normally, if you send integer data through the mixer, the bits are changed on output, as the mixer uses floating point math, with a conversion on input and output. This renders the DolbyE stream unusable. The DolbyE data goes through the audio mixer unconverted to floating point along the way.

Copyright © 2026 Vizrt

```
ChannelDolbyEEnabled__0 = 1
ChannelDolbyEPos__0 = 0
ChannelDolbyEEnabled__1 = 1
ChannelDolbyEPos__1 = 2
ChannelDolbyEEnabled__2 = 1
ChannelDolbyEPos__2 = 4
ChannelDolbyEEnabled__3 = 1
ChannelDolbyEPos__3 = 6
ChannelDolbyEEnabled__4 = 1
ChannelDolbyEPos__4 = 8
ChannelDolbyEEnabled__5 = 1
ChannelDolbyEPos__5 = 10
ChannelDolbyEEnabled__6 = 1
ChannelDolbyEPos__6 = 12
ChannelDolbyEEnabled__7 = 1
ChannelDolbyEPos__7 = 14
DolbyEOutput = 0
DolbyEEnabled = 1
DolbyEMixInPosition = 0
DolbyELoop = 1
```

⚠ **Note:** For a simple pass-through no Dolby dongle is required.

Copyright © 2026 Vizrt

12.10 Channel Setup and Clip Channel Routing

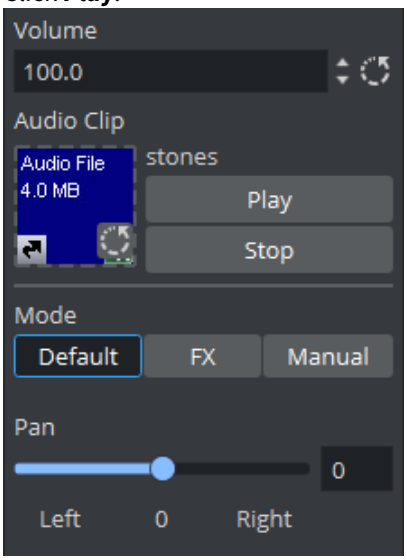
Channel setup is configured in the Audio Settings section in Viz Configuration. To get a correct mixing of clip channels, to the Viz Engine internal channels it is important to set the audio channels in a correct way.

The Audio Settings section of Viz Configuration is used to configure audio channels and channel routing, and To Add Multi-language Audio Channels. The latter allows the same scene with the same audio clips to output English, German, French and background music on three different machines. It is also possible to create 3D and other channel configurations for as many environments as needed. Stereo is configured by default.

From Viz Artist, a scene designer is able To Add Multiple Audio Channel Configurations, test the audio channel setup, and switch between the different local setups matching for example one or several remote Viz Engine audio setups. Configurations can also be tested separately or all together.

12.10.1 To Test Audio Channel Setup

1. Start Viz Artist.
2. Create a new Scene.
3. Add a group container to the Scene Tree.
4. Add the Audio Plug-in.
5. Open the Audio plug-in editor.
6. Add an audio clip to the *Test Clip* drop-zone.
7. Click **Play**.



Tip: Always have a set of test clips that provide audio for the different channel setups.

Copyright © 2026 Vizrt

12.11 Matrox Audio

The number of audio channels depends on the board. X.mio3 and DSX LE4 support 2, 4, 8 or 16 channels. X.mio5 and DSX LE5 support 1 to 8, 10, 12, 14, 16, 32 or 64 channels. The audio can be embedded into the video signal or be an external signal through the AES/EBU connectors. The available AES/EBU connectors depend on the Matrox version.

On the X.mio3 cards, there are balanced 75 Ohm connectors.

Note: AES/EBU audio is not available on X.mio5 and DSX LE5 boards.

12.11.1 To Enable Matrox Audio

1. Open **Viz Config**.
2. Click on Audio Settings.
3. In the **Various** tab, Enable Audio Active. This must be done for AES/EBU audio as well.
4. Click on the **Setup** tab.
5. Set a **Mode**. Select from:
    - Embedded
    - AES
    - Embedded AES
    - AES Embedded
    - Loop
6. Click on **Matrox**.
7. Select **VideoIn A** or **VideoIn B**.
8. In the **Audio** section:
    - Set **Audio** to `Active`.
    - Set the required **Channels**.
    - Set the required **Delay**.
9. Click **Save**.
10. Close **Viz Config**.

Tip: Use the provided configuration templates for a proper Audio Video delay. These templates are stored in **C:\Program Files\Vizrt\Viz3\Configuration Profiles**, you can easily access them by clicking on **Load** and **Installed Profile**.

Copyright © 2026 Vizrt

13 Viz Engine Shared Memory

A local VizCommunication.Map in each Viz Engine (as part of a cluster), collects and stores data. This data can be internal data, like a scene script pushing data to the map, or data from external control applications through TCP or UDP.

This section contains information on the following topics:

- Viz Engine SHM Synchronization
- Viz Engine SHM External Data Input
- Viz Engine Internal Data Interactive Scene
- Viz Engine SHM Snapshot

See Also

- Data Sharing (see the Viz Artist User Guide)
- Viz Engine REST Interface – read and write the shared memory maps over HTTP (`/api/v1/shm/...`).

Copyright © 2026 Vizrt

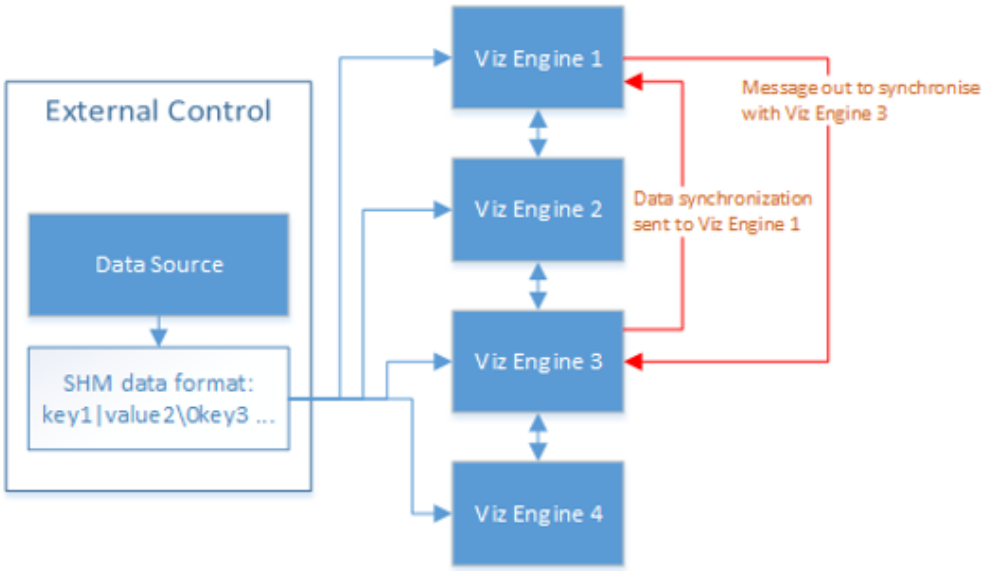
13.1 Viz Engine SHM Synchronization

If a Viz Engine is restarted or added to the cluster of Viz Engines for playout, the *VizCommunication.Map* data is not available on this Viz Engine. The local map on a new or restarted Viz Engine can be synchronized with the local map on another running Viz Engine in the same cluster.

A start-up Viz Engine can be synchronized through:

- TCP
- UDP
- An External Control Application
- Commands

Example: If Viz Engine 1 restarts it looks to Viz Engine 3 to update its local *VizCommunication.Map*.



This section contains information on the following topics:

- TCP and UDP Synchronization
  - To Synchronize a Viz Engine with TCP or UDP
- External Control Synchronization
  - From a Command Interface
  - Through TCP Communication
  - Through UDP Communication
- Command Synchronization
  - To Synchronize a Start-up Viz Engine with Commands

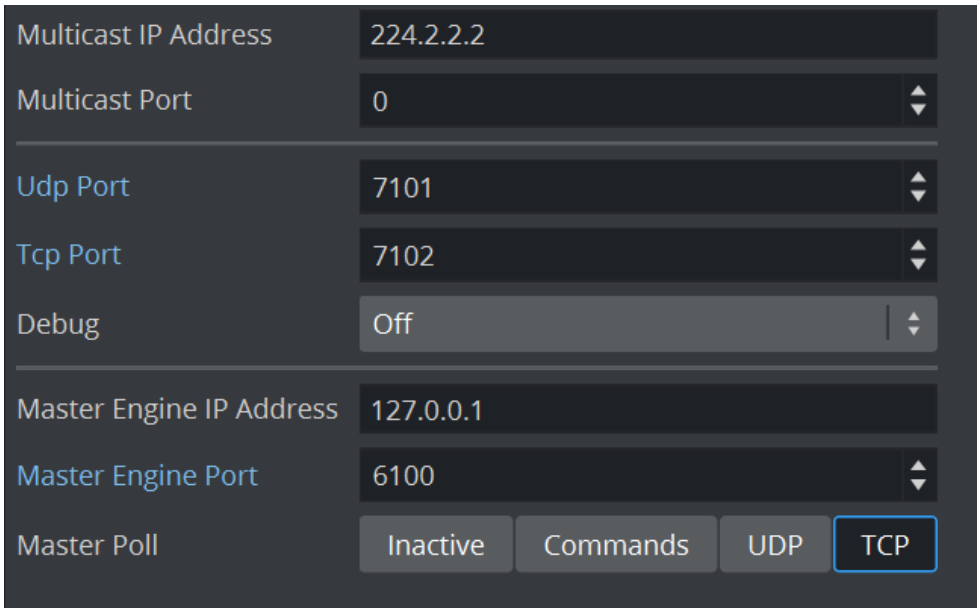
Copyright © 2026 Vizrt

13.1.1 TCP and UDP Synchronization

Use the procedure detailed below to synchronize a restarted or added Viz Engines, in a cluster, with the TCP or UDP protocol:

- **TCP:** The recommended communication protocol to synchronize maps on start-up because it is reliable and efficient (see SHM over TCP).
- **UDP:** For fast communication, but has drawbacks. For instance, packets can get lost (see SHM over UDP).

To Synchronize a Viz Engine with TCP or UDP



1. Go to Viz Configuration.
2. Click on Communication.
3. Click on the **Shared Memory Properties** tab.
4. Enter the TCP or UDP port number.

> **Note:** Always make sure that the selected port number is not in use by any other program on the same subnet. This is important for systems running multiple instances of Viz Engine.

5. In **Master Engine IP Address**, type the IP address of a running Viz Engine to synchronize with. Example *<IP/ hostname of Viz Engine 3>*.

> **Note:** Must be the IP address of a running Viz Engine in the same cluster system.

6. In **Master Engine Port**, type the port number to be used (default `6100`).This port sends the single startup synchronization command to start synchronizing. It should be the same as the general communication port of the Viz Engine set in **Master Engine IP Address**.
7. In **Master Poll**, select **UDP** or **TCP** as selected in step 4.

Copyright © 2026 Vizrt

8. Click **Save**.

13.1.2 External Control Synchronization

Synchronization can also be done from an External Control Application. The following command has to be sent to the Viz Engine which has the memory map populated:

From a Command Interface

```
VIZ_COMMUNICATION SYNCHRONIZE_TO <engine port>
```

where the Viz Engine is the engine which receives the data through the Command Interface. The port is usually `6100` (standard command interface port).

> **Note:** This is not recommended as a method for synchronization.

Through TCP Communication

```
VIZ_COMMUNICATION SYNCHRONIZE_SMMTCP <engine port>
```

where the Viz Engine is the engine which receives the data, and the port is the one that was specified for incoming TCP key|value pairs on the Viz Engine which is to receive the data (see SHM over TCP).

Through UDP Communication

```
VIZ_COMMUNICATION SYNCHRONIZE_UDP <engine port>
```

where the Viz Engine is the engine which receives the data, and the port is the one that was specified for incoming UDP key|value pairs on the Viz Engine which is to receive the data (see SHM over UDP).

13.1.3 Command Synchronization

Another communication option is by Commands. Here each map entry is synchronized through commands. It is reliable, but very slow and blocks the engine for quite an amount of time, if the map is large.

> **Note:** This is not recommended as a method for synchronization.

To Synchronize a Start-up Viz Engine with Commands

1. Go to Viz Configuration.
2. Click on Communication.
3. Click on the **Shared Memory Properties** tab.

Copyright © 2026 Vizrt

4. In the Shared Memory panel set these parameters:
    - **Master Engine Port:** Enter the communication port of the primary Engine (default is `6100`).
    - **Master Poll:** Click on **Commands**.
5. Click **Save**.

Copyright © 2026 Vizrt

13.2 Viz Engine SHM External Data Input

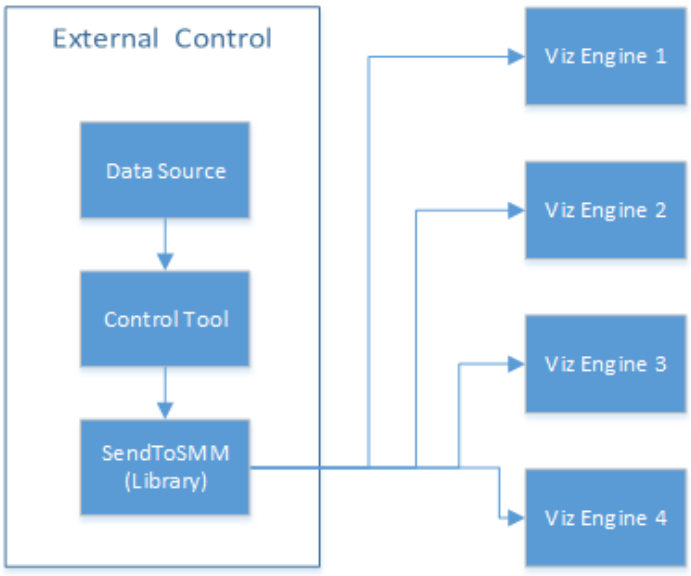
Data fed into the Shared Memory (SHM) should be done through the dedicated UDP or TCP IP ports for the SHM. Vizrt provides a set of components, *SendToSMM*, to make this task easier.

Note: Go to **<Viz Install Directory> > Tools > SendToSMM**, for more information about SendToSMM.

Data can also be sent to SHM through a Command Interface. Data sent through the Command Interface may be seen as a good option because the data would need to be sent to one Viz Engine only, and this Viz Engine engine would then distribute the data to the other Viz Engines. But data sent to SHM through the Command Interface has problems:

- Data sent through the Command Interface blocks the render queue of the receiving engine causing potential frame drops. Since the data needs to be sent through a command significant more bytes are transferred over the network.
- This Viz Engine is also a single point of failure.
- The data arrives at this one Viz Engine sooner than on all other Viz Engines.
- The notification method of the Graphic Hub is used to distribute the data and can cause additional load for the Graphic Hub.

The preferred method to send data is to use the *SendToSMM* library (or an equivalent) to send the data to the individual Viz Engines.



The communication protocol for the import of Shared Memory data depends on the type and final output of the data. There are set protocols to use with large amounts of data, in which all of the data must reach its destination graphic, and also where large amounts of data must be received quickly, but some loss of data is acceptable.

Copyright © 2026 Vizrt

Note: It is also possible to import data through Multicast. This method is not recommended as it can pollute the network.

This page contains information on the following:

- SHM over TCP
  - To Use TCP for SHM
- SHM over UDP
  - To Use UDP for SHM
- Plug-in API
  - Command Interface
  - Command Examples
- REST Interface

13.2.1 SHM over TCP

The SHM TCP communication protocol guarantees a reliable delivery of packages. It is a much more efficient than the Command Interface, but not as fast as SHM over UDP.

Use cases for a TCP connection could be finance stocks and currencies, or election result information, where the requirement is to deal with large amounts of information, and all of this data must reach its destination graphic. A single piece of lost data can have economic consequences, falsify charts, show mathematically wrong results, etc.

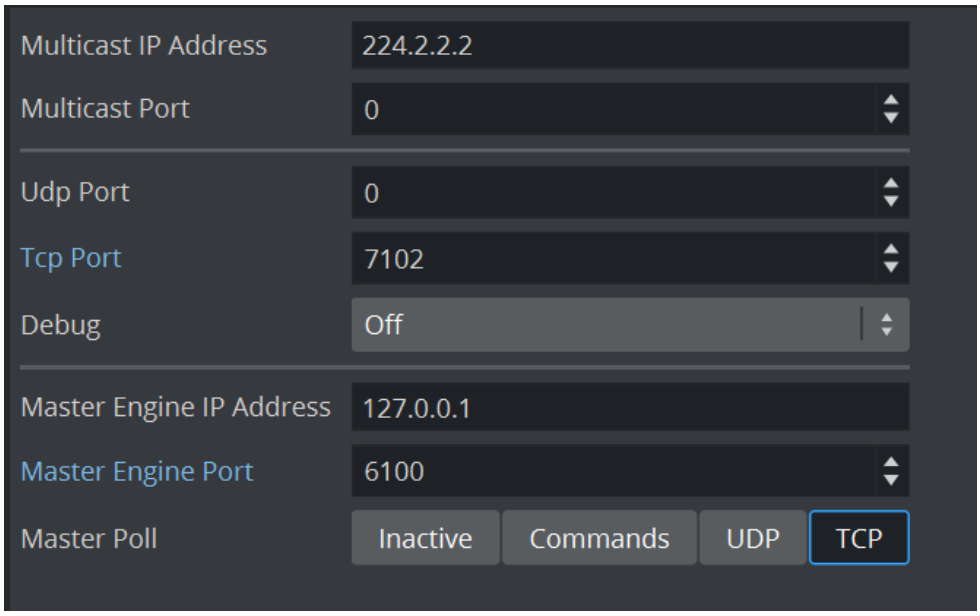
A TCP connection to a Viz Engine can be held open for a long time (this is recommended), and should not be closed and reopened between sending variables.

Note: The default number of connections is `1` and the maximum number of TCP Shared Memory connections is limited to `255`. Within this number of connections, a user-defined limit of maximum connections can be set in the configuration file with *smm_thread_count*.

IMPORTANT! The external program which provides the data, must connect and send the data to each Viz Engine individually. Vizrt provides a C# library, *SendToSMM* (part of the Viz install), for this purpose.

Copyright © 2026 Vizrt

To Use TCP for SHM



1. Go to the Configuring Viz.
2. Click on Communication.
3. Click on the **Shared Memory Properties** tab.
4. In the Shared Memory Panel, set **TCP Port.** There is no specific recommended port. Always make sure that the selected port is not in use by any other program on the same subnet.
5. Click **Save**.

The syntax for the key-value pairs is:

- key|value\0

Multiple key-value pairs can be sent, at once, as well. To do this make sure that each pair is terminated with `0`.

- key1|value1\0key2|value2\0key...

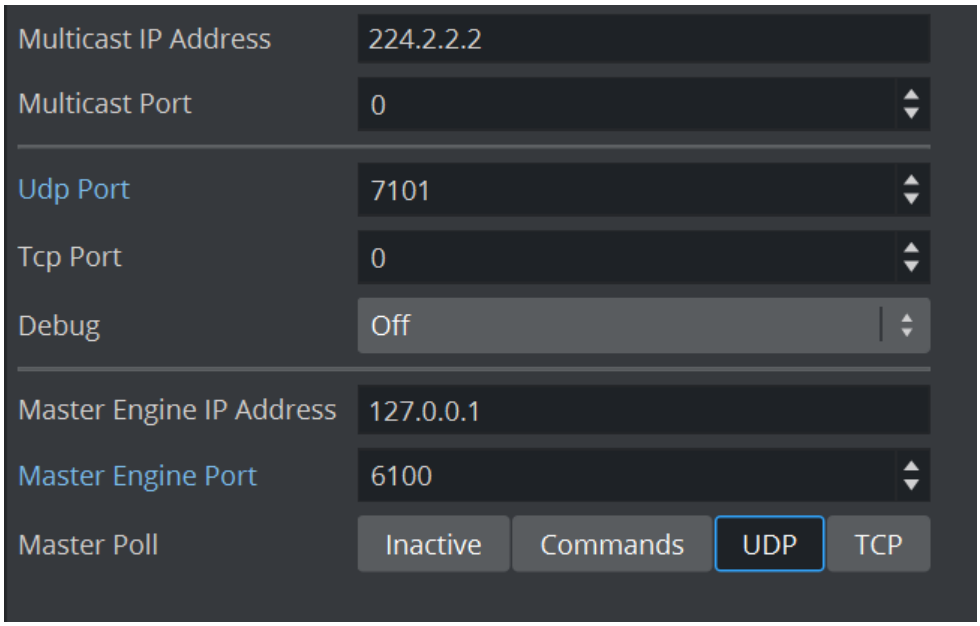
13.2.2 SHM over UDP

The SHM UDP communication protocol should be used for the delivery of volatile data. It is quicker than the SHM over TCP protocol, but less reliable, and is much more efficient than the Command Interface.

A use case for UDP would be Motor Sports, where data like speed, velocity, etc., is required. This is where there is a requirement to deal with large amounts of data, but not all of this data must reach its destination. A single piece of lost data does not affect the constant data update.

Copyright © 2026 Vizrt

To Use UDP for SHM



1. Go to the Configuring Viz.
2. Click on Communication.
3. Click on the **Shared Memory Properties** tab.
4. In the Shared Memory Panel, set **UDP Port.** There is no specific recommended port. Always make sure that the selected port is not in use by any other program on the same subnet.
5. Click **Save**.

The syntax for sending key-value pairs is the same as for TCP and UDP Synchronization.

13.2.3 Plug-in API

An option to manipulate data in SHM is by a plug-in interface. A use case would be where a TCP or UDP connection can not be used, or is not to be used. It is possible to write a plug-in to import data (for example, from an XML file, another database, etc.) and push it to SHM.

Another use case would be an interactive Scene (see Viz Engine Internal Data Interactive Scene).

> ⚠ **Note:** The Plug-in API documentation is included with the Viz installation (go to **Start > All Programs > Viz Artist x.x > Plugin SDK Documentation > Classes > Class List > SHARED_MEMORY**).

Command Interface

For small and single value changes the Command Interface of Viz Engine can be used. For example, to update a headline in a Scene.

Copyright © 2026 Vizrt

IMPORTANT! A command operation can block the renderer. If there are too many commands, within a small time, or commands containing a large amount of data, are sent, this can result in not rendering real-time anymore.

Note: Vizrt do not recommend this as a method for data import.

Any external program should consider the performance of the single or all connected Viz Engines. If there is a burst of thousands of SHM variables this can have implications on the Viz Engine rendering performance (Current (CUR) and Maximum (MAX)). A full list of commands is located at *<viz install folder>\Documentation\viz-cmd-doc-webhelp\index.html*

Note: From the list of commands, the commands, `CLEAR`, `DELETE_ELEMENT` and `PURGE_ELEMENT` only works when sent through the command interface of Viz Engine.

IMPORTANT! The command `CLEAR` must be run on each Engine where the MAP is to be reset (`VIZ_COMMUNICATION*MAP CLEAR`).

Whenever a new entry is made in the map (a new key-value pair) or values are changed, then the change is propagated to the other Viz Engines through a database messaging service to update the local copy of each Viz Engine's map (this only works when sent over the general communication port of Viz Engine. The default port is `6100`).

Command Examples

`VIZ_COMMUNICATION*MAP` can be used to access the map.

```
SET_DOUBLE_ELEMENT and GET_DOUBLE_ELEMENT
```

Example: `VIZ_COMMUNICATION*MAP SET_DOUBLE_ELEMENT "my_double" 1.2`

Example: `VIZ_COMMUNICATION*MAP GET_DOUBLE_ELEMENT "my_double"`

13.2.4 REST Interface

If the Viz Engine REST Interface is enabled, all three shared memory maps can also be read and written over HTTP:

Map	Command location	REST route
System.Map	`GLOBAL*MAP`	`/api/v1/shm/system`

Copyright © 2026 Vizrt

Map	Command location	REST route
VizCommunication.Map	`VIZ_COMMUNICATION*MAP`	`/api/v1/shm/distributed`
Scene.Map	`<scene>*MAP`	`/api/v1/shm/<scene uuid>` (UUID without brackets)

Append `/<key>` to address a single value, e.g. `http://localhost:61000/api/v1/shm/system/<key>`.

- On a map: `GET` reads it, `POST` adds a key, `PUT` replaces the whole map.
- On a key: `GET` reads it, `PUT` sets it, `DELETE` removes it.

Request bodies and status codes are described in the generated reference at `http://<host>:<port>/api/v1/documentation`. An OpenAPI description is installed at `<viz install folder>\library\REST\openapi.json`.

Copyright © 2026 Vizrt

13.3 Viz Engine Internal Data Interactive Scene

When data is modified on the distributed shared memory map on one Viz Engine through a script or through a plug-in, the data change gets reflected on the other Viz Engines automatically if they are configured to listen to the same shared memory map key. A use case could be a touch screen scene which modifies data, which is also used for HD-SDI Viz Engines or Viz Engines driving a Video Wall.

This synchronization uses Graphic Hub as a relay. Therefore, it is important that all Viz Engines which are to receive the data are connected to the same Graphic Hub, which use the same user or at least the same group.

13.3.1 Example

```
sub onInit()
VizCommunication.Map.RegisterChangedCallback("game")
end sub

sub OnSharedMemoryVariableChanged(map As SharedMemory, mapKey As String)
if mapKey="game" then
dim val = VizCommunication.map["game"]
println "new value:"+val
end if
end sub
```

If you now change the value of "game" on any of the connected clients running a scene with the above script, all Viz Engines print the new value into the console window.

Copyright © 2026 Vizrt

13.4 Viz Engine SHM Snapshot

It is also possible to take a data snapshot (save the whole content of the map) at any time. One Viz Engine can save the whole content of the map by calling the map's *SaveToDb* procedure, and another client can read it by calling *LoadFromDb.* The downside of this approach is that these functions block the renderer and can cause poor performance of the Graphic Hub Manager database, if the map is stored repeatedly.

IMPORTANT! The resulting SHM map objects are replicated as well. So saving the map each field or every minute can result in serious problems for the Graphic Hub Manager database (replication failing, etc.).

Note: For more information see the script function documentation. Go to, **Start > All Programs > Viz Artist x.x > Script Documentation > Data Types and procedures > Shared Memory**, or **<Viz Install Directory> > ScriptDoc > DataTypeSharedMemory**.

See Also

- Viz Engine SHM External Data Input
- Viz Engine SHM Synchronization

Copyright © 2026 Vizrt

14 Video Wall Configuration

Viz Engine supports output to multiple monitors configured as one large screen, commonly referred to as a video wall. This section details the hardware and software requirements, and how to configure Viz Engine to use a Video Wall.

Viz Engine video walls are based on the NVIDIA Mosaic technology for multiple displays, in an array of configurations. This allows for resolutions as high as up to 16k by 16k from one Viz Engine, fully synchronized with the house sync signal, with displays in landscape, portrait or arbitrary rotation layouts at the same time.

Please take a look at the Troubleshooting section prior to setting up NVIDIA Mosaic for the first time, as it provides solutions for common issues that may arise during Mosaic setup.

This chapter contains information on the following topics:

- Hardware Requirements and Recommendations
- Performance Considerations
- Troubleshooting Video Wall Configurations
- HDR Window Configuration
- Video Wall Setup Instructions

Copyright © 2026 Vizrt

14.1 Hardware Requirements and Recommendations

This section has information on the following topics:

- Minimum Hardware Configuration for Video Walls
- Recommended Configuration for Video Walls
- UHD Configuration with X.mio5 Boards
- IP-based Video Walls

The decision about which hardware to use when setting up a video wall depends a great deal on how the video wall can be used. In the most basic sense, any multi-display setup can be used. However, when in need of a an increased number of displays, clip playback capabilities, live streams, or complex graphics, the demands on the hardware increases significantly. A video board is required for clip playback and input capabilities.

Utilizing the NVIDIA Quadro P6000 graphics cards and the Matrox X.mio3 video board, Viz Engine support video wall configurations with up to 16k by 16k output resolution, and up to eight SDI inputs. Depending on the selected configuration, Viz Engine currently supports the following inputs and outputs for video walls:

- Four to eight SDI inputs, with resolutions ranging from SD to UHD, with a maximum of eight 1080i or one UHD concurrent input. (X.mio3).
- Up to six UHD inputs on X.mio5 SDI (and Gen-4 PCI).
- Four to 16 DisplayPort or DVI outputs, with up to UHD resolution per output.

Note: More GPUs require more bandwidth and therefore decrease the performance. Therefore, it is strongly recommended to use only one physical GPU and split the outputs (4*5K on P6000) with additional hardware like a Datapath Device.

14.1.1 Minimum Hardware Configuration for Video Walls

- HP Z8-series Desktop Workstation
- NVIDIA RTX A6000 graphics card or RTX 5000 Ada
- Matrox X.mio5 video board or Matrox DSX LE5 video board
- NVIDIA RTX Pro Sync synchronization card

14.1.2 Recommended Configuration for Video Walls

- HP Z8 Fury Desktop Workstation
- NVIDIA RTX Pro 5000 Blackwell or 6000 Max-Q Blackwell graphics card
- Matrox X.mio5 video board or Matrox DSX LE5 video board
- NVIDIA RTX Pro Sync synchronization card

Note: Videowall configurations require a DVI Max Resolution license.

Copyright © 2026 Vizrt

14.1.3 UHD Configuration with X.mio5 Boards

With the high bandwidth required to transform UHD surfaces to the GPU, any system that requires more than three UHD surfaces (no matter if live or clip) needs to be a **Gen-4 PCI** equipped system (like a Lenovo P620, HP G5 or G6i and a Gen-4 GPU, or like an A5000 or A6000). This guarantees playback of eight UHD surfaces (live or clips).

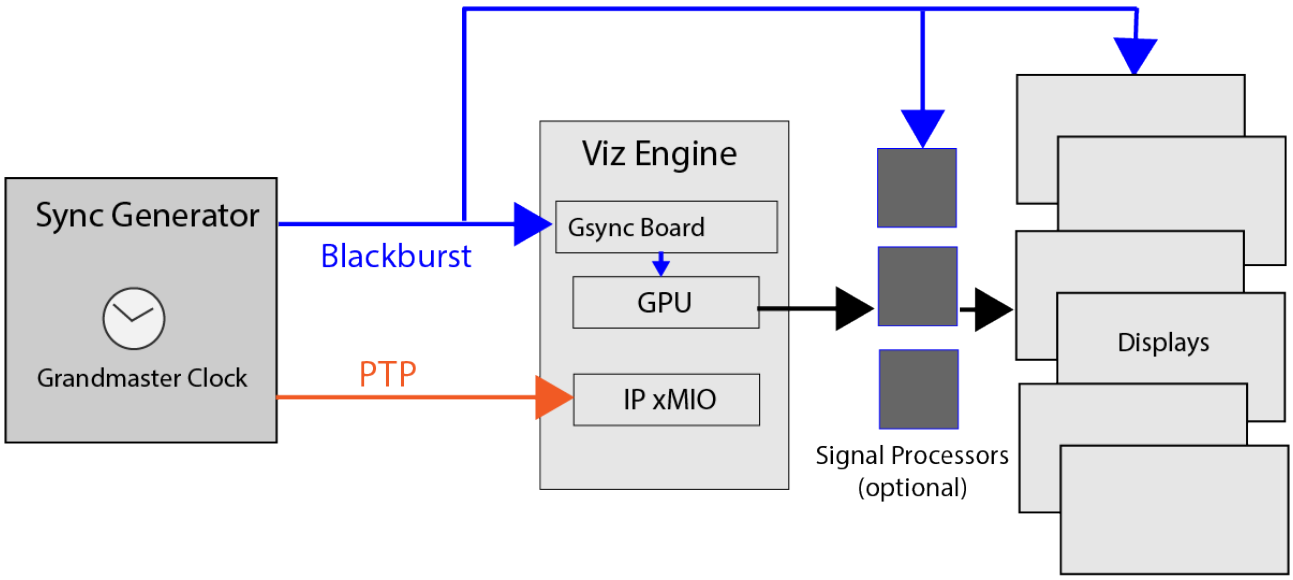
14.1.4 IP-based Video Walls

X.mio5 based topology boards do not have a genlock connector anymore, they only synced using PTP (older X.mio3 IP boards still have a Genlock connector). To correctly synchronize a videowall using X.mio5 boards, you need:

- A valid PTP signal.
- A blackburst signal to synchronize the NVIDIA gSync board.
- A blackburst signal for any additional device (for example, Signal processors or the displays).

All of these signals need to be driven from the same source. Therefore, it is highly recommended to use a system generating PTP and Genlock sources from the same clock.

The following diagram illustrates a basic synchronization flow:



Copyright © 2026 Vizrt

14.2 Performance Considerations

For maximum performance, the hardware described in the Recommended Configuration for Video Walls section should be used. In addition, Viz Engine should be running without GUI. To do this, start Viz Engine with the following command line options: `<viz install folder>\viz.exe -n -w`

14.2.1 Hardware Considerations

As Viz Engine operates in real time, rendering the output field by field, a wide range of factors may influence performance. Each field is rendered within 20 or 16.67 milliseconds, depending on the output format being 50 Hz (PAL) or 59.94 Hz (NTSC), respectively. This means that any requirements added to the render process reduces the time available for rendering the final output. Such requirements can be, but are not limited to:

- Copying between GPUs on a multi-GPU system.
- Copying between CPUs on a multi-CPU system.
- The system bus transfer rate on the system's motherboard.
- The amount of data to be copied and finally rendered.

In other words, adding more GPUs to the system decreases the overall performance, because of the time required for copying information between the GPUs. In configurations with multiple graphics cards, the cards should be connected to the same CPU (refer to the motherboard specifications). As a result, single GPU setups are *always* recommended over multi-GPU setups for performance reasons. By adding the Datapath Fx4 display wall controller, a 4K signal can be split into four HD signals. This allows to run, for example, a video wall with a total of 16 displays on one P6000 graphics card, with four Datapath Fx4 units.

On a multi-CPU system with two graphics cards, assigning them both to the same CPU can increase performance with as much as 30 percent.

14.2.2 Clip Playback

Utilizing several clip channels, either as insert or as backdrop is one of the main purposes of a videowall. The following points are important if one wants to playback clips without any jittering or stuttering on videowalls:

- Each clip channel requires several resources like buffers, codec initialization, file handles etc. Please activate only the required amount of clip players going to be used.
- Clips do have a pending player in background, meaning a clip is loaded into the pending player and when ready, it becomes the active player. This is not required anymore when using Super Channels, please deactivate the pending clip player.
- To avoid jittering when going On Air, use a warmup scene to utilize the clip channels. This warmup scene should have all required clip channels available as textures.
- The warmup scene should load a clip into its clip channels. This ensures all buffers are prepared and filled up. The codec of the clip should match the codec used later on (switching to a different codec should not affect the clip player).

Information: You can use the flag *Matrox.Clip.DelayedInitialization = 0*, to make sure all clip channels are initialized on Viz Engine startup. The loading of clips is still required by the warmup scene. This increases your startup and shutdown time! This also showed improved performance to the initial playout of clips in Videowall setups with multiple clip channels enabled.

Copyright © 2026 Vizrt

- Playback of AVC/h264 UHD clips requires an installed M264 board. Any other UHD clip playback requires a strong multicore CPU.

14.2.3 Scene Design Considerations

When designing scenes for a video wall, the design and graphics should be tested on the actual video wall. The scenes can be tested on computer monitors as well, in which case the monitor layout should preferably be as close to that of the video wall the scenes are being designed for, even down to make and model. This is to avoid different configurations, as well as the look and feel of the test configuration.

For the best result, performance tests should always be performed on the actual video wall before taking a new scene On Air.

Super Channels should be used as replacement of GFX Channels.

Since Viz Artist 3.8.2, the resolution of GFX Channels is by default set to the configured output resolution. To increase performance, limit the GFX channel resolution to the maximum resolution needed, for example `1920` x `1080` for HD resolution. The resolution setting depends on the scenes to be shown and how memory intensive they are.

In Multi-GPU configurations, GFX Channels should be added as Texture rather than DVE assets, as the performance improves significantly.

Copyright © 2026 Vizrt

14.3 Troubleshooting Video Wall Configurations

Please take a look at this section prior to configuring a Video Wall for the first time. It provides solutions for common issues which may arise during setup, and covers the following topics:

- Performance Issues
- Steps to Recover from Severe NVIDIA Mosaic Driver Related Issues
- Experiencing BSOD or System Freeze while Setting up Mosaic
- Only Some Displays of the Video Wall display an Image
- Missing NVIDIA Control Panel Settings
- NVIDIA Control Panel Crashes
- Mosaic Configuration Not Supported Error
- G-Sync Status LEDs or Topology Reports Indicate a Synchronization Issue
- Poor Performance when Using GFX Channels as DVE
- Blending Issues Using Classic Scenes in Superchannels with 16-bits per Channel Rendering
- Jittering on HP Z840
- Other Synchronization Issues

Note: Run Viz Engine in Engine mode and not in Artist mode. The startup should be `viz.exe -n -w`. The Engine is correctly configured only if Viz starts with a full black window with `-w` parameter.

14.3.1 Performance Issues

Create a simple scene with a bar moving from left to right and back in an endless loop, and run this scene on the newly configured Video Wall to spot issues such as jittering, tearing effects, etc. When troubleshooting performance issues, always make sure to check the following before investigating further:

- Make sure that the system meets the Minimum Hardware Configuration for Video Walls.
- Make sure that the Pre-Requirements for all Setups are met.
- Check that graphics and video devices are on the same CPU bus. The GPUs should always have priority.
- Make sure that the power connector to the graphics device is connected properly. The power connector must be 8-pin or use the 6-to-8-pin adapter included with the graphics device.
- Check that the HP Z8 G4 or **BIOS Settings** are correctly configured.
- Check the Viz Engine Video Wall Configuration Settings.
- Configure the NVIDIA driver for Video Wall.

If performance problems persist, a possible solution may be to increase the **Reactivation Delay** value in Viz Configuration. Please refer to the setting Video Input: Clip Input.

14.3.2 Steps to Recover from Severe NVIDIA Mosaic Driver Related Issues

1. Remove the NVIDIA drivers.
2. Shutdown the system.
3. Ensure that all monitor cables are connected properly to the graphics card.
4. Boot the system.
5. Reinstall the NVIDIA drivers.

Copyright © 2026 Vizrt

14.3.3 Experiencing BSOD or System Freeze while Setting up Mosaic

While operating in Mosaic mode, any changes to the physical setup, such as adding or removing monitors, can cause a system crash resulting in a Blue Screen of Death or complete system freeze. There is a chance to run into follow-up issues (mentioned in this chapter) after the system has rebooted.

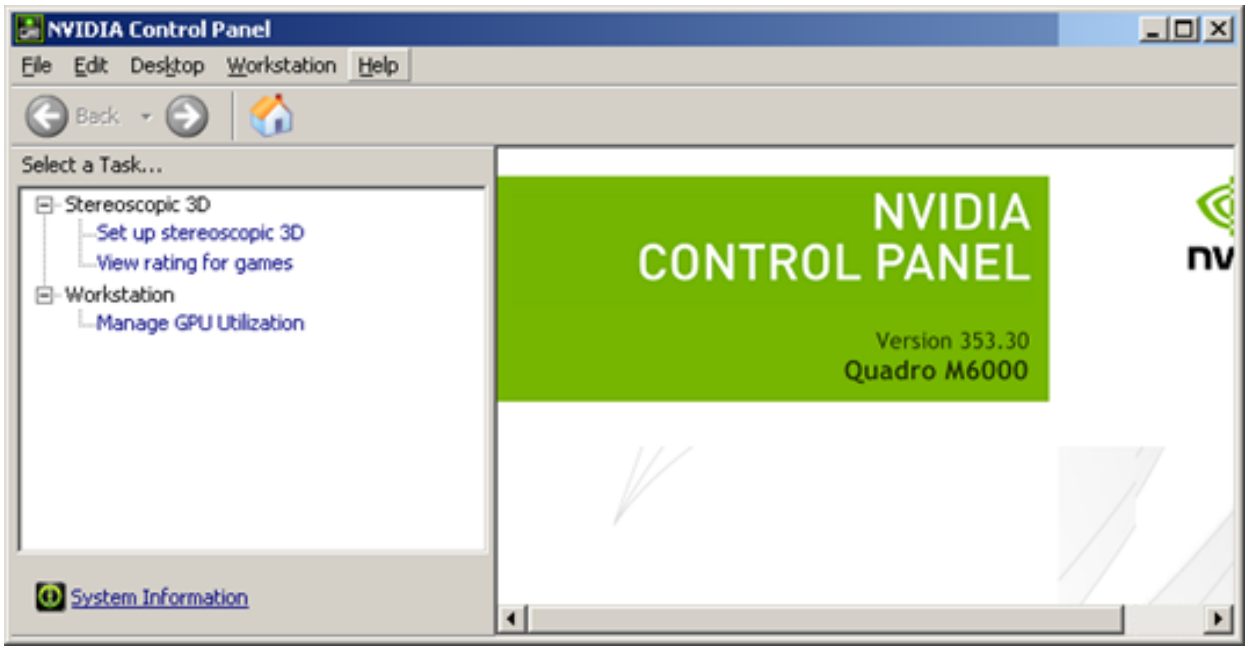
14.3.4 Only Some Displays of the Video Wall display an Image

There can be multiple reasons for this. Please ensure that:

- All monitors are connected properly to the graphics device and that they are active.
- Delete and re-create your Mosaic setup in the NVIDIA Control Panel.
- Please refer to section Steps to recover from severe NVIDIA Mosaic driver related issues in case the issue persists.

14.3.5 Missing NVIDIA Control Panel Settings

Please perform a re-installation of the NVIDIA driver as mentioned in section Steps to recover from severe NVIDIA Mosaic driver related issues. A known cause for this behavior are physical changes to an existing video-wall setup, for instance adding or removing monitors, while operating in Mosaic mode.



14.3.6 NVIDIA Control Panel Crashes

There is a possibility that the NVIDIA Control Panel crashes during Mosaic configuration, although this is not considered a reliable indicator on whether a driver re-installation is required or not. Sometimes the NVIDIA Control Panel can be restarted manually, and usually rebooting the machine should suffice.

Copyright © 2026 Vizrt

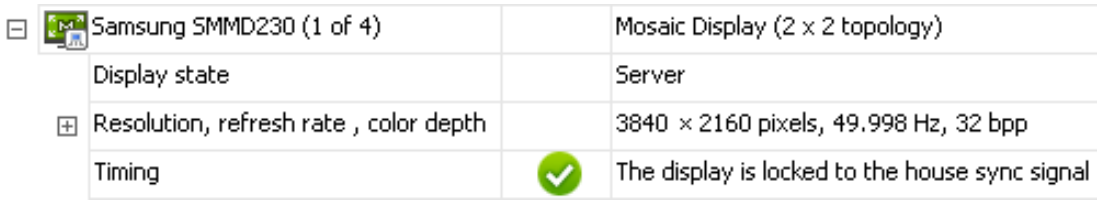
Please refer to section Steps to recover from severe NVIDIA Mosaic driver related issues if the issue persists.

14.3.7 Mosaic Configuration Not Supported Error

If running with two or more cards, make sure the same outputs are used on each card. For example, if using two DisplayPort outputs and one DVI output on the first card, the same outputs must be used on the second card, etc. Failure to do so may result in Mosaic setup failure, with an error message stating that the configuration is not supported.

14.3.8 G-Sync Status LEDs or Topology Reports Indicate a Synchronization Issue

If the status LEDs on the NVIDIA G-Sync status LEDs indicate an issue, or the topology reports "The display is locked to the house sync signal" for all displays, the topology view may indicate that everything is fine at first sight. However, when taking a closer look at the topology, it reveals that all monitors seem to sync to the **house sync signal** instead of reporting that the display is locked to the **frame lock sync** pulse:



This indicates that there exists an synchronization issue which needs to be fixed. Solving this may require going through one or several of the following steps:

1. **Re-establish synchronization**
   a. Disable display synchronization in section **Synchronize Displays**.
   b. Re-establish synchronization.
   See the NVIDIA Mosaic Configuration for 1080i50 or NVIDIA Mosaic Configuration for 1080i60M sections for more detailed instructions.
2. **Reconnect G-SYNC signal cables**
   a. Physically disconnect the sync cable from the G-SYNC card.
   b. Reconnect the sync cable.
   c. Switch synchronization back on in the NVIDIA Control Panel. Go to **Synchronize Displays** and set the radio button to **An external house sync signal**. See the NVIDIA Mosaic Configuration for 1080i50 or NVIDIA Mosaic Configuration for 1080i60M sections for more detailed instructions.
   d. Verify the LEDs on the NVIDIA Quadro Sync, or in the Topology Inspector of the NVIDIA Control Panel.
   e. If this does not help, try to reboot the machine.

In case this does not fix the issue:

1. Disable display synchronization in section Synchronize Displays.
2. Shutdown the machine.
3. Check cabling and signal sources.
4. Reboot the machine.
5. Re-establish synchronization as mentioned in section Synchronize Displays.

Copyright © 2026 Vizrt

14.3.9 Poor Performance when Using GFX Channels as DVE

On high resolution Video Walls in particular, poor performance may be experienced when using GFX channels as DVEs. This is caused by anti-aliased "blitting". To disable anti-aliasing on GFX channels, open the Viz configuration file, locate the `gfx_channels_antialiased` string, and set the value to `0`. The default value is `1`.

Example: `gfx_channels_antialiased = 1`

14.3.10 Blending Issues Using Classic Scenes in Superchannels with 16-bits per Channel Rendering

To correctly use Classic Scenes with 16-bit per Channel support (for example, for Superwhite and Superblack Support) and Superchannels (for example, Multiplay presets), the configuration flag `classic_allow_extended_range` needs to be set to `0`.

14.3.11 Jittering on HP Z840

Jittering on the HP Z840 is most commonly caused by a missing swapgroup. Make sure to run the `RENDERER_JOIN_SWAPGROUP 1` command to the Engine.

14.3.12 Other Synchronization Issues

When attaching the connector cables between the graphics card and the G-SYNC card, make sure that the red line on all of the connector cables are facing the card’s mounting bracket, towards the back of the case. Failure to do so can result in synchronization issues. In rare cases, syncing with only the NVIDIA G-SYNC can result in more stable synchronization. If all else fails, try disconnecting the reference signal from the Matrox board.

Copyright © 2026 Vizrt

14.4 HDR Window Configuration

14.4.1 Introduction

Viz Engine can be configured to create window contexts with enabled HDR compatibility. The content color space must be sRGB, which is then displayed using an adjustable brightness level in scRGB using a 16-bit float per channel framebuffer.

Warning: Rendering to HDR-enabled windows only works in combination with Viz Engine Render Pipeline scenes and does not work with the Classic Render Pipeline.

14.4.2 Prerequisites

- Microsoft Windows 11 or later.
- An HDR compatible monitor with HDR enabled via Microsoft Windows Display Settings.

Information: It is recommended that all monitors used are capable of running HDR, because if Microsoft Windows decides to create a window on a non-HDR monitor, HDR cannot be enabled at all.

14.4.3 Configuration

Viz Engine

Open Viz Engine Configuration file (for example, *C:\ProgramData\vizrt\VizEngine\VizEngine-0.cfg*) and set the flag `hdr_preview_mode` to `1` . All Viz Engine Renderer window contexts are then created with HDR compatibility enabled.

For each of them, the console shows on which display it was created, the current HDR enablement state and the maximum `WhiteLevel` . If those lines are not printed, HDR window rendering is not available at all.

Example

```
Display Info:
o Name: \\.\DISPLAY2
o HDR: Enabled
o WhiteLevel: 264
```

Windows Display Settings

To adjust the brightness of the SDR content, Viz Engine reads the display setting called **HDR/SDR brightness balance**. The slider can be found under **Display Settings > Windows HD Color settings** for any HDR enabled display. The rendering output reflects any change to the slider value immediately.

Copyright © 2026 Vizrt

14.4.4 References

- Prerequisites and Configuration Details | Setup HDR Preview
- https://learn.microsoft.com/en-us/windows/win32/direct3darticles/high-dynamic-range
- https://en.wikipedia.org/wiki/ScRGB

Copyright © 2026 Vizrt

14.5 Video Wall Setup Instructions

This section has information on the following topics:

- Pre-Requirements for All Setups
- Configure the NVIDIA Driver for Video Wall
- Order of Steps to Set Up NVIDIA Mosaic
- NVIDIA Quadro Sync
    - To Check the Status LEDs of the G-Sync Device

Additional information:

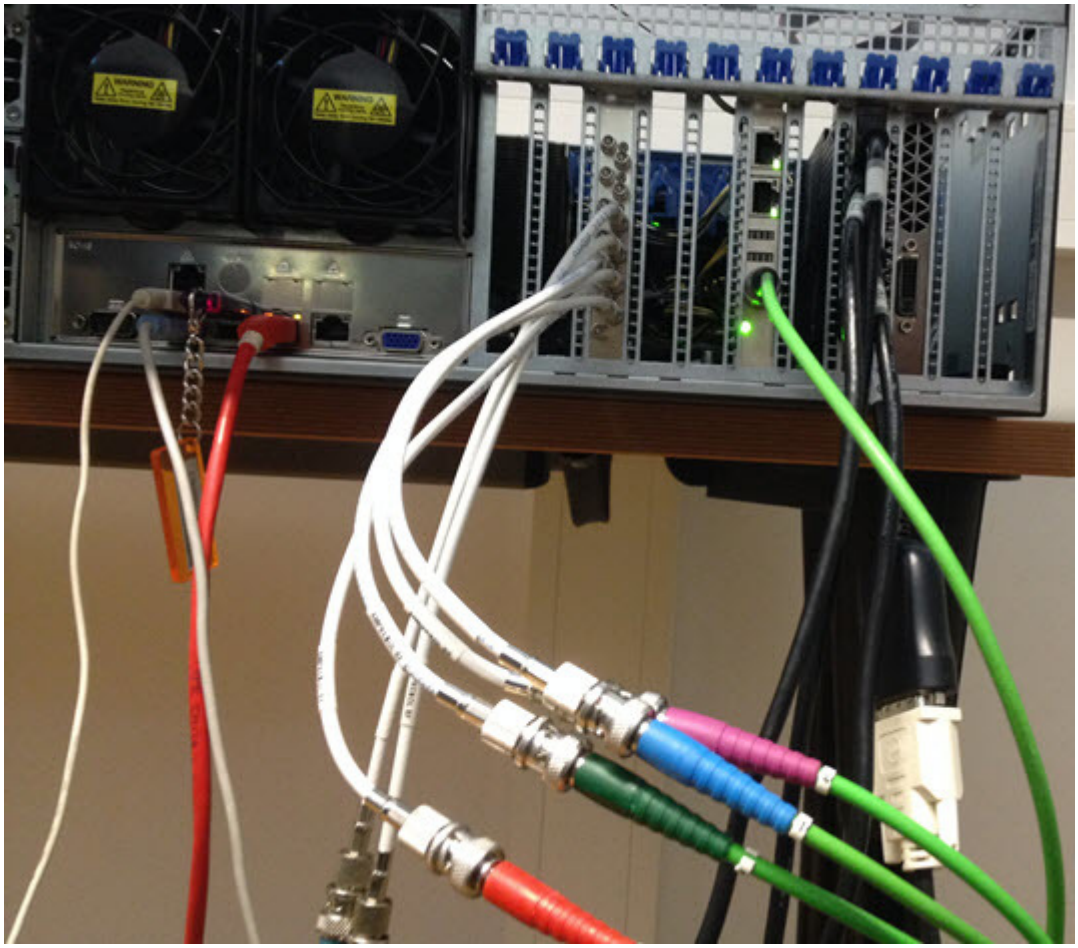
- Multiplay Troubleshooting Guide
- www.datapath.co.uk for Datapath Devices.

It is recommended to use only one GPU. Using more than one GPU results in a performance drop. If you are required to use more than one GPU, make sure that the monitor cables are the same for all cards. For example, a video wall configuration with six monitors on two cards, should use three outputs on each card with the same connectors on each card. If required, adapters can be used to connect the displays to the machine.

- Make sure that both GPU and Matrox boards are bound to the same CPU per PCI slot. The CPU assignment is usually listed in the cover or documentation of your machine. If possible, also bind a M264 board to this CPU.
- Make sure the NVIDIA G-SYNC card receives the same reference signal as the video board:

Info: A G-sync board is always required, no matter how many GPUs are installed.

Copyright © 2026 Vizrt



14.5.1 Pre-Requirements for All Setups

1. Shut down the machine.
2. Install the graphics device, NVIDIA G-Sync board and Matrox X.mio board. Please refer to the related documentation included with the hardware. For multiple CPU setups with two graphics devices, make sure that both devices are on the same CPU.

IMPORTANT! NVIDIA GPUs must be installed using the 8-pin power adapter which is included with the card. Installing the card using a 6-pin power connector has a negative impact on performance and can lead to unexpected system behavior with reports of a PCI error.

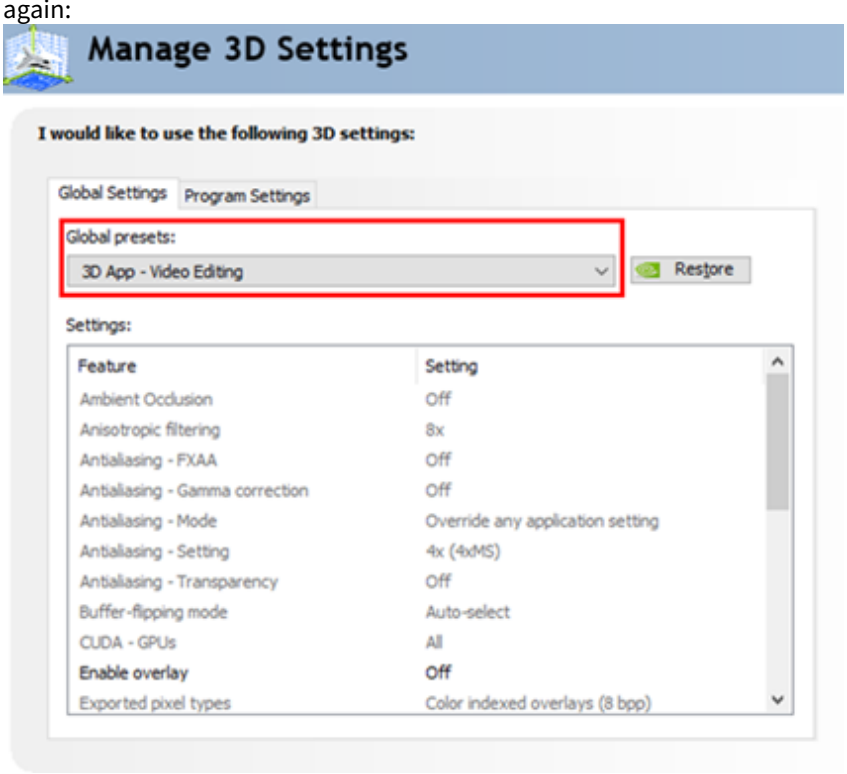
3. Connect the NVIDIA G-Sync and Matrox X.mio to the Genlock source. When attaching the connector cables between the graphics card and the G-SYNC card, make sure that the red line on all of the connector cables are facing the card's mounting bracket, towards the back of the case. Failure to do so can result in synchronization issues.
4. Connect the Matrox X.mio video input and output jacks.
5. Connect one monitor only to the graphics card, for the initial setup.
6. Boot the machine.
7. Install NVIDIA drivers and Matrox DSX.TopologyUtils. Reboot the machine as required.
8. Shut down the computer and connect all remaining video wall displays to the graphics card. For configurations with more than one graphics card, make sure to use the same outputs on all cards.

Copyright © 2026 Vizrt

9. Boot the machine and proceed with the Order of Steps to Set Up NVIDIA Mosaic.

14.5.2 Configure the NVIDIA Driver for Video Wall

1. Using the standard NVIDIA settings for Viz Engine, change the following parameter: **Vertical Sync:** `On` .
2. Set the power management to **Prefer maximum power.**
3. Apply the changes, then select **3D App - Video Editing** from the profile drop-down menu, and apply changes again:



14.5.3 Order of Steps to Set Up NVIDIA Mosaic

Please take a look at the Troubleshooting section prior to setting up NVIDIA Mosaic for the first time, as it provides solutions for common issues that may arise during Mosaic setup.

IMPORTANT! For 50 Hz setups, make sure that there is no **EDID-file** loaded. Then start the Mosaic configuration, there is no need to perform more steps. For 59.94 Hz, load the EDID-file and make sure it is applied properly by verifying every single monitor in the **View System Topology** panel.

If running with two or more cards, make sure the same outputs are used on each card. For example, if using two DisplayPort outputs and one DVI output on the first card, the same outputs must be used on the second card, etc. Failure to do so may result in Mosaic setup failure, with an error message stating that the configuration is not supported.

1. Enter Mosaic configuration in the NVIDIA control panel and setup Mosaic for the required refresh rate:
   - NVIDIA Mosaic Configuration for 1080i50
   - NVIDIA Mosaic Configuration for 1080i60M

Copyright © 2026 Vizrt

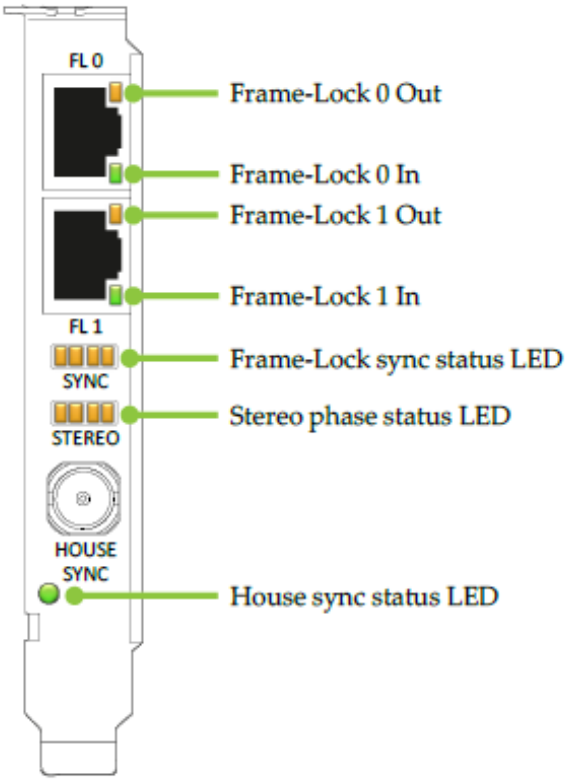
2. Synchronize the GPU to the internal house-sync, by configuring the NVIDIA G-Sync device in the **Synchronize displays** panel of the NVIDIA Control Panel:
    - Video Wall Setup Instructions. Proceed with adjusting the Viz Engine Video Wall Configuration Settings to finalize the video wall configuration.

14.5.4 NVIDIA Quadro Sync

The NVIDIA Quadro sync card is used to synchronize the graphics cards with the house clock, and is required for video wall configurations.

V-sync must still be set in the NVIDIA driver to ensure that the OpenGL SwapBuffer operation takes place at the vertical retrace, to avoid tearing between two frames. The Quadro Sync board synchronizes the vertical retrace of all the displays, it does not lock the OpenGL SwapBuffer operation to the vertical retrace.

To Check the Status LEDs of the G-Sync Device



1. Ensure the House sync status LED indicates that a proper sync signal is connected.
2. Ensure that the Frame-Lock sync status LED is shining green. It must not blink or shine orange. When sync is stable, the Stereo phase status LED should also be lit green.

Copyright © 2026 Vizrt

14.5.5 Configuration Using Datapath Devices

This section has some basic information on how to configure a videowall using Datapath devices:

- Prepare the Layout
- Prepare the Input
- Attach the Device(s)
- Troubleshooting

For detailed instructions on how to use these devices, please refer to https://www.datapathdocuments.co.uk/downloads/.

A Maxwell or Pascal GPU supports up to 4K (5K Pascal) resolution per head. This allows for a maximum resolution of 16Kx16K on one GPU. As single GPU installations are preferred for performance issues one can utilize a multi display distributor from Datapath to grab the incoming signal and split it up into four individual HD signals. A single head drives four HD outputs, four heads can drive up to 16 HD outputs. The Datapath devices are recognized by the NVIDIA driver as one single monitor with its full resolution. Additionally, each Datapath device can be synced on a house signal, usually the same sync used for synchronizing the GSync board and the Matrox devices.

Please follow these steps to setup a videowall based on Datapath devices. It is recommended to perform each of the following steps one after the other:

- Prepare the Datapath environment.
- Setup NVIDIA Mosaic.
- Synchronize your Mosaic.

Bezel correction: A bezel correction needs to be done on Datapath side, as the NVIDIA driver only sees one big display and is not aware of any bezel information.

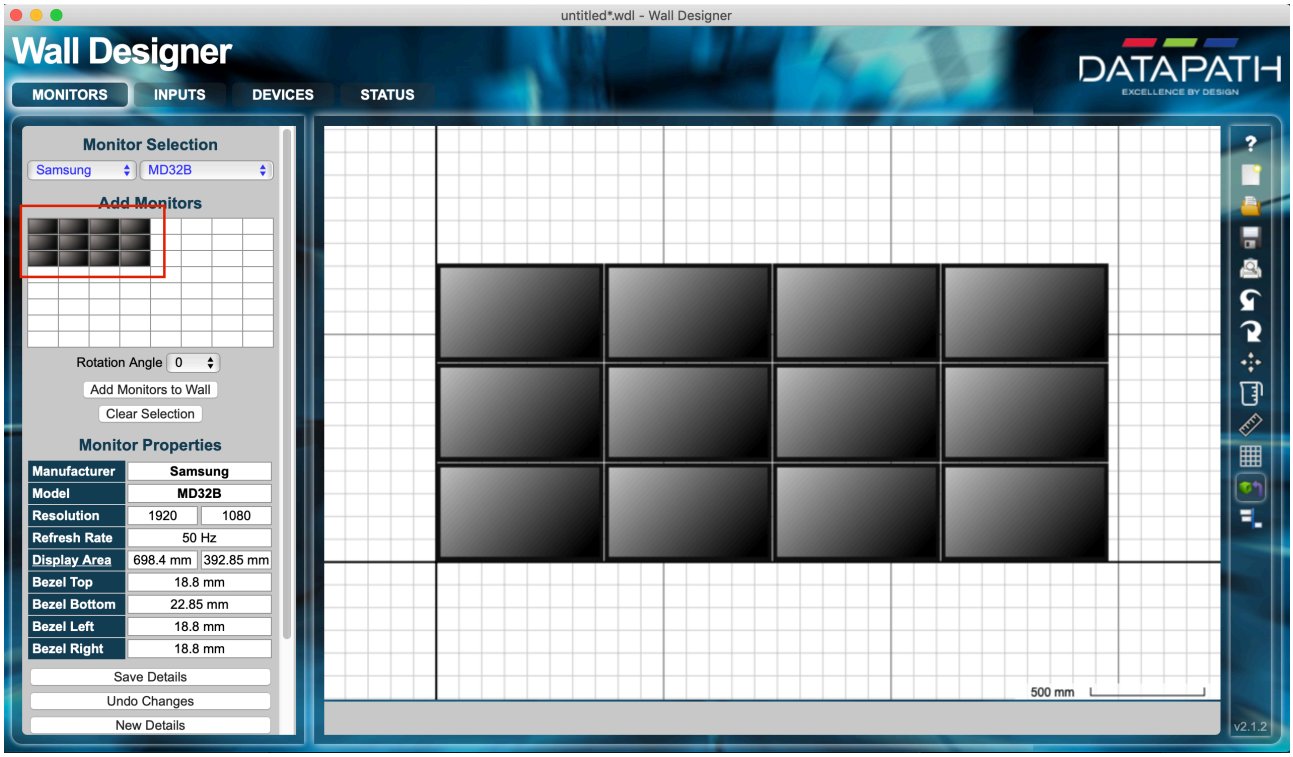
Prepare the Layout

Choose your monitor vendor and add the number of screens you want to utilize.

Please make sure the resolution is the correct native resolution of your monitors and the refresh rate matches your final refresh rate (50Hz or 59.94Hz). After that, press **Add Monitors to Wall** button. This represents the physical alignment of your monitors. Perform any modification of your alignment in this screen.

Note: You can also rotate and flip monitors within this view.

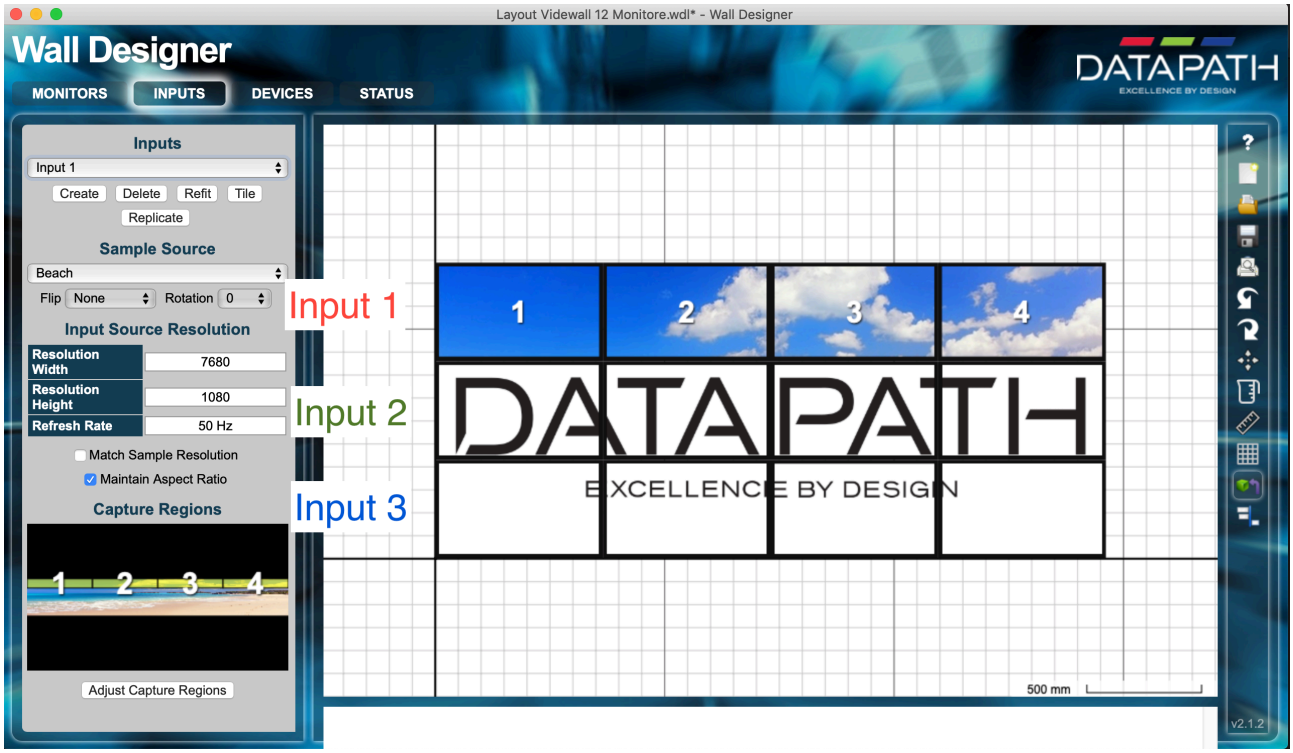
Copyright © 2026 Vizrt



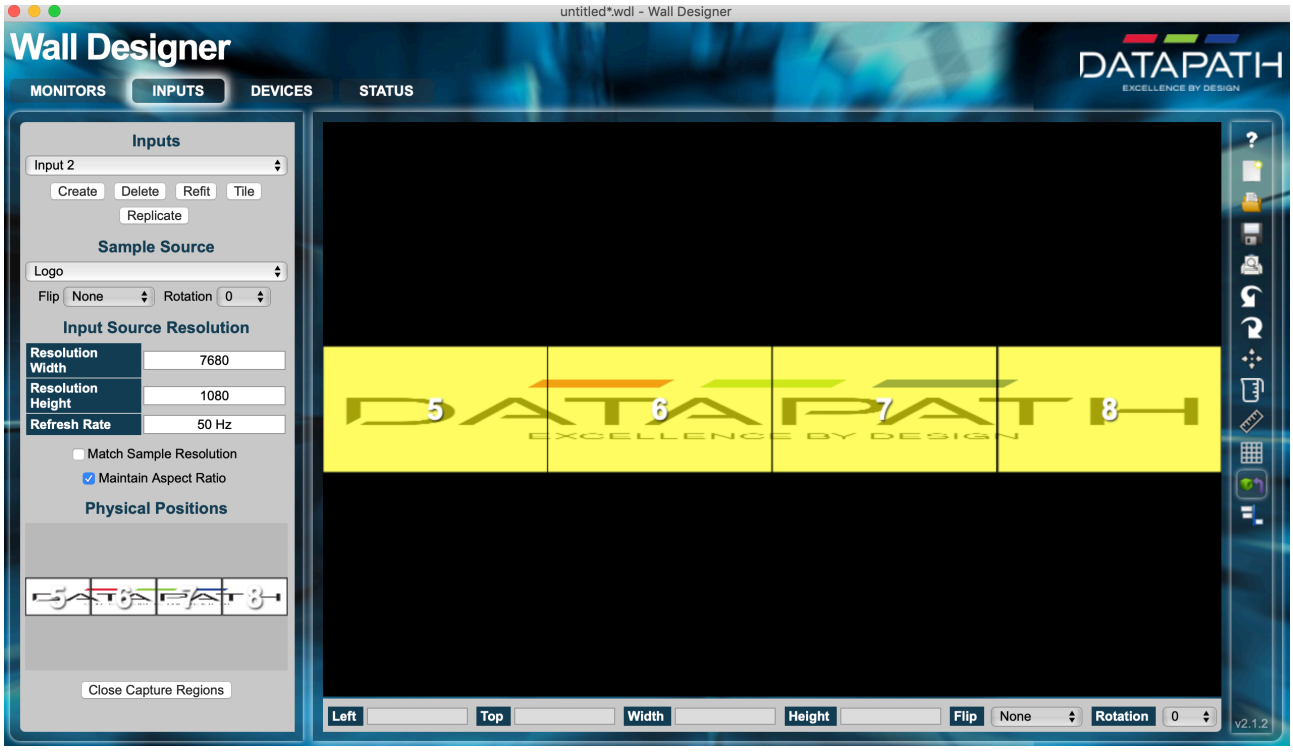
Prepare the Input

In this step, we create a virtual device recognized by the NVIDIA GPU as monitor(s). Create as many inputs as heads (=Datapath devices) physically connected to your NVIDIA GPU. In our example, we use three devices, each one of them is driving one row. One input is 4x1920px and 1080 pixels height. The input needs to be configured to be 7680px * 1080px. This is the same resolution as NVIDIA needs to recognize them.

Copyright © 2026 Vizrt



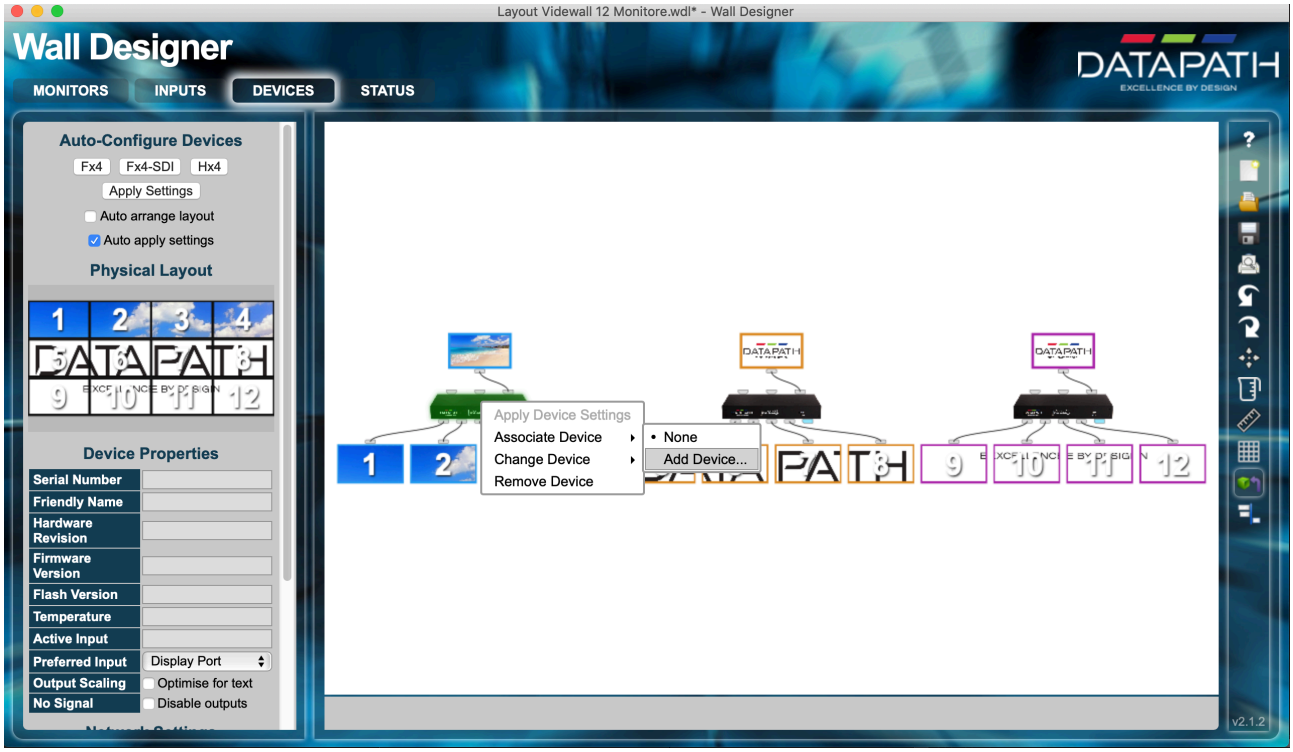
Press **Adjust Capture Regions** if you need to adjust the area the input captured by the various monitors. The alignment of monitors can be fine tuned pixel accurate. Any bezel correction needs also be done here. Make sure the displays align correctly to the configured input.



Copyright © 2026 Vizrt

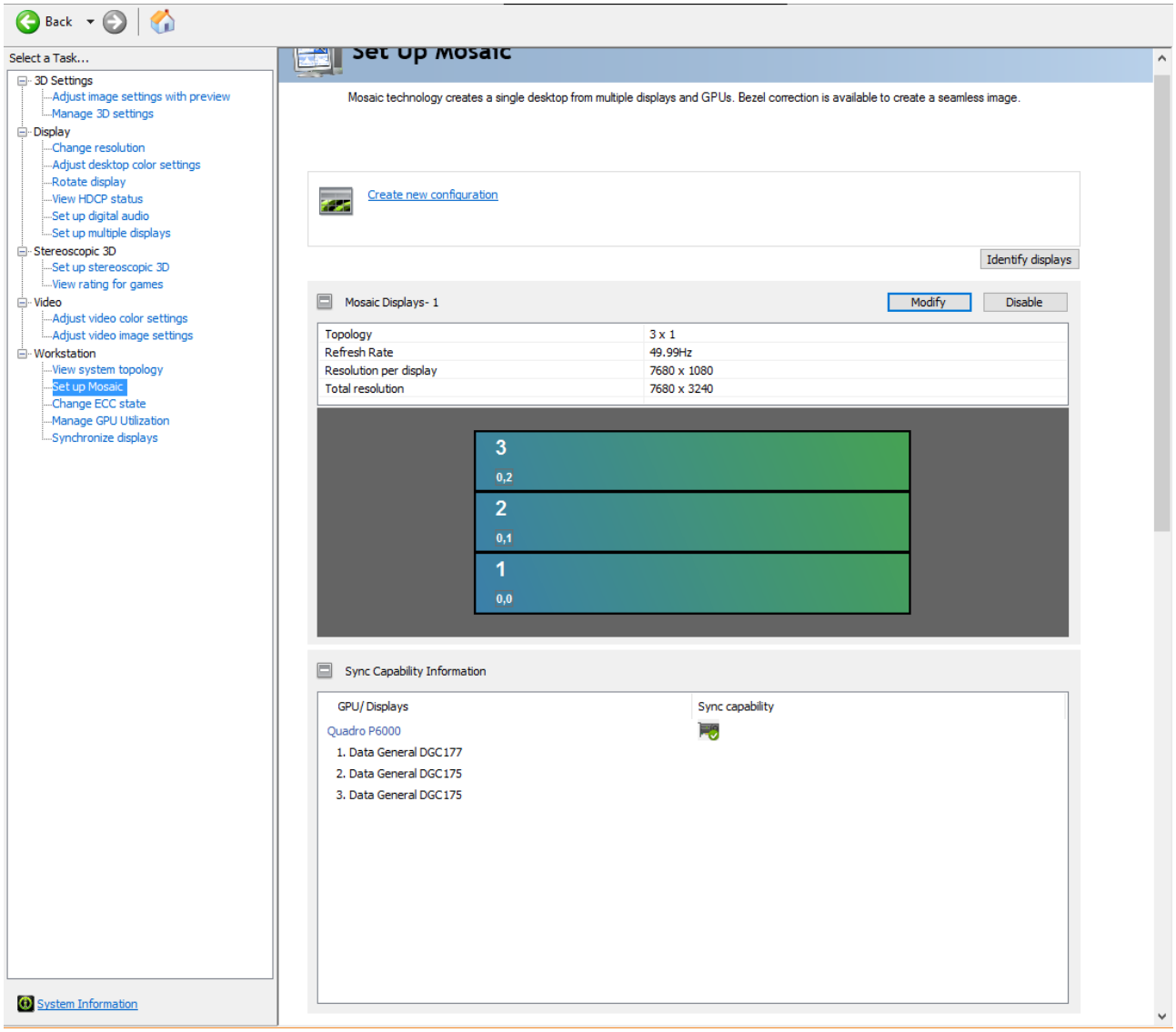
Attach the Device(s)

We need to connect the input to our physical Datapath devices. This is done in the devices section of Wall Designer. If you hit the **Auto Configure Device** on your current device, it generates an input, a Datapath device and the number of monitors. Now, we need to assign the Datapath devices to the one available. Make sure your environment discovered all your Datapath devices either on the network or connected via USB. This can be done in the status page. If no devices are present, they must be added manually.



Right click on the **Datapath** and assign the correct one. If it is missing from the list, please manually add the IP number of your device. (This can be obtained by connecting it via USB). Once you click on the **Apply Settings** button, the configuration activates and the NVIDIA should recognize three virtual monitors with a resolution of 7680x1080 each. Verify the input by clicking on the input image(s). Once this step has been completed, you can continue by creating your NVIDIA mosaic:

Copyright © 2026 Vizrt



Troubleshooting

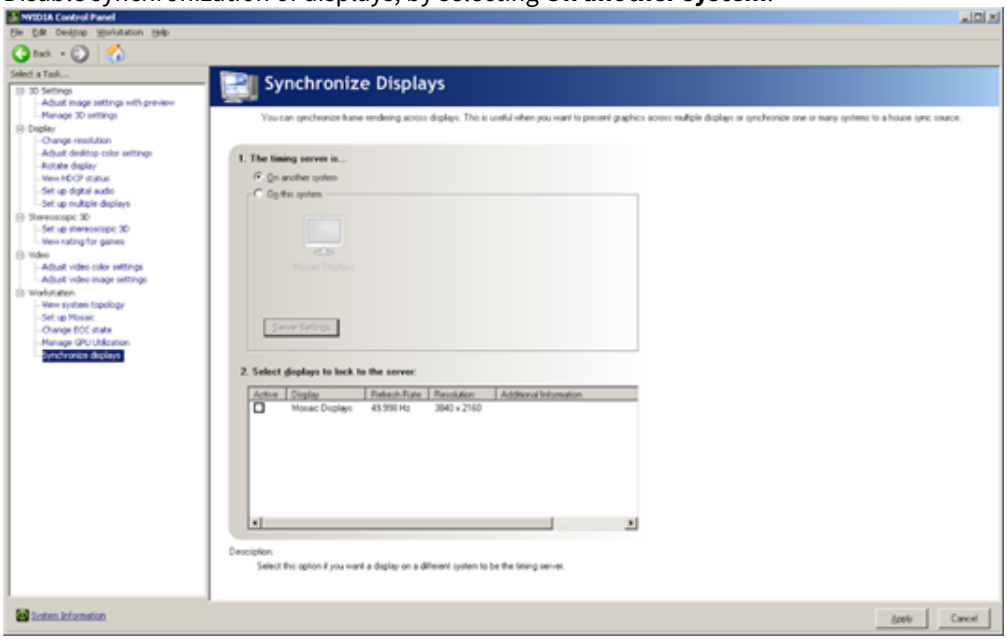
Detailed setting of each Datapath device is available by opening its built-in webpage. It allows setting proper name, network settings, correct timing and most important correct sync settings. Open the webpage and verify the correct timing for the sync and the refresh rate for each monitor. It can also display a test pattern instead of the NVIDIA output, which makes troubleshooting much more easier. If monitors remain black, but give a proper image when using a test pattern, verify in the NVIDIA control panel by checking the color depth per display. It must be set to eight bits/pixel, not six bits/pixel.

Copyright © 2026 Vizrt

14.5.6 NVIDIA Mosaic Configuration for 1080i50

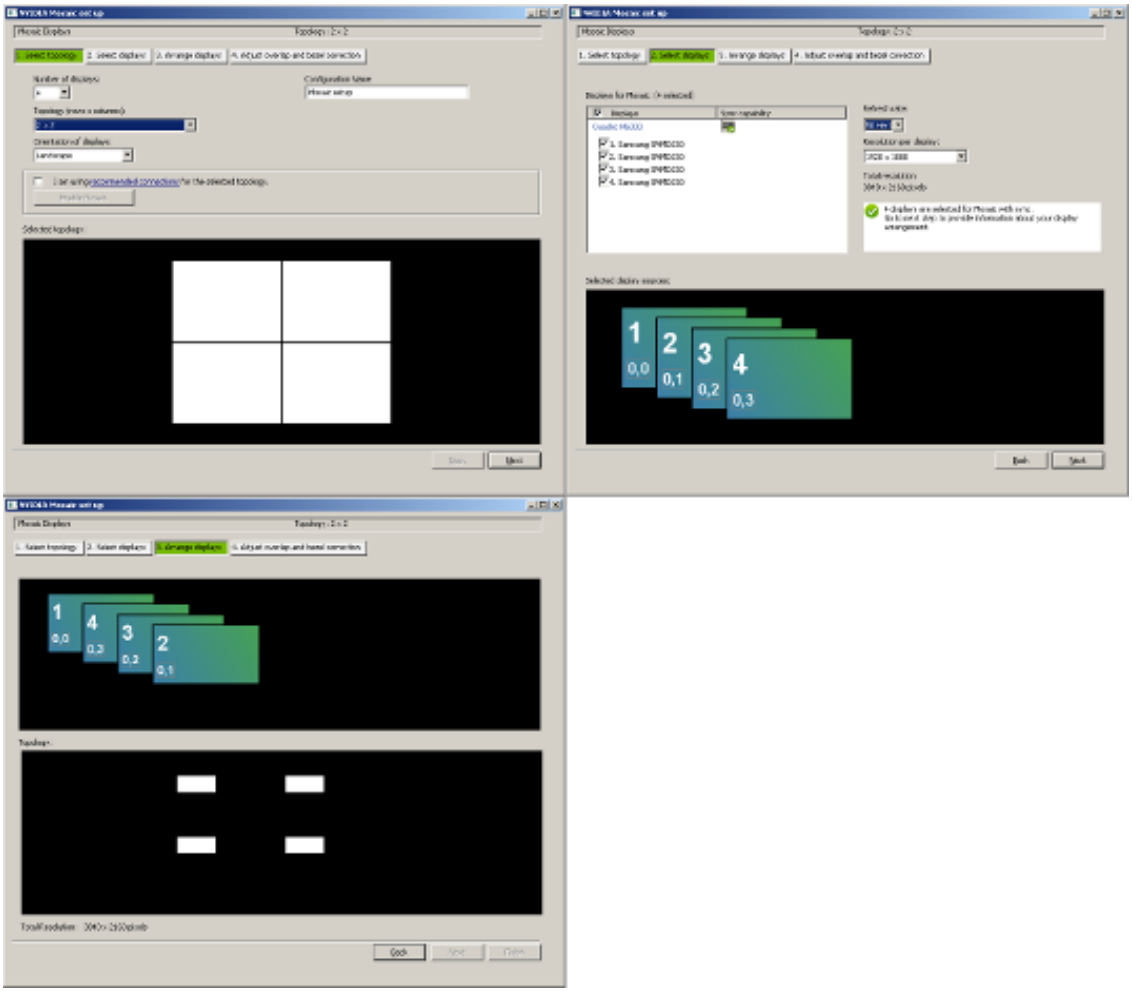
The procedure described below assumes that the NVIDIA Driver Configuration has been completed during initial setup of the computer running Viz Engine. If not, refer to the Graphics Boards chapter for details on setting up the NVIDIA hardware correctly for use with Viz Engine.

1. Enter NVIDIA Control Panel.
2. Disable synchronization of displays, by selecting **On another system**.



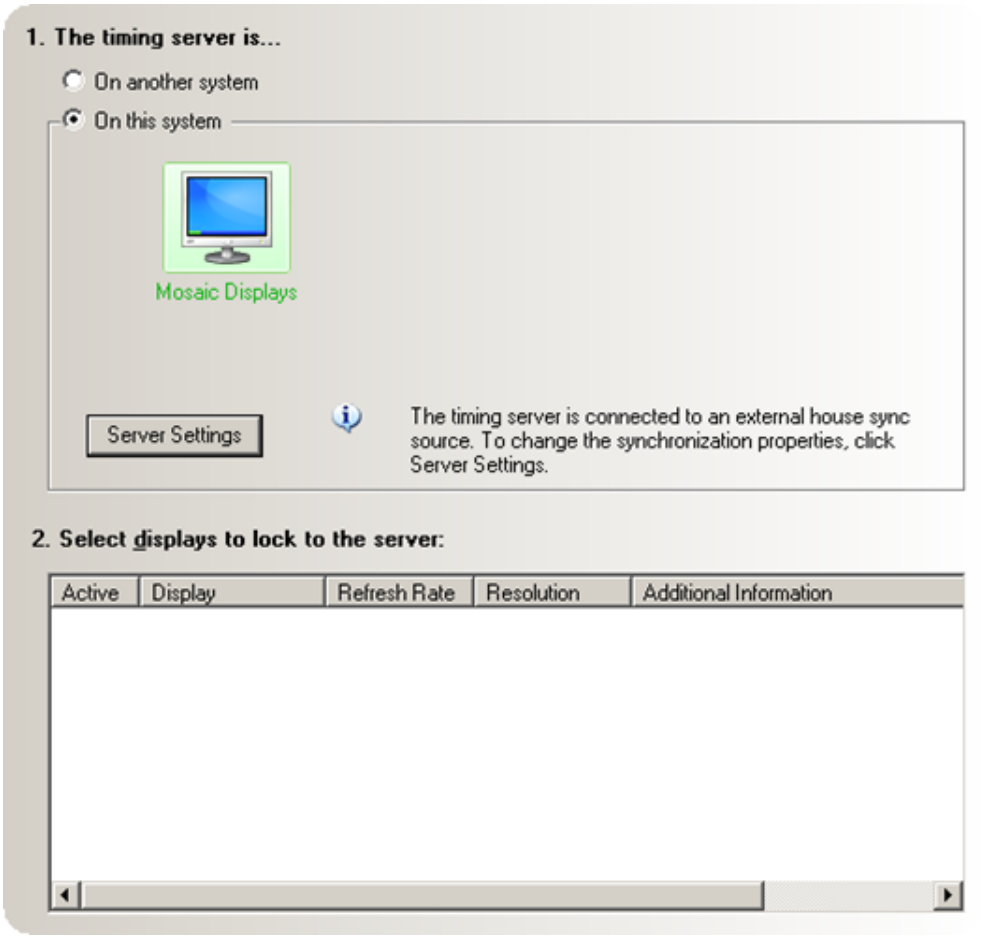
3. Establish a 50Hz Mosaic setup by selecting **Set up Mosaic** from the **Workstation** panel:
   - In step **1. Select topology**, make sure to leave the **I am using recommended connections for the selected topology** check box unchecked.
   - In step **2. Select displays**, select all displays that are to be used for the video wall, for example 1, 2, 3 and 4, and select **50 Hz** as frequency rate.
   - In step **3. Arrange displays**, drag and drop the displays according to the physical arrangement, not by the **On Screen Display** numbering. This can result in the monitors being arranged as 1, 4, 3, and 2, ensuring proper layout.

Copyright © 2026 Vizrt



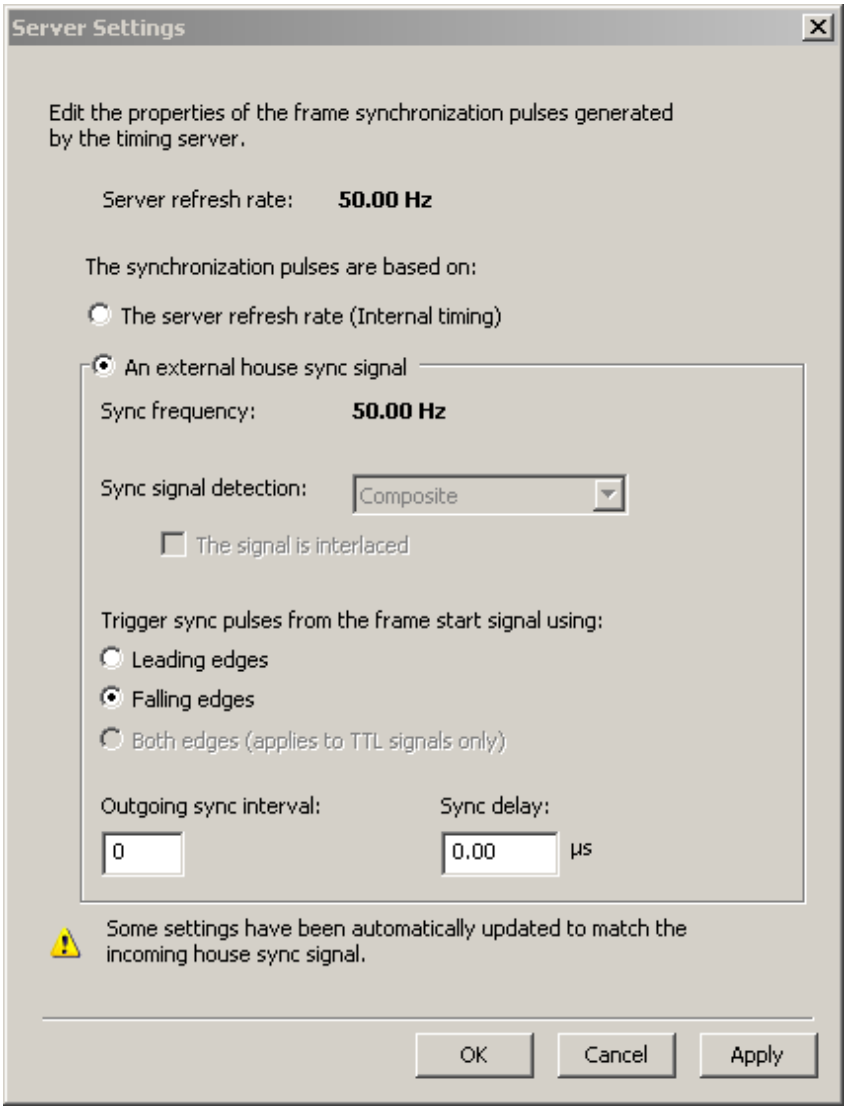
4. Enter section **Synchronize Displays** in the NVIDIA Control Panel and choose **On this system** for the question regarding the timing server, then click the **Server Settings** button.

Copyright © 2026 Vizrt

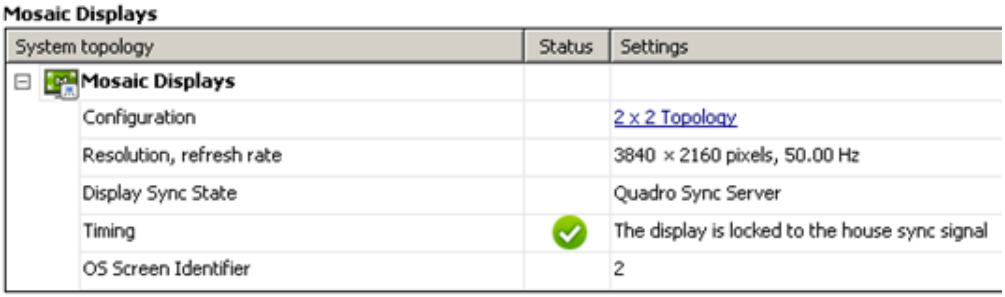


Make sure the presented server refresh rate matches the one of the incoming Genlock signal. Choose **An external house sync signal**. Leave all other settings as they are, and apply the changes.

Copyright © 2026 Vizrt

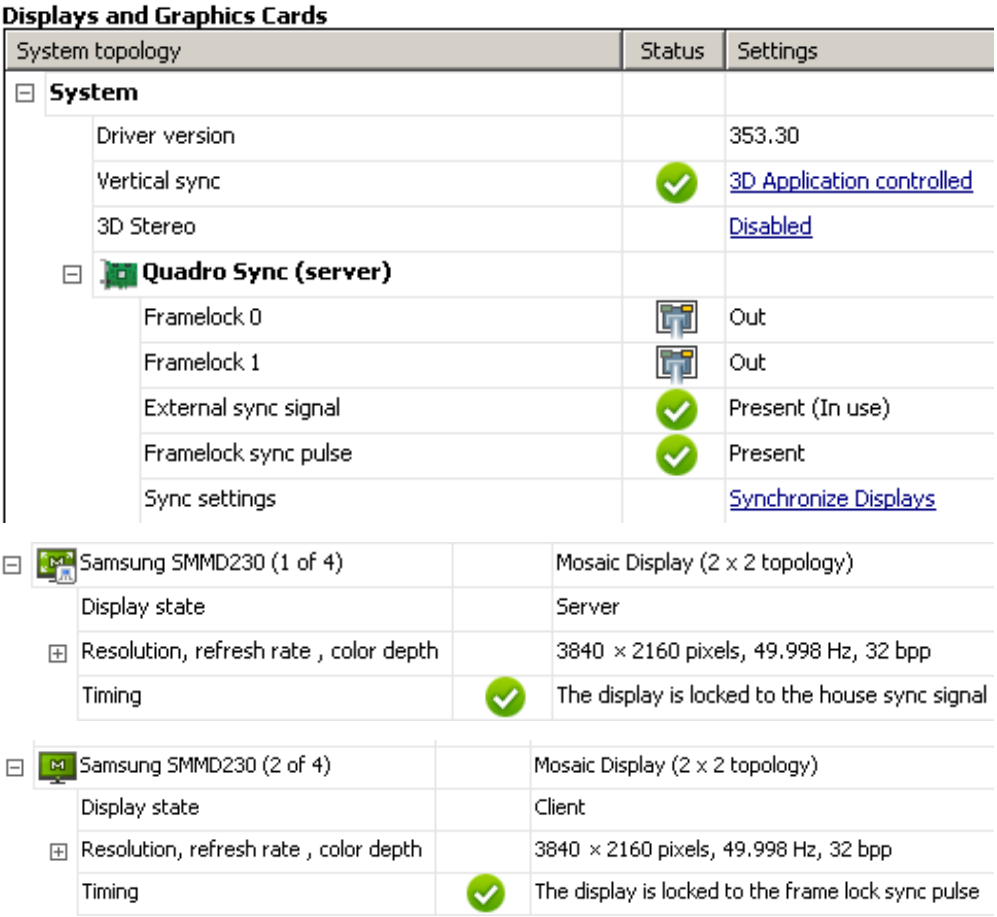


5. Check the **Topology** in the NVIDIA Control Panel:



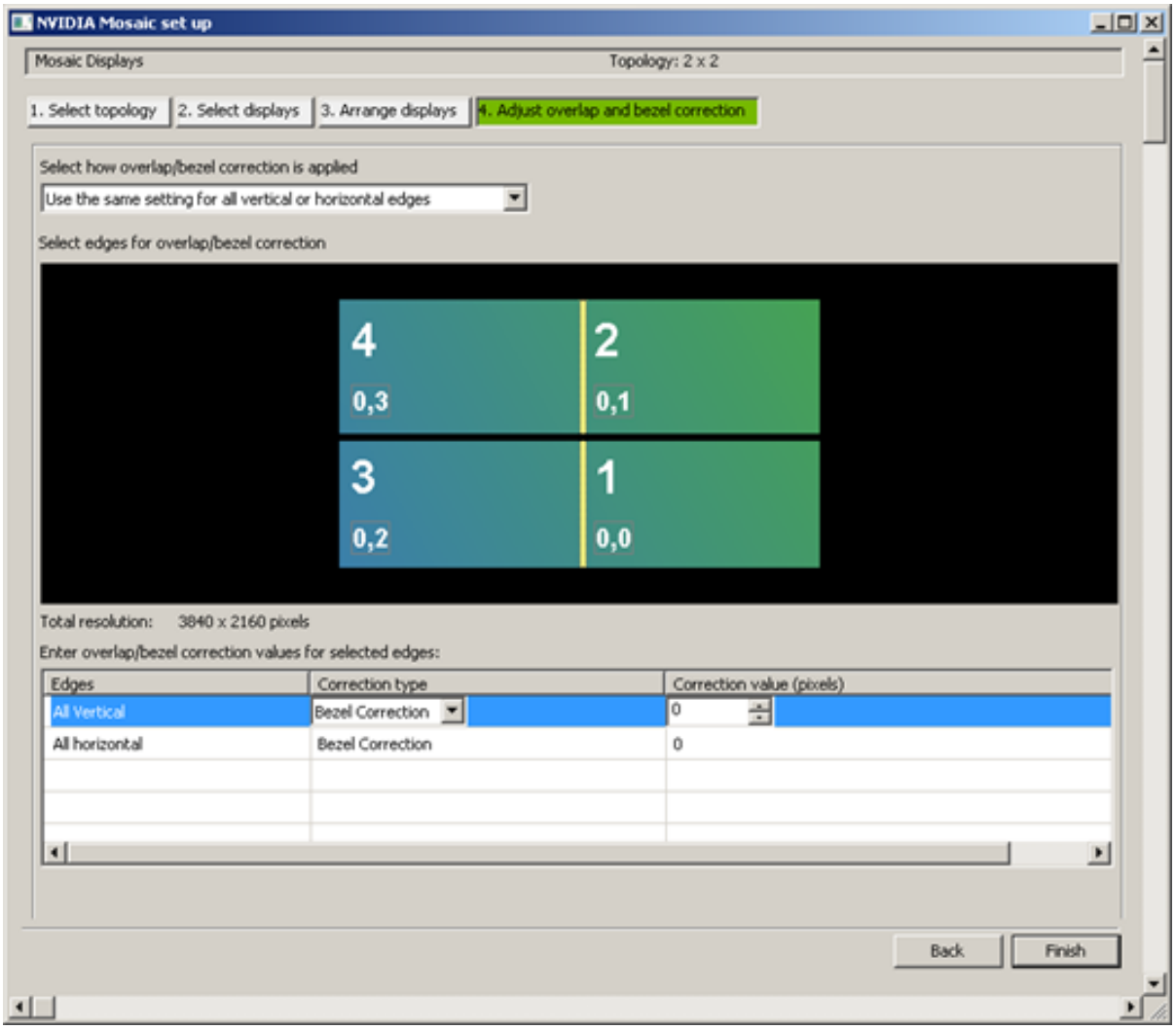
The **View System Topology** panel reports that **The display is locked to the house sync signal** for one of the connected displays. For the other connected displays, the report states that **The display is locked to the frame lock sync pulse**.

Copyright © 2026 Vizrt

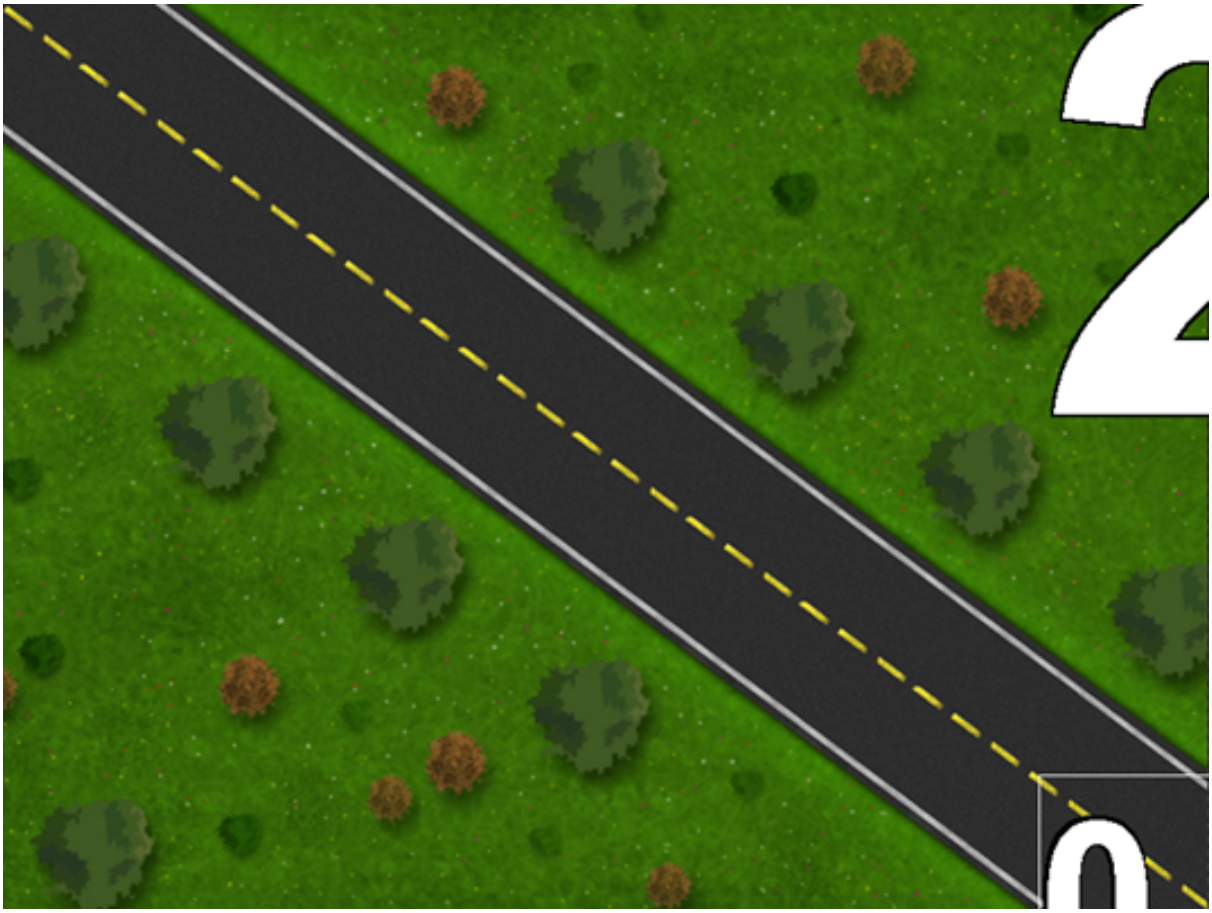


6. Check the LEDs on the NVIDIA Quadro Sync.
7. Configure Bezel/Overlap for Mosaic in the NVIDIA Control Panel:

Copyright © 2026 Vizrt



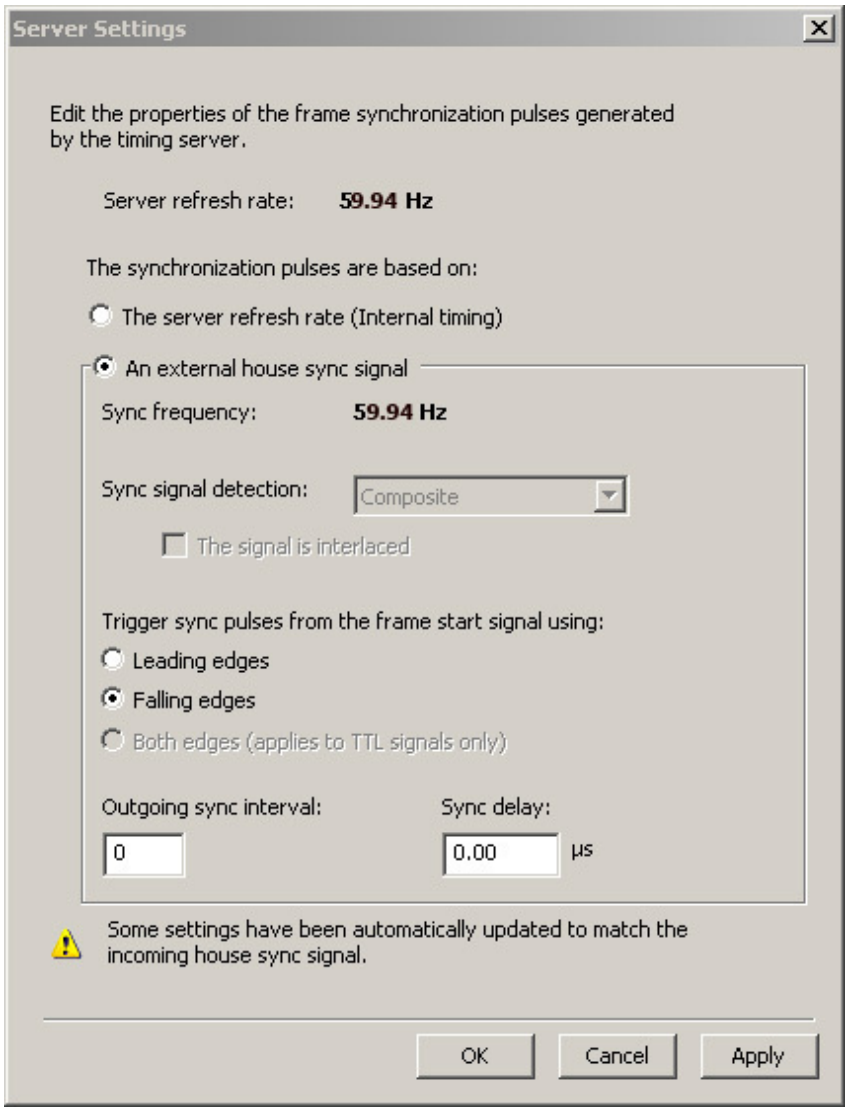
Copyright © 2026 Vizrt



Copyright © 2026 Vizrt

14.5.7 NVIDIA Mosaic Configuration for 1080i60M

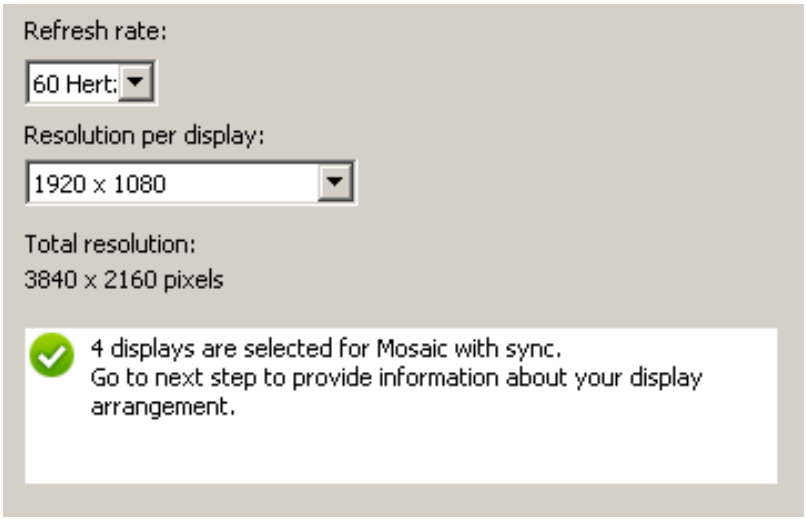
In case the video-wall displays in use natively supports a refresh rate of 59.94 Hz, the setup procedure described in NVIDIA Mosaic Configuration for 1080i50 can be followed, only selecting a refresh-rate of 59.94 Hz while creating the Mosaic:



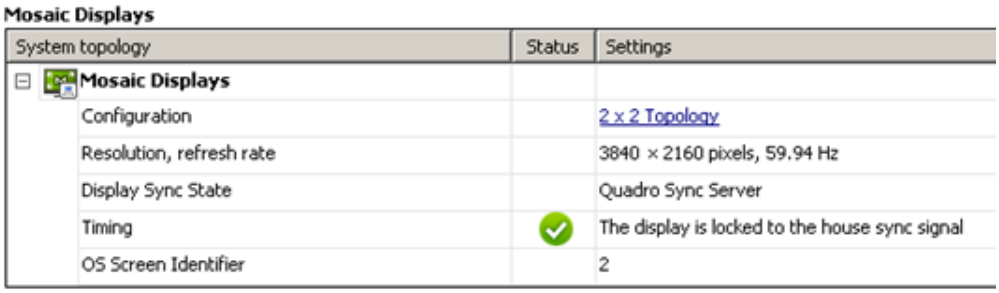
If this is not the case, a number of steps need to be performed in advance, as described in the section To Create a Custom 59.94 Hz Resolution. An **EDID-file** is *always* required to run monitors that do not natively support 59.94 Hz frequency at that frequency. Contact the display hardware vendor to obtain the correct EDID file for the monitors in use.

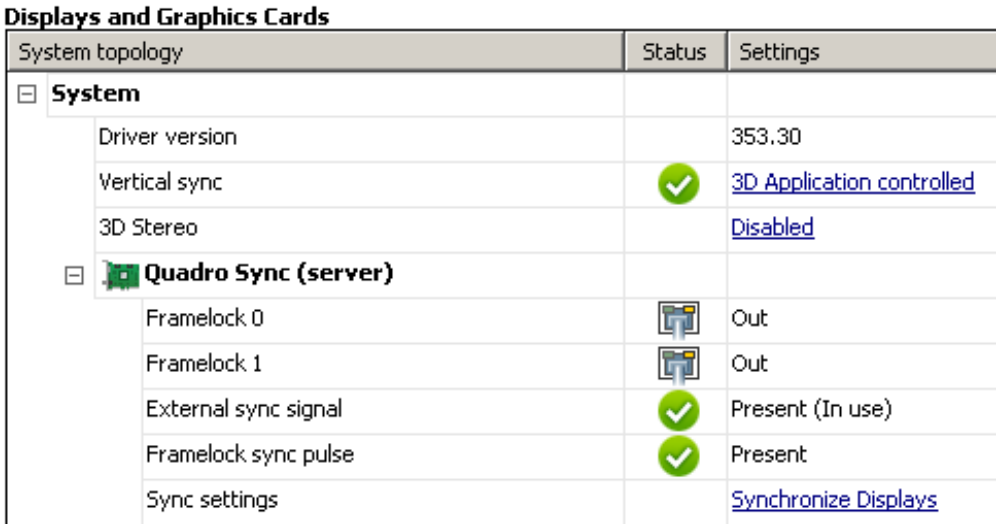
1. To configure Mosaic for 1080i60M, follow the steps described in the NVIDIA Mosaic Configuration for 1080i50 section. Make sure to select a refresh-rate of 59.94 Hz during step 2 of the Mosaic creation process. Depending on the NVIDIA driver installer, sometimes 59.94 Hz is not available from the **Refresh rate** drop-down list, even if the loaded EDID-file enables this frequency rate. If this is the case, select 60 Hz:

Copyright © 2026 Vizrt



2. Once Mosaic is configured and the G-Sync device has been locked to the house-sync, the topology in the NVIDIA control panel should look as follows, with the report for one of the connected displays stating **The display is locked to the house sync signal**.



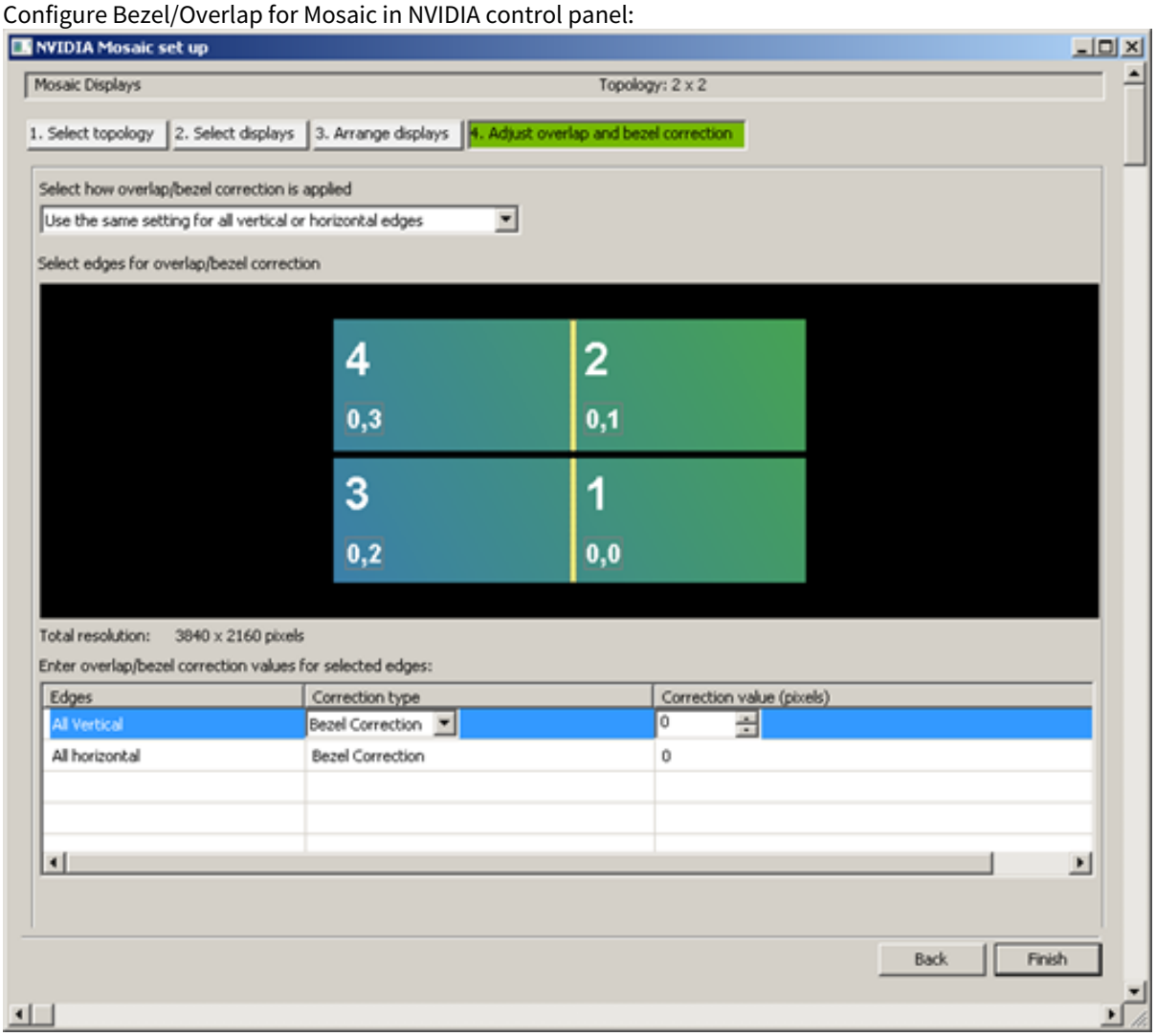




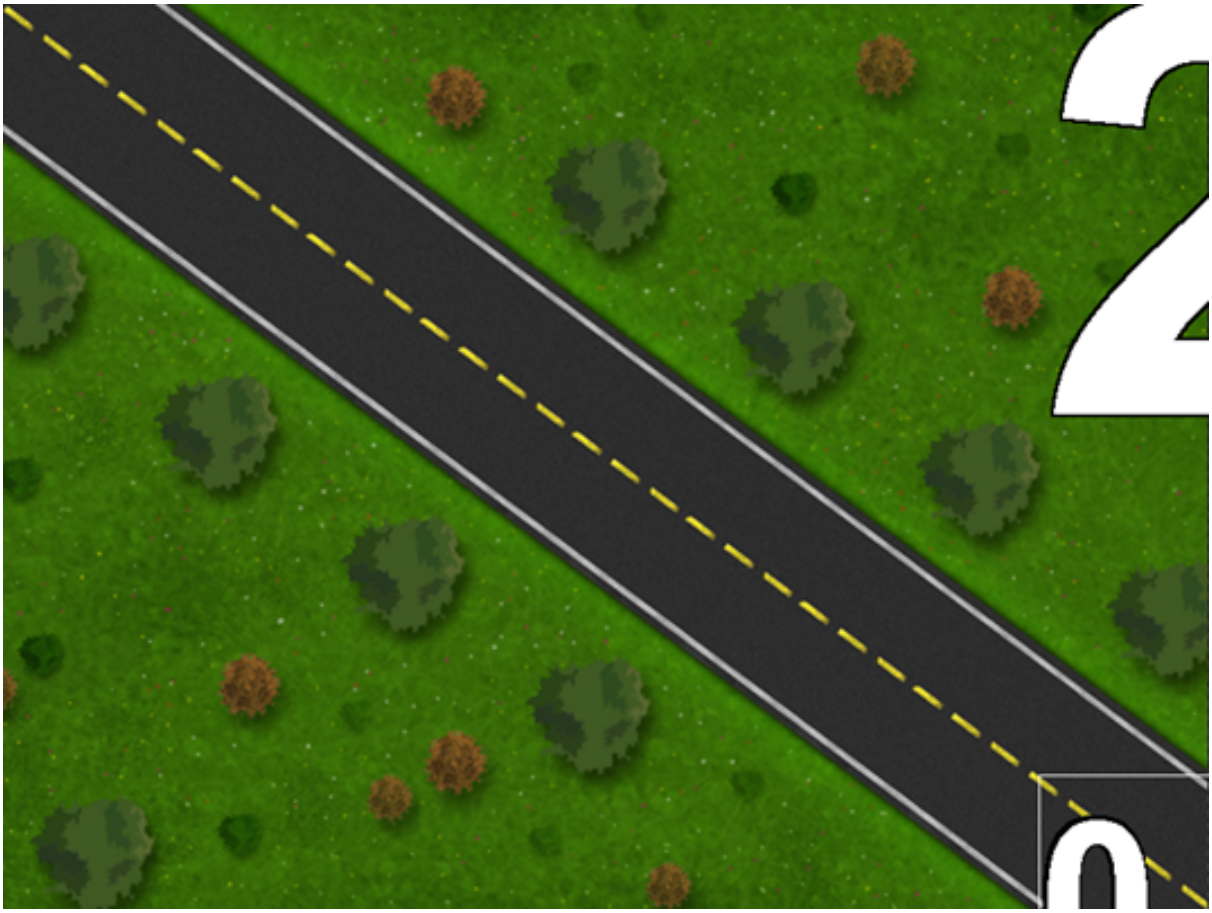
The report for all other displays should be that **The display is locked to the frame lock sync pulse**.

Samsung SMMD230 (3 of 4)		Mosaic Display (2 x 2 topology)
Display state		Client
Resolution, refresh rate , color depth		3840 × 2160 pixels, 59.939 Hz, 32 bpp
Timing	✓	The display is locked to the frame lock sync pulse

3. Check the LEDs on the NVIDIA Quadro Sync.
4. Configure Bezel/Overlap for Mosaic in NVIDIA control panel:



Copyright © 2026 Vizrt



5. Proceed with configuring the required Viz Engine Video Wall Configuration Settings.

14.5.8 Custom Resolution for 59.94 Hz Refresh Rate

Video-wall displays may not natively support a refresh rate of 59.94 Hz. Such displays *always* require an **EDID-file** to be able to run at that frequency. Contact the display hardware vendor to obtain the correct EDID file for the monitors in use. Furthermore, a number of steps must be carried out before initiating the NVIDIA Mosaic Configuration for 1080i60M.

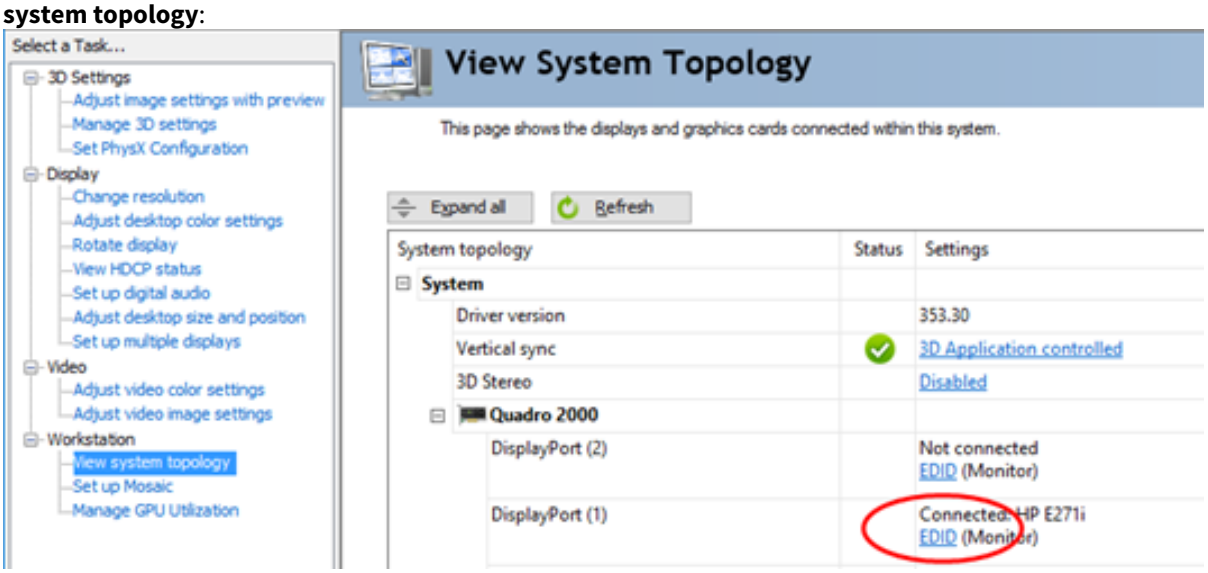
This section contains information on the following topics:

- To Load an EDID File
- To Create a Custom 59.94 Hz Resolution
- To Change from 59.94 Hz to 50 Hz Refresh Rate

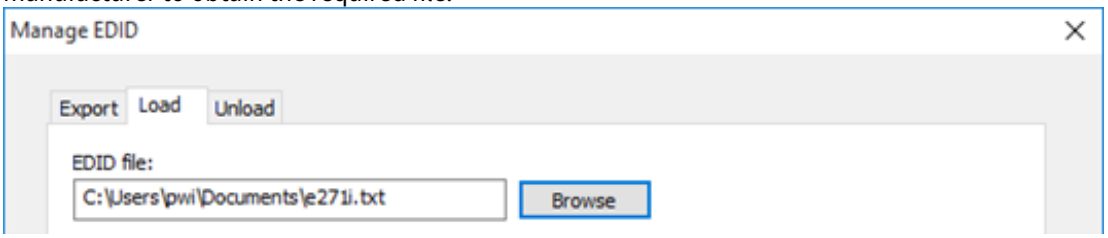
Caution: Loading unsupported or corrupted EDID settings for a display may render the source device unable to output any video signal to the display. Please refer to the information supplied by the display hardware vendor for information on the displays capabilities or limitations.

To Load an EDID File

1. In the **NVIDIA Control Panel**, expand the **Workstation** section of the **Select a Task...** menu and click **View system topology**:



2. Click the **EDID** link on one monitor, and select the **Load** tab.
3. Click the Browse button and select the EDID file to be loaded. This file is usually included on the CD or DVD containing drivers, utilities and documentation for the display. If not, please contact the display manufacturer to obtain the required file.



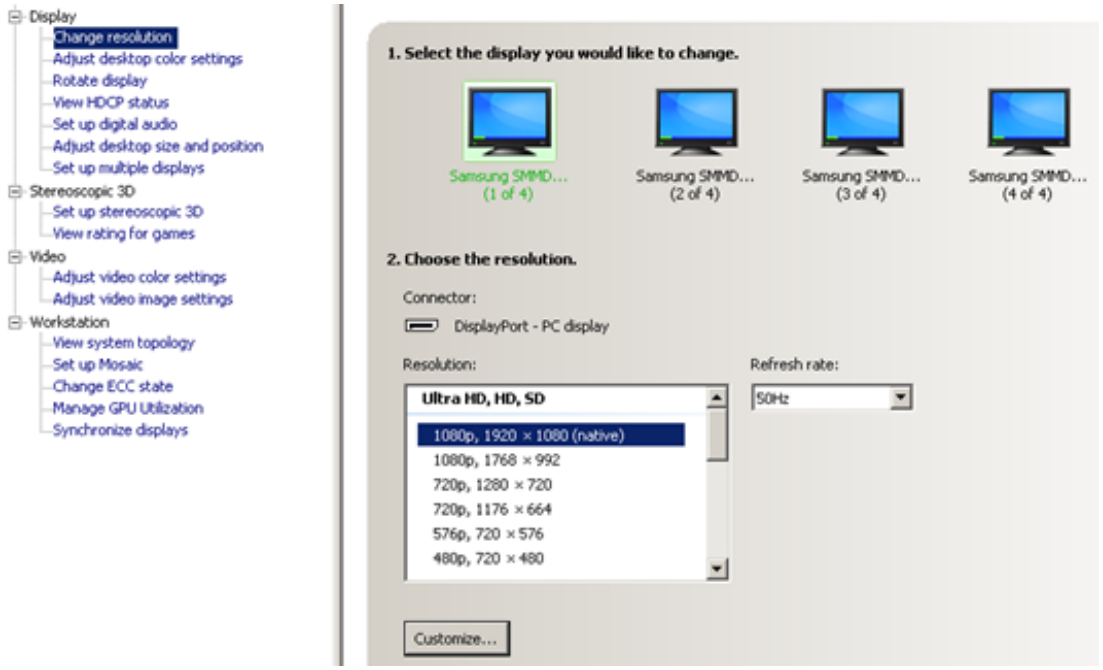
4. In the **Select Connector to force EDID** panel, check all connected displays, then click **Load EDID**.

Copyright © 2026 Vizrt

5. When the EDID file has been applied to all displays, check the properties of every display in the **View System Topology** section. They must all run on 59.94 Hz.
6. Proceed To Create a Custom 59.94 Hz Resolution.

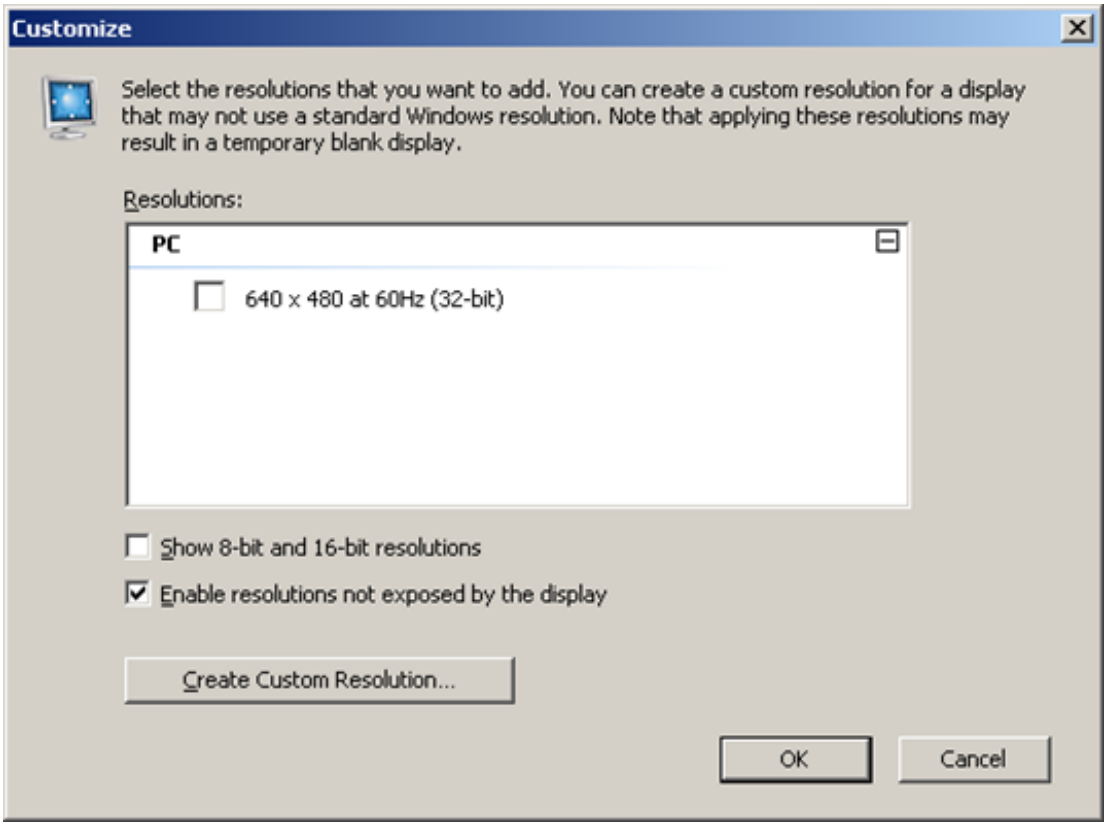
To Create a Custom 59.94 Hz Resolution

1. Enter the **Change resolution** section of the NVIDIA control panel.
2. In the first section, **Select the display you would like to change**, select display number 1 and click the **Customize...** button:

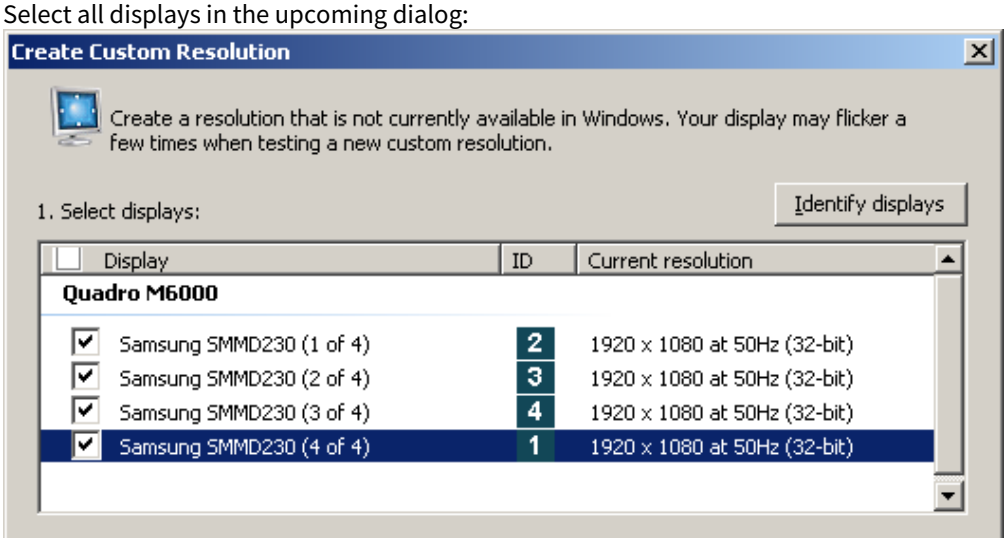


3. In the upcoming dialog, tick the **Enable resolutions not exposed by the display** check box, and click **Create Custom Resolution...**.

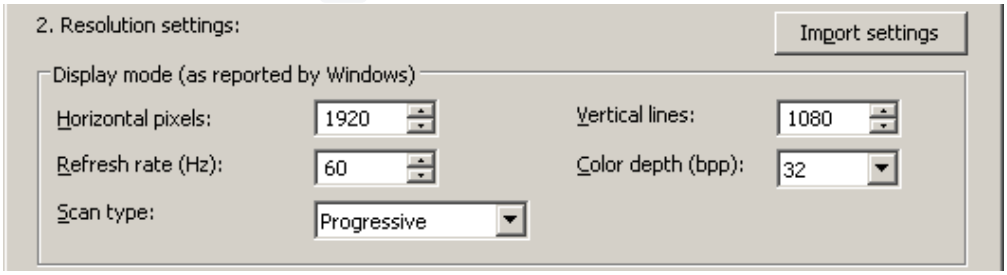
Copyright © 2026 Vizrt



4. Select all displays in the upcoming dialog:

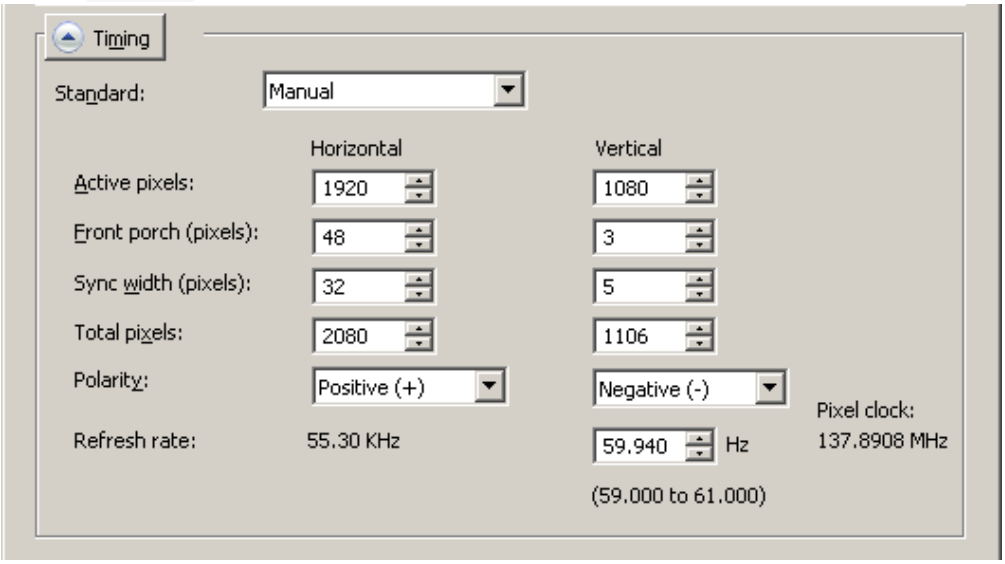


5. Enter a **Refresh rate (Hz)** of `60` Hz and leave all other settings to their default values:



Copyright © 2026 Vizrt

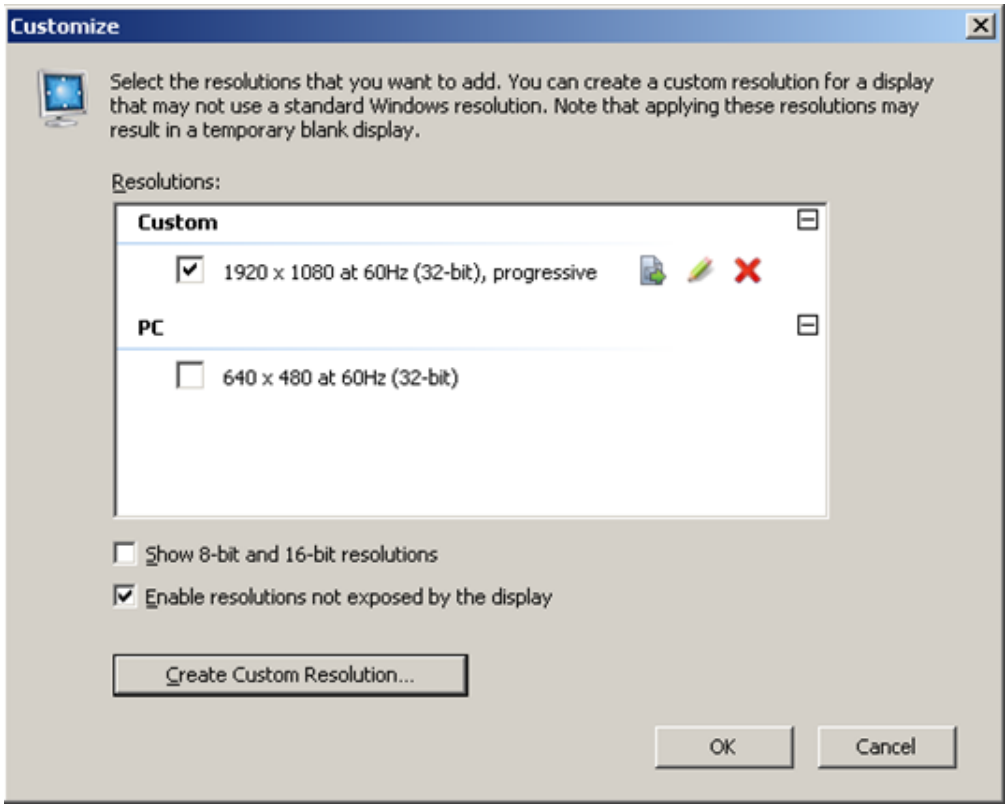
In the Timing panel, select `Manual` timing adjustment from the **Standard** drop-down list. Set the **Refresh rate** to `59.940` Hz, and leave all other settings to their respective default values:



6. Click the **Test** button. The connected monitors might go black for several seconds while adjusting to the new refresh rate. Do not press any key on the keyboard or buttons on the mouse while this goes on. When the screens returns to normal, acknowledge the new settings by pressing **Yes** in the upcoming dialog.
7. The newly created resolution now shows up as illustrated below and has been applied to all displays.

IMPORTANT! Please note that for 59.94 Hz setups, the server refresh rate might report to be 60 Hz when the external house sync is 59.94 Hz. Even though the custom resolution states to be 60 Hz, the correct value under the hood is 59.94 Hz.

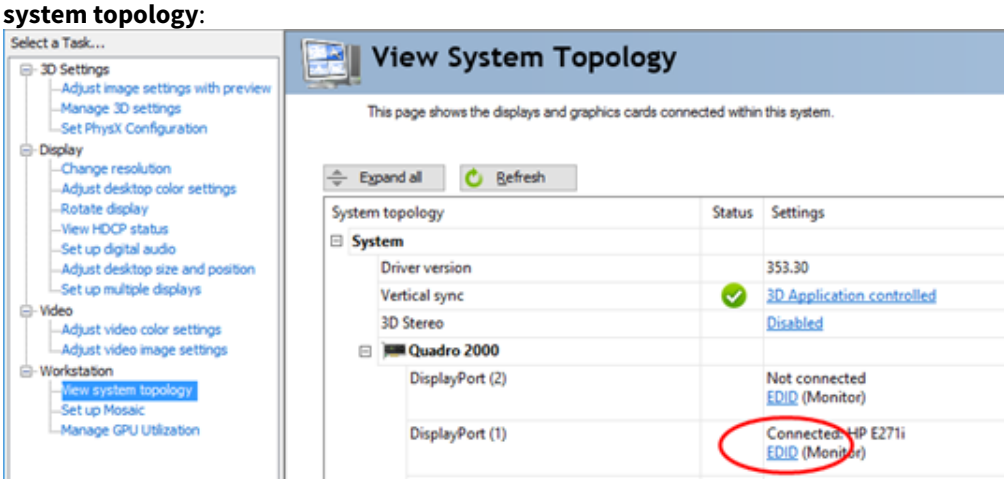
Copyright © 2026 Vizrt



To Change from 59.94 Hz to 50 Hz Refresh Rate

If changing the frequency from 59.94 Hz back to 50 Hz, the EDID file must be unloaded for all connected displays, followed by rebooting the machine. Otherwise, Mosaic can not be applied.

1. In the **NVIDIA Control Panel**, expand the **Workstation** section of the **Select a Task...** menu and click **View system topology**:



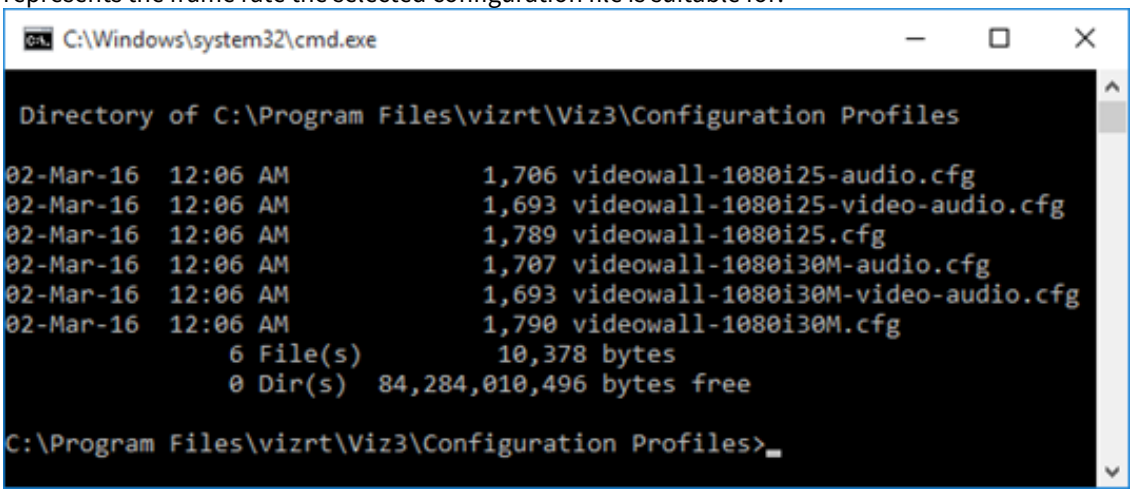
2. Click the **EDID** link on one monitor, and select the **Unload** tab.
3. Select all connected displays and click the **Unload** button.
4. Repeat the step above for all monitors one at a time.
5. Restart the computer and proceed with the NVIDIA Mosaic Configuration for 1080i50.

Copyright © 2026 Vizrt

14.5.9 Viz Engine Video Wall Configuration Settings

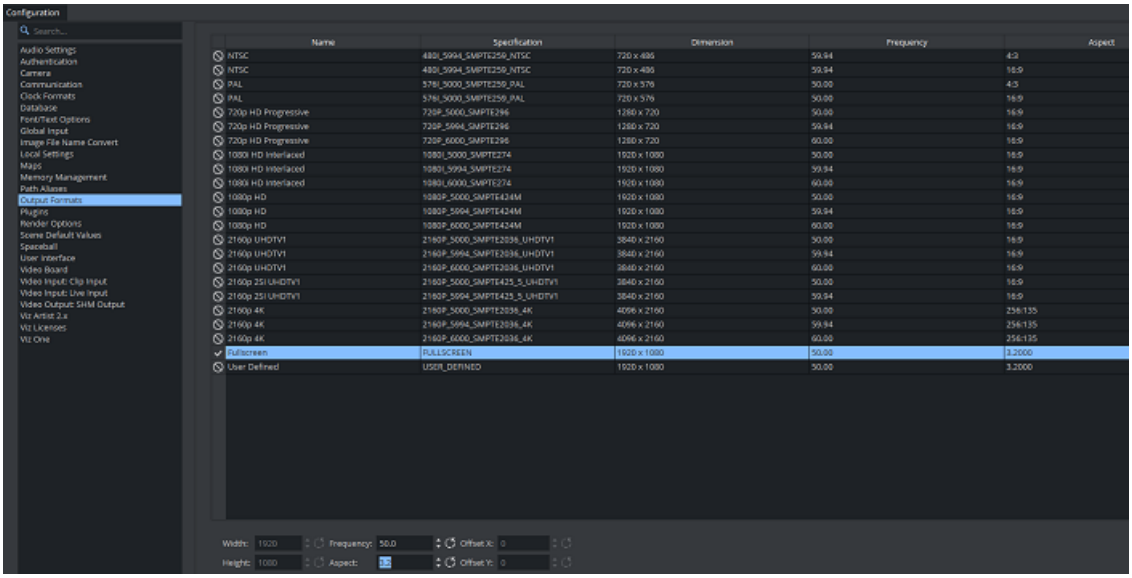
To Configure Viz Engine for Video Wall

1. Open Viz Configuration, and create a backup of the existing configuration settings by saving them to a new file.
2. Import a configuration profile for the desired input format from the selection of Installed Configuration Profiles. These ship with the Engine installation and are located in the *<viz install folder>\Configuration Profiles* folder. The input format files are named *video-resolution-*.cfg*, where the asterisk represents the frame rate the configuration is suitable for.
3. Next, import a Video wall configuration profile, where the frame rate matches the one loaded for the video hardware in the previous step. As with the input format profiles, these are located in the *<viz install folder>\Configuration Profiles* folder. These configuration files are named *videowall-*.cfg*, where the asterisk represents the frame rate the selected configuration file is suitable for:



4. Adjust the Video Output in Viz Config:
   - Verify that *Fullscreen* is the selected format in the output format specification list. The output format MUST be *Fullscreen* and not *User Defined*. Only **Fullscreen** and **Videowall mode on** forces the engine to go into Videowall Mode.
   - The frame rate should have been set when loading the *videowall-*.cfg* configuration profile in step two. Ensure that the correct values are set for the required output. For PAL systems, set the **Frame** value to `50`. For NTSC systems, set the **Frame** value to `59.94`.
   - Set the **Aspect** value to the aspect ratio of the video wall. The aspect ratio decimal value is found by dividing the number of pixels, width by height. This value depends on the actual video wall display layout and needs to be adjusted if changes to the display layout are made.

Copyright © 2026 Vizrt



5. Verify the following Video Output settings:
   - **Video wall/Multi-display** should be set to Active.
   - Click the **Show Video Output Config. Editor** button and verify that the required Genlock is set. For configurations with the Matrox X.Mio3 video board, Genlock must be set to **Auto**. Other configurations can be set to **Auto**, **Blackburst** or **Tri-Level**, as required.
6. Under Render Options, disable **On Air Mouse Cursor**.
7. In the Video Input section, enable the required number of **Live** and **Clip Channel** sources. Unused live inputs should be deactivated, as activating them could have an impact on performance.
8. In the Database section, specify which Viz Graphic Hub to use and provide the login credentials.
9. Start Viz Engine without a User Interface, by executing the following command from the command line: `<viz install folder>\viz.exe -n -w`.

IMPORTANT! On Windows 10, it might happen that Viz renders at half the speed (25fps) and renders full speed if the focus is on another window. In this case, change the startup parameter to `<viz install folder>\viz.exe -n -y`.

11. Load a scene containing a single live-video or clip source as a texture and scale it up to full screen. If running Viz Artist, switch to On Air Mode. Watch for drops in the video wall output within the first 60 seconds.
12. Verify the configuration of the NVIDIA driver by checking the Topology Inspector, and check the Viz Engine configuration. This is especially important when setting up a 59.94 Hz video-wall, as a wrong refresh-rate of 60 Hz could get applied to the displays by mistake.
13. In general, a proper Mosaic setup is automatically synchronized with Viz Engine. However, it can be necessary to force Viz Engine to synchronize. This is necessary for all setups with multiple graphics cards. To force synchronization, issue the following command in the Engine Console: `send RENDERER JOIN_SWAPGROUP 1`.

IMPORTANT! Do not execute this command unless needed, as this may have an impact on performance.

Copyright © 2026 Vizrt

14. If the computer is rebooted or Viz Engine restarted, the command needs to be sent again. To avoid doing this manually every time the Engine starts, set the `swapgroup` value in the **RENDER_OPTIONS** section of the configuration file to `1`. The default value is `0`.

Tip: This synchronization can also remedy certain tearing effects. Create a simple scene with a bar running left to right. If any of the connected monitors display tearing effects, try sending the above command.

Audio Output

For performance reasons, in a Video Wall setup the video out channels are usually set to **Unused**. To configure a Video Wall with embedded or AES audio output, the video out channel must be mapped to the selected Matrox channel. Set **Map to Viz Channel** in the Matrox **VideoOut Properties** in Viz Configuration.

Note: When running a Video Wall with audio out, even though the video out channel is mapped, the SDI output is black.

It is highly recommended to not use the audio output of the installed (SDI) video board. Audio can usually also be grabbed from one of the HDMI/Display Port connectors of the NVIDIA board.

For Audio Output in a ST2110 environment, please see Large Canvas Output.

NDI

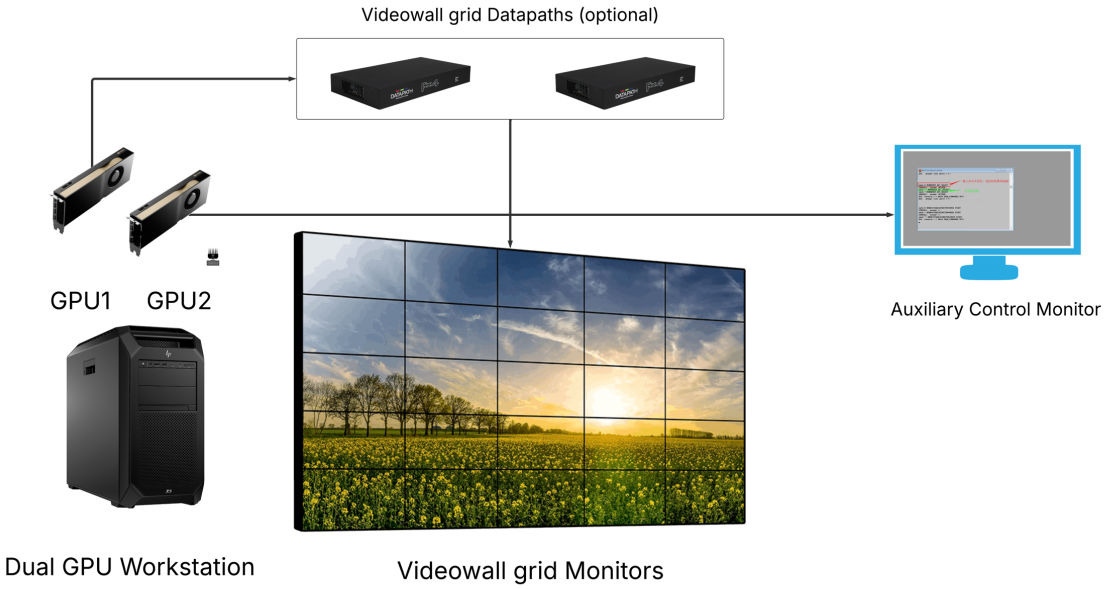
Please note, that NDI input is currently not tested on videowall setups.

Copyright © 2026 Vizrt

14.5.10 Mosaic + 1 Setup

Concept

A standard Mosaic video wall setup, using multiple displays connected to NVIDIA GPUs, can be extended with **an additional monitor** driven by a separate GPU as a control or auxiliary display. This extra monitor operates **outside** the Mosaic grid, but still extends the desktop, allowing you to run applications, monitoring tools, or system utilities (like a Viz Engine console, performance monitor, or task manager) without interfering with the main videowall content.



Setup Steps

Connect Hardware

- Connect your primary videowall displays to NVIDIA Quadro GPUs (via Datapath: Optional, in case the number of monitors is larger than the number of display ports).
- Connect the auxiliary monitor to a separate GPU.
- If NVIDIA G-sync is used in the setup, make sure it is connected to the grid monitor's GPU.

> ⚠ **Note:** The GPUs must be from the same family.

Install Drivers and Configure Mosaic

- Install the recommended NVIDIA driver corresponding to the installed Viz Engine version.

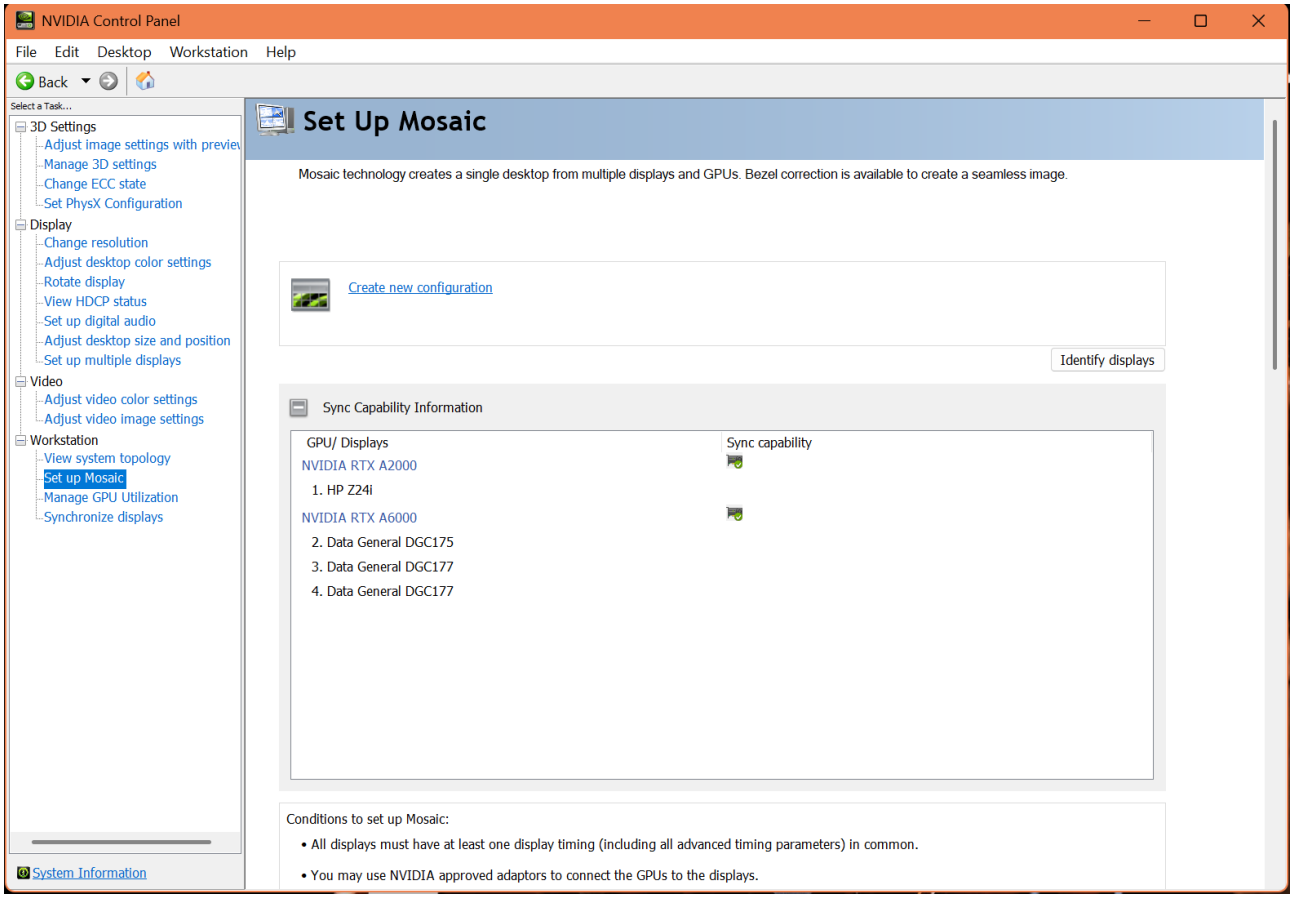
Copyright © 2026 Vizrt

- Configure NVIDIA Mosaic excluding the additional monitor on the second GPU.

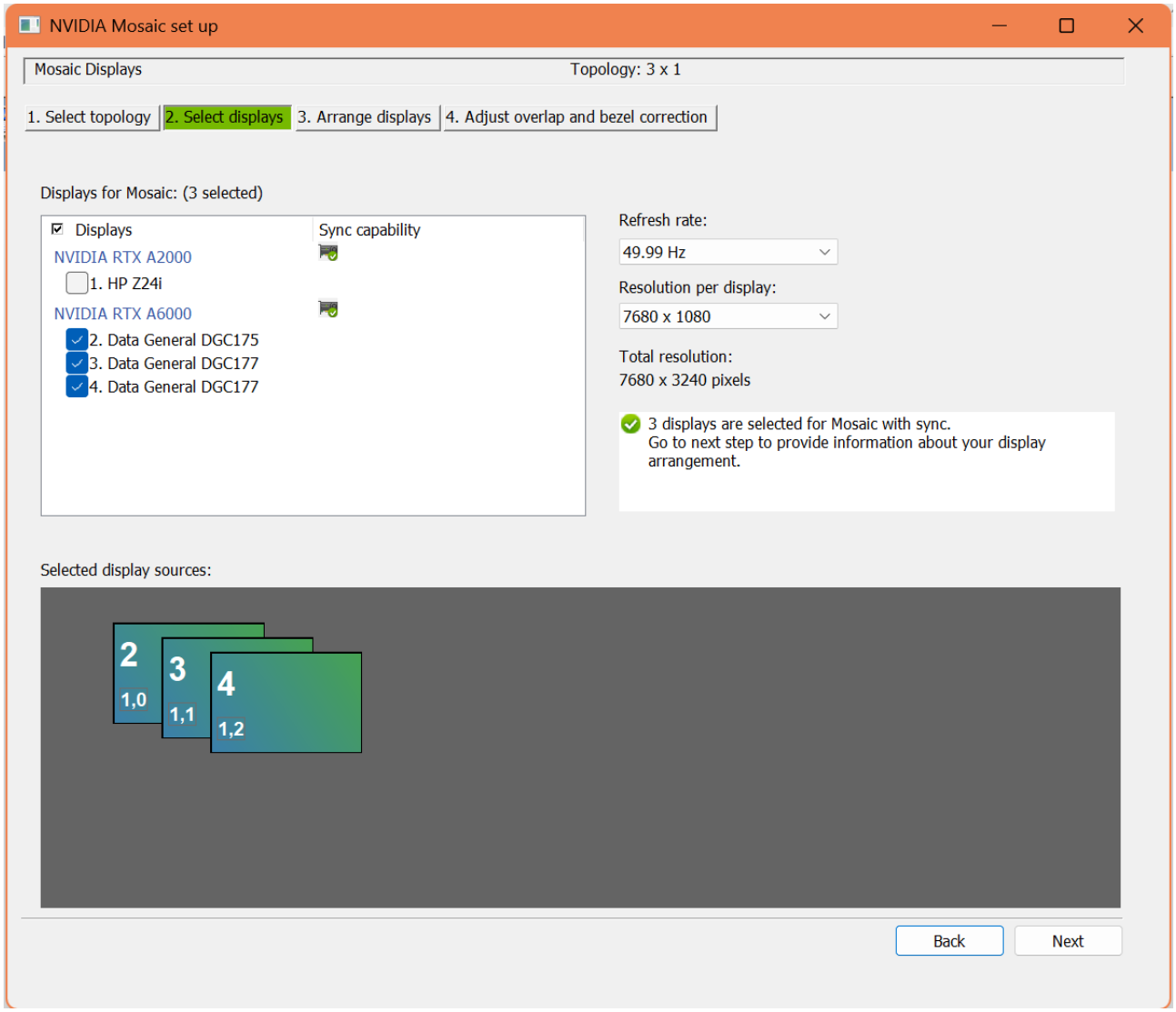
Note: If the NVIDIA GPU driver is already installed and an additional GPU is added, the additional GPU may not be detected until the NVIDIA GPU driver is reinstalled.

Identify Display Layout

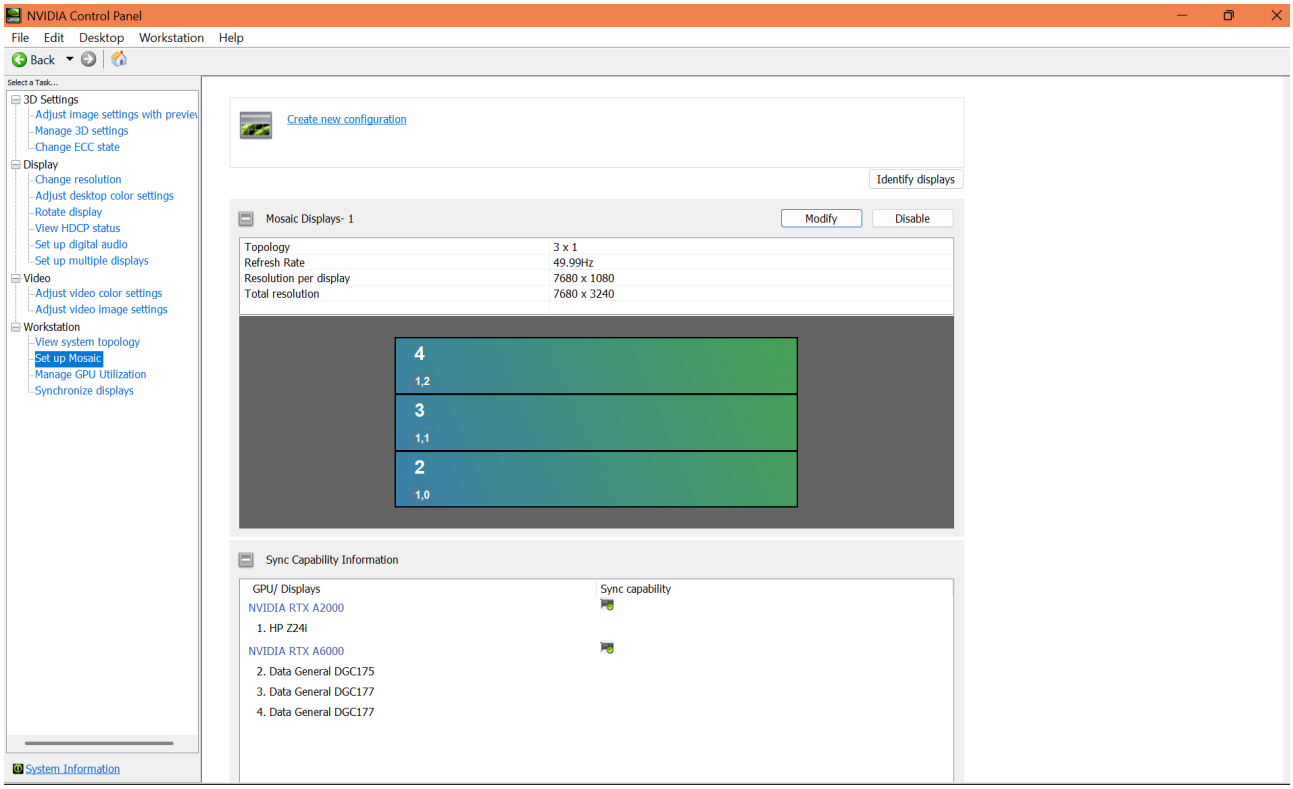
- Open the **NVIDIA Control Panel > Workstation > Set up Mosaic**.
- Select Identify Displays to determine which display is on which output.



Copyright © 2026 Vizrt



Copyright © 2026 Vizrt



Apply and Confirm Layout

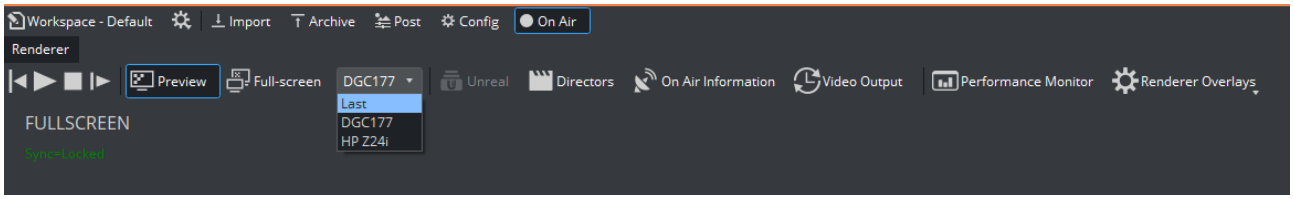
- Wait for the layout to apply, and the screens refresh (may take a few seconds).
- Verify that:
  - The Mosaic videowall functions as a unified display, and the mosaic looks as expected.
  - If you are using a G-sync, ensure all monitors are synchronized (**Workstation > View System Topology**).
  - The control monitor appears as a separate extended desktop, distinct from the mosaic.

Using the Control / Auxiliary mMonitor

The Viz Engine should be started on the mosaic videowall grid GPU:

- Starting Viz Engine in video wall mode using the starting parameters `-n -w -G<n>`.
  - The renderer window appears on the video wall monitors.
  - The Viz Engine console can be moved to the control monitor.
- Starting Viz Engine in Viz Artist mode using the starting parameters `-u1 -y -G<n>`.
  - The Viz Artist and console windows can be moved between monitors.
  - When switching to On Air mode, make sure to select the correct monitor

Copyright © 2026 Vizrt



Information: <*n*> represents the index of the GPU connected to the Mosaic, depending on how Windows detects the GPU, this can be 1 or 2.

Information: When Viz Artist is hosted in the control monitor, it was observed that the scene tree panel is not resizable, as a workaround, detach the scene tree group, or the scene editor group.

Information: Specific to Mosaic +1 setup, changing *Vulkan/OpenGL present method* setting in the NVIDIA Control Panel to *Prefer layered on DXGI swapchain* showed stability in FPS metrics.

See Also

- NVIDIA Support
- Mosaic Utility Driver

Copyright © 2026 Vizrt

14.5.11 Large Canvas Output

Prerequisites

The feature adds support for custom non-broadcast resolution canvases targeting SMPTE ST 2110-driven video walls.

> **Note:** Make sure your required output is covered by the appropriate license(s). For example, 4K, 8K, …

Supported Video Hardware

- Matrox X.mio5 D25
- Matrox DSX LE5 (D25, Q25)
- Matrox DSX LE6

Known Limitations

- The canvas resolution cannot exceed 8192 * 4320 pixels for supported X.mio5 and DSX LE5 boards, and 15360 * 2160 pixels for DSX LE6 boards.
- The canvas resolution has a minimum resolution of the selected output flow resolution, 1280x720 pixels with 720p.
- When using output flows with different resolutions, the larger output flow dictates the minimum resolution.
- The canvas width needs to be a multiple of 64 and the height needs to be even.
- The single cutout flows need to be progressive and in broadcast resolution.
- There are at most eight cutout flows.
- The minimum value for the horizontal position is `0` pixels. The maximum value is the canvas resolution width, minus the output resolution width. The position must be a multiple of two pixels. Odd values are rounded to the lower even value.
- The minimum value for the vertical position is `0` lines. The maximum value is the canvas resolution height, minus the output resolution height.
- The D25 cards can only handle a maximum of two UHD outputs.

Configuration

There are several settings necessary to enable the large canvas feature. These settings must be set directly in the Viz Engine configuration file (in *%PROGRAMDATA%\vizrt\VizEngine*).

1. Write all related settings into the configuration file set `VerboseConfig = 1`, then start Viz Artist/Viz Engine once, and shut it down again.
2. Set `Matrox.LargeCanvasMode = 1` to enable the feature.
3. Set the output format to *USER DEFINED* and the desired resolution using the dimensions and frequency of your video wall (for example, `output_system = USER_DEFINED 7680 1080 50.00 0 0`).
4. Depending on the number of outputs required to cover the entire canvas, configure each in the following way (example):

Copyright © 2026 Vizrt

The output types must be set to IP, and the output content must be set to *1* (Program), for all needed output flows. Set the video system to the broadcast resolution required and the position of the upper left corner of the cutout. Numbers in configuration setting names correspond to the used Viz Engine output channel.

- *out_type1 = IP1*
- *Output1.Content = 1*
- *Matrox0.VideoOut1.CanvasVideoSystem = 1080P_5000_SMPTE424M*
- *Matrox0.VideoOut1.CanvasHorizontalPosition = 0*
- *Matrox0.VideoOut1.CanvasVerticalPosition = 0*

5. When using full resolution video walls it is recommended to set *output_bpc = 10*.

Important: Step 5 is mandatory for video walls running at 59.94 fps.

Key Output

Fill and Key output with large canvas is supported and enabled with the setting `Matrox.LargeCanvas.ContainsAlpha = 1` in the configuration file. This enables a key signal on all IP output flows. The usual per-output alpha configuration is ignored. The key signal uses the same video system and canvas position as the associated fill.

Important: Only odd-numbered IP outputs can be used because the even-numbered connectors are used for the key signal. This affects all settings mentioned in the Configuration section.

Attention: The additional bandwidth required means that the overall resolution (numbers of pixels) of a fill and key system is limited to half of that of a fill-only system.

Supported Features

Feature	X.mio5 D25	DSX LE5 D25	DSX LE5 Q25	DSX LE6
Large Canvas support	✅ Yes	✅ Yes	✅ Yes	✅ Yes
Maximum canvas resolution	**8192 × 4320**	**8192 × 4320**	**8192 × 4320**	**15360 × 2160**
Wall orientation best suited	UHD / 16:9 Linear	UHD / 16:9 Linear	UHD / 16:9 Linear	**Very wide / panoramic**
Maximum number of Large Canvas outputs	**8**	**8**	**8**	**8**
UHD outputs in Large Canvas	**2**	**2**	**4**	**4**

Copyright © 2026 Vizrt

Vertical UHD stacking (2 × 2160 = 4320 px)	✅ Supported	✅ Supported	✅ Supported	❌ Not supported (height limit 2160)
UHD 2×2 grid (4 independent UHD outputs)	❌ Not supported	❌ Not supported	✅ Supported	❌ Not supported
Mixed HD / UHD on same canvas	✅ Yes	✅ Yes	✅ Yes	✅ Yes
Progressive outputs only	✅ Yes	✅ Yes	✅ Yes	✅ Yes
Broadcast resolutions only	✅ Yes	✅ Yes	✅ Yes	✅ Yes
Canvas geometry constraints	Width × 64 Even height	Same	Same	Same
Fill & Key with Large Canvas	✅ Yes	✅ Yes	✅ Yes	✅ Yes
Fill & Key impact	**1 logical output = 2 flows (Fill + Key); halves the number of usable outputs**	Same	Same	Same

Copyright © 2026 Vizrt

15 Vizrt Viz Engine Monitor

This monitor is a replacement for the SMNP component of Viz Engine. It uses OpenTelemetry metrics and OpenTelemetry logs to provide the information from OS shared memory, written by a Viz Engine instance if configured to do so.

- **Application Name:** Viz Engine Monitor
- **Enterprise ID:** `27566.3.1.7.` % `N`, where % `N` is the instance number without leading zeros.

15.1 Installation

```
msiexec /i "VizEngineMonitor-xxx.msi" /qn
```

Using the installer installs the application as a service. The service is installed as first instance by default. For further instances please refer to Install Service.

Before running the application, the configuration files must be adjusted for OpenTelemetry endpoints and logging destination.

15.2 Configuration

- Viz Engine has to be configured to write the status information.
- The Viz Engine Monitor requires OpenTelemetry endpoints for logs and metrics, that are up and running.
- These endpoints need to be configured in the Viz Engine Monitor Configuration.

15.2.1 Viz Engine Configuration

Viz Engine needs to be configured to write the OS shared memory. Please make sure `shm_system_status_enabled` is set to `1` in the Viz Engine configuration file.

15.2.2 OpenTelemetry Endpoints

The Viz Engine Monitor expects end endpoint for logs, and an endpoint for metrics for OpenTelemetry HTTP transport. They usually look like:

- *http://hostname:4318/v1/logs*
- *http://hostname:4318/v1/metrics*

15.2.3 Viz Engine Monitor Configuration

The configuration files are located at *%PROGRAMDATA%\\Vizrt\\Engine_Monitor\\cfg*. The location may be changed with application arguments. See also usage below.

Copyright © 2026 Vizrt

The configuration is done in *vizeng_monitor%N.json*, or, if it cannot find this file, in *vizeng_monitor.json*. The `%N` is replaced by the instance number with a two digit format.

For the first instance, the configuration file is searched at *vizeng_monitor01.json* and then at *vizeng_monitor.json*. If the instance number is not found, the first instance is used.

Example configuration file *vizeng_monitor01.json*:

```
{
  "version": "1.0",
  "sleep_time_ms": 100,
  "verbose_messages": false,
  "report_expected_messages": false,
  "metric_config": "otel_metrics_1"
}
```

Parameters for Engine Monitor Configuration

- **version:** The version of the configuration file. This is used to check if the configuration file is compatible.
- **sleep_time_ms:** The time in milliseconds between two runs. Default is *100* milliseconds.
- **verbose_messages:** Logs more details in each log message, when set to *true*. Default is *false*.
- **report_expected_messages:** Emits expected failures as well as regular info logs, when set to *true*. Default is *false*.
- **metric_config:** The `id` to use in the metrics array of Viz component logger configuration. Default is `otel_metrics_1`.

Usually an instance of this application is installed for each Viz Engine to monitor.

15.2.4 Viz Component Log Configuration

The configuration for logging is located in *vcl_config%N.json*, or, if it cannot find this file, in *vcl_config.json*. The `%N` is replaced by the instance number with a two digit format. If the instance number is not found, the first instance is used.

For example, for the third instance, the configuration file is searched at *vcl_config03.json* and then at *vcl_config.json*.

OpenTelemetry Logs Endpoint

In the default configuration, the metrics endpoint is located at `backend/concrete/otel_log/exporter/http/endpoint`.

```
"backend": [
  {
    "id": "backend_id_otel",
    "concrete": {
      "otel_log": {
        "exporter": {
```

```
          "type": "http",
          "http": {
            "endpoint": "http://127.0.0.1:4318/v1/logs",
            ...
```

OpenTelemetry Metrics Endpoint

In the default configuration, the metrics endpoint is located at `metrics/concrete/otel_metrics/exporter/http/endpoint`.

```
"metrics": [
  {
    "id": "otel_metrics_1",
    "concrete": {
      "otel_metrics": {
        "exporter": {
          "type": "http",
          "http": {
            "endpoint": "http://localhost:4318/v1/metrics",
            ...
```

15.3 Usage

```
"%ProgramFiles%\Vizrt\Engine_Monitor\vizeng_monitor.exe" -h
```

The application can be started as a service or as a console application. The service is by default installed as first instance, by the installer.

For debugging, the console application can be started with the `-c` parameter. Usually, the console mode is combined with a path to the configuration and log files, `-p`.

```
vizeng_monitor.exe -c -p "some\where\" -n 3
```

In this case, the configuration files are searched at *some\\where\\cfg\\license_monitor03.json*, *some\\where\\cfg\\license_monitor.json* and the log files are stored at *some\\where\\log*.

The service may also be installed, uninstalled, started, or stopped.

15.3.1 Install Service

To install a service manually (for example, for multiple instances on one host), the `-n` parameter is used to specify the instance number.

Copyright © 2026 Vizrt

```
vizeng_monitor.exe -i [-n instance]
```

- `vizeng_monitor.exe -i -n 3` registers a service with the name.
- `Vizrt Viz Engine Monitor_3` for the third instance.

15.3.2 Uninstall Service

To uninstall a service manually (for example, for multiple instances on one host), the `-n` parameter is used to specify the instance number.

```
vizeng_monitor.exe -u [-n instance]
```

- `vizeng_monitor.exe -u` unregisters the service with the name.
- `Vizrt Viz Engine Monitor_1` for the first instance.

15.3.3 Start Installed Service

To start a registered service manually (for example, for multiple instances on one host), the `-n` parameter is used to specify the instance number.

```
vizeng_monitor.exe -r [-n instance]
```

- `vizeng_monitor.exe -r -n 2` starts the registered service with the name.
- `Vizrt Viz Engine Monitor_2` for the second instance.

15.3.4 Stop Installed Service

To stop a registered service manually, the `-n` parameter is used to specify the instance number.

```
vizeng_monitor.exe -k [-n instance]
```

- `vizeng_monitor.exe -k -n 2` stops a running service with the name.
- `Vizrt Viz Engine Monitor_2` for the second instance.

15.4 Troubleshooting

The application uses a bootstrap logger that writes to the console, and this logger is used to log errors during startup. In case the logs are not emitted where they should be, the bootstrap logger shows any log configuration

Copyright © 2026 Vizrt

errors. For this to work, the application must be started in *console* mode. A registered service does not need to be unregistered first, but it is recommended to stop the service before starting the application in console mode.

```
vizeng_monitor.exe -c ...
```

Using a log file instead of OpenTelemetry is a good way to check if the application is running correctly. See configuration above.

15.5 Logged Information

For each iteration, the following log records are logged.

Common to all log records, are these key/value pairs in the `default` group:

- **host.name:** The hostname of the origin of the message.
- **service.instance.id:** The instance number of the application.
- **service.name:** The name of the application, `license_monitor`.
- **app_version:** The version of the application.
- **vizrt.service.oid:** The OID of the application, `27566.3.1.7.` % `N`, where % `N` is the instance number without leading zeros.

15.5.1 Shared Memory Information Logs

Logged as `shm metric` in the `vizrt.details` group:

- **metrics_for_oid:** The OID for the app to which this log belongs to. `27566.3.3.1.` % `N`, where % `N` is the instance number without leading zeros.
- **TCPPort:** TCP port the application is listening to.
- **ComputerType:** String identifying the computer type.
- **SystemID:** String identifying the system ID.
- **VizVersionStr:** Version of Viz Engine.
- **SceneFront:** Scene loaded into the front layer.
- **SceneMid:** Scene loaded into the middle layer.
- **SceneBack:** Scene loaded into the back layer.
- **ScenePostFront:** Scene loaded into the front layer in post mode.
- **ScenePostMid:** Scene loaded into the middle layer in post mode.
- **ScenePostBack:** Scene loaded into the back layer in post mode.

15.5.2 Shared Memory Information Metrics

Emitted with %N replaced by the instance id.

- `shm.27566.3.3.1.` %N `.OnAir_gauge`: True if On-Air.
- `shm.27566.3.3.1.` %N `.ConnectedToDb_gauge`: True if connected to a Graphic Hub.
- `shm.27566.3.3.1.` %N `.GenlockStatus_gauge`: Status of Genlock.

Copyright © 2026 Vizrt

- `shm.27566.3.3.1.%N.MaxFrameRate_gauge` : Max framerate in frames per second.
- `shm.27566.3.3.1.%N.VideoIn.1_gauge` : Status of Video Input 1.
- `shm.27566.3.3.1.%N.VideoIn.2_gauge` : Status of Video Input 2.
- `shm.27566.3.3.1.%N.VideoIn.3_gauge` : Status of Video Input 3.
- `shm.27566.3.3.1.%N.VideoIn.4_gauge` : Status of Video Input 4.
- `shm.27566.3.3.1.%N.VideoIn.5_gauge` : Status of Video Input 5.
- `shm.27566.3.3.1.%N.VideoIn.6_gauge` : Status of Video Input 6.
- `shm.27566.3.3.1.%N.VideoIn.7_gauge` : Status of Video Input 7.
- `shm.27566.3.3.1.%N.VideoIn.8_gauge` : Status of Video Input 8.
- `shm.27566.3.3.1.%N.ClipIn.1_gauge` : Status of Clip Input 1.
- `shm.27566.3.3.1.%N.ClipIn.2_gauge` : Status of Clip Input 2.
- `shm.27566.3.3.1.%N.ClipIn.3_gauge` : Status of Clip Input 3.
- `shm.27566.3.3.1.%N.ClipIn.4_gauge` : Status of Clip Input 4.
- `shm.27566.3.3.1.%N.ClipIn.5_gauge` : Status of Clip Input 5.
- `shm.27566.3.3.1.%N.ClipIn.6_gauge` : Status of Clip Input 6.
- `shm.27566.3.3.1.%N.ClipIn.7_gauge` : Status of Clip Input 7.
- `shm.27566.3.3.1.%N.ClipIn.8_gauge` : Status of Clip Input 8.
- `shm.27566.3.3.1.%N.TextureMemory.Free_gauge` : Free texture memory in KB.
- `shm.27566.3.3.1.%N.TextureMemory.Total_gauge` : Total texture memory in KB.
- `shm.27566.3.3.1.%N.TextureMemory.Used_gauge` : Used texture memory in %.
- `shm.27566.3.3.1.%N.Post_gauge` : Post mode is active.
- `shm.27566.3.3.1.%N.IsNLE_gauge` : NLE mode is active.
- `shm.27566.3.3.1.%N.ReadyForConnect_gauge` : Viz Engine is initialized and ready to receive commands.
- `shm.27566.3.3.1.%N.RetraceCounterMoving_gauge` : true if retrace counter changed since the last run.

Where Status for Genlock, Video and Clip Inputs is one of the following:

- **0:** OK
- **1:** Wrong Format
- **2:** Bad
- **3:** Dropped Frames
- **255:** Not Available

15.6 Complete Engine Configuration File

Example configuration file *vcl_config.json*:

```
{
```

Copyright © 2026 Vizrt

```
"async": [
  {
    "id": "async_1_id",
    "num_items": 1200,
    "num_workers": 4,
    "overflow_policy": "overrun_oldest"
  }
],
"attribute": [
  {
    "id": "vcl_config_logger_attributes_id",
    "multi_attributes": [
      {
        "sdid_name": "config",
        "single_attributes": [
          {
            "param_name": "facility",
            "param_value": "vcl config"
          }
        ]
      }
    ]
  }
],

"formatter": [
  {
    "id": "formatter_txt",
    "concrete": {
      "formatter_simple": {
        "attr_prefix": "\n    ",
        "attr_assign": "=",
        "attr_postfix": "",
        "attr_separator": "",
        "attr_name_escape": "\n\r\t\"\\{} ",
        "attr_value_escape": "\n\r\t\"",
        "group_prefix": "\n  ",
        "group_assign": ": ",
        "group_postfix": "",
        "group_separator": "",
        "group_name_escape": "\n\r\t\"\\{} ",
        "groups_prefix": "",
        "groups_postfix": "",
        "payload_prefix": " <<",
        "payload_postfix": ">>",
        "payload_escape": "\n\r\t\""
      }
    }
  },
  {
    "id": "formatter_compact",
    "concrete": {
      "formatter_simple": {
```

```
          "attr_prefix": "{\"",
          "attr_assign": "\"=\"",
          "attr_postfix": "\"}",
          "attr_separator": ", ",
          "attr_name_escape": "\n\r\t\"\\{} ",
          "attr_value_escape": "\n\r\t\"",
          "group_prefix": "{ \"",
          "group_assign": "\": ",
          "group_postfix": " }",
          "group_separator": "; ",
          "group_name_escape": "\n\r\t\"\\{} ",
          "groups_prefix": "[",
          "groups_postfix": "]",
          "payload_prefix": " <<",
          "payload_postfix": ">> ",
          "payload_escape": "\n\r\t\""
        }
      }
    }
  ],

  "backend": [
    {
      "id": "collection_id_console",
      "concrete": {
        "backend_collection": {
          "backend_refs": [
            "backend_cout",
            "backend_id_logfile",
            "backend_id_otel"
          ]
        }
      }
    },
    {
      "id": "collection_id_service",
      "concrete": {
        "backend_collection": {
          "backend_refs": [
            "backend_id_logfile",
            "backend_id_otel"
          ]
        }
      }
    },
    {
      "id": "collection_id_control",
      "concrete": {
        "backend_collection": {
          "backend_refs": [
            "backend_id_control"
          ]
        }
```

```
    }
  },
  {
    "id": "backend_vcl_config",
    "concrete": {
      "spdlog": {
        "attribute_ref": "",
        "formatter_ref": "formatter_compact",
        "spdlog_format": "%v",
        "spdlog_time_type": "utc",
        "spdlog_eol": "default",
        "dup_filter_max_skip_duration": 10,
        "spdlog_log_level": "debug",
        "spdlog_flush_level": "debug",
        "rotating_file_sink": {
          "base_filename": "vcl_config_%N",
          "max_size_MB": 1024,
          "max_files": 20,
          "rotate_on_open": false
        }
      }
    }
  },
  {
    "id": "backend_cout_terse",
    "concrete": {
      "cout_terse": {
        "attribute_ref": ""
      }
    }
  },
  {
    "id": "backend_cout",
    "concrete": {
      "cout": {
        "formatter_ref": "formatter_txt"
      }
    }
  },
  {
    "id": "backend_id_logfile",
    "concrete": {
      "spdlog": {
        "attribute_ref": "",
        "formatter_ref": "formatter_txt",
        "spdlog_format": "%v",
        "spdlog_time_type": "utc",
        "spdlog_eol": "default",
        "dup_filter_max_skip_duration": 10,
        "spdlog_log_level": "debug",
        "spdlog_flush_level": "debug",
        "rotating_file_sink": {
          "base_filename": "id_%N_monitor",
```

```
        "max_size_MB": 1024,
        "max_files": 20,
        "rotate_on_open": false
      }
    }
  }
},
{
  "id": "backend_id_otel",
  "concrete": {
    "otel_log": {
      "exporter": {
        "type": "http",
        "grpc": {
          "endpoint": "localhost:4317",
          "use_ssl_credentials": false,
          "ssl_ca_cert_path": "",
          "max_threads": 4,
          "timeout_seconds": 1
        },
        "http": {
          "endpoint": "http://127.0.0.1:4318/v1/logs",
          "content_type": "",
          "use_json_name": false,
          "console_debug": false,
          "timeout_seconds": 1,
          "max_concurrent_requests": 0,
          "max_requests_per_connection": 0,
          "ssl_insecure_skip_verify": true,
          "ssl_ca_cert_path": "",
          "ssl_ca_cert_string": "",
          "ssl_client_key_path": "",
          "ssl_client_key_string": "",
          "ssl_client_cert_path": "",
          "ssl_client_cert_string": "",
          "ssl_min_tls": "",
          "ssl_max_tls": "",
          "ssl_cipher": "",
          "ssl_cipher_suite": "",
          "compression": "",
          "retry_policy_max_attempts": 0,
          "retry_policy_initial_backoff": 1.0,
          "retry_policy_max_backoff": 60.0,
          "retry_policy_backoff_multiplier": 1.5,
          "api_key": ""
        }
      }
    }
  }
},
{
  "id": "backend_id_control",
  "concrete": {
```

```
      "spdlog": {
        "attribute_ref": "",
        "formatter_ref": "formatter_compact",
        "spdlog_format": "%v",
        "spdlog_time_type": "utc",
        "spdlog_eol": "default",
        "dup_filter_max_skip_duration": 10,
        "spdlog_log_level": "debug",
        "spdlog_flush_level": "debug",
        "rotating_file_sink": {
          "base_filename": "id_%N_control",
          "max_size_MB": 1024,
          "max_files": 20,
          "rotate_on_open": false
        }
      }
    },
    {
      "id": "backend_id_mapping",
      "concrete": {
        "spdlog": {
          "attribute_ref": "",
          "formatter_ref": "formatter_compact",
          "spdlog_format": "%v",
          "spdlog_time_type": "utc",
          "spdlog_eol": "default",
          "dup_filter_max_skip_duration": 10,
          "spdlog_log_level": "debug",
          "spdlog_flush_level": "debug",
          "rotating_file_sink": {
            "base_filename": "id_%N_mapping",
            "max_size_MB": 1024,
            "max_files": 20,
            "rotate_on_open": false
          }
        }
      }
    }
  ],

  "xlate_logger": [
    { "": "default_logger_id_%N" },
    { "vizrt/vlc_config": "vcl_config_logger" },
    { "xx_vizrt/os_lib": "logger_id_control" },
    { "xx_vizrt/wibu": "logger_id_control" },
    { "vizrt/app/viz_engine/monitor": "logger_id_control" },
    { "vizrt/app/viz_engine/monitor/": "logger_id_control" },
    { "vizrt/app/viz_engine/monitor/console": "logger_id_console" },
    { "vizrt/app/viz_engine/monitor/service": "logger_id_service" },
    { "vizrt/app/viz_engine/monitor/console/mapping": "logger_id_mapping" },
    { "vizrt/app/viz_engine/monitor/service/mapping": "logger_id_mapping" }
  ],
```

Copyright © 2026 Vizrt

```
"logger": [
  {
    "id": "null_logger",
    "async_ref": "",
    "concrete": {
      "logger_simple": {
        "backend_ref": "",
        "attribute_ref": "",
        "should_log_filter_ref": "",
        "repeat_filter_ref": ""
      }
    }
  },
  {
    "id": "vcl_config_logger",
    "async_ref": "",
    "concrete": {
      "logger_simple": {
        "backend_ref": "backend_vcl_config",
        "attribute_ref": "vcl_config_logger_attributes_id",
        "should_log_filter_ref": "",
        "repeat_filter_ref": ""
      }
    }
  },
  {
    "id": "logger_id_control",
    "async_ref": "",
    "concrete": {
      "logger_simple": {
        "backend_ref": "collection_id_control",
        "attribute_ref": "",
        "should_log_filter_ref": "minimum_debug_id",
        "repeat_filter_ref": ""
      }
    }
  },
  {
    "id": "logger_id_console",
    "async_ref": "",
    "concrete": {
      "logger_simple": {
        "backend_ref": "collection_id_console",
        "attribute_ref": "",
        "should_log_filter_ref": "minimum_debug_id",
        "repeat_filter_ref": ""
      }
    }
  },
  {
```

```
      "id": "logger_id_service",
      "async_ref": "",
      "concrete": {
        "logger_simple": {
          "backend_ref": "collection_id_service",
          "attribute_ref": "",
          "should_log_filter_ref": "minimum_info_id",
          "repeat_filter_ref": ""
        }
      }
    },
    {
      "id": "logger_id_mapping",
      "async_ref": "",
      "concrete": {
        "logger_simple": {
          "backend_ref": "backend_id_mapping",
          "attribute_ref": "",
          "should_log_filter_ref": "minimum_info_id",
          "repeat_filter_ref": ""
        }
      }
    }
  ],


  "repeat_filter": [
    {
      "id": "repeat_filter_id",
      "concrete": {
        "repeat_error_code": {
          "minimum_level": "error",
          "max_repeat_count": 100,
          "max_age": 10
        }
      }
    },
    {
      "id": "repeat_param_value_id",
      "concrete": {
        "repeat_error_code": {
          "minimum_level": "warning",
          "max_repeat_count": 20,
          "max_age": 30
        }
      }
    }
  ],


  "should_log_filter": [
    {
      "id": "minimum_debug_id",
```

Copyright © 2026 Vizrt

```
      "concrete": {
        "minimum_severity": {
          "minimum_level": "debug"
        }
      }
    },
    {
      "id": "minimum_debug_fixed_id",
      "concrete": {
        "minimum_severity_fixed": {
          "minimum_level": "debug"
        }
      }
    },
    {
      "id": "minimum_info_id",
      "concrete": {
        "minimum_severity": {
          "minimum_level": "info"
        }
      }
    },
    {
      "id": "minimum_info_fixed_id",
      "concrete": {
        "minimum_severity_fixed": {
          "minimum_level": "info"
        }
      }
    }
  ],
  "metrics": [
    {
      "id": "otel_metrics_1",
      "concrete": {
        "otel_metrics": {
          "export_interval_ms": 1000,
          "export_timeout_ms": 500,
          "exporter": {
            "type": "http",
            "grpc": {
              "endpoint": "localhost:4317",
              "use_ssl_credentials": false,
              "ssl_ca_cert_path": "",
              "max_threads": 4,
              "timeout_seconds": 1,
              "aggregation_temporality": "cumulative"
            },
            "http": {
              "endpoint": "http://localhost:4318/v1/metrics",
              "content_type": "",
              "use_json_name": false,
              "console_debug": false,
```

```
              "timeout_seconds": 1,
              "max_concurrent_requests": 0,
              "max_requests_per_connection": 0,
              "ssl_insecure_skip_verify": true,
              "ssl_ca_cert_path": "",
              "ssl_ca_cert_string": "",
              "ssl_client_key_path": "",
              "ssl_client_key_string": "",
              "ssl_client_cert_path": "",
              "ssl_client_cert_string": "",
              "ssl_min_tls": "",
              "ssl_max_tls": "",
              "ssl_cipher": "",
              "ssl_cipher_suite": "",
              "compression": "",
              "retry_policy_max_attempts": 0,
              "retry_policy_initial_backoff": 1.0,
              "retry_policy_max_backoff": 60.0,
              "retry_policy_backoff_multiplier": 1.5,
              "api_key": "",
              "aggregation_temporality": "cumulative"
            }
          }
        }
      }
    }
  ],
  "version": "1.0"
}
```

15.6.1 Parameters for Viz Component Log Configuration

The loggers used are found under `xlate_logger`:

- `vizrt/vlc_config` logger for Viz component logger. Leave it to vcl_config_logger to have a bootstrap log file.
- `vizrt/os_lib` logger for os lib component. This entry is missing by default to inherit the logger.
- `vizrt/wibu` logger for WIBU component. This entry is missing by default to inherit the logger.
- `vizrt/app/viz_engine/monitor` and `vizrt/app/viz_engine/monitor/` logger for the control mode operations, such as *install as a service*.
- `vizrt/app/viz_engine/monitor/console` logger when running in console mode.
- `vizrt/app/viz_engine/monitor/service` logger when running in service mode.
- `vizrt/app/viz_engine/monitor/console/mapping` logger to map obstructed sensitive data as hash to real data when running in console mode. Not yet used.
- `vizrt/app/viz_engine/monitor/service/mapping` logger to map obstructed sensitive data as hash to real data when running in service mode. Not yet used.

The usual customization points are then:

Copyright © 2026 Vizrt

- `vizrt/app/viz_engine/monitor/console` This logs by default to console and log files.
- `vizrt/app/viz_engine/monitor/service` This logs by default to log files and Opentelemetry endpoints.

Look up the translated logger and follow any backend_ref or backend_refs until you reach a concrete backend, such as otel, ... and configure their endpoints.

Example Backend otel Log

```
{
  "id": "backend_id_otel",
  "concrete": {
    "otel_log": {
      "exporter": {
        "type": "http",
        "http": {
          "endpoint": "<http://127.0.0.1:4318/v1/logs>",
          "timeout_seconds": 1,
          "max_concurrent_requests": 0,
          "max_requests_per_connection": 0,
          "ssl_insecure_skip_verify": true,
        }
      }
    }
  }
}
```

Copyright © 2026 Vizrt